David Derler

A Modular Framework for Privacy-Enhancing Signatures: Generalizations, Extensions, and Novel Building Blocks

DISSERTATION

zur Erlangung des akademischen Grades

Doktor der technischen Wissenschaften

eingereicht an der

Technischen Universität Graz

Gutachter

Prof. Dr. Christian Rechberger (Technische Universität Graz)

Prof. Dr. Jens Groth (University College London)

Graz, Juli 2017

Eidesstattliche Erklärung

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst, andere als die angegebenen Quellen/Hilfsmittel nicht benutzt, und die den benutzten Quellen wörtlich und inhaltlich entnommenen Stellen als solche kenntlich gemacht habe. Das in TUGRAZonline hochgeladene Textdokument ist mit der vorliegenden Dissertation identisch.

Graz, Juli 2017

Kurzfassung

Neue IT-Paradigmen wie Cloud-Computing haben in den letzten Jahren signifikante Aufmerksamkeit auf sich gezogen und viele Unternehmen lagern mittlerweile Daten, Berechnungen, oder Dienste zu Cloud-Service-Providern aus um Kostenreduktionen und/oder Effizienzsteigerungen zu erreichen. Gleichzeitig tragen wir portable Computer in unseren Taschen, die den ubiquitären Zugriff auf die durch diesen Trend entstehenden verteilten Infrastrukturen und Dienste ermöglichen. Während diese neuen Paradigmen viele Vorteile bieten, bringen sie auch eine Vielzahl an offenen Fragen mit sich. Viele dieser Fragen ergeben sich daraus, dass die involvierten Daten von (typischerweise nicht vollständig vertrauenswürdigen) Dritten verarbeitet werden und betreffen die Sicherheit der Daten und die Privatspähre der Nutzer.

In dieser Arbeit adressieren wir die Frage wie man mit kryptographischen Methoden die Authentizität der verarbeiteten Daten sicherstellen kann, während man gleichzeitig die Privatsphäre der involvierten Parteien bestmöglich schützt. Dabei legen wir den Fokus auf zwei (sich teilweise überschneidende) Aspekte von Privatsphäre. Zum einen adressieren wir die Privatsphäre der Partei, welche die Daten authentisiert (im folgenden Signator). In diesem Zusammenhang kommt es häufig vor, dass der Signator mehr persönliche Informationen preisgibt als für die Authentisierung notwendig wären. Um dies zu unterbinden schlagen wir kryptographische Schemen und Protokolle vor, welche inhärent sicherstellen, dass – abgesehen davon dass die authentisierten Daten von einem Mitglied einer authorisierten Gruppe stammen – keine Informationen über den Signator preisgegeben werden. Zum Anderen zielen wir auf Privatsphärenaspekte im Kontext der authentisierten Daten selbst ab. Authentisierte (signierte) Daten enthalten oft sensible Informationen, und deren Weitergabe an unauthorisierte Parteien kann schwerwiegende Verletzungen der Privatsphäre darstellen. Aus diesem Grund ist es wichtig Möglichkeiten zur Verfügung zu stellen um sensible Teile vor der Weitergabe der Daten zu entfernen oder zu ersetzen. Gleichzeitig ist es jedoch wichtig die Authentizität der nicht entfernten oder ersetzten Teile der Daten garantieren zu können. Unsere Arbeit in diese Richtung beschäftigt sich mit kryptographischen Schemen und Protokollen die eine Modifikation von authentisierten Daten in einer kontrollierten, vom Signator definierten Art und Weise erlauben, während die Authentizitätsgarantien erhalten bleiben.

Von einem technischen Blickwinkel erweitert diese Arbeit den aktuellen Wissensstand bezüglich kryptographischen Schemen und Protokollen in den oben genannten Interessensgebieten. Wir präsentieren sowohl neue Paradigmen und Konstruktionen, als auch Generalisierungen und Erweiterungen existierender Pa-

radigmen und Konstruktionen. Dabei folgen wir einem modularen Ansatz, welcher unsere Konstruktionen konzeptionell einfach und leicht verständlich macht. Während Modularität oft auf Kosten reduzierter Effizienz erreicht wird, sind die Paradigmen die unseren Konstruktionen unterliegen auch von einem praktischen Blickwinkel attraktiv.

Abstract

New computing paradigms such as cloud computing attracted significant attention in recent years and meanwhile numerous enterprises outsource data, computations or services to cloud computing providers for flexibility and/or cost efficiency reasons. In addition, we carry computing devices in our pockets, which allow to ubiquitously access the distributed infrastructures and services emerging from this trend. While these novel paradigms have many advantages, they raise open questions in various directions. Many of those questions are related to security and privacy and arise from processing data at (typically not fully trusted) third parties.

We address the question as how to cryptographically ensure the authenticity of processed data, while at the same time maintaining the privacy of the involved parties. Thereby, we focus on two (partially intersecting) aspects of privacy. First, we target privacy with respect to the party who authenticates the data, i.e., the signer. There are many scenarios where signers reveal more personal information than actually required upon authentication. Our work in this direction aims to counter this by cryptographic schemes and protocols, which inherently ensure that—beyond the fact that the authenticated data stems from a member of some authorized group—nothing is revealed about the signer. Second, we target privacy with respect to the authenticated data itself. In particular, authenticated data often contains sensitive information and disclosing this information to unauthorized parties may pose severe privacy issues. To this end, it is important to have means to remove or replace the privacy sensitive parts before disclosure. At the same time, it is important to guarantee authenticity of the parts of the data which were not removed or replaced. Our work in this direction thus covers cryptographic schemes and protocols which allow modifications of authenticated data in a controlled, signer-defined manner, while still upholding the authenticity guarantees.

From a technical point of view, this thesis improves upon the state of the art of provably secure cryptographic schemes and protocols within the areas of interest outlined above. More precisely, we present novel paradigms and constructions, as well as generalizations and extensions of existing paradigms and constructions. Thereby we follow a modular approach which makes our constructions conceptually simple and easy to understand. While modularity often comes at the cost of reduced efficiency, we stress that the paradigms underlying our constructions are also appealing from a practical point of view.

Acknowledgements

The completion of a PhD project requires the support of many people. I am very grateful that I have received this support and want to express my gratitude below. First and foremost, I would like to thank my colleague and friend Daniel Slamanig for inspiring me to start working in the field wherein I work now, for providing me with the necessary tools to be able to conduct research in this field, for serving as my mentor throughout the whole PhD project, and for all the fun time in general. Furthermore, I would like to thank my advisor Christian Rechberger for his guidance and support. I would also like to thank Stefan Mangard, who served as my advisor before Christian joined IAIK. Moreover, I would like to thank Jens Groth for his excellent feedback on a preliminary version of this thesis and for serving as the external assessor of this thesis. In addition, I would like to express my thanks to all my co-authors for the great collaborations, and all my co-authors and colleagues at IAIK for providing an inspiring and enjoyable environment. I would also like to thank Tibor Jager for hosting me at University Paderborn for a research visit. Finally, I would like to thank my family and friends for providing the supportive environment which made this PhD project possible at all.

David Derler
Graz, July 2017

Table of Contents

List of Publications

Refereed Conference Proceedings

[c1] Michael Till Beck, Jan Camenisch, David Derler, Stephan Krenn, Henrich C. Pöhls, Kai Samelin, and Daniel Slamanig. Practical strongly invisible and strongly accountable sanitizable signatures. In *Information Security and Privacy - 22nd Australasian Conference, ACISP 2017, Auckland, New Zealand, July 3-5, 2017, Proceedings, Part I*, pages 437–452, 2017. Full Version: *IACR Cryptology ePrint Archive*, 2017:445.

[c2] Jan Camenisch, David Derler, Stephan Krenn, Henrich C. Pöhls, Kai Samelin, and Daniel Slamanig. Chameleon-Hashes with Ephemeral Trapdoors and Applications to Invisible Sanitizable Signatures. In *Public-Key Cryptography - PKC 2017 - 20th IACR International Conference on Practice and Theory in Public-Key Cryptography, Amsterdam, The Netherlands, March 28-31, 2017, Proceedings, Part II*, pages 152–182, 2017. Full Version: *IACR Cryptology ePrint Archive*, 2017:11.

[c3] David Derler, Sebastian Ramacher, and Daniel Slamanig. Homomorphic proxy re-authenticators and applications to verifiable multi-user data aggregation. In *Financial Cryptography and Data Security - 21st International Conference, FC 2017, Sliema, Malta, April 3-7, 2017, Revised Selected Papers*, 2017. To Appear. Full Version: *IACR Cryptology ePrint Archive*, 2017:86.

[c4] Olivier Blazy, David Derler, Daniel Slamanig, and Raphael Spreitzer. Non-interactive plaintext (in-)equality proofs and group signatures with verifiable controllable linkability. In *Topics in Cryptology - CT-RSA 2016 - The Cryptographers' Track at the RSA Conference 2016, San Francisco, CA, USA, February 29 - March 4, 2016, Proceedings*, pages 127–143, 2016. Full Version: *IACR Cryptology ePrint Archive*, 2016:82.

[c5] David Derler, Stephan Krenn, and Daniel Slamanig. Signer-anonymous designated-verifier redactable signatures for cloud-based data sharing. In *Cryptology and Network Security - 15th International Conference, CANS 2016, Milan, Italy, November 14-16, 2016, Proceedings*, pages 211–227, 2016. Full Version: *IACR Cryptology ePrint Archive*, 2016:1064.

[c6] David Derler, Christian Hanser, Henrich C. Pöhls, and Daniel Slamanig. Towards authenticity and privacy preserving accountable workflows. In *Privacy and Identity Management. Time for a Revolution? - 10th IFIP WG 9.2, 9.5, 9.6/11.7, 11.4, 11.6/SIG 9.2.2 International Summer School, Edinburgh, UK, August 16-21, 2015, Revised Selected Papers*, pages 170–186, 2015.

[c7] David Derler, Christian Hanser, and Daniel Slamanig. A new approach to efficient revocable attribute-based anonymous credentials. In *Cryptography and Coding - 15th IMA International Conference, IMACC 2015, Oxford, UK, December 15-17, 2015. Proceedings*, pages 57–74, 2015.

[c8] David Derler, Christian Hanser, and Daniel Slamanig. Revisiting cryptographic accumulators, additional properties and relations to other primitives. In *Topics in Cryptology - CT-RSA 2015, The Cryptographer's Track at the RSA Conference 2015, San Francisco, CA, USA, April 20-24, 2015. Proceedings*, pages 127–144, 2015. Full Version: *IACR Cryptology ePrint Archive*, 2015:87.

[c9] David Derler, Henrich C. Pöhls, Kai Samelin, and Daniel Slamanig. A general framework for redactable signatures and new constructions. In *Information Security and Cryptology - ICISC 2015 - 18th International Conference, Seoul, South Korea, November 25-27, 2015, Revised Selected Papers*, pages 3–19, 2015. Full Version: *IACR Cryptology ePrint Archive*, 2015:1059.

[c10] David Derler and Daniel Slamanig. Rethinking privacy for extended sanitizable signatures and a black-box construction of strongly private schemes. In *Provable Security - 9th International Conference, ProvSec 2015, Kanazawa, Japan, November 24-26, 2015, Proceedings*, pages 455–474, 2015. Full Version: *IACR Cryptology ePrint Archive*, 2015:843.

[c11] David Derler, Christian Hanser, and Daniel Slamanig. Blank digital signatures: Optimization and practical experiences. In *Privacy and Identity Management for the Future Internet in the Age of Globalisation - 9th IFIP WG 9.2, 9.5, 9.6/11.7, 11.4, 11.6/SIG 9.2.2 International Summer School, Patras, Greece, September 7-12, 2014, Revised Selected Papers*, pages 201–215, 2014.

[c12] David Derler, Christian Hanser, and Daniel Slamanig. Privacy-enhancing proxy signatures from non-interactive anonymous credentials. In *Data and Applications Security and Privacy XXVIII - 28th Annual IFIP WG 11.3 Working Conference, DBSec 2014, Vienna, Austria, July 14-16, 2014. Proceedings*, pages 49–65, 2014. Full Version: *IACR Cryptology ePrint Archive*, 2014:285.

[c13] David Derler, Klaus Potzmader, Johannes Winter, and Kurt Dietrich. Anonymous ticketing for NFC-enabled mobile phones. In *Trusted Systems - Third International Conference, INTRUST 2011, Beijing, China, November 27-29, 2011, Revised Selected Papers*, pages 66–83, 2011.

Articles

[j1] Moritz Horsch, David Derler, Christof Rath, Hans-Martin Haase, and Tobias Wich. Open Source für Europäische Signaturen. *Datenschutz und Datensicherheit*, 38(4):237–241, 2014.

Preprints/Manuscripts

[i1] Melissa Chase, David Derler, Steven Goldfeder, Claudio Orlandi, Sebastian Ramacher, Christian Rechberger, Daniel Slamanig, and Greg Zaverucha. Post-quantum zero-knowledge and signatures from symmetric-key primitives. *IACR Cryptology ePrint Archive*, 2017:279, 2017. Merge of [i3] and [GCZ16].

[i2] David Derler, Stephan Krenn, Thomas Lorünser, Sebastian Ramacher, Daniel Slamanig, and Christoph Striecks. Forward-secure proxy re-encryption and relations to HIBE and puncturable encryption, 2017. Manuscript.

[i3] David Derler, Claudio Orlandi, Sebastian Ramacher, Christian Rechberger, and Daniel Slamanig. Digital signatures from symmetric-key primitives. *IACR Cryptology ePrint Archive*, 2016:1085, 2016.

[i4] David Derler and Daniel Slamanig. Fully-anonymous short dynamic group signatures without encryption. *IACR Cryptology ePrint Archive*, 2016:154, 2016.

[i5] David Derler and Daniel Slamanig. Key-homomorphic signatures and applications to multiparty signatures and non-interactive zero-knowledge. *IACR Cryptology ePrint Archive*, 2016:792, 2016.

[i6] David Derler and Daniel Slamanig. Practical witness encryption for algebraic languages or how to encrypt under Groth-Sahai proofs. *IACR Cryptology ePrint Archive*, 2015:1073, 2015.

Theses

[t1] David Derler. On the optimization of two recent proxy-type digital signature schemes and their efficient implementation in Java. Master's thesis, Graz University of Technology, 2013.

Selected Project Deliverables

[d1] Johannes Buchmann, Denise Demirel, David Derler, Lucas Schabhüser, Daniel Slamanig, and Thomas Gross. Overview of verifiable computing

techniques providing private and public verification. PRISMACLOUD D5.8, 2015. https://prismacloud.eu.

[d2] Denise Demirel, David Derler, Christian Hanser, Henrich Pöhls, Daniel Slamanig, and Giulia Traverso. Overview of functional and malleable signature schemes. PRISMACLOUD D4.4, 2015. https://prismacloud.eu.

1 Introduction

In recent years, computing has experienced a paradigm shift. On the one hand, enterprises increasingly follow the trend to outsource data, computations, or services to third party infrastructure (i.e., to cloud computing providers) for cost efficiency and/or flexibility reasons. On the other hand, computing became ubiquitous. That is, we face computers in our daily lives, which allow us to access the distributed infrastructures and services emerging from this trend from virtually everywhere. Both aforementioned paradigms are well accepted, many interesting business models exist in both contexts, and a wide variety of solutions are already deployed. Yet, they pose new challenges in various directions. Many of those challenges are related to security and privacy (see, e.g., [LRD+15]) and arise from storing and processing data at some typically not fully trusted third party. In this thesis we address the question as how to reduce the trust assumptions in the entities processing the data. In particular, we aim to cryptographically ensure the authenticity of the processed data, while at the same time maintaining the privacy of the involved parties. Our contributions target two different aspects of privacy in this context, which we will further discuss below.

Privacy with Respect to the Signer. When authenticating (signing) data, one often reveals more information than necessary for the respective task. In particular, in many scenarios it is sufficient to demonstrate that the authenticated data stems from a member of some authorized group, while not revealing any other potentially privacy sensitive information about the actual signer. Prominent examples for such scenarios can, e.g., be found in the steadily increasing area of intelligent transportation systems. This area spans private as well as public transportation and includes systems for vehicle to vehicle communica-

tion, floating car data, electronic toll systems, electronic parking, and smart ticketing. Unfortunately, practical implementations of such systems often put privacy features aside in favor of other features—even though the requirement to ensure privacy of users in those scenarios is meanwhile explicitly stated in EU Directive 2010/40/EU. To foster a broad acceptance and usage of privacy respecting systems it is important to develop solutions which produce as little overhead regarding computational complexity and bandwidth as possible. Privacy-enhancing cryptographic primitives like anonymous credentials [Cha85], group signature schemes [CvH91], or ring signature schemes [RST01] are useful tools to cryptographically address the privacy issues in this context. Anonymous credentials allow organizations to issue credentials (e.g., digital ID cards) to users, who can then anonymously demonstrate possession of such a credential to every third party that trusts the respective organization. Thereby, modern credential systems consider credentials as a collection of attributes, and upon authentication users are able to reveal subsets of these attributes or even to prove certain predicates with respect to non-revealed attributes (e.g., the attribute birthdate has a value such that it holds that age > 18).[1] Group signatures are a very related concept and essentially allow a group manager to set up some group so that every member of this group can anonymously issue signatures on behalf of the group. Still, in case of a dispute, a dedicated entity called the opening authority can re-identify the actual signer for a given signature. In a sense, group signatures can thus informally be viewed as non-interactive counterparts to anonymous credentials with a single attribute certifying group membership. Finally, also ring signatures are useful in applications where signer anonymity is important. They are similar to group signatures, but signatures are generated with respect to ad-hoc groups (rings) and the actual signer remains unconditionally anonymous.[2]

Privacy with Respect to the Signed Data. When outsourcing data storage or processing to third-party infrastructure, one faces some obstacles. Among them is the question as how to ensure the authenticity of outsourced data. While conventional digital signatures provide a potential solution to the static setting where data are simply stored by third parties, we are interested in a dynamic setting. Here, the goal is to outsource data and to allow third parties to process them, i.e., to modify them in some controlled manner so that it is later possible to verify whether certain modifications were from the class of "allowed" modifications. Useful tools in this context are malleable signatures, i.e., signatures that allow to modify signed messages in a controlled manner and without invalidating the respective signatures. The expressiveness of the restrictions on modifications thereby varies from simply "blacking out" [JMSW02] or replacing [ACdMT05]

[1] We note that even though some anonymous credential systems are defined with respect to an interactive showing procedure, we categorize them as variants of signature schemes. For simplicity, we also use the term signer to refer to the party who performs the showing.

[2] Note that there are also concepts between group signatures and ring signatures. For instance, in accountable ring signatures [XY04, BCC$^+$15], signatures are generated with respect to ad-hoc groups, yet there is a dedicated opening authority which is able to re-identify the signer.

message parts, to the possibility to define rather generic policies that every signed message needs to fulfill [BF14], or even allowing to evaluate (arbitrary) functions on the signed messages [BF11, BGI14, GVW15].[3] In general, one can group malleable signature schemes into *functional signatures*, where the allowed computations can only be performed by a designated entity, and *homomorphic signatures*, where anyone can perform the allowed computations. When using malleable signatures to ensure authenticity of outsourced data, a signature can be issued on the data so that the service provider can apply allowed modifications while still upholding the validity of the signature. Another somewhat related application is the authentic documentation of outsourced processes for accountability and/or auditing reasons. While conventional signatures already deliver a means to hold the executing party accountable for its actions, i.e., by requiring a valid signature created by the executing party on a report documenting its actions, malleable signatures additionally allow the outsourcing party to predefine the structure of such a report (and possibly even certain restrictions).[4] In both scenarios sketched above, it is crucial that signatures on the modified data do not reveal information on the data that was removed by the modifications, followingly subsumed as privacy.

Intersections of Both. While the privacy aspects discussed above can be viewed in isolation in some applications, there are also scenarios where both aspects are relevant simultaneously. That is, scenarios where the identity of the signer as well as the document content may reveal privacy sensitive information. As an example, consider a health information sharing system. In such a system, doctors sign medical records of patients so that the patients can distribute authentic subsets of their records to further stakeholders such as other doctors or employers. Clearly, different stakeholders only need to see different portions of the signed documents and therefore one needs the ability to black out (remove) certain document parts while still maintaining the authenticity guarantees of the remaining parts. Orthogonal to that, one may observe that it will often be the case that only knowing the specialization of the signing doctor (which can be deduced from the doctor's identity) allows to infer privacy sensitive knowledge about the patient's illness, e.g., in case of an oncologist. Therefore, the doctor's identity is also an asset that deserves protection in such a system for the sake of increased patient privacy.

1.1 Results and Outline

Figure 1.1 clusters our results with respect to the areas of interest in this thesis and also incorporates pointers to the chapters/sections containing the actual contributions. In a nutshell, our results can be organized in different layers building upon each other.

[3] For an extensive overview of malleable signatures and related concepts, see [d2].
[4] Refer to [c6] for an extensive discussion from the application perspective.

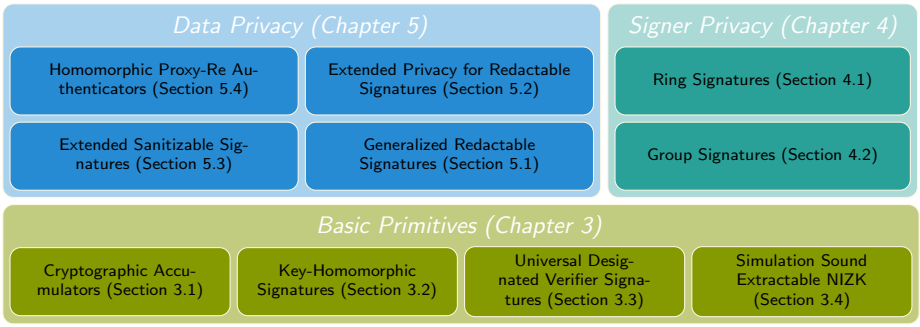

Figure 1.1: Overview of contributions.

At the bottom layer we focus on basic primitives. They can be seen as building blocks for schemes and protocols in the areas of interest in this thesis. Yet, they are also suitable for a broader range of applications in general. Building upon this layer, we have two clusters focusing on the two topics of interest posed above, i.e., authentication primitives with privacy features for the signer and authentication primitives with privacy features with respect to the authenticated data.

In the remainder of this section we give a high-level overview of our results. The more technical discussion is postponed to the introductions of the respective chapters/sections. Note that our overview partly borrows formulations from the abstracts/introductions of the referenced papers.

Basic Primitives. Below we subsume our results on basic primitives.

Cryptographic Accumulators [c8]. Cryptographic accumulators, firstly introduced in [BdM93], allow to succinctly represent a set $\mathcal{X}$ of values as a so-called accumulator $\Lambda_{\mathcal{X}}$. Thereby, for each member $x \in \mathcal{X}$, one can efficiently compute a witness attesting the membership of $x \in \Lambda_{\mathcal{X}}$, while this should be computationally infeasible for values $x \notin \mathcal{X}$ (*collision freeness*). Further, *dynamic* accumulators allow to dynamically add/delete values to/from the accumulator, whereas *universal* accumulators additionally provide means to obtain non-membership witnesses for non-members (i.e., values $x \notin \mathcal{X}$). In case of *universal* accumulators, collision-freeness also covers the infeasibility to compute non-membership witnesses for accumulated values. Besides that, accumulators may provide *undeniability* [BLL00, BLL02, Lip12] and *indistinguishability*. While undeniability can be seen as a stronger collision-freeness notion, indistinguishability requires that neither the accumulator nor corresponding witnesses leak information about the accumulated set. Cryptographic accumulators received quite some attention from the cryptographic community and many different formalizations considering different security notions exist.

We introduce a unified formalization of accumulators, where—in addition to collision freeness—we also consider undeniability and introduce the first mean-

ingful formalization of indistinguishability.[5] Such a formalization of accumulators yields a solid basis for generic constructions of primitives that build upon accumulators and they are used as central building blocks for three of the malleable signature schemes presented in this thesis. In addition to our unifying formalization, we formally analyze the relations between the different security properties of accumulators and classify existing constructions with respect to their security in our unified model. It thereby turns out that no existing construction provides indistinguishability in our model. To resolve this issue, we propose the first indistinguishable, dynamic accumulator scheme. In addition, we present a simple, light-weight generic transformation that can be applied to existing accumulators to obtain a slightly weaker form of indistinguishability. Furthermore, we analyze relations of accumulators to other primitives. Most interestingly, one of the relations we prove also yields the first construction of an undeniable, indistinguishable, universal accumulator.

Key-Homomorphic Signatures and Applications [i5]. As mentioned above, one of our main areas of interest is in the field of malleable signatures, i.e., signatures which possess certain homomorphic properties on their message space. For this result we turn to a related issue, namely signatures with homomorphic properties on their *key space*. Interestingly, signature schemes have not yet been explicitly investigated in this context so far.

We initiate the study of key-homomorphic signatures by introducing a framework that generalizes larger classes of existing signature schemes with respect to various natural variants of key-homomorphisms. Such a framework is especially interesting as it allows to make general statements about schemes from those classes and to obtain generic constructions building upon schemes from those classes. Our results on extended privacy for redactable signatures in Section 5.2 include one immediate example of a generic construction building upon our framework. Besides that, our results include elegant and simple compilers from classes of schemes admitting certain types of key homomorphisms to other interesting cryptographic primitives. In particular, we present multiple applications of our framework to different variants of (privacy-enhancing) signature schemes involving multiple parties, as well as an application to non-interactive proof systems providing strong security guarantees. In addition, we analyze various signature schemes with respect to their key-homomorphic properties, and thereby directly obtain various instantiations of our generic constructions being favorable regarding conceptual simplicity and efficiency when compared to existing constructions. Note that the applications of our framework which we deem to be more interesting in the context of this thesis are presented in separate sections (Section 3.3, Section 3.4, and Section 4.1). We also want to note that our results go beyond the scope of this thesis and refer the reader to Section 1.1.2 for a brief discussion of additional results.

[5] Subsequently to our work this notion has even been strengthened in [GOP+16]. Nevertheless, our indistinguishability notion is better suited for the applications where we use accumulators.

Signatures with Signer Privacy. Our results on schemes providing privacy features for the signer presented in this thesis include a ring signature scheme and a group signature scheme. The ring signature scheme constitutes one application of our framework for key-homomorphic signatures discussed above. To this end, we do not discuss it again and directly turn to our group signature scheme.

Group Signatures [i4]. Initial work on group signatures only introduced "intuitive" security notions and no comprehensive security model existed. To this end [BMW03], introduced the first formal security model. Subsequently, [BSZ05] extended the model in [BMW03] by considering the possibility to dynamically add/remove group members.[6] Their model is very strong and anonymity needs to hold even if the adversary sees arbitrary key exposures and arbitrary openings of other group signatures. Existing constructions achieving this strong security notion follow the so-called sign-encrypt-prove (SEP) paradigm [CS97], where creating a group signature involves proving knowledge of an encrypted membership certificate (issued by the group manager). An elegant alternative to this paradigm is the so-called sign-randomize-prove (SRP) paradigm [BCN+10]. Here, the group manager uses a signature scheme with randomizable signatures (e.g., [CL02a, PS16]) to sign a commitment to the user's secret key. Creating a group signature involves the randomization of the signature and proving knowledge of the signed secret key. While the SRP paradigm seems to be beneficial regarding efficiency, unfortunately all known constructions following this paradigm are proven secure in a model which is much weaker than the BSZ model.

We tackle the question whether schemes being secure in the strong BSZ model while following the SRP paradigm even exist at all. We answer it to the affirmative by presenting a surprisingly efficient construction. Our construction allows us to further push the computational efficiency limits regarding signature generation and verification with respect to all known constructions being secure in such strong models. Besides those efficiency gains, we even beat the popular BBS short group signature scheme [BBS04] regarding signature size. Compared to previous work, our construction uses slightly stronger, yet very plausible assumptions.

Signatures with Data Privacy. Below we review our results on signature schemes providing privacy features with respect to the signed data. Our results include various variants of malleable signature schemes.

Generalized Redactable Signatures [c9]. Redactable signature schemes allow to sign messages, where certain parts of the message may later be removed (i.e., redacted) by any party without signer interaction and without invalidating the signature. Redactable signatures were initially introduced in [SBZ01, JMSW02] and subsequently extended to support more sophisticated data structures such as trees [BBD+10, SPB+12a] or graphs [KB13]. All these constructions make

[6] Also note that [KY05] introduced an alternative formalization of dynamic group signatures providing similar guarantees as [BSZ05].

the data structure of the messages to be signed explicit in their security model and are therefore not usable to look at redactable signature schemes from a more general viewpoint. Besides the work cited above, much other work on redactable signatures exists. Unfortunately, nearly every proposed construction also introduces its own security model, yielding a rather messy terminology in the field of redactable signatures. Besides the usual unforgeability guarantees, an important requirement in most applications of redactable signatures is that redacted signatures do not reveal information about the redacted message parts. In the literature, this requirement is termed *privacy*. A related, stronger requirement is that signatures do not even reveal whether redactions have taken place. This property is called *transparency* in [BBD+10], while [CLX09] consider it as a variant of privacy.

We deem redactable signatures to be very useful tools to cryptographically address the questions we are interested in. To this end, we propose a generalized security model for redactable signatures that does not depend on the concrete data structure of the signed messages. Thereby, we closely align our notions and terminology with the standard security model for the related notion of sanitizable signatures [BFF+09] (as it is done in [BBD+10] for redactable signatures for trees) and take care that our new model is compatible with existing schemes. In addition, we introduce the notion of designated redactors, where the signer can (optionally) hand over some extra piece of information to the redactor(s). This extra piece of information often allows to obtain more efficient schemes. Furthermore, we present three generic constructions of redactable signatures for sets and linear documents which make—among others—black-box usage of indistinguishable cryptographic accumulators. Besides yielding a flexible framework with many possible instantiations, two of our constructions can also be viewed as generalizations of existing lines of work.

Extended Privacy for Redactable Signatures [c5]. While plain redactable signatures already address important issues in the context of privacy preserving processing of authentic documents, one may observe that—depending on the type of processed document—there may be other privacy sensitive assets which deserve protection. In particular, redacting certain parts of a document might not be as useful as intended if already the identity of the signer allows to infer privacy sensitive information. Thus another important issue to address in this context is to conceal the identity of the signer, as it is for example known from group signatures. Besides that, one may observe that everyone getting to hold a valid (redacted) signature will be able to convince others of the authenticity of the associated document by simply publishing the signature. Again, this might be too privacy invasive in certain scenarios. It would be better to have a similar feature as provided by universal designated verifier (UDV) signatures [SBWP03]. In UDV signatures, the owner of the data can convert publicly verifiable signatures to signatures which are *only* useful to convince one particular designated entity of the authenticity of the respective document.

We address the extension of redactable signatures according to the requirements sketched above. While the features we would require are provided by

distinct cryptographic schemes which have been studied in isolation, it is well known that a naïve combination of different cryptographic primitives to a larger system is often problematic. In particular, without a thorough specification and analysis of the resulting combined scheme, subtle issues often remain undetected. To this end, we rigorously formalize our desired primitive which we dub *signer-anonymous designated-verifier redactable signatures*. Based on our formalization, we present two constructions and prove them secure in our newly introduced model. The first one is a generic construction which is more of theoretical interest. In contrast, for the second construction we build upon our generic framework for redactable signatures and also leverage our results on key-homomorphic signature schemes to obtain a particularly efficient instantiation. We also confirm our efficiency claims with implementation results.

Extended Sanitizable Signatures [c10]. Sanitizable signatures were initially introduced in [ACdMT05] and later formalized in [BFF+09]. They allow to sign messages, where some dedicated party (the sanitizer) may later change (sanitize) certain predefined parts of the signed message without invalidating the signature and without signer interaction. Essentially, sanitizable signatures are required to provide the following security properties. Firstly, only the signer and the sanitizer should be able to create valid signatures (unforgeability) and the sanitizer should only be able to modify the parts of the message which were initially defined to be modifiable (immutability). Secondly, one should not be able to distinguish between signatures created by the signer and signatures created by the sanitizer (transparency) and it should be infeasible to recover sanitized information (privacy). Finally, neither signers nor sanitizers should be able to repudiate the creation of signatures (accountability). Subsequent to the initial formalization, the additional property (strong) unlinkability was introduced [BFLS10, BPS13] as a stronger privacy notion.

To make sanitizable signature schemes even more expressive, Klonowski and Lauks [KL06] introduced several extensions to plain sanitizable signatures. Most interestingly, they introduced the feature to limit the possible modifications per modifiable message block to signer-defined sets (`LimitSet`). While [KL06] did not provide any formal security definitions, [CJ10] extended the security model for sanitizable signatures to also cover their extensions. Unfortunately, the definition of privacy in [CJ10] contradicts privacy in the original sense of sanitizable signatures. That is, it does not require the admissible changes within the `LimitSet` blocks to remain concealed upon verification of the signature.

We address this issue and review the notion of unlinkability in the context of `LimitSet`. We conclude that—while such a notion would fix the aforementioned issue—it seems to be too strong to obtain practically efficient schemes. Consequently, we introduce a stronger version of privacy (denoted *strong privacy*) that captures privacy in the original sense while still allowing practically efficient instantiations. In addition, we set our newly introduced privacy notion in the context of the existing privacy notions and show that privacy is strictly weaker than strong privacy, that unlinkability is strictly stronger than strong privacy and that strong privacy is independent of transparency. Finally, we provide a

black-box construction of schemes supporting the `LimitSet` extension from plain sanitizable signatures in the models of [BFF+09, GQZ10] and cryptographic accumulators, and show that the so-obtained construction provides strong privacy if the accumulator scheme is indistinguishable.

Homomorphic Proxy-Re Authenticators [c3]. Proxy re-cryptography [BBS98] targets the setting where a semi-trusted proxy can transform cryptographic objects under one key to cryptographic objects under another key using so called re-keys which enable this transformation. While the domain of proxy re-encryption has proven to be useful in a plethora of applications (cf. Section 5.4), proxy re-cryptography in the context of authentication primitives seems to inherently require a combination with homomorphic properties (malleability) on the message space to be really useful. Interestingly, however, this direction has received no attention so far.

We address this issue and propose the notion of homomorphic proxy-re authenticators. Here, data sources authenticate data items under their own secret keys and send them to an aggregator. The aggregator—who is in possession of re-keys from all the single sources to a receiver—can then transform the single signatures to a message authentication code (MAC) under the receiver's key. In addition, the resulting MAC provides homomorphic properties on the message space, so that the aggregator can report authentic evaluations of arithmetic circuits (functions) on the data items provided by the sources to the receiver. Our framework considers two flavors of privacy—input privacy and output privacy. Input privacy requires that a MAC which authenticates the evaluation of a function f on a set of data items does not reveal more information than the description of f and the evaluation result would reveal on their own. Output privacy requires that the aggregator neither learns the data items provided by the sources, nor the result of the evaluation of f on them. We present two modular constructions of homomorphic proxy-re authenticators for linear functions—the first providing input privacy and the second providing input and output privacy. On our way to achieve output privacy, we formalize the notion of homomorphic proxy re-encryption and present an instantiation for linear functions.

1.1.1 Bibliographical Remarks

The technical parts following this introduction are directly taken from the full versions of the author's publications as referenced above (mostly verbatim). We made adaptations to achieve notational consistency within this thesis, rearranged the order of some sections, and made some other minor changes and clarifications. Furthermore, we isolated the required preliminaries within a dedicated chapter to avoid redundancies. Likewise to the technical parts, the preliminaries chapter reuses the preliminaries from the author's publications (also including the ones where the contribution is not included in this thesis) with some notational adaptations for consistency reasons. For all included parts the author contributed as one of the main authors.

1.1.2 Other Contributions

We now briefly discuss additional contributions of the author which are not included in this thesis. For the papers where the author did not contribute as one of the main authors we make the author's contributions explicit. Again, we partly borrow formulations from the abstracts/introductions of the referenced papers.

Further Results on Basic Primitives. We now discuss our additional results on basic primitives.

Post-Quantum Signatures [i1]. Shor's polynomial time algorithm for computing discrete logarithms and factoring [Sho94] made it evident that having a sufficiently powerful quantum computer will allow to break virtually all public key cryptosystems being currently in place. While statements about the feasibility of building such a quantum computer within the next few decades are rather speculative, it is important to be prepared. We introduce signature schemes and zero-knowledge proofs which are solely based on symmetric key primitives. Those primitives are conjectured to resist attacks by quantum computers. Furthermore, we present implementations confirming the practicability of our results. This work is a merge of [i3] and [GCZ16]. The author contributed as one of the main authors of [i3].

Forward-Secure Proxy Re-Encryption [i2]. In this work we turn to another issue in the context of proxy re-cryptography, namely *forward secure* proxy re-encryption. Forward security is a notion which aims to prevent that a key leakage at a certain point in time affects previous usages of the key. For example, in the domain of encryption schemes this would mean that leaking a key at a certain point in time does not help in decrypting ciphertexts which were created before the key leakage. The tricky part in such a setting is that one requires a mechanism which allows to update the secret key while keeping the associated public key constant and compact. We formalize forward-security in the context of proxy re-encryption and provide two modular constructions based on different building blocks related to hierarchical identity based encryption [GS02]. Some techniques we use are also inspired by our work on key-homomorphic signatures. The author mainly contributed in the conception phase and in the final writing phase.

Practical Witness Encryption [i6]. Witness encryption is an alternative encryption paradigm where one encrypts with respect to an instance of some problem (a word x in an $\mathbf{NP}$-language L) so that everyone who knows a solution to this problem instance (a witness w attesting the membership of $x \in L$) can later decrypt. We construct witness encryption for a class of algebraic languages being particularly interesting in the design of cryptographic schemes and protocols. Furthermore, we show how to use our witness encryption scheme to encrypt with respect to a zero-knowledge proof, so that only the prover can decrypt. Finally, we discuss how to apply our techniques in the context of privacy-preserving communication.

Multi-Key Homomorphic Signatures [i5]. In addition to our results on key-homomorphic signatures discussed before, we introduce the notion of multi-key homomorphic signatures in [i5]. Essentially, this notion combines homomorphic properties on the message space with homomorphic properties on the key space. We discuss applications of key homomorphisms in this context and present first results regarding the feasibility of achieving certain flavors of multi-key homomorphic signatures.

Further Results on Signatures with Signer Privacy. Below, we revisit our additional results on variants of signature schemes which provide privacy features for the signer.

Linking of Group Signatures [c4]. We extend group signatures with a feature that allows a dedicated entity, called linking authority, to prove whether two group signatures were created by the same or different group members, while not learning anything about the group member's identity. Our construction complements so-called controllably linkable group signatures [HLhC+11, HLC+13, SSU14, HCCN15]—where the linking authority only outputs a decision bit—with publicly verifiable proofs. We call group signatures providing this feature *group signatures with verifiable controllable linkability* (VCL-GS). We construct a novel proof system to prove statements about the equality and inequality of two encrypted plaintexts. Based on this, we formalize the security requirements for VCL-GS and present a construction using our plaintext (in-)equality proofs. The author contributed to the construction of the proof system and to the security proofs of the VCL-GS extension.

Revocation of Anonymous Credentials [c7]. When using signature variants where the signers remain anonymous, one may observe that it is a non-trivial task to revoke their signing rights. In this work, we focus on revocation in the context of attribute based anonymous credential systems. In particular, we introduce a security model which explicitly considers revocation together with two revocation mechanisms for the credential system in [HS14, FHS15b]. Both revocation mechanisms are based on cryptographic accumulators.

Anonymous Credentials on a Java Card [c13]. We demonstrate the feasibility of using Brands' attribute based anonymous credential system [Bra00]—which is well known due to being used within Microsoft's U-Prove [PZ13]—on a standard Java card. This work subsumes the results of the author's bachelor's thesis.

Further Results on Signatures with Data Privacy. Below, we review further results on signature schemes providing privacy features with respect to the signed data.

Invisible Santizable Signatures [c2]. In this work, we tackle the long standing open question as how to construct sanitizable signature, where no one except the signer and the sanitizer is able to decide whether a block is admissibly

changeable or not. On our way, we introduce a new building block dubbed chameleon hashes with ephemeral trapdoors (CHETs) and present four constructions thereof. The author contributed to the security proofs of the new sanitizable signature scheme as well as the security proofs of the two CHETs in the known-order group setting.

Stronger Security for Invisible Sanitizable Signatures [c1]. It remained open in [c2] whether it is possible to achieve stronger invisibility and signer-accountability definitions. In this work we strengthen the definitions from [c2] and present an extension of the construction in [c2] which achieves these stronger notions as well as an implementation confirming its practicality. The author contributed to fine-tuning some details in the construction and the security proofs.

Authentic Documentation of Outsourced Workflows [c6]. Besides end-to-end authenticity of outsourced modifiable data, we see the authentic documentation of outsourced processes and/or workflows as one central application of malleable signatures. We define a framework which formally captures the requirements for this application. On top of that, we study the suitability of different flavors of malleable signatures to cover (subsets of) the defined requirements.

Black-Box Constructions of Proxy-Type Signatures [c12]. Proxy-type signatures are a variant of malleable signatures with similarities to (extended) sanitizable signatures regarding the expressiveness of the allowed modifications. The main difference to sanitizable signatures is that the delegator does never issue a signature on its own, but only computes a delegation key which allows a designated party (the proxy) to issue signatures on behalf of the delegator. We focus on the two proxy-type signature variants called blank digital signatures [HS13a] and warrant-hiding proxy signatures [HS13b]. In particular, we present black-box constructions of those two signature variants from non-interactive anonymous credentials. In addition, we present instantiations from well known CL [CL04] and Brands credentials [Bra00] as used within IBM's Identity Mixer [CH02] and Microsoft's U-Prove [PZ13], respectively.

Optimization and Implementation of Blank Digital Signatures [c11]. We present optimizations of blank digital signatures [HS13a] and a library integrating our optimized version into the Java Cryptography Architecture [Ora]. In addition, we provide means to integrate the keying material into X.509 certificates [CSF+08]. Our implementation provides support for XML and PDF documents. This work subsumes parts of the author's master's thesis [t1].

1.2 Cryptographic Aspects

To conclude the introduction, we informally discuss two aspects we consider important from a cryptographic point of view.

Modularity vs. Efficiency. When designing complex cryptographic schemes and protocols with a broad range of different security requirements, one will notice that it is hard to reach maximum efficiency and maximum modularity at the same time. To this end, it is required to find a suitable tradeoff between those two goals. The most extreme path with respect to efficiency would be to go for ad hoc constructions. Here, one typically buys efficiency at the cost of reduced modularity, and, thus, reduced conceptual simplicity. In contrast, the most extreme path regarding modularity would be to build upon composability frameworks such as universal composability [Can01] or constructive cryptography [Mau11, MR11], which, roughly speaking, directly consider arbitrary compositions of building blocks proven secure in these frameworks. Here one often needs to trade efficiency for compatibility with the framework itself.

We take a path which is somewhere in between those two extreme directions. That is, we aim to modularly construct our schemes from other well-defined smaller building blocks. This allows us to achieve flexibility with respect to possible instantiations and underlying cryptographic hardness assumptions. Yet, we do not target arbitrary compositions of our building blocks. This, in turn, gives us more flexibility regarding efficiency. Note that our design goal is also reflected in the structure of this thesis, starting with basic primitives and then moving on to more sophisticated schemes, (partly) building upon them.

Provable Security. To argue about the security of our constructions, we use reductionist security proofs. To this end, we informally discuss the idea of reductionist security proofs and some concepts which are of particular interest for our work. Henceforth, we use the term *efficient* to refer to algorithms that run in polynomial time in the length of their input. Furthermore, we use the term *negligible* to refer to functions that vanish faster than every inverse polynomial in their input. The goal of provable security is to relate the security of cryptographic schemes to the hardness of solving well studied and presumably hard problems.[7] In doing so, one carefully defines a so-called security model, which consists of an interface definition of the cryptographic scheme itself and formal definitions of the required security properties. To prove the security of a scheme, one usually uses a reductionist argument to prove security by contradiction. That is, one shows that an efficient adversary against a certain security property can efficiently be turned into an efficient adversary against some presumably hard problem, contradicting the existence of an efficient adversary against this security property. With increasing complexity of the cryptographic schemes and their corresponding security models, also reductionist security proofs become more complex and are often hard to follow. In this context, sequences of games (cf. [Sho04] for an excellent overview) turn out to be a very useful tool to obtain clear and easy to comprehend security proofs. The basic idea is to change the behavior of the challenger that interacts with the adversary in a sequence of game transitions, where each transition changes the winning probability to a certain

[7] That is, problems where it is assumed that—for every efficient adversary that can also make random steps during execution—the probability of solving them is negligible in the security parameter.

extent. Ultimately, these transitions yield to a game where one can easily bound the winning probability of the adversary (e.g., because it is equal to 0 or it is straightforward to come up with a reduction). Using the changes of the winning probability in each transition and the winning probability in the final game, one can then obtain a bound for the winning probability in the original game. In the simplest case the probabilities to detect the game transitions are negligible in the security parameter. Then, the winning probability in the original game is negligible in the security parameter if the winning probability in the final game is negligible in the security parameter (as long as the number of game changes is polynomially bounded). For a more formal treatment of reductionist security and sequences of games, we refer the reader to [Gol08] and [Sho04], respectively.

2 Preliminaries

In this chapter, we introduce our notation and establish the required preliminaries. We, thereby, assume familiarity with computational complexity theory and basic cryptographic knowledge, and refer the reader to [KL07, Gol08] for an excellent treatment of these topics. Most of the notions and definitions in this chapter are rather standard and can, e.g., be found in [Gol01, KL07, Gol08, Kat10]. For less standard notions and definitions we explicitly include references.

2.1 Notation

We use κ to denote the security parameter and we use sans-serif letters, e.g., A, B, to denote algorithms. If not stated otherwise, all algorithms are required to be efficient, i.e., their running time can be bounded by a polynomial in their input length. Furthermore, all algorithms return a special symbol $\perp$ on error. By $y \leftarrow \mathsf{A}(1^\kappa, x)$, we denote that y is assigned the output of the potentially probabilistic algorithm A on input x and fresh random coins. If A is a probabilistic algorithm, we use $\mathsf{A}(1^\kappa, x;\ r)$ to make the random coins r explicit. For the algorithms representing the adversaries in the security games we will use calligraphic letters, e.g., $\mathcal{A}$. We assume 1^κ to be an implicit input to all algorithms, and, thus, may omit 1^κ as an explicit input parameter. Similar to our notation in the context of algorithms, we use $y \xleftarrow{R} S$ to denote that an element is sampled uniformly at random from a finite set S and assigned to y. We let $[n] := \{1, \ldots, n\}$ and may use the concatenation operator, e.g., $(a_i)_{i=1}^n || (b_i)_{i=1}^m := (a_1, \ldots, a_n, b_1, \ldots, b_m)$. Thereby, we assume that concatenated sequences can later be uniquely decomposed (even when concatenating elements of different types and lengths). We write $\Pr[\Omega : \mathcal{E}]$ to denote the prob-

ability of an event $\mathcal{E}$ over the probability space Ω. We use $\mathcal{C}$ to denote challengers of security experiments, and $\mathcal{C}_\kappa$ to make the security parameter explicit. A function $\varepsilon(\cdot) : \mathbb{N} \to \mathbb{R}_{\geq 0}$ is called negligible, iff it vanishes faster than every inverse polynomial, i.e., $\forall\, k : \exists\, n_k : \forall\, n > n_k : \varepsilon(n) < n^{-k}$. Furthermore, we may also use the term "practically efficient". Unlike the previous definitions, this term is more informal and means that an algorithm can be executed on any state of the art machine within reasonable time (we deem computation times of $\approx 1s$ to be reasonable).

2.2 Cryptographic Hardness Assumptions

Our results include black-box constructions which are independent of the concrete assumptions, as well as constructions in the known-order group setting (with bilinear pairings). As we also require the strong RSA assumption in hidden-order groups for our overview of existing accumulator constructions in Section 3.1 we include it here for completeness.

2.2.1 Known-Order Groups

In this section we present the required definitions and assumptions in the known-order group setting.

Definition 2.1 (Group Generator)**.** *Let* GGen *be an algorithm which takes a security parameter* κ *as input, and generates a group description*

$$\mathsf{G} \coloneqq (p, \mathbb{G}, g),$$

where the group order of the group $\mathbb{G}$ *is a prime* p *of bitlength* κ*, and* g *is a generator of* $\mathbb{G}$*.*

Definition 2.2 (Discrete Logarithm Assumption (DL))**.** *The DL assumption holds relative to* GGen*, if for all PPT adversaries* $\mathcal{A}$ *there exists a negligible function* $\varepsilon(\cdot)$ *such that:*

$$\Pr\left[\mathsf{G} \leftarrow \mathsf{GGen}(1^\kappa),\ r \xleftarrow{R} \mathbb{Z}_p,\ r^\star \leftarrow \mathcal{A}(\mathsf{G}, g^r)\ :\ r = r^\star\right] \leq \varepsilon(\kappa).$$

Definition 2.3 (Computational Diffie-Hellman Assumption (CDH))**.** *The CDH assumption holds relative to* GGen*, if for all PPT adversaries* $\mathcal{A}$ *there exists a negligible function* $\varepsilon(\cdot)$ *such that:*

$$\Pr\left[\mathsf{G} \leftarrow \mathsf{GGen}(1^\kappa),\ r, s \xleftarrow{R} \mathbb{Z}_p,\ h^\star \leftarrow \mathcal{A}(\mathsf{G}, g^r, g^s)\ :\ h^\star = g^{rs}\right] \leq \varepsilon(\kappa).$$

Definition 2.4 (Decisional Diffie-Hellman Assumption (DDH))**.** *The DDH assumption holds relative to* GGen*, if for all PPT adversaries* $\mathcal{A}$ *there exists a negligible function* $\varepsilon(\cdot)$ *such that:*

$$\Pr\left[\begin{array}{l}\mathsf{G} \leftarrow \mathsf{GGen}(1^\kappa),\ b \xleftarrow{R} \{0,1\},\ r, s, t \xleftarrow{R} \mathbb{Z}_p, \\ b^\star \leftarrow \mathcal{A}(\mathsf{G}, g^r, g^s, g^{b\cdot(rs)+(1-b)\cdot t})\end{array}\ :\ b = b^\star\right] \leq 1/2 + \varepsilon(\kappa).$$

Definition 2.5 (Decision Linear Assumption [BBS04] (DLIN))**.** *The DLIN assumption holds relative to* GGen*, if for all PPT adversaries* $\mathcal{A}$ *there exists a negligible function* $\varepsilon(\cdot)$ *such that:*

$$\Pr\left[\begin{array}{l}\mathsf{G} \leftarrow \mathsf{GGen}(1^\kappa),\ u, v \xleftarrow{R} \mathbb{G},\ b \xleftarrow{R} \{0,1\},\ r, s, t \xleftarrow{R} \mathbb{Z}_p, \\ b^\star \leftarrow \mathcal{A}(\mathsf{G}, u, v, u^r, v^s, g^{b\cdot(r+s)+(1-b)\cdot t})\end{array} \ :\ b = b^\star\right] \leq 1/2 + \varepsilon(\kappa).$$

Definition 2.6 (t-strong Diffie-Hellman Assumption [BB04b] (t-SDH))**.** *The t-SDH assumption holds relative to* GGen*, if for all PPT adversaries* $\mathcal{A}$ *there exists a negligible function* $\varepsilon(\cdot)$ *such that:*

$$\Pr\left[\begin{array}{l}\mathsf{G} \leftarrow \mathsf{GGen}(1^\kappa),\ \alpha \xleftarrow{R} \mathbb{Z}_p, \\ (c^\star, h^\star) \leftarrow \mathcal{A}(\mathsf{G}, g^\alpha, \ldots, g^{\alpha^t})\end{array} \ :\ \begin{array}{r} h^\star = g^{\frac{1}{c^\star+\alpha}} \ \wedge \\ c^\star \in \mathbb{Z}_p \setminus \{-\alpha\} \ \wedge \\ t \leq \mathsf{poly}(\kappa)\end{array}\right] \leq \varepsilon(\kappa).$$

Definition 2.7 (t-Diffie-Hellman Exponent Assumption [GJM02, CKS09] (t-DHE))**.** *The t-DHE assumption holds relative to* GGen*, if for all PPT adversaries* $\mathcal{A}$ *there exists a negligible function* $\varepsilon(\cdot)$ *such that:*

$$\Pr\left[\begin{array}{l}\mathsf{G} \leftarrow \mathsf{GGen}(1^\kappa),\ \gamma \xleftarrow{R} \mathbb{Z}_p, \\ h^\star \leftarrow \mathcal{A}(\mathsf{G}, g^\gamma, \ldots, g^{\gamma^t}, g^{\gamma^{t+2}}, \ldots, g^{\gamma^{2t}})\end{array} \ :\ \begin{array}{r} h^\star = g^{\gamma^{t+1}} \ \wedge \\ t \leq \mathsf{poly}(\kappa)\end{array}\right] \leq \varepsilon(\kappa).$$

Bilinear Maps. Let $\mathbb{G}_1 = \langle g \rangle$, $\mathbb{G}_2 = \langle \hat{g} \rangle$, and $\mathbb{G}_T$ be groups of prime order p. A bilinear map (paring) is an efficiently computable map $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ with the following two properties:

Bilinearity: $e(g^a, \hat{g}^b) = e(g, \hat{g})^{ab} \quad \forall\ (a, b) \in \mathbb{Z}_p^2$.

Non-degeneracy: $e(g, \hat{g}) \neq 1_{\mathbb{G}_T}$, i.e., $e(g, \hat{g})$ generates $\mathbb{G}_T$.

We distinguish three different settings. In the Type-1 setting, we have that $\mathbb{G}_1 = \mathbb{G}_2$. In the Type-2 setting, we have that $\mathbb{G}_1 \neq \mathbb{G}_2$, yet, there exists an efficiently computable isomorphism $\psi : \mathbb{G}_2 \to \mathbb{G}_1$. Finally, in the Type-3 setting, we also have that $\mathbb{G}_1 \neq \mathbb{G}_2$ but no efficiently computable isomorphism is known. Throughout this thesis, we will denote elements in $\mathbb{G}_T$ by boldface letters, e.g., $\mathbf{g} = e(g, \hat{g})$.

Definition 2.8 (Bilinear Group Generator)**.** *Let* BGGen *be an algorithm which takes a security parameter* κ *and a type* $\mathsf{T} \in \{1, 2, 3\}$*, and generates a bilinear group description* BG *in the Type-*T *setting, where* BG *is defined as*

$$\mathsf{BG} \coloneqq \begin{cases} (p, \mathbb{G}, \mathbb{G}_T, e, g) & \text{if } \mathsf{T} = 1, \\ (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g, \hat{g}, \psi) & \text{if } \mathsf{T} = 2, \\ (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g, \hat{g}) & \text{if } \mathsf{T} = 3. \end{cases}$$

Thereby, the common group order of the groups $\mathbb{G}$ *and* $\mathbb{G}_T$ *(resp.* $\mathbb{G}_1$*,* $\mathbb{G}_2$*, and* $\mathbb{G}_T$*) is a prime* p *of bitlength* κ*,* e *is a pairing,* g *and* $\hat{g}$ *are generators of* $\mathbb{G}_1$ *and* $\mathbb{G}_2$*, respectively, and* ψ *is an isomorphism* $\mathbb{G}_2 \to \mathbb{G}_1$*.*

Note that a bilinear paring serves as a DDH oracle for the group $\mathbb{G}$ in the Type-1 setting and for the group $\mathbb{G}_2$ in the Type-2 setting. Thus, if the CDH assumption holds in the respective groups, they represent Gap Diffie-Hellman groups [OP01, BLS04].

While Gap Diffie-Hellman groups are useful in protocol design, also the absence of a DDH oracle may be useful. This absence is captured by the following assumptions. In the Type-2 setting, the external Diffie-Hellman (XDH) problem is assumed to be hard.

Definition 2.9 (XDH)**.** *The XDH assumption holds relative to* BGGen *if the DDH assumption holds in* $\mathbb{G}_1$*.*

Due to the absence of the isomorphism in the Type-3 setting, the symmetric external Diffie-Hellman (SXDH) problem is assumed to be hard.

Definition 2.10 (SXDH)**.** *The SXDH assumption holds relative to* BGGen *if the DDH assumption holds in* $\mathbb{G}_1$ *and* $\mathbb{G}_2$*.*

We henceforth may also use *symmetric setting* to refer to the Type-1 setting. Likewise, when it is clear from the context, we may use *asymmetric setting* to refer to the Type-2 or the Type-3 setting.

Finally, we require the bilinear decisional Diffie-Hellman assumption in Type-1 bilinear groups.

Definition 2.11 (Bilinear Decisional Diffie-Hellman Assumption [BB04a] (BDDH))**.** *The BDDH assumption holds relative to* BGGen*, if for all PPT adversaries* $\mathcal{A}$ *there is a negligible function* $\varepsilon(\cdot)$ *such that*

$$\Pr\left[\begin{array}{l}\mathsf{BG} \leftarrow \mathsf{BGGen}(1^\kappa, 1),\ r, s, t, u \xleftarrow{R} \mathbb{Z}_q,\ b \xleftarrow{R} \{0,1\},\\ b^\star \leftarrow \mathcal{A}\left(\mathsf{BG}, g^r, g^s, g^t, \mathbf{g}^{b \cdot rst + (1-b)u}\right)\end{array} \ :\ b = b^\star\right] \leq 1/2 + \varepsilon(\kappa).$$

Instantiation of Bilinear Pairings. The currently most prominent choice to instantiate bilinear pairings is to use a Type-3 pairing defined over Barreto-Naehrig (BN) curves [BN05]. We, however, note that it is important to consider the recent progress [KB16, SS16] in improving the asymptotic bounds for solving discrete logarithms in extension fields of prime characteristic ($\mathbb{G}_T$ is $\mathbb{F}_p^{12}$ for BN-curves) when choosing the key sizes. A recent work by Menezes et al. [MSS16] assesses (among others) the impact of these advances and suggests to increase the bitlength of p from 256 to 383 as a conservative choice for the 128-bit security level using BN curves. Another recent and even more conservative estimate [BD17] suggests to use a bitlength of 462 for p. These results also suggest that in light of [KB16, SS16] other curve types such as BLS12 [BLS02] or KSS16 [KSS08] may be the better choice to instantiate parings.

2.2.2 Hidden-Order Groups

Although hidden-order groups are not in the focus of this thesis, we include the strong RSA assumption [BP97] for completeness.

Definition 2.12 (RSA Instance Generator)**.** *Let* RSAGen *be an algorithm which on input of a security parameter* κ *outputs* (N, p, q)*, where* $N = pq$ *is an RSA modulus, and* p *and* q *are two random primes of bitlength* κ*.*

Definition 2.13 (Strong RSA assumption (s-RSA))**.** *The s-RSA assumption holds relative to* RSAGen*, if for all PPT adversaries* $\mathcal{A}$ *there is a negligible function* $\varepsilon(\cdot)$ *such that*

$$\Pr\left[\begin{array}{l}(N,p,q) \leftarrow \mathsf{RSAGen}(1^\kappa),\\ u \xleftarrow{R} \mathbb{Z}_N^*,\ (v,w) \leftarrow \mathcal{A}(u,N)\end{array} \ : \ v^w \equiv u \pmod{N} \ \wedge \ w > 1\right] \leq \varepsilon(\kappa).$$

2.3 Computational Idealizations

In a reductionist security proof one ideally directly relates the hardness of breaking a cryptographic scheme or protocol to the hardness of breaking a particular hardness assumption. If this is possible we talk about a security reduction in the *standard model*. While finding a security reduction in the standard model is most desirable, it is often required to make further idealizing assumptions when proving the security of a cryptographic scheme or protocol. We will briefly discuss potential idealizations below.

Common Reference String Model. In the common reference string (CRS) model, all parties rely on a CRS set up by some trusted third party. In security reductions, one can then indistinguishably set up the CRS in a way that it embeds problem instances, or so that certain secrets (trapdoors) are known. When constructions proven secure using this idealization are used in practice, one carefully needs to choose the party who sets up the CRS. Sometimes it can also be possible to use multi-party computation protocols which inherently ensure that the CRS is jointly set up by multiple parties so that the security expectations are met.

Generic Group Model. In the generic group model (GGM) [Sho97], one idealizes groups by assuming that an adversary only has access to group operation oracles but can not exploit any structural properties of the representation of the group elements. A proof in the GGM should be seen as a means to gain confidence in the plausibility of an assumption rather than a proof that the assumption is universally true. We refer the reader to [KM07] for an excellent overview of different opinions on the GGM within the community.

Random Oracle Model. In the random oracle model (ROM) [BR93], one idealizes hash functions by modeling them as oracles which return uniformly random values upon each new query, while consistently repeating the responses for queries which have already been answered previously. If it does not suffice that a RO returns uniformly random values to argue about the security of a scheme, one can additionally program the random oracle so that it returns a particular value upon a particular query. We refer the reader to [KM15] for an excellent discussion of the advantages and disadvantages of using the random oracle model. Their conclusion is that there is currently no evidence that a proof in the ROM points to a real-world security weakness (even though there are some artificial counterexamples for the existence of ROs, e.g., [CGH04]).

2.4 Cryptographic Building Blocks

In this section we formally recall the required cryptographic building blocks.

2.4.1 One-Way Functions

Below, we recall the notion of one-way functions.

Definition 2.14. *A function $f : \{0,1\}^* \to \{0,1\}^*$ is called a one-way function, if (1) there exists a PPT algorithm $\mathcal{A}_1$ so that $\forall\, x \in \{0,1\}^* : \mathcal{A}_1(x) = f(x)$, and if (2) for every PPT algorithm $\mathcal{A}_2$ there is a negligible function $\varepsilon(\cdot)$ such that it holds that*

$$\Pr\left[x \xleftarrow{R} \{0,1\}^\kappa,\ x^\star \leftarrow \mathcal{A}_2(1^\kappa, f(x)) \ :\ f(x) = f(x^\star)\right] \leq \varepsilon(\kappa).$$

2.4.2 Signature Schemes

We now recall the definition of signature schemes.

Definition 2.15. *A signature scheme Σ is a tuple* $(\mathsf{PGen}, \mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ *of PPT algorithms, which are defined as follows:*

$\mathsf{PGen}(1^\kappa)$: *This algorithm takes a security parameter κ as input and outputs public parameters* pp*. We assume that* pp *are implicitly included in all public keys.*

$\mathsf{KeyGen}(\mathsf{pp})$: *This algorithm takes public parameters* pp *as input and outputs a secret (signing) key* sk *and a public (verification) key* pk *with associated message space $\mathcal{M}$ (we may omit to make the message space $\mathcal{M}$ explicit).*

$\mathsf{Sign}(\mathsf{sk}, m)$: *This algorithm takes a secret key* sk *and a message $m \in \mathcal{M}$ as input and outputs a signature σ.*

$\mathsf{Verify}(\mathsf{pk}, m, \sigma)$: *This algorithm takes a public key* pk*, a message $m \in \mathcal{M}$ and a signature σ as input and outputs a bit $b \in \{0,1\}$.*

We explicitly include the parameter generation algorithm PGen in our definition as we also want to cover schemes where many independently generated public keys are with respect to the same parameters pp, e.g., some elliptic curve group parameters. The usual definition without PGen is a special case of our definition. That is for schemes without an explicit parameter generation the PGen algorithm simply returns $\mathsf{pp} \leftarrow 1^\kappa$, i.e., one can directly run KeyGen on the security parameter.

Besides the usual correctness property, Σ needs to provide some unforgeability notion. Below, we present two standard notions required in our context (ordered from weak to strong). We start with universal unforgeability under no message attacks ($\mathsf{UUF\text{-}NMA}$ security).

Definition 2.16 ($\mathsf{UUF\text{-}NMA}$)**.** *A signature scheme Σ is $\mathsf{UUF\text{-}NMA}$ secure, if for all PPT adversaries $\mathcal{A}$ there is a negligible function $\varepsilon(\cdot)$ such that*

$$\Pr\left[\begin{array}{l|r} \mathsf{pp} \leftarrow \mathsf{PGen}(1^\kappa),\ (\mathsf{sk},\mathsf{pk}) \leftarrow \mathsf{KeyGen}(\mathsf{pp}), & \\ m^\star \xleftarrow{R} \mathcal{M},\ \sigma^\star \leftarrow \mathcal{A}(\mathsf{pk}, m^\star) & \mathsf{Verify}(\mathsf{pk}, m^\star, \sigma^\star) = 1 \end{array}\right] \leq \varepsilon(\kappa).$$

The most common notion is existential unforgeability under adaptively chosen message attacks ($\mathsf{EUF\text{-}CMA}$ security).

Definition 2.17 ($\mathsf{EUF\text{-}CMA}$)**.** *A signature scheme Σ is $\mathsf{EUF\text{-}CMA}$ secure, if for all PPT adversaries $\mathcal{A}$ there is a negligible function $\varepsilon(\cdot)$ such that*

$$\Pr\left[\begin{array}{l|r} \mathsf{pp} \leftarrow \mathsf{PGen}(1^\kappa),\ (\mathsf{sk},\mathsf{pk}) \leftarrow \mathsf{KeyGen}(\mathsf{pp}), & \mathsf{Verify}(\mathsf{pk}, m^\star, \sigma^\star) = 1 \\ (m^\star, \sigma^\star) \leftarrow \mathcal{A}^{\mathsf{Sign}(\mathsf{sk},\cdot)}(\mathsf{pk}) & \wedge\ m^\star \notin \mathcal{Q}^{\mathsf{Sign}} \end{array}\right] \leq \varepsilon(\kappa),$$

where the environment keeps track of the queries to the signing oracle via $\mathcal{Q}^{\mathsf{Sign}}$.

2.4.3 Homomorphic MACs

Our subsequent definition of homomorphic message authentication codes (MACs) is inspired by [AB09] but tailored to our concrete requirements.

Definition 2.18. *A homomorphic MAC for a family of function classes $\{\mathcal{F}_{\mathsf{pp}}\}$ is a tuple $(\mathsf{Gen}, \mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Comb}, \mathsf{Verify})$ of algorithms defined as:*

$\mathsf{Gen}(1^\kappa, \ell)$: *Takes a security parameter κ and an upper bound ℓ on the vector length as input and outputs public parameters pp, determining the message space $\mathcal{M}^\ell$, the function class $\mathcal{F}_{\mathsf{pp}}$ containing functions $f : (\mathcal{M}^\ell)^n \to \mathcal{M}^\ell$ with $\ell, n \leq \mathsf{poly}(\kappa)$, as well as a tag space being exponentially large in κ.*

$\mathsf{KeyGen}(\mathsf{pp})$: *Takes public parameters pp as input and outputs a secret key sk.*

$\mathsf{Sign}(\mathsf{sk}, \vec{v}, \mathsf{id}, \tau)$: *Takes a MAC key sk, a vector $\vec{v}$, an identifier id, and a tag τ as input, and outputs a MAC μ.*

$\mathsf{Comb}(\mathsf{pp}, f, (\mu_i)_{i\in[n]})$: *Takes a function $f \in \mathcal{F}_{\mathsf{pp}}$ and a sequence of valid MACs $(\mu_i)_{i\in[n]}$ on vectors $(\vec{v}_i)_{i\in[n]}$ as input, and outputs a MAC μ on $\vec{v} = f((\vec{v}_i)_{i\in[n]})$.*

$\mathsf{Verify}(\mathsf{sk}, \vec{v}, \mu, \tau, (\mathsf{id}_i)_{i\in[n]}, f)$: *Takes a MAC key* sk*, a vector* $\vec{v}$*, a MAC* μ*, a tag* τ*, a sequence of identifiers* $(\mathsf{id}_i)_{i\in[n]}$*, and a function* $f \in \mathcal{F}_{\mathsf{PP}}$ *as input, and outputs a bit.*

A homomorphic MAC is required to be correct and unforgeable. While we omit the obvious correctness definition, we recall the unforgeability definition. The environment maintains a list `SIG` which is initially empty.

Definition 2.19. *A homomorphic MAC for a family of function classes* $\{\mathcal{F}_{\mathsf{PP}}\}$ *is unforgeable if for all PPT adversaries* $\mathcal{A}$ *there exists a negligible function* $\varepsilon(\cdot)$ *such that:*

$$\Pr\left[\begin{array}{l} \mathsf{PP} \leftarrow \mathsf{Gen}(1^\kappa, \ell),\ \mathsf{sk} \leftarrow \mathsf{KeyGen}(\mathsf{PP}), \\ (\vec{y}^\star, \mu^\star, \tau^\star, (\mathsf{id}_i^\star)_{i\in[n]}, f^\star) \leftarrow \mathcal{A}^{\mathsf{Sig}(\cdot,\cdot,\cdot)}(\mathsf{PP}) \end{array} : \begin{array}{r} \mathsf{Verify}(\mathsf{sk}, \vec{y}^\star, \mu^\star, \tau^\star, (\mathsf{id}_i^\star)_{i\in[n]}, f^\star) = 1 \ \wedge \ f^\star \in \mathcal{F}_{\mathsf{PP}} \ \wedge \\ (\nexists (\vec{v}_j)_{j\in[n]} : (\forall j \in [n] : (\vec{v}_j, \mathsf{id}_j^\star) \in \mathcal{S}[\tau^\star]) \ \wedge \ f^*((\vec{v}_j)_{j\in[n]}) = \vec{y}^\star) \end{array}\right] \leq \varepsilon(\kappa),$$

where the sign oracle Sig *is defined as:*

$\mathsf{Sig}((\vec{v}_i)_{i\in[n]}, (\mathsf{id}_i)_{i\in[n]}, \tau)$: *If* $\mathtt{SIG}[\tau] \neq \bot$ *or there exists* $u, v \in [n], u \neq v$ *so that* $\mathsf{id}_u = \mathsf{id}_v$ *return* $\bot$*. Otherwise, compute* $\mu_i \leftarrow \mathsf{Sign}(\mathsf{sk}, \vec{v}_i, \mathsf{id}_i, \tau)$ *for* $i \in [n]$*, set* $\mathtt{SIG}[\tau] \leftarrow \{(\vec{v}_i, \mathsf{id}_i)\}_{i\in[n]}$ *and return* $(\mu_i)_{i\in[n]}$*.*

2.4.4 Signatures on Equivalence Classes

We briefly recall structure-preserving signatures on equivalence classes (SPS-EQ) as presented in [HS14, FHS15b]. Therefore, let p be a prime and $\ell > 1$; then $\mathbb{Z}_p^\ell$ is a vector space and one can define a projective equivalence relation on it, which propagates to $\mathbb{G}_i^\ell$ and partitions $\mathbb{G}_i^\ell$ into equivalence classes. Let $\sim_{\mathcal{R}}$ be this relation, i.e., for $m, n \in \mathbb{G}_i^\ell : m \sim_{\mathcal{R}} n \Leftrightarrow \exists\, s \in \mathbb{Z}_p^* : m = n^s$, where n^s denotes component wise exponentiation of each vector component of n with s. An SPS-EQ scheme now signs an equivalence class $[m]_{\mathcal{R}}$ for $m \in (\mathbb{G}_i^*)^\ell$ by signing a representative m of $[m]_{\mathcal{R}}$. One of the design goals of SPS-EQ is to guarantee that two message-signature pairs from the same equivalence class cannot be linked. Let us recall the formal definition of an SPS-EQ scheme below.

Definition 2.20. *An* SPS-EQ *on* $\mathbb{G}_i^*$ *(for* $i \in \{1, 2\}$*) consists of the following PPT algorithms:*

$\mathsf{BGGen}_{\mathcal{R}}(1^\kappa)$*: This algorithm on input of a security parameter* κ *outputs a bilinear group* BG*.*

$\mathsf{KeyGen}_{\mathcal{R}}(\mathsf{BG}, \ell)$*: This algorithm on input of a bilinear group* BG *and a vector length* $\ell > 1$ *outputs a key pair* $(\mathsf{sk}, \mathsf{pk})$*.*

$\mathsf{Sign}_{\mathcal{R}}(m, \mathsf{sk})$*: This algorithm on input a representative* $m \in (\mathbb{G}_i^*)^\ell$ *and a secret key* sk *outputs a signature* σ *for the equivalence class* $[m]_{\mathcal{R}}$*.*

$\mathsf{ChgRep}_{\mathcal{R}}(m, \sigma, \rho, \mathsf{pk})$*: This algorithm on input of a representative* $m \in (\mathbb{G}_i^*)^\ell$ *of class* $[m]_{\mathcal{R}}$*, a signature* σ *for* m*, a scalar* ρ *and a public key* pk *returns an updated message-signature pair* (m', σ')*, where* $m' = m^\rho$ *is the new representative and* σ' *its updated signature.*

$\mathsf{Verify}_{\mathcal{R}}(m, \sigma, \mathsf{pk})$*: This algorithm on input of a representative* $m \in (\mathbb{G}_i^*)^\ell$*, a signature* σ *and a public key* pk *outputs a bit* $b \in \{0, 1\}$*.*

$\mathsf{VKey}_{\mathcal{R}}(\mathsf{sk}, \mathsf{pk})$ *This algorithm on input a secret key* sk *and a public key* pk *outputs a bit* $b \in \{0, 1\}$*.*

For security, one requires the following properties.

Definition 2.21 (Correctness)**.** *An* SPS-EQ *scheme on* $(\mathbb{G}_i^*)^\ell$ *is called* correct *if for all security parameters* $\kappa \in \mathbb{N}$*,* $\ell > 1$*,* $\mathsf{BG} \leftarrow \mathsf{BGGen}_{\mathcal{R}}(1^\kappa)$*,* $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KeyGen}_{\mathcal{R}}(\mathsf{BG}, \ell)$*,* $m \in (\mathbb{G}_i^*)^\ell$ *and* $\rho \in \mathbb{Z}_p^*$*:*

$$\mathsf{VKey}_{\mathcal{R}}(\mathsf{sk}, \mathsf{pk}) = 1 \quad \wedge \quad \Pr\left[\mathsf{Verify}_{\mathcal{R}}(m, \mathsf{Sign}_{\mathcal{R}}(m, \mathsf{sk}), \mathsf{pk}) = 1\right] = 1 \quad \wedge$$
$$\Pr\left[\mathsf{Verify}_{\mathcal{R}}(\mathsf{ChgRep}_{\mathcal{R}}(m, \mathsf{Sign}_{\mathcal{R}}(m, \mathsf{sk}), \rho, \mathsf{pk}), \mathsf{pk}) = 1\right] = 1.$$

For EUF-CMA security, outputting a valid message-signature pair, corresponding to an unqueried equivalence class, is considered to be a forgery:

Definition 2.22 (EUF-CMA)**.** *An* SPS-EQ *over* $(\mathbb{G}_i^*)^\ell$ *is* existentially unforgeable under adaptively chosen-message attacks*, if for all PPT adversaries* $\mathcal{A}$ *with access to a signing oracle* $\mathcal{O}^{\mathsf{Sign}_{\mathcal{R}}}$*, there is a negligible function* $\varepsilon(\cdot)$ *such that:*

$$\Pr\left[\begin{array}{l} \mathsf{BG} \leftarrow \mathsf{BGGen}_{\mathcal{R}}(1^\kappa), \\ (\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KeyGen}_{\mathcal{R}}(\mathsf{BG}, \ell), \\ (m^\star, \sigma^\star) \leftarrow \mathcal{A}^{\mathcal{O}^{\mathsf{Sign}_{\mathcal{R}}(\mathsf{sk}, \cdot)}}(\mathsf{pk}) \end{array} : \begin{array}{r} [m^\star]_{\mathcal{R}} \neq [m]_{\mathcal{R}} \quad \forall\, m \in \mathcal{Q}^{\mathsf{Sign}_{\mathcal{R}}} \quad \wedge \\ \mathsf{Verify}_{\mathcal{R}}(m^\star, \sigma^\star, \mathsf{pk}) = 1 \end{array}\right] \leq \varepsilon(\kappa),$$

where $\mathcal{Q}^{\mathsf{Sign}_{\mathcal{R}}}$ *is the set of queries that* $\mathcal{A}$ *has issued to the signing oracle* $\mathcal{O}^{\mathsf{Sign}_{\mathcal{R}}}$*.*

Besides EUF-CMA security, an additional security property for SPS-EQ was introduced in [FHS15a].

Definition 2.23 (Perfect Adaption of Signatures)**.** *An* SPS-EQ *scheme on* $(\mathbb{G}_i^*)^\ell$ perfectly adapts signatures *if for all tuples* $(\mathsf{sk}, \mathsf{pk}, M, \sigma, \rho)$ *where it holds that* $\mathsf{VKey}_{\mathcal{R}}(\mathsf{sk}, \mathsf{pk}) = 1$*,* $\mathsf{Verify}_{\mathcal{R}}(m, \sigma, \mathsf{pk}) = 1$*,* $m \in (\mathbb{G}_i^*)^\ell$*, and* $\rho \in \mathbb{Z}_p^*$*, the distributions* $(m^\rho, \mathsf{Sign}_{\mathcal{R}}(m^\rho, \mathsf{sk}))$ *and* $\mathsf{ChgRep}_{\mathcal{R}}(m, \sigma, \rho, \mathsf{pk})$ *are identical.*

An instantiation providing all above security properties is provided in [FHS15a, FHS15b]. Here, assuming the DDH assumption to hold on the message space yields that different message-signature pairs from the same equivalence class cannot be linked.

2.4.5 Static Group Signatures

In the following, we recall the established model for static group signatures from [BMW03].

Definition 2.24. *A group signature scheme* GS *is a tuple* $(\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify}, \mathsf{Open})$ *of PPT algorithms which are defined as follows:*

$\mathsf{KeyGen}(1^\kappa, n)$: *Takes a security parameter* κ *and the group size* n *as input. It generates and outputs a group verification key* gpk*, a group opening key* gok*, as well as a list of group signing keys* $\mathsf{gsk} = \{\mathsf{gsk}_i\}_{i\in[n]}$.

$\mathsf{Sign}(\mathsf{gsk}_i, m)$: *Takes a group signing key* gsk_i *and a message* m *as input and outputs a signature* σ.

$\mathsf{Verify}(\mathsf{gpk}, m, \sigma)$: *Takes a group verification key* gpk*, a message* m *and a signature* σ *as input, and outputs a bit* b.

$\mathsf{Open}(\mathsf{gok}, m, \sigma)$: *Takes a group opening key* gok*, a message* m *and a signature* σ *as input, and outputs an identity* i.

The GS security properties are formally defined as follows (we omit correctness).

Definition 2.25 (Anonymity)**.** *A* GS *is anonymous, if for all PPT adversaries* $\mathcal{A}$ *there exists a negligible function* $\varepsilon(\cdot)$ *such that*

$$\Pr\left[\begin{array}{l|rl} & (\mathsf{gpk},\mathsf{gok},\mathsf{gsk}) \leftarrow \mathsf{KeyGen}(1^\kappa, n), & \\ & b \xleftarrow{R} \{0,1\},\ \mathcal{O} \leftarrow \{\mathsf{Open}(\mathsf{gok},\cdot,\cdot)\}, & \\ & (i_0^\star, i_1^\star, m^\star, \mathsf{ST}) \leftarrow \mathcal{A}^{\mathcal{O}}(\mathsf{gpk},\mathsf{gsk}), & : \quad b = b^\star \ \wedge \\ & \sigma \leftarrow \mathsf{Sign}(\mathsf{gsk}_{i_b^\star}, m^\star),\ b^\star \leftarrow \mathcal{A}^{\mathcal{O}}(\sigma, \mathsf{ST}) & (m^\star,\sigma) \notin \mathcal{Q}_2^{\mathsf{Open}} \end{array}\right] \leq \varepsilon(\kappa),$$

where $\mathcal{A}$ *runs in two stages and* $\mathcal{Q}_2^{\mathsf{Open}}$ *records the* Open *queries in stage two.*

Definition 2.26 (Traceability)**.** *A* GS *is traceable, if for all PPT adversaries* $\mathcal{A}$ *there exists a negligible function* $\varepsilon(\cdot)$ *such that*

$$\Pr\left[\begin{array}{l|rl} & (\mathsf{gpk},\mathsf{gok},\mathsf{gsk}) \leftarrow \mathsf{KeyGen}(1^\kappa, n), & \\ & \mathcal{O} \leftarrow \{\mathsf{Sig}(\cdot,\cdot), \mathsf{Key}(\cdot)\}, & \\ & (m^\star, \sigma^\star) \leftarrow \mathcal{A}^{\mathcal{O}}(\mathsf{gpk},\mathsf{gok}), & : \quad \mathsf{Verify}(\mathsf{gpk}, m^\star, \sigma^\star) = 1 \ \wedge \\ & i \leftarrow \mathsf{Open}(\mathsf{gok}, m^\star, \sigma^\star) & (i = \perp \ \vee \ (i \notin \mathcal{Q}^{\mathsf{Key}} \ \wedge \\ & & (i, m^\star) \notin \mathcal{Q}^{\mathsf{Sig}})) \end{array}\right] \leq \varepsilon(\kappa),$$

where $\mathsf{Sig}(i, m)$ *returns* $\mathsf{Sign}(\mathsf{gsk}_i, m)$*,* $\mathsf{Key}(i)$ *returns* gsk_i*, and* $\mathcal{Q}^{\mathsf{Sig}}$ *and* $\mathcal{Q}^{\mathsf{Key}}$ *record the queries to the signing and key oracle respectively.*

We call a GS *secure*, if it is correct, anonymous and traceable.

2.4.6 Public Key Encryption

We also require public key encryption, which we recall below.

Definition 2.27. *A public key encryption scheme* Ω *is a triple* $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ *of PPT algorithms, which are defined as follows:*

$\mathsf{KeyGen}(1^\kappa)$: *This algorithm takes a security parameter* κ *as input and outputs a secret decryption key* sk *and a public encryption key* pk *(and we assume that the message space* $\mathcal{M}$ *is implicitly defined by* pk*).*

$\mathsf{Enc}(\mathsf{pk}, m)$: *This algorithm takes a public key* pk *and a message* $m \in \mathcal{M}$ *as input and outputs a ciphertext* c*.*

$\mathsf{Dec}(\mathsf{sk}, c)$: *This algorithm takes a secret key* sk *and a ciphertext* c *as input and outputs a message* $m \in \mathcal{M}$ *or* $\perp$*.*

Besides the obvious correctness property, we require a public key encryption scheme to be $\mathsf{IND\text{-}T}$ secure as formally recalled below.

Definition 2.28 ($\mathsf{IND\text{-}T}$ Security)**.** *Let* $\mathsf{T} \in \{\mathsf{CPA}, \mathsf{CCA2}\}$*. A public key encryption scheme* Ω *is* $\mathsf{IND\text{-}T}$ *secure, if for all PPT adversaries* $\mathcal{A}$ *there exists a negligible function* $\varepsilon(\cdot)$ *such that*

$$\Pr\left[\begin{array}{l|rl} (\mathsf{sk},\mathsf{pk}) \leftarrow \mathsf{KeyGen}(1^\kappa), & & \\ (m_0^\star, m_1^\star, \mathsf{ST}) \leftarrow \mathcal{A}^{\mathcal{O}_\mathsf{T}}(\mathsf{pk}), & & b = b^\star \ \wedge \\ b \xleftarrow{R} \{0,1\},\ c \leftarrow \mathsf{Enc}(\mathsf{pk}, m_b^\star), & : & c \notin \mathcal{Q}^{\mathsf{Dec}} \ \wedge \ |m_0^\star| = |m_1^\star| \\ b^\star \leftarrow \mathcal{A}^{\mathcal{O}_\mathsf{T}}(c, \mathsf{ST}) & & \end{array}\right] \leq 1/2 + \varepsilon(\kappa),$$

where the adversary runs in two stages,

$$\mathcal{O}_\mathsf{T} \leftarrow \begin{cases} \emptyset & \text{if} \quad \mathsf{T} = \mathsf{CPA}, \text{ and} \\ \{\mathcal{O}^{\mathsf{Dec}}(\mathsf{sk}, \cdot)\} & \text{if} \quad \mathsf{T} = \mathsf{CCA2}, \end{cases}$$

$\mathcal{Q}^{\mathsf{Dec}}$ *denotes the list of queries to* $\mathcal{O}^{\mathsf{Dec}}$ *in stage two and we set* $\mathcal{Q}^{\mathsf{Dec}} \leftarrow \emptyset$ *if* $\mathsf{T} = \mathsf{CPA}$*.*

2.4.7 Proxy Re-Encryption

A proxy re-encryption (PRE) scheme is an encryption scheme that allows a proxy to transform a message m encrypted under public key rpk_A of party A into a ciphertext to m under rpk_B for another party B, so that the proxy learns nothing about m. For our formal definitions, we largely follow [AFGH06].

Definition 2.29. *A* PRE *is a tuple* $(\mathsf{PGen}, \mathsf{KeyGen}, \vec{\mathsf{Enc}}, \vec{\mathsf{Dec}}, \mathsf{ReGen}, \mathsf{ReEnc})$ *of algorithms, where* $\vec{\mathsf{Enc}} = (\mathsf{Enc}^i)_{i\in[2]}$ *and* $\vec{\mathsf{Dec}} = (\mathsf{Dec}^i)_{i\in[2]}$*, which are defined as follows:*

$\mathsf{PGen}(1^\kappa)$: *Takes a security parameter* κ *and outputs parameters* PP*.*

$\mathsf{KeyGen}(\mathsf{pp})$: *Takes parameters* pp *and outputs a key pair* $(\mathsf{rsk}, \mathsf{rpk})$.

$\mathsf{ReGen}(\mathsf{rsk}_A, \mathsf{rpk}_B)$: *Takes a secret key* rsk_A *and a public key* rpk_B*, and outputs a re-encryption key* $\mathsf{rk}_{A\to B}$.

$\mathsf{Enc}^i(\mathsf{rpk}, m)$: *Takes a public key* rpk *and a message* m*, and outputs a ciphertext* c.

$\mathsf{ReEnc}(\mathsf{rk}_{A\to B}, c_A)$: *Takes a re-encryption key* $\mathsf{rk}_{A\to B}$ *and a ciphertext* c_A *under* rpk_A*, and outputs a re-encrypted ciphertext* c_B *for* rpk_B.

$\mathsf{Dec}^i(\mathsf{rsk}, c)$: *Takes a secret key* rsk *and a ciphertext* c*, and outputs* m.

A PRE scheme needs to be correct. This notion requires that for all security parameters $\kappa \in \mathbb{N}$, all honestly generated parameters $\mathsf{pp} \leftarrow \mathsf{PGen}(1^\kappa)$, all key pairs $(\mathsf{rsk}_A, \mathsf{rpk}_A) \leftarrow \mathsf{KeyGen}(\mathsf{pp})$, $(\mathsf{rsk}_B, \mathsf{rpk}_B) \leftarrow \mathsf{KeyGen}(\mathsf{pp})$, all re-encryption keys $\mathsf{rk}_{A\to B} \leftarrow \mathsf{ReGen}(\mathsf{rsk}_A, \mathsf{rpk}_B)$, all messages m it holds with probability one that

$$\begin{aligned}
&\forall\, i \in [2]\ \exists\, j \in [2]\ : \mathsf{Dec}^j(\mathsf{rsk}_A, \mathsf{Enc}^i(\mathsf{rpk}_A, m)) = m, \text{ and}\\
&\exists\, i \in [2]\ \exists\, j \in [2]\ : \mathsf{Dec}^j(\mathsf{rsk}_B, \mathsf{ReEnc}(\mathsf{rk}_{A\to B}, \mathsf{Enc}^i(\mathsf{pk}_A, m))) = m.
\end{aligned}$$

Thereby i and j determine the level of the ciphertexts. We will henceforth use the following semantics: first-level ciphertexts (Enc^1) cannot be re-encrypted by a proxy, whereas second-level ciphertexts (Enc^2) can be re-encrypted.

In addition, a PRE needs to be IND-CPA secure. We, henceforth, only require a relaxed IND-CPA notion which we term IND-CPA$^-$. It is clearly implied by the original IND-CPA notion from [AFGH06] (some oracles are omitted and the adversary only gets to see a second-level ciphertext).

Definition 2.30 (IND-CPA$^-$)**.** *A* PRE *is* IND-CPA$^-$ *secure, if for all PPT adversaries* $\mathcal{A}$ *there is a negligible function* $\varepsilon(\cdot)$ *such that*

$$\Pr\left[\begin{array}{l|l}
\mathsf{pp} \leftarrow \mathsf{PGen}(1^\kappa),\ b \xleftarrow{R} \{0,1\}, & \\
(\mathsf{sk}_t, \mathsf{pk}_t) \leftarrow \mathsf{KeyGen}(\mathsf{pp}), & \\
(\mathsf{sk}_h, \mathsf{pk}_h) \leftarrow \mathsf{KeyGen}(\mathsf{pp}), & \\
\mathsf{rk}_{t\to h} \leftarrow \mathsf{ReGen}(\mathsf{sk}_t, \mathsf{pk}_h), & \\
(m_0^\star, m_1^\star, \mathsf{st}) \leftarrow \mathcal{A}(\mathsf{pp}, \mathsf{pk}_t, \mathsf{pk}_h, \mathsf{rk}_{t\to h}), & : \ b = b^\star \\
c \leftarrow \mathcal{E}^2(m_b^\star, \mathsf{pk}_t),\ b^\star \leftarrow \mathcal{A}(\mathsf{st}, c) &
\end{array}\right] \leq 1/2 + \varepsilon(\kappa).$$

We remark that ReGen as defined in [AFGH06] also takes sk_h to cover schemes which require an interaction upon re-key generation. As we only deal with non-interactive ones, we omit it.

2.4.8 Non-Interactive Commitments

Below we recall the notion of non-interactive commitments. Henceforth, we may simply use the term commitment to refer to non-interactive commitments.

Definition 2.31. *A non-interactive commitment scheme is a tuple of PPT algorithms* $(\mathsf{PGen}, \mathsf{Commit}, \mathsf{Open})$*, which are defined as follows:*

$\mathsf{PGen}(1^\kappa)$: *Takes a security parameter* κ *as input and outputs public parameters* pp *(we note that* pp *implicitly define the message space* $\mathcal{M}$*).*

$\mathsf{Commit}(\mathsf{pp}, m)$: *Takes public parameters* pp *and a message* m *as input and outputs a commitment* C *together with a corresponding opening information* O*.*

$\mathsf{Open}(\mathsf{pp}, C, O)$: *Takes public parameters* pp *and a commitment* C *with corresponding opening information* O *as input, and outputs message* $m \in \mathcal{M} \cup \{\perp\}$*.*

We may sometimes omit pp as an input parameter for the Commit and Open algorithms to ease the notation. We call a non-interactive commitment scheme secure, if it is correct, (computationally) binding and (computationally) hiding. While correctness is straightforward and therefore omitted, the remaining security properties are defined as follows.

Definition 2.32 (Binding)**.** *A non-interactive commitment scheme is* binding*, if for all PPT adversaries* $\mathcal{A}$ *there is a negligible function* $\varepsilon(\cdot)$ *such that*

$$\Pr\left[\begin{array}{l|r} \mathsf{pp} \leftarrow \mathsf{PGen}(1^\kappa),\ (C^\star, O^\star, O'^\star) \leftarrow \mathcal{A}(\mathsf{pp}), & m \neq m' \wedge \\ m \leftarrow \mathsf{Open}(\mathsf{pp}, C^\star, O^\star),\ m' \leftarrow \mathsf{Open}(\mathsf{pp}, C^\star, O'^\star) & : \ m \neq \perp \wedge \\ & m' \neq \perp \end{array}\right] \leq \varepsilon(\kappa).$$

Definition 2.33 (Hiding)**.** *A non-interactive commitment scheme is* hiding*, if for all PPT adversaries* $\mathcal{A}$ *there is a negligible function* $\varepsilon(\cdot)$ *such that*

$$\Pr\left[\begin{array}{l|r} \mathsf{pp} \leftarrow \mathsf{PGen}(1^\kappa),\ (m_0^\star, m_1^\star, \mathsf{st}) \leftarrow \mathcal{A}(\mathsf{pp}), & b = b^\star \wedge \\ b \xleftarrow{R} \{0,1\},\ (C, O) \leftarrow \mathsf{Commit}(\mathsf{pp}, m_b^\star), & : \ m_0^\star \in \mathcal{M} \wedge \\ b^\star \leftarrow \mathcal{A}(C, \mathsf{st}) & m_1^\star \in \mathcal{M} \end{array}\right] \leq \frac{1}{2} + \varepsilon(\kappa).$$

Homomorphic Commitments. We call a commitment scheme *homomorphic* if for any $\kappa \in \mathbb{N}$, for any $\mathsf{pp} \leftarrow \mathsf{PGen}(1^\kappa)$, for any $m, m' \in \mathcal{M}$ we have $m \oplus m' = \mathsf{Open}(\mathsf{pp}, \mathsf{Commit}(\mathsf{pp}, m) \otimes \mathsf{Commit}(\mathsf{pp}, m'))$ for some binary operations $\oplus$ and $\otimes$.

We emphasize that any perfectly correct $\mathsf{IND\text{-}CPA}$ secure public key encryption schemes yields perfectly binding and computationally hiding commitments, e.g., ElGamal [Gam85], which is also homomorphic.

2.4.9 Σ-Protocols

Let L be an **NP**-language with statements x living in domain X and associated witness relation R so that $L = \{x \mid \exists w : R(x, w) = 1\} \subseteq \mathsf{X}$. A Σ-protocol for language L is defined as follows.

Definition 2.34. *A* Σ*-protocol for language* L *is an interactive three-move protocol between a PPT prover* $\mathsf{P} = (\mathsf{Commit}, \mathsf{Prove})$ *and a PPT verifier* $\mathsf{V} = (\mathsf{Challenge}, \mathsf{Verify})$*, where* P *makes the first move and transcripts are of the form* $(\mathsf{a}, \mathsf{e}, \mathsf{z}) \in \mathsf{A} \times \mathsf{E} \times \mathsf{Z}$*. Additionally they satisfy the following properties*

Completeness. *A Σ-protocol for language L is complete, if for all security parameters κ, and for all $(x, w) \in R$, it holds that*

$$\Pr[\langle \mathsf{P}(1^\kappa, x, w), \mathsf{V}(1^\kappa, x)\rangle = 1] = 1.$$

s-Special Soundness. *A Σ-protocol for language L is s-special sound if there exists a PPT extractor E so that for all x, and for all sets of accepting transcripts $\{(\mathsf{a}, \mathsf{e}_i, \mathsf{z}_i)\}_{i\in[s]}$ with respect to x where $\forall i, j \in [s], i \neq j : \mathsf{e}_i \neq \mathsf{e}_j$, generated by any algorithm with polynomial runtime in κ, it holds that*

$$\Pr\left[\ w \leftarrow \mathsf{E}(1^\kappa, x, \{(\mathsf{a}, \mathsf{e}_i, \mathsf{z}_i)\}_{i\in[s]}) \ : \ (x, w) \in R\ \right] \geq 1 - \varepsilon(\kappa).$$

Special Honest-Verifier Zero-Knowledge. *A Σ-protocol is special honest-verifier zero-knowledge, if there exists a PPT simulator S so that for every $x \in L$ and every challenge e from the challenge space, it holds that a transcript $(\mathsf{a}, \mathsf{e}, \mathsf{z})$, where $(\mathsf{a}, \mathsf{z}) \leftarrow \mathsf{S}(1^\kappa, x, \mathsf{e})$ is computationally indistinguishable from a transcript resulting from an honest execution of the protocol.*

The s-special soundness property gives an immediate bound for the soundness of the protocol: if no witness exists then (ignoring a negligible error) the prover can successfully answer at most to $^{s-1}/_{t}$ challenges, where $t = |\mathsf{E}|$ is the size of the challenge space. In case this value is too large, it is possible to reduce the soundness error using well known properties of Σ-protocols. Below we restate some properties for completeness (see, e.g., [Dam10, Sch17]).

Lemma 2.1. *The properties of Σ-protocols are invariant under parallel repetition. In particular, the ℓ fold parallel repetition of a Σ-protocol for relation R with challenge length t yields a new Σ-protocol with challenge length ℓt.*

Lemma 2.2. *If there exists a Σ-protocol for R with challenge length t, then there exists a Σ-protocol for R with challenge length t' for any t'.*

Below, we recall another well known fact about the AND composition of Σ-protocols.

Lemma 2.3. *Let L_1 and L_2 be two languages with associated witness relations R_1 and R_2, respectively. Further, let Σ_1 and Σ_2 be two Σ-protocols with identical challenge space so that Σ_1 is for L_1 and Σ_2 is for L_2. Then a Σ-protocol for the conjunction of L_1 and L_2, i.e., $L_1 \wedge L_2 := \{(x_1, x_2) \mid \exists\ w_1, w_2 : (x_1, w_1) \in R_1 \ \wedge\ (x_2, w_2) \in R_2\}$ is obtained by running Σ_1 and Σ_2 in parallel using a single common challenge e.*

Finally, we note that an equality (EQ) composition of Σ-protocols resulting in a language $L = \{(x_1, x_2) \mid \exists\ w : (x_1, w) \in R_1 \ \wedge\ (x_2, w) \in R_2\}$ can be achieved as a special case of an AND composition, where besides an identical challenge e also the same random tape is used for the prover in both instances.

The Fiat-Shamir Transform. The Fiat-Shamir (FS) transform [FS86] is a frequently used tool to convert Σ-protocols $\langle \mathsf{P}, \mathsf{V} \rangle$ to their non-interactive counterparts. Essentially, the transform removes the interaction between P and V by using a hash function $H : \mathsf{A} \times \mathsf{X} \to \mathsf{E}$, which is modeled as a RO, to obtain the challenge e.[1] That is, one uses a PPT algorithm $\mathsf{Challenge}'(1^\kappa, \mathsf{a}, x)$ which obtains $\mathsf{e} \leftarrow H(\mathsf{a}, x)$ and returns e. Then, the prover can locally obtain the challenge e *after* computing the initial message a. Starting a verifier $\mathsf{V}' = (\mathsf{Challenge}', \mathsf{Verify})$ on the same initial message a will then yield the same challenge e. More formally, we obtain the non–interactive PPT algorithms $(\mathsf{P}_H, \mathsf{V}_H)$ indexed by the used RO:

$\mathsf{P}_H(1^\kappa, x, w)$: Start P on $(1^\kappa, x, w)$, obtain the first message a, answer with $\mathsf{e} \leftarrow H(\mathsf{a}, x)$, and finally obtain z. Return $\pi \leftarrow (\mathsf{a}, \mathsf{z})$.

$\mathsf{V}_H(1^\kappa, x, \pi)$: Parse π as (a, z). Start V' on $(1^\kappa, x)$, send a as first message to the verifier. When V' outputs e, reply with z and output 1 if V accepts and 0 otherwise.

2.4.10 Non-Interactive Proof Systems

Now, we recall a standard definition of non-interactive proof systems (Π). Therefore, let L be an **NP**-language with witness relation R defined as $L = \{x \mid \exists\, w : R(x, w) = 1\}$.[2]

Definition 2.35. *A non-interactive proof system Π for language L is a tuple of algorithms* (Setup, Proof, Verify)*, which are defined as follows:*

$\mathsf{Setup}(1^\kappa)$: *This algorithm takes a security parameter κ as input, and outputs a common reference string* crs*.*

$\mathsf{Proof}(\mathsf{crs}, x, w)$: *This algorithm takes a common reference string* crs*, a statement x, and a witness w as input, and outputs a proof π.*

$\mathsf{Verify}(\mathsf{crs}, x, \pi)$: *This algorithm takes a common reference string* crs*, a statement x, and a proof π as input, and outputs a bit $b \in \{0, 1\}$.*

Below, we formally recall the security notions which are required in the context of this thesis.

Definition 2.36 (Completeness)**.** *A non-interactive proof system Π for language L is* complete*, if for every adversary $\mathcal{A}$ it holds that*

$$\Pr\left[\begin{array}{l|r} \mathsf{crs} \leftarrow \mathsf{Setup}(1^\kappa),\ (x^\star, w^\star) \leftarrow \mathcal{A}(\mathsf{crs}), & \mathsf{Verify}(\mathsf{crs}, x^\star, \pi) = 1 \\ \pi \leftarrow \mathsf{Proof}(\mathsf{crs}, x^\star, w^\star) & \vee\ (x^\star, w^\star) \notin R \end{array}\right] = 1.$$

[1] This is a stronger variant of FS (cf. [FKMV12, BPW12]). The original weaker variant of the FS transform does not include the statement x in the challenge computation.

[2] We want to acknowledge [Gro06, GS08, BGI14], who inspired our notation.

Definition 2.37 (Soundness)**.** *A non-interactive proof system* Π *for language* L *is* sound*, if for every PPT adversary* $\mathcal{A}$ *there is a negligible function* $\varepsilon(\cdot)$ *such that*

$$\Pr\left[\begin{array}{l|r} \mathsf{crs} \leftarrow \mathsf{Setup}(1^\kappa),\ (x^\star, \pi^\star) \leftarrow \mathcal{A}(\mathsf{crs}) & \mathsf{Verify}(\mathsf{crs}, x^\star, \pi^\star) = 1 \\ & \wedge\ x^\star \notin L \end{array}\right] \leq \varepsilon(\kappa).$$

If we quantify over all adversaries $\mathcal{A}$ and require $\varepsilon = 0$, we have perfect soundness, but we present the definition for computationally sound proofs (arguments).

Definition 2.38 (Adaptive Witness-Indistinguishability)**.** *A non-interactive proof system* Π *for language* L *is* adaptively witness-indistinguishable*, if for every PPT adversary* $\mathcal{A}$ *there is a negligible function* $\varepsilon(\cdot)$ *such that*

$$\Pr\left[\mathsf{crs} \leftarrow \mathsf{Setup}(1^\kappa),\ b \xleftarrow{R} \{0,1\},\ b^\star \leftarrow \mathcal{A}^{\mathcal{P}(\mathsf{crs},\cdot,\cdot,\cdot,b)}(\mathsf{crs}) \ :\ b = b^\star\right] \leq \varepsilon(\kappa),$$

where $\mathcal{P}(\mathsf{crs}, x, w_0, w_1, b) \coloneqq \mathsf{Proof}(\mathsf{crs}, x, w_b)$*, and* $\mathcal{P}$ *returns* $\perp$ *if* $(x, w_0) \notin R \ \vee\ (x, w_1) \notin R$*.*

If $\varepsilon = 0$, we have perfect adaptive witness-indistinguishability.

Definition 2.39 (Adaptive Zero-Knowledge)**.** *A non-interactive proof system* Π *for language* L *is* adaptively zero-knowledge*, if there exists a PPT simulator* $(\mathsf{S}_1, \mathsf{S}_2)$ *such that for every PPT adversary* $\mathcal{A}$ *there is a negligible function* $\varepsilon(\cdot)$ *such that*

$$\left|\begin{array}{l} \Pr\left[\mathsf{crs} \leftarrow \mathsf{Setup}(1^\kappa) \ :\ \mathcal{A}^{\mathcal{P}(\mathsf{crs},\cdot,\cdot)}(\mathsf{crs}) = 1\right] - \\ \Pr\left[(\mathsf{crs}, \tau) \leftarrow \mathsf{S}_1(1^\kappa) \ :\ \mathcal{A}^{\mathcal{S}(\mathsf{crs},\tau,\cdot,\cdot)}(\mathsf{crs}) = 1\right] \end{array}\right| \leq \varepsilon(\kappa),$$

where, τ *denotes a simulation trapdoor. Thereby,* $\mathcal{P}(\mathsf{crs}, x, w)$ *and* $\mathcal{S}(\mathsf{crs}, \tau, x, w)$ *return* $\perp$ *if* $(x, w) \notin R$ *or* $\pi \leftarrow \mathsf{Proof}(\mathsf{crs}, x, w)$ *and* $\pi \leftarrow \mathsf{S}_2(\mathsf{crs}, \tau, x)$*, respectively, otherwise.*

If $\varepsilon = 0$, we have perfect adaptive zero-knowledge.

Definition 2.40 (Proof of Knowledge)**.** *A non-interactive proof system* Π *for language* L *admits proofs of knowledge, if there exists a PPT extractor* $\mathsf{E} = (\mathsf{E}_1, \mathsf{E}_2)$ *such that for every PPT adversary* $\mathcal{A}$ *there is a negligible function* $\varepsilon_1(\cdot)$ *such that*

$$\left|\begin{array}{l} \Pr\left[\mathsf{crs} \leftarrow \mathsf{Setup}(1^\kappa) \ :\ \mathcal{A}(\mathsf{crs}) = 1\right] - \\ \Pr\left[(\mathsf{crs}, \xi) \leftarrow \mathsf{E}_1(1^\kappa) \ :\ \mathcal{A}(\mathsf{crs}) = 1\right] \end{array}\right| \leq \varepsilon_1(\kappa),$$

and for every PPT adversary $\mathcal{A}$ *there is a negligible function* $\varepsilon_2(\cdot)$ *such that*

$$\Pr\left[\begin{array}{l|r} (\mathsf{crs}, \xi) \leftarrow \mathsf{E}_1(1^\kappa),\ (x^\star, \pi^\star) \leftarrow \mathcal{A}(\mathsf{crs}), & \mathsf{Verify}(\mathsf{crs}, x^\star, \pi^\star) = 1\ \wedge \\ w \leftarrow \mathsf{E}_2(\mathsf{crs}, \xi, x^\star, \pi^\star) & (x^\star, w) \notin R \end{array}\right] \leq \varepsilon_2(\kappa).$$

Definition 2.41 (Simulation Sound Extractability)**.** *An adaptively zero-knowledge non-interactive proof system* Π *for language* L *is simulation sound extractable, if there exists a PPT extractor* (S, E) *such that for every adversary* $\mathcal{A}$ *it holds that*

$$\left| \begin{array}{l} \Pr\left[(\mathsf{crs}, \tau) \leftarrow \mathsf{S}_1(1^\kappa) \ : \ \mathcal{A}(\mathsf{crs}, \tau) = 1\right] - \\ \Pr\left[(\mathsf{crs}, \tau, \xi) \leftarrow \mathsf{S}(1^\kappa) \ : \ \mathcal{A}(\mathsf{crs}, \tau) = 1\right] \end{array} \right| = 0,$$

and for every PPT adversary $\mathcal{A}$ *there is a negligible function* $\varepsilon_2(\cdot)$ *such that*

$$\Pr\left[\begin{array}{l} (\mathsf{crs}, \tau, \xi) \leftarrow \mathsf{S}(1^\kappa), \\ (x^\star, \pi^\star) \leftarrow \mathcal{A}^{\mathcal{S}(\mathsf{crs}, \tau, \cdot)}(\mathsf{crs}), \\ w \leftarrow \mathsf{E}(\mathsf{crs}, \xi, x^\star, \pi^\star) \end{array} \ : \ \begin{array}{r} \mathsf{Verify}(\mathsf{crs}, x^\star, \pi^\star) = 1 \ \wedge \\ (x^\star, \pi^\star) \notin \mathcal{Q}_\mathsf{S} \ \wedge \ (x^\star, w) \notin R \end{array}\right] \leq \varepsilon_2(\kappa),$$

where $\mathcal{S}(\mathsf{crs}, \tau, x) \coloneqq \mathsf{S}_2(\mathsf{crs}, \tau, x)$ *and* $\mathcal{Q}_\mathsf{S}$ *keeps track of the queries to and answers of* $\mathcal{S}$*.*

Note that the definition of simulation sound extractability of [Gro06] is stronger than ours in the sense that the adversary also gets the trapdoor ξ as input. However, in our context this weaker notion (previously also used in other works such as [DHLW10, ADK+13]) is sufficient.

Definition 2.42 (Weak Simulation Sound Extractability)**.** *An adaptively zero-knowledge non-interactive proof system* Π *for language* L *is weakly simulation sound extractable, if it satisfies Definition 2.41 with the following modified winning condition:*

$$\mathsf{Verify}(\mathsf{crs}, x^\star, \pi^\star) = 1 \ \wedge \ (x^\star, \cdot) \notin \mathcal{Q}_\mathsf{S} \ \wedge \ (x^\star, w) \notin R.$$

The Groth-Sahai (GS) Proof System [GS08, GS07]. GS proofs are non-interactive witness-indistinguishable (NIWI) and zero-knowledge (NIZK) proofs for the satisfiability of various types of equations defined over bilinear groups. In this thesis we only require proofs for the satisfiability of pairing product equations (PPEs) of the form

$$\prod_{i=1}^{n} e(a_i, \underline{\hat{y}_i}) \cdot \prod_{i=1}^{m} e(\underline{x_i}, \hat{b}_i) \cdot \prod_{i=1}^{m} \prod_{j=1}^{n} e(\underline{x_i}, \underline{\hat{y}_j})^{\gamma_{ij}} = t_T. \tag{2.1}$$

Such a pairing product equation implicitly defines an **NP** relation, where the vectors $(x_1, \dots x_m) \in \mathbb{G}_1^m$, $(\hat{y}_1, \dots, \hat{y}_n) \in \mathbb{G}_2^n$ constitute the witness with respect to the statement $(a_1, \dots, a_n) \in \mathbb{G}_1^n, (\hat{b}_1, \dots, \hat{b}_m) \in \mathbb{G}_2^m, (\gamma_{ij})_{i \in [m], j \in [n]} \in \mathbb{Z}_p^{n \cdot m}$, and $t_T \in \mathbb{G}_T$. To conduct a proof, one commits to the vectors $(x_i)_{i \in [m]}$ and $(\hat{y}_i)_{i \in [n]}$, and uses the commitments instead of the actual values in the PPE. Loosely speaking, the proof π is used to "cancel out" the randomness used in the commitments. However, this does not directly work when using the groups $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$, but requires to map the involved elements to vector spaces

associated to these groups. For instance, in the SXDH setting those vector spaces are $\mathbb{G}_1^2$, $\mathbb{G}_2^2$ and $\mathbb{G}_T^4$ and one uses the corresponding bilinear map $F : \mathbb{G}_1^2 \times \mathbb{G}_2^2 \to \mathbb{G}_T^4$ to prove the satisfiability of PPE.

We do not require further details on the instantiation of GS proofs in the context of this thesis and refer the reader to [GS07] for an in-depth treatment. Nevertheless, we want to explicitly point out three properties of GS proofs:

CRS Indistinguishability: The CRS in the GS proof system can be set up in two different "modes". The first mode yields unconditional soundness, whereas the second mode yields unconditional witness indistinguishability (WI)/zero-knowledge (ZK). The important point is that both modes are computationally indistinguishable which is why both soundness and WI/ZK can be shown to hold at the same time.

Requirements to Obtain Zero-Knowledge: Recall that zero-knowledge requires a PPT simulator which is able to simulate proofs without knowing any witness. In the context of the GS framework, such a simulator is only known to exist for a special form of the PPE [GS07, EG14]. That is, a form of the PPE where the simulator can efficiently find a satisfying witness on its own (e.g., if the PPE is so that one can use $(1, \ldots, 1) \in \mathbb{G}_1^m, (1, \ldots, 1) \in \mathbb{G}_2^n$ as a trivial satisfying witness). It is important to note that such a form of the PPE often requires additional measures to prevent the actual prover from using those "trivial" witnesses (e.g., disjunctions of multiple equations [Gro06, BFG13] as discussed below).

Expressing Disjunctions of PPE*s:* Groth [Gro06] showed that one can express disjunctions of PPEs. We illustrate how this is done using a simple example, which is tailored to our ring signature scheme in Section 4.1. For a general description we refer the reader to [Gro06]. Assume that we have ℓ PPEs of the form $\{e(\underline{x}_i, \hat{g}) = e(h, \hat{x}_i)\}_{i \in [\ell]}$ and want to prove that we know a satisfying assignment for at least one i. To do so, we prove the equations

$$\{e(\underline{x}_i, \hat{g}) = e(h, \underline{\hat{x}}_i)\}_{i \in [\ell]}, \tag{2.2}$$

$$e(g^{-1} \prod_{i \in [\ell]} \underline{g}_i, \hat{g}) = 1, \text{ and} \tag{2.3}$$

$$\{e(\underline{g}_i, (\hat{x}_i)^{-1} \underline{\hat{x}}_i) = 1\}_{i \in [\ell]}. \tag{2.4}$$

Equation 2.2 modifies the original set of equations so that one additionally proves knowledge of $\hat{x}_i$. This equation can now be trivially satisfied. To ensure that the prover, however, uses a non-trivial witness for at least one of the equations one needs two additional equations. Equation 2.3 constitutes a selector equation, which can only be satisfied if at least one $\underline{g}_i$ is a commitment to $g_i \neq 1$. Equation 2.4 additionally ensures that the commitment $\underline{\hat{x}}_i$ actually contains $\hat{x}_i$ for each i where $\underline{g}_i$ is with respect to a $g_i \neq 1$. We also note that [BFG13] show a more efficient technique tailored to a disjunction of two PPEs which have a special form. This technique may, e.g., be useful to instantiate our schemes in Section 3.3 and Section 3.4.

Converting Σ-protocols to Non-Interactive Proof Systems. One can obtain a non-interactive proof system satisfying the properties above by applying the FS transform to any Σ-protocol where the min-entropy α of the commitment a sent in the first phase is so that $2^{-\alpha}$ is negligible in the security parameter κ and the challenge space E is exponentially large in the security parameter. Formally, $\mathsf{Setup}(1^\kappa)$ fixes a hash function $H : \mathsf{A} \times \mathsf{X} \to \mathsf{E}$, which is modeled as a RO, sets $\mathsf{crs} \leftarrow (1^\kappa, H)$ and returns crs. The algorithms Proof and Verify are defined as follows:

$$\mathsf{Proof}(\mathsf{crs}, x, w) := \mathsf{P}_H(1^\kappa, x, w), \quad \mathsf{Verify}(\mathsf{crs}, x, \pi) := \mathsf{V}_H(1^\kappa, x, \pi).$$

Combining [FKMV12, Thm. 1, Thm. 2, Prop. 1] (among others) shows that a so-obtained proof system is complete, sound, and adaptively zero-knowledge if the underlying Σ-protocol is 2-special sound and the commitments sent in the first move are unconditionally binding. Furthermore, [FKMV12, Thm. 2] shows, that such a proof system is also simulation sound if the Σ-protocol additionally provides quasi-unique responses [Fis05], i.e., it is computationally infeasible to find a new valid response when given an accepting proof. Finally, [FKMV12, Thm. 3] shows that such a poof system is even simulation sound extractable when allowing the extractor to rewind the adversary.

2.4.11 Signatures of Knowledge.

Below we recall signatures of knowledge (SoKs) [CL06], where L is as above. For the formal notions we follow [BCC+15] and use a generalization of the original extraction property termed f-extractability. A signature of knowledge (SoK) for L is defined as follows.

Definition 2.43. *A* SoK *for language* L *is a tuple of PPT algorithms* $(\mathsf{Setup}, \mathsf{Sign}, \mathsf{Verify})$*, which are defined as follows:*

$\mathsf{Setup}(1^\kappa)$: *This algorithm takes a security parameter* κ *as input and outputs a common reference string* crs*. We assume that the message space* $\mathcal{M}$ *is implicitly defined by* crs*.*

$\mathsf{Sign}(\mathsf{crs}, x, w, m)$: *This algorithm takes a common reference string* crs*, a word* x*, a witness* w*, and a message* m *as input and outputs a signature* σ*.*

$\mathsf{Verify}(\mathsf{crs}, x, m, \sigma)$: *This algorithm takes a common reference string* crs*, a word* x*, a message* m*, and a signature* σ *as input and outputs a bit* $b \in \{0, 1\}$*.*

Definition 2.44 (Correctness)**.** *A* SoK *with respect to* L *is correct, if there exists a negligible function* $\varepsilon(\cdot)$ *such that for all* $x \in L$*, for all* w *such that* $(x, w) \in R$*, and for all* $m \in \mathcal{M}$ *it holds that*

$$\Pr\left[\mathsf{crs} \leftarrow \mathsf{Setup}(1^\kappa),\ \sigma \leftarrow \mathsf{Sign}(\mathsf{crs}, x, w, m) : \mathsf{Verify}(\mathsf{crs}, x, m, \sigma) = 1\right] \geq 1 - \varepsilon(\kappa).$$

Definition 2.45 (Simulatability)**.** *A* SoK *with respect to L is simulatable, if there exists a PPT simulator $\mathcal{S} = (\mathsf{SimSetup}, \mathsf{SimSign})$ such that for all PPT adversaries $\mathcal{A}$ there exists a negligible function $\varepsilon(\cdot)$ such that it holds that*

$$\left| \begin{array}{l} \Pr\left[\mathsf{crs} \leftarrow \mathsf{Setup}(1^\kappa),\ b \leftarrow \mathcal{A}^{\mathsf{Sign}(\mathsf{crs},\cdot,\cdot,\cdot)}(\mathsf{crs}) \ :\ b = 1\right] \ - \\ \Pr\left[(\mathsf{crs},\tau) \leftarrow \mathsf{SimSetup}(1^\kappa),\ b \leftarrow \mathcal{A}^{\mathsf{Sim}(\mathsf{crs},\tau,\cdot,\cdot,\cdot)}(\mathsf{crs}) \ :\ b = 1\right] \end{array} \right| \leq \varepsilon(\kappa),$$

where we set $\mathsf{Sim}(\mathsf{crs},\tau,x,w,m) \coloneqq \mathsf{SimSign}(\mathsf{crs},\tau,x,m)$ *and* Sim *only responds if $(x,w) \in R$.*

Definition 2.46 (f-Extractability)**.** *A* SoK *with respect to L is f-extractable, if in addition to $\mathcal{S}$ there exists a PPT extractor* Extract*, such that for all PPT adversaries $\mathcal{A}$ there exists a negligible function $\varepsilon(\cdot)$ such that it holds that*

$$\Pr\left[\begin{array}{l} (\mathsf{crs},\tau) \leftarrow \mathsf{SimSetup}(1^\kappa), \\ (x,m,\sigma) \leftarrow \mathcal{A}^{\mathsf{Sim}(\mathsf{crs},\tau,\cdot,\cdot,\cdot)}(\mathsf{crs}), \\ y \leftarrow \mathsf{Extract}(\mathsf{crs},\tau,x,m,\sigma) \end{array} \ :\ \begin{array}{r} \mathsf{Verify}(\mathsf{crs},x,m,\sigma) = 0 \ \vee \\ (x,m,\sigma) \in \mathcal{Q}^{\mathsf{Sim}} \ \vee \\ (\exists\ w : (x,w) \in R \ \wedge \\ y = f(w)) \end{array}\right] \geq 1 - \varepsilon(\kappa),$$

where $\mathcal{Q}^{\mathsf{Sim}}$ denotes the queries (resp. answers) of Sim*.*

We note that, as illustrated in [BCC+15], this notion is a generalization of the original extractability notion from [CL06] which implies the original extractability notion if f is the identity. In this case, we simply call the f-extractability property *extractability*. Analogous to [BCC+15], we require the used SoK to be at the same time extractable and *straight-line f-extractable* with respect to some f other than the identity, where straight-line as usual says that the extractor runs without rewinding the adversary [Fis05].

3
Basic Primitives

This chapter is dedicated to our results on basic primitives. They can be seen as building blocks for schemes and protocols in the areas of interest in this thesis. Yet, they are also suitable for a broader range of applications in general. The results include generalisations of and new notions for existing primitives. In particular, we target cryptographic accumulators in Section 3.1 and a framework capturing key-homomorphic properties for digital signatures and applications of this framework in Section 3.2, Section 3.3, and Section 3.4.

3.1 Cryptographic Accumulators

A (static) cryptographic accumulator scheme allows to accumulate a finite set $\mathcal{X} = \{x_1, \ldots, x_n\}$ into a succinct value $\Lambda_{\mathcal{X}}$, the so called accumulator. For every element $x_i \in \mathcal{X}$, one can efficiently compute a so called witness wit_{x_i} to certify the membership of x_i in $\Lambda_{\mathcal{X}}$. However, it should be computationally infeasible to find a witness for any non-accumulated value $y \notin \mathcal{X}$ (*collision freeness*). Dynamic accumulators additionally allow the dynamic addition/deletion of values to/from a given accumulator and to update existing witnesses accordingly (without the need to fully recompute these values on each change of the accumulated set). Universal accumulators additionally provide means to obtain non-membership witnesses for non-members $y \notin \mathcal{X}$. Here, collision freeness also covers that it is computationally infeasible to create non-membership witnesses for values $x_i \in \mathcal{X}$. Over time, further security properties, that is, *undeniability* and *indistinguishability*, have been proposed. Undeniability is specific to universal accumulators and says that it should be computationally infeasible to compute two contradicting witnesses for $z \in \mathcal{X}$ and $z \notin \mathcal{X}$. Indistinguishability

says that neither the accumulator nor the witnesses leak information about the accumulated set $\mathcal{X}$ and, thus, requires randomized accumulator schemes.

Applications. Accumulators were originally proposed for timestamping purposes [BdM93], i.e., to record the existence of a value at a particular point in time. Over time, other applications such as membership testing, distributed signatures, accountable certificate management [BLL00] and authenticated dictionaries [GTH02] have been proposed. Accumulators are also used as building blocks in redactable (cf. Section 5.1), sanitizable (cf. Section 5.3), P-homomorphic signatures [ABC+12], anonymous credentials [SNF11], group signatures [TX03], privacy-preserving data outsourcing [Sla12] as well as for authenticated data structures [GOT15]. Moreover, accumulator schemes that allow to prove the knowledge of a (non-membership) witness for an unrevealed value in zero-knowledge (introduced for off-line e-cash in [STY00]) are now widely used for revocation of group signatures and anonymous credentials [CL02b]. Quite recently, accumulators were also used in Zerocoin [MGGR13], an anonymity extension to the Bitcoin cryptocurrency.

Since their introduction, numerous accumulator schemes with somewhat different features have been proposed. Basically, the major lines of work are schemes in hidden-order groups (RSA), known-order groups (DL) and hash-based constructions (which may use, but typically do not require number theoretic assumptions).

Hidden-Order Groups. The original RSA-based scheme of Benaloh and de Mare [BdM93] has been refined by Barić and Pfitzmann [BP97], who strengthened the original security notion to *collision freeness*. In [San99], Sander proposed to use RSA moduli with unknown factorization to construct trapdoor-free accumulators. Camenisch and Lysyanskaya [CL02b] extended the scheme in [BP97] with capabilities to dynamically add/delete values to/from the accumulator, which constituted the first *dynamic accumulator* scheme. Their scheme also supports *public updates* of existing witnesses, that is, updates without the knowledge of any trapdoor. Later, Li et al. [LLX07] added support for non-membership witnesses to [CL02b] and, therefore, obtained *universal dynamic accumulators*. They also proposed an optimization for more efficient updates of non-membership witnesses, for which, however, weaknesses have been identified later [PB10, MV13]. Lipmaa [Lip12] generalized RSA accumulators to modules over Euclidean rings. In all aforementioned schemes, the accumulation domain is restricted to primes to guarantee collision freeness. In [TX03], Tsudik and Xu proposed a variation of [CL02b], which allows to accumulate semiprimes. This yields a collision-free accumulator under the assumption that the used semiprimes are hard to factor and their factorization is not publicly known. Moreover, in [WWP07] an accumulator scheme that allows to accumulate arbitrary integers and supports batch updates of witnesses has been proposed. Yet, this scheme was broken in [CH10].

Known-Order Groups. In [Ngu05], Nguyen proposed a dynamic accumulator scheme which works in pairing-friendly groups of prime order p. It is secure under the t-SDH assumption and allows to accumulate up to t values from the domain $\mathbb{Z}_p$. Later, Damgård and Triandopoulos [DT08] as well as Au et al. [ATSM09] extended Nguyen's scheme with universal features. Quite recently, Acar and Nguyen [AN11] eliminated the upper bound t on the number of accumulated elements of the t-SDH accumulator. To this end, they use a set of accumulators, each containing a subset of the whole set to be accumulated. An alternative accumulator scheme for pairing friendly groups of prime order has been introduced by Camenisch et al. [CKS09]. It supports public updates of witnesses and the accumulator and its security relies on the t-DHE assumption.

Hash-Based Constructions. Buldas et al. [BLL00, BLL02] presented the very first universal dynamic accumulator that satisfies *undeniability* (termed as undeniable attester and formalized in the context of accumulators in [Lip12]). Their construction is based on collision-resistant hashing and the use of hash-trees. Another hash-tree based construction of a universal accumulator that satisfies a notion similar to undeniability has been proposed in [CHKO08] (the scheme is called a strong universal accumulator). Quite recently, another accumulator based on hash-trees, which uses commitments based on bivariate polynomials modulo RSA composites as a collision-resistant hash function, has been introduced in [BC14].

For the sake of completeness, we also mention the construction of static accumulators in the random oracle model based on Bloom filters, proposed by Nyberg [Nyb96a, Nyb96b].

Subsequent Work. We also want to note that there exists follow-up work on various aspects of accumulators. In [RY16], Reyzin and Yakoubov aim to mitigate the requirement to keep the accumulator and the witnesses synchronized upon modifications of the accumulated set. They introduce a formal model and present a corresponding construction which—within some natural limits—allows to verify outdated witnesses with respect to an up to date accumulator and vice versa. In [GOP+16], Gosh et al. strengthen our indistinguishability[1] notion, further investigate the relation of accumulators to other similar primitives, and extend the accumulator model regarding more expressive operations on the accumulated set. In [ZKP17] Zhang et al. present further expressiveness extensions to accumulators (others than the ones in [GOP+16]). In [BCD+17], Baldimtsi et al. establish a more modular view on accumulators and show how to combine accumulators with more basic features and weaker security guarantees to obtain accumulators with more sophisticated features and stronger security guarantees.

Contribution. The contributions presented in this section are as follows:

[1] Note that indistinguishability is better suited for all applications we target in this thesis.

- While some papers [BdM93, BP97, CL02b, ATSM09, Ngu05] do not explicitly formalize accumulator schemes, formal definitions are given in [CKS09, WWP07, FN02, LLX07, CF13, Lip12, CHKO08, AN11]. However, these models are typically tailored to the functionalities of the respective scheme. While they widely match for the basic notion of (static) accumulators (with the exception of considering randomized accumulators), they differ when it comes to dynamic and universal accumulators. To overcome this issue, we propose a unified formal model for accumulators, which is especially valuable when treating accumulators in a black-box fashion. We, thereby, also include the notion of undeniability [BLL00, BLL02, Lip12] and a strengthened version of the recent indistinguishability notion [dMLPP12]. Besides, we also confirm the intuition and show that undeniability is a strictly stronger notion than collision freeness.
- We provide an exhaustive classification of existing accumulator schemes and show that most existing accumulator schemes are distinguishable in our model. To resolve this issue, we propose a simple, light-weight generic transformation that allows to add indistinguishability to existing dynamic accumulators and prove the security of the so-obtained schemes. As this transformation, however, comes at the cost of reduced collision freeness, we additionally propose the first indistinguishable scheme that does not suffer from this shortcoming.
- Since accumulators are somehow related to commitments to sets [KZG10, FLZ14], commitments to vectors [CF13] and to zero-knowledge sets [MRK03], it is interesting to study their relationship. We formally show that indistinguishable accumulators imply non-interactive commitment schemes. Furthermore, we formally show that zero-knowledge sets imply indistinguishable, undeniable universal accumulators, yielding the first construction of such accumulators.

Henceforth, we use Λ to denote an accumulator and if we want to make the accumulated set $\mathcal{X} = \{x_1, \ldots, x_n\}$ explicit, we write $\Lambda_{\mathcal{X}}$. Given an accumulator $\Lambda_{\mathcal{X}}$, a membership witness for an element $x_i \in \mathcal{X}$ is denoted by wit_{x_i}, whereas a non-membership witness for an element $y_j \notin \mathcal{X}$ is denoted by $\underline{\mathsf{wit}}_{y_j}$. The accumulator secret key (trapdoor) is denoted by sk_Λ, while the public key is denoted by pk_Λ.

3.1.1 A Unified Model for Cryptographic Accumulators

In the original sense, accumulator schemes were defined by the following properties (see, e.g., [CL02b, LLX07]).

Efficient generation: There is an efficient algorithm that, on input of a security parameter κ, selects a functionality $f : \mathcal{Z}_A^\kappa \times \mathcal{Z}_I^\kappa \to \mathcal{Z}_A^\kappa$ from a family of families of functions $\{\mathcal{F}_\kappa\}$, i.e., generates the accumulator specific key pair $(\mathsf{sk}_\Lambda, \mathsf{pk}_\Lambda)$, where sk_Λ is a trapdoor for f.

Efficient evaluation: There is an efficient algorithm that computes $f(\Lambda, x)$.

Quasi-commutativity: It holds that $f(f(\Lambda, x_1), x_2) = f(f(\Lambda, x_2), x_1)\ \forall\ x_1, x_2 \in \mathcal{Z}_I^\kappa, \Lambda \in \mathcal{Z}_A^\kappa$.

Assuming that it is computationally infeasible to invert f without knowing sk_Λ, the quasi-commutativity directly yields a way to define witnesses. For instance, $f(\Lambda, x_1)$ can serve as witness for the accumulation of x_2. Nonetheless, it is more meaningful to provide a more abstract algorithmic definition of accumulators as done below, since there are several constructions that do not fit into this characterization (for instance, hash-tree constructions do not require the quasi-commutativity property).

Trusted vs. Non-Trusted Setup. Known accumulators that rely on number theoretic assumptions require a trusted setup, i.e., a TTP runs the setup algorithm Gen and discards the trapdoor sk_Λ afterwards. Here, access to sk_Λ allows to break collision freeness (and its stronger form: undeniability). Consequently, correctness of the accumulator scheme also needs to hold if sk_Λ is omitted in all algorithms, which is the case for all existing schemes. In contrast, in constructions relying on collision-resistant hash functions (not based on number theoretic assumptions) there is no trapdoor at all and, therefore, no trusted setup is required. To study number theoretic accumulators without trusted setup, Lipmaa [Lip12] proposed a modified model which divides the Gen algorithm into a Setup and a Gen algorithm. In this model, the adversary can control the randomness used inside Setup and, thus, knows the trapdoor. Nevertheless, it can neither access nor influence the randomness of the Gen algorithm. This model, however, still requires a partially trusted setup and also does not fit to the known order group setting, which makes it not generally applicable.[2] Consequently, when considering the state of the art it seems most reasonable to define a security model with respect to a trusted setup. We emphasize that this model is compatible with all existing constructions. Nevertheless, it remains a challenging open issue to design accumulators based on standard assumptions which are secure without any trusted setup.

Definitions. In the following, we provide a definition for (static) accumulators, which we adapt from [WWP07, FN02]. In contrast to previous models, we explicitly consider randomized accumulator schemes. Then, we extend this model in order to formalize dynamic accumulators. It is similar to [CKS09, CF13], but avoids shortcomings such as missing private updates. Based on this, we define universal and universal dynamic accumulators and propose a suitable security model. Furthermore, we discuss undeniable and indistinguishable accumulators, give formalizations for these properties, and, investigate relationships between security properties.

[2] This model is tailored to the hidden order group setting, where Setup produces a composite modulus N. Gen chooses a random generator g of a large subgroup of $\mathbb{Z}_N^*$. Then, the adversary knows the factorization of N but does not control the choice of g. RSA accumulators are obviously insecure in this setting, but Lipmaa provides secure solutions based on modules over an Euclidean ring, which, however, rely on rather unstudied assumptions.

We call accumulators that have an upper bound t on the number of accumulated values *t-bounded accumulators* and *unbounded* otherwise. To model this, our Gen algorithm takes an additional parameter t, where $t = \infty$ is used to indicate that the accumulator is unbounded. For the sake of completeness, we model the algorithms such that they support an optional input of the trapdoor (denoted as $\mathsf{sk}_\Lambda^\sim$) since this often allows to make the algorithms more efficient. However, we stress that we consider the trusted setup model and, hence, adversaries are not given access to the trapdoor sk_Λ. Consequently, if $\mathsf{sk}_\Lambda^\sim$ is set, the party running the algorithm needs to be fully trusted.

Definition 3.1 (Static Accumulator)**.** *A static accumulator is a tuple of efficient algorithms* (Gen, Eval, $\mathsf{WitCreate}$, Verify) *which are defined as follows:*

$\mathsf{Gen}(1^\kappa, t)$: *This algorithm takes a security parameter κ and a parameter t. If $t \neq \infty$, then t is an upper bound on the number of elements to be accumulated. It returns a key pair $(\mathsf{sk}_\Lambda, \mathsf{pk}_\Lambda)$, where $\mathsf{sk}_\Lambda = \emptyset$ if no trapdoor exists.*

$\mathsf{Eval}((\mathsf{sk}_\Lambda^\sim, \mathsf{pk}_\Lambda), \mathcal{X})$: *This (probabilistic)*[3] *algorithm takes a key pair $(\mathsf{sk}_\Lambda^\sim, \mathsf{pk}_\Lambda)$ and a set $\mathcal{X}$ to be accumulated and returns an accumulator $\Lambda_\mathcal{X}$ together with some auxiliary information aux.*

$\mathsf{WitCreate}((\mathsf{sk}_\Lambda^\sim, \mathsf{pk}_\Lambda), \Lambda_\mathcal{X}, \mathsf{aux}, x_i)$: *This algorithm takes a key pair $(\mathsf{sk}_\Lambda^\sim, \mathsf{pk}_\Lambda)$, an accumulator $\Lambda_\mathcal{X}$, auxiliary information aux and a value x_i. It returns $\perp$, if $x_i \notin \mathcal{X}$, and a witness wit_{x_i} for x_i otherwise.*

$\mathsf{Verify}(\mathsf{pk}_\Lambda, \Lambda_\mathcal{X}, \mathsf{wit}_{x_i}, x_i)$: *This algorithm takes a public key pk_Λ, an accumulator $\Lambda_\mathcal{X}$, a witness wit_{x_i} and a value x_i. It returns 1 if wit_{x_i} is a witness for $x_i \in \mathcal{X}$ and 0 otherwise.*

Henceforth, we call an accumulator *randomized* if the Eval algorithm is probabilistic. Based on Definition 3.1, we can now formalize *dynamic accumulators*. We widely align our definitions with [WWP07, FN02], but, in addition, we need to consider that the various dynamic accumulator schemes proposed so far differ regarding the *public updatability* of witnesses and the accumulator.

Definition 3.2 (Dynamic Accumulator)**.** *A dynamic accumulator is a static accumulator that additionally provides efficient algorithms* (Add, Delete, $\mathsf{WitUpdate}$) *which are defined as follows:*

$\mathsf{Add}((\mathsf{sk}_\Lambda^\sim, \mathsf{pk}_\Lambda), \Lambda_\mathcal{X}, \mathsf{aux}, x_i)$: *This algorithm takes a key pair $(\mathsf{sk}_\Lambda^\sim, \mathsf{pk}_\Lambda)$, an accumulator $\Lambda_\mathcal{X}$, auxiliary information aux, as well as a value x_i to be added. If $x_i \in \mathcal{X}$, it returns $\perp$. Otherwise, it returns the updated accumulator $\Lambda_{\mathcal{X}'}$ with $\mathcal{X}' \leftarrow \mathcal{X} \cup \{x_i\}$, the updated auxiliary information aux', as well as some auxiliary witness update information aux_u.*

[3] If Eval is probabilistic, the internally used randomness is denoted as r. If we want to make the randomness used by the Eval algorithm explicit, we will write Eval_r.

$\mathsf{Delete}((\mathsf{sk}_\Lambda^{\sim}, \mathsf{pk}_\Lambda), \Lambda_\mathcal{X}, \mathsf{aux}, x_i)$: *This algorithm takes a key pair* $(\mathsf{sk}_\Lambda^{\sim}, \mathsf{pk}_\Lambda)$*, an accumulator* $\Lambda_\mathcal{X}$*, auxiliary information* aux*, as well as a value* x_i *to be removed. If* $x_i \notin \mathcal{X}$*, it returns* $\perp$*. Otherwise, it returns the updated accumulator* $\Lambda_{\mathcal{X}'}$ *with* $\mathcal{X}' \leftarrow \mathcal{X} \setminus \{x_i\}$ *and auxiliary information* aux'*, as well as some auxiliary witness update information* aux_u.

$\mathsf{WitUpdate}((\mathsf{sk}_\Lambda^{\sim}, \mathsf{pk}_\Lambda), \mathsf{wit}_{x_i}, \mathsf{aux}_\mathsf{u}, x_j)$: *This algorithm takes a key pair* $(\mathsf{sk}_\Lambda^{\sim}, \mathsf{pk}_\Lambda)$*, a witness* wit_{x_i} *to be updated, auxiliary update information* aux_u *and a value* x_j *which was added/deleted to/from the accumulator. It returns an updated witness* wit'_{x_i} *on success and* $\perp$ *otherwise.*

Below, we define *universal accumulators* and emphasize that features provided by universal accumulators can be seen as supplementary features for both, *static* and *dynamic* accumulators.

Definition 3.3 (Universal Accumulator)**.** *A universal accumulator is a static or a dynamic accumulator with the following properties. For static accumulator schemes the algorithms* WitCreate *and* Verify *take an additional boolean parameter* type*, indicating whether the given witness is a membership* ($\mathsf{type} = 0$) *or non-membership* ($\mathsf{type} = 1$) *witness. For dynamic accumulator schemes this additionally applies to* WitUpdate*.*

Security Model. Now, we introduce our unified security model. It is adapted from [LLX07] and further extended by undeniability and indistinguishability.

Classic Notion. A secure accumulator scheme is required to be *correct* and *collision-free*. Correctness says that for all honestly generated keys, and for all honestly computed accumulators and witnesses, the Verify algorithm will always return 1. This also needs to hold after additions, deletions, and witness updates. We note that the algorithm Eval returns some auxiliary information aux which is updated by the Add and Delete algorithms (i.e., they return aux'). This information needs to be provided to the algorithms WitCreate, Add, and Delete. Furthermore, the algorithms Add and Delete output some auxiliary witness update information aux_u, which is required to update the witnesses via WitUpdate after Add and Delete. Once again, we stress that correctness also needs to hold when all algorithms are executed without sk_Λ (denoted by $\mathsf{sk}_\Lambda^{\sim}$ in the definitions above) and the output distribution of the algorithms must not be affected by the presence/absence of sk_Λ.

We omit to formally define correctness, and continue with collision freeness. Collision freeness informally states that it is neither feasible to find a witness for a non-accumulated value nor feasible to find a non-membership witness for an accumulated value. More formally:

Definition 3.4 (Collision Freeness)**.** *A cryptographic accumulator of type* $\mathsf{t} \in \{\mathsf{static}, \mathsf{dynamic}\}$ *and* $\mathsf{u} \in \{\mathsf{universal}, \mathsf{non\text{-}universal}\}$ *is* collision-free*, if for all*

PPT adversaries $\mathcal{A}$ there is a negligible function $\varepsilon(\cdot)$ such that:

$$\Pr\left[\begin{array}{ll} (\mathsf{sk}_\Lambda, \mathsf{pk}_\Lambda) \leftarrow \mathsf{Gen}(1^\kappa, t), & (\mathsf{Verify}(\mathsf{pk}_\Lambda, \Lambda^\star, \mathsf{wit}^\star_{x_i}, x_i^\star, 0) = 1 \\ \mathcal{O} \leftarrow \{\mathcal{O}^{\mathsf{t}}, \mathcal{O}^{\mathsf{u}}\}, \ (\mathsf{wit}^\star_{x_i}/\underline{\mathsf{wit}}^\star_{x_i}, & : \quad \wedge\ x_i^\star \notin \mathcal{X}^\star) \ \vee \ (x_i^\star \in \mathcal{X}^\star \ \wedge \\ x_i^\star, \mathcal{X}^\star, r^\star) \leftarrow \mathcal{A}^{\mathcal{O}}(\mathsf{pk}_\Lambda) & \mathsf{Verify}(\mathsf{pk}_\Lambda, \Lambda^\star, \underline{\mathsf{wit}}^\star_{x_i}, x_i^\star, 1) = 1) \end{array}\right] \leq \varepsilon(\kappa),$$

where $\Lambda^\star \leftarrow \mathsf{Eval}_{r^\star}((\mathsf{sk}_\Lambda, \mathsf{pk}_\Lambda), \mathcal{X}^\star)$ and $\mathcal{A}$ has oracle access to $\mathcal{O}^{\mathsf{t}}$ and $\mathcal{O}^{\mathsf{u}}$ which are defined as follows:

$$\mathcal{O}^{\mathsf{t}} := \begin{cases} \{\mathcal{O}^{\mathsf{E}(\cdot,\cdot,\cdot)}\} & \text{if } \mathsf{t} = \mathsf{static}, \\ \{\mathcal{O}^{\mathsf{E}(\cdot,\cdot,\cdot)}, \mathcal{O}^{\mathsf{A}(\cdot,\cdot,\cdot,\cdot)}, \mathcal{O}^{\mathsf{D}(\cdot,\cdot,\cdot,\cdot)}\} & \text{otherwise.} \end{cases}$$

$$\mathcal{O}^{\mathsf{u}} := \begin{cases} \{\mathcal{O}^{\mathsf{W}(\cdot,\cdot,\cdot,\cdot)}, \mathcal{O}^{\underline{\mathsf{W}}(\cdot,\cdot,\cdot,\cdot)}\} & \text{if } \mathsf{u} = \mathsf{universal}, \\ \{\mathcal{O}^{\mathsf{W}(\cdot,\cdot,\cdot,\cdot)}\} & \text{otherwise.} \end{cases}$$

Thereby, $\mathcal{O}^{\mathsf{E}}, \mathcal{O}^{\mathsf{A}}$ and $\mathcal{O}^{\mathsf{D}}$ represent the oracles for the algorithms $\mathsf{Eval}, \mathsf{Add}$, and Delete, respectively. An adversary is allowed to query them an arbitrary number of times. In case of randomized accumulators the adversary outputs randomness $r^\star$, whereas $r^\star$ is omitted for deterministic accumulators. Likewise, the adversary can control the randomness r used by $\mathcal{O}^{\mathsf{E}}$ for randomized accumulators. Therefore, $\mathcal{O}^{\mathsf{E}}$ takes an additional parameter for r (which is missing for deterministic accumulators). The oracles $\mathcal{O}^{\mathsf{W}}$ and $\mathcal{O}^{\underline{\mathsf{W}}}$ allow the adversary to obtain membership witnesses for members and non-membership witnesses for non-members, respectively. Thereby, the environment keeps track of all oracle queries (and answers) and lets the respective oracle return $\perp$ if calls to it are not consistent with respect to previous queries. Furthermore, we assume that the adversary outputs either a membership witness $\mathsf{wit}^\star_{x_i}$ or a non-membership witness $\underline{\mathsf{wit}}^\star_{x_i}$ (denoted by $\mathsf{wit}^\star_{x_i}/\underline{\mathsf{wit}}^\star_{x_i}$). If the accumulator is non-universal, one simply omits the non-membership related parts.

One distinction to previous models is that we explicitly give access to all algorithms via oracles. In doing so we ensure that security proofs take the requirement that everything can be simulated in the absence of sk_Λ into account. This is vital and could be overseen otherwise.

Definition 3.5 (Secure Accumulator)**.** *A cryptographic accumulator is* secure *if it is correct and collision-free.*

Undeniable Accumulators. In [Lip12], Lipmaa formalized undeniability for accumulators. A universal accumulator is *undeniable* if it is computationally infeasible to find a membership as well as a non-membership witness for the same value—independently of whether it is contained in an accumulator or not. More formally undeniability is defined as:

Definition 3.6 (Undeniability)**.** *A universal cryptographic accumulator of type* $\mathsf{t} \in \{\mathsf{static}, \mathsf{dynamic}\}$ *is* undeniable*, if for all PPT adversaries $\mathcal{A}$ there is a*

negligible function $\varepsilon(\cdot)$ such that:

$$\Pr\left[\begin{array}{l|rl}(\mathsf{sk}_\Lambda,\mathsf{pk}_\Lambda)\leftarrow\mathsf{Gen}(1^\kappa,t), & & \mathsf{Verify}(\mathsf{pk}_\Lambda,\Lambda^\star,\mathsf{wit}^\star_{x_i},x_i^\star,0)=\\ (\mathsf{wit}^\star_{x_i},\underline{\mathsf{wit}}^\star_{x_i},x_i^\star,\Lambda^\star)\leftarrow\mathcal{A}^{\mathcal{O}^{\mathsf{t}}}(\mathsf{pk}_\Lambda) & : & \mathsf{Verify}(\mathsf{pk}_\Lambda,\Lambda^\star,\underline{\mathsf{wit}}^\star_{x_i},x_i^\star,1)=\\ & & 1\end{array}\right]\leq\varepsilon(\kappa),$$

where, $\mathcal{A}$ has oracle access to $\mathcal{O}^{\mathsf{t}}$ which is defined as follows:

$$\mathcal{O}^{\mathsf{t}}:=\begin{cases}\{\mathcal{O}^{\mathsf{E}(\cdot,\cdot,\cdot)},\mathcal{O}^{\mathsf{W}(\cdot,\cdot,\cdot,\cdot)},\mathcal{O}^{\underline{\mathsf{W}}(\cdot,\cdot,\cdot,\cdot)}\} & \text{if } \mathsf{t}=\mathsf{static},\\ \{\mathcal{O}^{\mathsf{E}(\cdot,\cdot,\cdot)},\mathcal{O}^{\mathsf{A}(\cdot,\cdot,\cdot,\cdot)},\mathcal{O}^{\mathsf{D}(\cdot,\cdot,\cdot,\cdot)},\mathcal{O}^{\mathsf{W}(\cdot,\cdot,\cdot,\cdot)},\mathcal{O}^{\underline{\mathsf{W}}(\cdot,\cdot,\cdot,\cdot)}\} & \text{otherwise.}\end{cases}$$

Notice that the definition of the oracles is as in the definition of collision freeness for universal accumulators.

Definition 3.7. *A universal accumulator is* undeniable *if it is a secure accumulator satisfying the undeniability property.*

Indistinguishable Accumulators. Li et al. [LLX07] pointed out informally (without giving any formalizations) that the accumulation of an additional random value may render guessing the accumulated set infeasible. Later, de Meer et al. [dMLPP12] tried to formalize this intuition via an additional *indistinguishability* property. Unfortunately, there are some issues with their notion. Firstly, it only covers static accumulators and, secondly, indistinguishability in the vein of [LLX07] weakens collision resistance. Basically, one can easily generate a membership witness for the random value. Secondly, the security game in [dMLPP12] allows to prove indistinguishability of deterministic accumulators, which are clearly not indistinguishable. In particular, the random value is chosen and accumulated within the security game. However, this non-determinism is not required to be part of the accumulator construction itself. Consequently, a deterministic accumulator can satisfy this notion while being trivially distinguishable. From this, we conclude that the non-determinism must be intrinsic to the Eval algorithm.[4]

Definition 3.8 (Indistinguishability)**.** *A cryptographic accumulator of type $\mathsf{t}\in\{\mathsf{static},\mathsf{dynamic}\}$ and $\mathsf{u}\in\{\mathsf{universal},\mathsf{non\text{-}universal}\}$ is* indistinguishable*, if for all PPT adversaries $\mathcal{A}$ there is a negligible function $\varepsilon(\cdot)$ such that:*

$$\Pr\left[\begin{array}{l}(\mathsf{sk}_\Lambda,\mathsf{pk}_\Lambda)\leftarrow\mathsf{Gen}(1^\kappa,t),\ b\xleftarrow{R}\{0,1\},\\ \mathcal{O}\leftarrow\{\mathcal{O}^{\mathsf{t}},\mathcal{O}^{\mathsf{u}}\},\ (\mathcal{X}_0,\mathcal{X}_1,\mathsf{ST})\leftarrow\mathcal{A}(\mathsf{pk}_\Lambda),\\ (\Lambda_{\mathcal{X}_b},\mathsf{aux})\leftarrow\mathsf{Eval}((\mathsf{sk}_\Lambda,\mathsf{pk}_\Lambda),\mathcal{X}_b),\\ b^\star\leftarrow\mathcal{A}^{\mathcal{O}}(\mathsf{pk}_\Lambda,\Lambda_{\mathcal{X}_b},\mathsf{ST})\end{array}\ :\ b=b^\star\right]\leq\frac{1}{2}+\varepsilon(\kappa),$$

[4] Independently of our work, this observation was quite recently also made in [dMPPS14a] by the authors of [dMLPP12]: The insertion of the random value has been removed from the game and the Eval algorithm is now required to be non-deterministic.

where $\mathcal{X}_0$ and $\mathcal{X}_1$ are two distinct subsets of the accumulation domain and $\mathcal{O}^{\mathsf{t}}$ as well as $\mathcal{O}^{\mathsf{u}}$ are defined as follows:

$$\mathcal{O}^{\mathsf{t}} := \begin{cases} \emptyset & \text{if } \mathsf{t} = \mathsf{static}, \\ \{\mathcal{O}^{\mathsf{A}_{\cup}(\cdot,\cdot,\mathsf{aux},\cdot)}, \mathcal{O}^{\mathsf{D}_{\cap}(\cdot,\cdot,\mathsf{aux},\cdot)}\} & \text{otherwise.} \end{cases}$$

$$\mathcal{O}^{\mathsf{u}} := \begin{cases} \{\mathcal{O}^{\mathsf{W}(\cdot,\cdot,\mathsf{aux},\cdot)}, \mathcal{O}^{\underline{\mathsf{W}}(\cdot,\cdot,\mathsf{aux},\cdot)}\} & \text{if } \mathsf{u} = \mathsf{universal}, \\ \{\mathcal{O}^{\mathsf{W}(\cdot,\cdot,\mathsf{aux},\cdot)}\} & \text{otherwise.} \end{cases}$$

If the probability above is exactly 1/2 we have unconditional *indistinguishability, whereas we have* computational *indistinguishability if the probability is negligibly different from 1/2.*

Here, the oracles can only be called for the challenge accumulator. We require that the input parameter aux for the oracles is kept up to date and is consistently provided by the environment, since the knowledge of aux would allow the adversary to trivially win the game. Furthermore, note that this game does not allow the adversary to control the randomness used for the evaluation of $\Lambda_{\mathcal{X}_b}$. For the definitions of the remaining oracles, we use $\mathcal{X}_{\cup} := \mathcal{X}_0 \cup \mathcal{X}_1$ and $\mathcal{X}_{\cap} := \mathcal{X}_0 \cap \mathcal{X}_1$ to restrict the adversary from oracle queries which would trivially allow to win the game. $\mathcal{O}^{\mathsf{A}_{\cup}}$ as well as $\mathcal{O}^{\mathsf{D}_{\cap}}$ allow the adversary to execute the Add and Delete algorithms. Thereby, $\mathcal{O}^{\mathsf{A}_{\cup}}$ only allows queries for values $x_i \notin \mathcal{X}_{\cup}$, whereas $\mathcal{O}^{\mathsf{D}_{\cap}}$ only allows queries for values $x_i \in \mathcal{X}_{\cap}$. Upon every Add and Delete the sets $\mathcal{X}_{\cup}$ and $\mathcal{X}_{\cap}$ are updated consistently. If the accumulator is t-bounded, the Add and Delete oracles additionally enforce that the difference of additions and deletions is always less than or equal $t - \max\{|\mathcal{X}_0|, |\mathcal{X}_1|\}$. Oracles $\mathcal{O}^{\mathsf{W}}$ and $\mathcal{O}^{\underline{\mathsf{W}}}$ are as above, with the difference that $\mathcal{O}^{\mathsf{W}}$ allows only queries for values $x_i \in \mathcal{X}_{\cap}$, while $\mathcal{O}^{\underline{\mathsf{W}}}$ allows only queries for values $y_j \notin \mathcal{X}_{\cup}$.

Remark 3.1. *We remark that indistinguishability is only intended to protect the initially accumulated set, and does not account for dynamic changes of the accumulated set in case of dynamic accumulators.*

As opposed to the approach where indistinguishability is achieved using an inherently randomized Eval algorithm, it was suggested in the literature that indistinguishability can be achieved by choosing random values from the accumulation domain and additionally accumulating them. We formally capture this suggestion by introducing a generic transformation for the latter approach (cf. Transformation 3.1), where we restrict ourselves to static accumulators and note that an extension to dynamic/universal accumulators is straight forward (the transformation only affects the Eval algorithm).

Transformation 3.1. *Let* $(\mathsf{Gen}, \mathsf{Eval}, \mathsf{WitCreate}, \mathsf{Verify})$ *be any static accumulator. The transformation defines a new static accumulator scheme* $(\mathsf{Gen}, \mathsf{Eval}', \mathsf{WitCreate}, \mathsf{Verify})$*, where* Eval' *is defined as follows*

$\mathsf{Eval}'((\mathsf{sk}_{\Lambda}^{\sim}, \mathsf{pk}_{\Lambda}), \mathcal{X})$: *This algorithm samples a uniformly random element x_r from some domain, compatible with the accumulation domain. It computes and returns the result of* $\mathsf{Eval}((\mathsf{sk}_{\Lambda}^{\sim}, \mathsf{pk}_{\Lambda}), \mathcal{X} \cup \{x_r\})$

Usually the domain where x_r is sampled from is the accumulation domain. However, we explicitly allow it to be different so that our transformation is general enough to also cover the RSA accumulators. As already noted above, collision freeness no longer holds for $\mathcal{X}$ but with respect to $\mathcal{X} \cup \{x_r\}$. To draw a line between inherently randomized constructions and such relying on Transformation 3.1, we differentiate between indistinguishability and collision-freeness-weakening (cfw) indistinguishability:

Definition 3.9 (cfw-Indistinguishability)**.** *A cryptographic accumulator is called* cfw*-indistinguishable, if it achieves indistinguishability by applying Transformation 3.1.*

Relation Between Security Properties. Intuitively, undeniability seems to be a strictly stronger security requirement than collision freeness. We confirm this intuition below:

Lemma 3.1. *Every undeniable universal accumulator is collision-free.*

We prove that every undeniable accumulator is also collision-free by showing that an efficient adversary $\mathcal{A}$ against the *collision freeness* of an accumulator scheme can be used to construct an efficient adversary $\mathcal{B}$ against its *undeniability*. Note that $\mathcal{B}$ can simulate all required oracles.

Proof. $\mathcal{B}$ gets input pk_Λ and runs $\mathcal{A}(\mathsf{pk}_\Lambda)$. There are two cases if $\mathcal{A}$ wins the collision freeness game: $\mathcal{A}$ outputs either $(\mathsf{wit}^\star_{x_i}, x_i^\star, \mathcal{X}^\star, r^\star)$ or $(\underline{\mathsf{wit}}^\star_{x_i}, x_i^\star, \mathcal{X}^\star, r^\star)$.

If $\mathcal{A}$ outputs $(\mathsf{wit}^\star_{x_i}, x_i^\star, \mathcal{X}^\star, r^\star)$ such that $x_i^\star \notin \mathcal{X}^\star$, $\mathcal{B}$ evaluates the accumulator with respect to $\mathcal{X}^\star$ and $r^\star$. Hence, it obtains $\Lambda_{\mathcal{X}^\star}$. Next, it computes $\underline{\mathsf{wit}}_{x_i}$ using the witness generation algorithm (which can always be done since $x_i \notin \mathcal{X}$) and outputs $(\mathsf{wit}^\star_{x_i}, \underline{\mathsf{wit}}_{x_i}, x_i^\star, \Lambda_{\mathcal{X}^\star})$.

If $\mathcal{A}$ outputs $(\underline{\mathsf{wit}}^\star_{x_i}, x_i^\star, \mathcal{X}^\star, r^\star)$ such that $x_i^\star \in \mathcal{X}^\star$, $\mathcal{B}$ evaluates the accumulator with respect to $\mathcal{X}^\star$ and $r^\star$. Hence, it obtains $\Lambda_{\mathcal{X}^\star}$. Next, it computes wit_{x_i} using the witness generation algorithm (which can always be done since $x_i^\star \in \mathcal{X}^\star$) and outputs $(\mathsf{wit}_{x_i}, \underline{\mathsf{wit}}^\star_{x_i}, x_i^\star, \Lambda_{\mathcal{X}^\star})$. Hence, whenever $\mathcal{A}$ wins the collision freeness game, $\mathcal{B}$ wins the undeniability game with exactly the same probability

As mentioned in [Lip12], a black-box reduction in the other direction is impossible. [BLL02] provides a collision-free universal accumulator that is not undeniable. Therefore, this proves the following lemma by counterexample:

Lemma 3.2. *Not every collision-free universal accumulator is undeniable.*

3.1.2 State of the Art and Categorization

In this section we first discuss the basic principles of existing constructions grouped by their underlying security assumptions. Subsequently, we provide a compact overview and exhaustive classification of existing approaches.

Strong RSA Setting. All schemes in this setting are extensions of [BdM93, BP97]. Here, the accumulator $\Lambda_{\mathcal{X}}$ is defined to be $\Lambda_{\mathcal{X}} \leftarrow g^{\prod_{x \in \mathcal{X}} x} \bmod N$, where N is an RSA modulus consisting of two large safe primes p, q and g is randomly drawn from the cyclic group of quadratic residues modulo N. Thus, we have $(\mathsf{sk}_\Lambda, \mathsf{pk}_\Lambda) = ((p,q),(g,N))$ and a witness for a value x_i is given by $\mathsf{wit}_{x_i} \leftarrow \Lambda_{\mathcal{X}}^{x_i^{-1}} \bmod N$. Clearly, if we were able to forge a witness $\mathsf{wit}_{x_i} \equiv \Lambda_{\mathcal{X}}^{x_i^{-1}} \pmod N$ for a value x_i not contained in $\Lambda_{\mathcal{X}}$, then we would also be able to break the strong RSA assumption. Due to the multiplicative relationship of the accumulated values in the exponent, the domain of accumulated values is restricted to prime numbers (or products of primes with unknown factorization [TX03]). Note that accumulating a composite number $a = b \cdot c$ would allow to derive witnesses for each of its factors, when given a witness wit_a for a (i.e., $\mathsf{wit}_b \equiv (\mathsf{wit}_a)^c \pmod N$). Hence, to accumulate sets from more general domains, one needs a suitable mapping from these domains to prime numbers (e.g., [STY00]).

Some accumulator schemes in this setting [CL02b, LLX07] also provide dynamic features. Adding a value to the accumulator can be done without any secret by a simple exponentiation of the accumulator and its witnesses. In contrast, if one wants to delete a value x_j, then one has to compute the x_j-th root of the accumulator, which is intractable without sk_Λ under the strong RSA assumption. Yet, one can still use an arithmetic trick to publicly update membership witnesses upon the deletion of a value. To update the witness for a value x_i in $\Lambda_{\mathcal{X} \setminus \{x_j\}}$, one finds $a, b \in \mathbb{Z}$ such that $ax_i + bx_j = 1$ and computes the new witness as $\mathsf{wit}'_{x_i} \leftarrow \mathsf{wit}_{x_i}{}^b \cdot \Lambda_{\mathcal{X} \setminus \{x_j\}}{}^a \bmod N$ from the old witness wit_{x_i} [CL02b].

Furthermore, the accumulator scheme in [LLX07] also provides universal features as it supports non-membership witnesses: Let $\Lambda_{\mathcal{X}}$ be an accumulator to the set $\mathcal{X}$ and let $y_j \notin \mathcal{X}$. Now, it holds that $\gcd(\prod_{x \in \mathcal{X}} x, y_j) = 1$ or, equivalently, $a \prod_{x \in \mathcal{X}} x + by_j = 1$ for $a, b \in \mathbb{Z}$. Consequently, we compute $d \leftarrow g^{-b} \bmod N$, where g is the initial value of the empty accumulator, and form a non-membership witness $\underline{\mathsf{wit}}_{y_j} \leftarrow (a, d)$. The verification of non-membership witnesses is, then, done by checking whether $\Lambda_{\mathcal{X}}{}^a \equiv d^{y_j} \cdot g \pmod N$ holds. In a similar way as it is done for membership witnesses, also non-membership witnesses can be updated publicly (cf. [LLX07]).

t-SDH Setting. All schemes in this settings are based on the t-bounded accumulator proposed by Nguyen [Ngu05], which uses a group $\mathbb{G}$ of prime order p generated by g with a bilinear map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$. Here, we have $\mathsf{pk}_\Lambda = (g, g^s, g^{s^2}, \ldots, g^{s^t}, u)$ and $\mathsf{sk}_\Lambda = s$. An accumulator $\Lambda_{\mathcal{X}}$ to a set $\mathcal{X} = \{x_1, \ldots, x_n\} \in \mathbb{Z}_p^n$ with $n \leq t$ is defined to be $\Lambda_{\mathcal{X}} \leftarrow g^{u \prod_{x \in \mathcal{X}} (x+s)}$ and a membership witness for a value $x_i \in \mathcal{X}$ is computed as $\mathsf{wit}_{x_i} \leftarrow g^{u \prod_{x \in \mathcal{X} \setminus \{x_i\}} (x+s)}$, where $u \xleftarrow{R} \mathbb{Z}_p^*$. Then, one checks whether a value x_i is contained in $\Lambda_{\mathcal{X}}$ by verifying whether $e(\Lambda_{\mathcal{X}}, g) = e(g^{x_i} g^s, \mathsf{wit}_{x_i})$ holds. This scheme allows to evaluate accumulators publicly, that is, by expanding the polynomial $h(X) = \prod_{x \in \mathcal{X}} (x + X) \in \mathbb{Z}_p[X]$ and evaluating it in $\mathbb{G}$ via pk_Λ, which results in $g^{h(s)}$. The public computation of a witness for x_i works likewise with regard to the set $\mathcal{X} \setminus \{x_i\}$. Furthermore, these witnesses can also be updated in constant time and without

knowing the secret key (cf. [Ngu05]).

In [DT08, ATSM09], Nguyen's scheme is extended by non-membership witnesses and the random value u is eliminated. The former work also shows how public updates of non-membership witnesses can be done in constant time. Note that these tweaks can also be applied to the latter. The computation of a non-membership witness for a value $y_j \notin \mathcal{X}$ in [ATSM09] exploits the fact that the polynomial division of $h(X) = \prod_{x \in \mathcal{X}}(x + X)$ by $(y_j + X)$ leaves a remainder $d \in \mathbb{Z}_p^*$. Such a witness has the form $(a, d) = (g^{(h(s)-d)/(y_j+s)}, d)$ and can be verified by checking whether $e(\Lambda_\mathcal{X}, g) = e(a, g^{y_j} g^s) e(g, g^d)$ holds. In [DT08], d is set to $h(-y_j)$, which also yields suitable non-membership witnesses.

t-DHE Setting. In [CKS09], Camenisch et. al give a t-bounded accumulator scheme based on the t-DHE assumption. Like the accumulators in the t-SDH setting, it uses a group $\mathbb{G}$ of prime order p generated by g with a bilinear map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$. In addition, it requires a signature scheme with a corresponding key pair $(\mathsf{sk}_\Sigma, \mathsf{pk}_\Sigma)$. Here, we have $\mathsf{sk}_\Lambda = \mathsf{sk}_{\mathsf{sig}}$ and the public key is $\mathsf{pk}_\Lambda = (g_1, \ldots, g_t, g_{t+2}, \ldots, g_{2t}, z, \mathsf{pk}_\Sigma) = (g^{\gamma^1}, \ldots, g^{\gamma^t}, g^{\gamma^{t+2}}, \ldots, g^{\gamma^{2t}}, e(g,g)^{\gamma^{t+1}}, \mathsf{pk}_\Sigma)$ with $\gamma \xleftarrow{R} \mathbb{Z}_p^*$. We can accumulate a set $\mathcal{X} = \{x_1, \ldots, x_m\}$ with $m \leq t$ by computing $\Lambda_\mathcal{X} \leftarrow \prod_{i=1}^m g_{t+1-i}$ and signing g_i together with x_i using sk_Σ, which assigns the value of x_i to g_i. A witness wit_{x_j} for $x_j \in \mathcal{X}$ is given as $\mathsf{wit}_{x_j} \leftarrow \prod_{i=1, i \neq j}^m g_{t+1-i+j}$. The membership of x_j can be verified by checking whether $e(g_j, \Lambda_\mathcal{X}) = z \cdot e(g, \mathsf{wit}_{x_j})$ holds and by verifying the signature on g_j and x_j under pk_Σ.

This scheme enables public updates of the witnesses and the accumulator upon delete, since this requires only pk_Λ. If we, however, want to add a value x_i to the accumulator, then we need the secret signing key sk_Λ to create a signature on g_i and x_i in order to link value x_i with parameter g_i. Consequently, public additions to the accumulator require to include a signature for every potential value to be accumulated into the public parameters. Obviously, with the exception of very small accumulation domains this seems impractical.

Collision-Resistant Hash Setting. In this setting, accumulators are built from collision-resistant hash functions and (sorted) Merkle hash-trees. Here, each leaf node represents an accumulated value and is labeled with the corresponding hash value. Every inner node is labeled with a hash value formed from its children's labels (and potentially some additional information). The accumulator Λ itself is the root node label (root hash) and a membership witness for an accumulated value x_i is its authentication path, i.e., all the labels of siblings on the path of the leaf to the root node. Verification of witnesses is done the obvious way, i.e., by recomputing the hash tree and comparing the root hashes. To prove non-membership, such schemes exploit the order on the leaf nodes [BLL00, BLL02, CHKO08] (alternatively, an order can also be enforced by a suitable encoding on the accumulated values [CHKO08]). Proving non-membership of a value y_j boils down to demonstrating membership witnesses for two values x_i, x_{i+1} corresponding to two consecutive leaves such that $x_i < y_j < x_{i+1}$ holds.

Accumulators from Vector Commitments. Catalano and Fiore [CF13] proposed a black-box construction of accumulators from vector commitments. A vector commitment allows to form a succinct commitment C to a vector $\mathcal{X} = (x_1, \ldots, x_n)$. Here, it should be computationally infeasible to open position i of C to a value x_i' different from x_i. The accumulation domain in the black-box construction in [CF13] is the set $D = \{1, \ldots, t\}$. The accumulator is modeled as a commitment to a binary vector of length t, that is, each bit i indicates the presence or absence of element $i \in D$ in the accumulator. Then, the (non-)membership of a value i can be proven by opening position i of a commitment to 1 or 0, respectively.

Categorizing Cryptographic Accumulators. Now, we give a comprehensive overview of existing accumulator schemes in Table 3.1. We categorize them regarding their static or dynamic nature and universal features and provide a characterization of their public updating capabilities (of witnesses and of accumulators, respectively). In particular, we tag an accumulator as dynamic, if witness and accumulator value updates can be performed in constant time, i.e., independent of the size of $\mathcal{X}$. If the same is possible without having access to the accumulator trapdoor, then we tag the accumulator as *publicly updatable*. Furthermore, the properties indistinguishability and undeniability have not been considered for most existing accumulator schemes so far. Therefore, we provide a classification regarding their indistinguishability (when using Transformation 3.1) and their undeniability, and provide the respective proofs subsequently. For the sake of completeness, our comparison also includes static accumulator schemes [BdM93, BP97, Nyb96a, Nyb96b].

In the following, we will analyze the undeniability and (in)distinguishability of several constructions. We will not reconsider the collision freeness of these constructions in our model, as it is compatible with previous models.

Undeniability of Universal Accumulators. We will now prove the subsequent lemma:

Lemma 3.3. *The universal accumulators in [DT08, ATSM09] are undeniable.*

Proof. We prove the undeniability of [DT08, ATSM09] by showing that an efficient adversary $\mathcal{A}$ against the undeniability can be turned into an efficient adversary $\mathcal{B}$ against t-SDH assumption. Note that $\mathcal{B}$ can simulate all oracles, since, due to the model, the accumulator and the witnesses can be generated without knowing the trapdoor sk_Λ. $\mathcal{B}$ gets a t-SDH instance $(g, g^s, \ldots, g^{s^t})$ as input, sets $\mathsf{pk}_\Lambda = (g, g^s, \ldots, g^{s^t})$ and starts $\mathcal{A}$ with pk_Λ.

Eventually, $\mathcal{A}$ outputs $(\mathsf{wit}^\star_{x_i}, \underline{\mathsf{wit}}^\star_{x_i}, x_i^\star, \Lambda^\star)$, with $\underline{\mathsf{wit}}^\star_{x_i} = (a, d)$ such that $e(\Lambda^\star, g) = e(g^{x_i^\star} g^s, \mathsf{wit}^\star_{x_i})$ and $e(\Lambda^\star, g) = e(g^{x_i^\star} g^s, a) e(g^d, g)$ holds. Then it holds that $e(g^{x_i^\star} g^s, \mathsf{wit}^\star_{x_i}) = e(g^{x_i^\star} g^s, a) e(g^d, g)$. With, $\mathsf{wit}_{x_i} = g^{\mathcal{X}'}$ and $a = g^{a'}$ for some unknown $\mathcal{X}', a'$, we know that the relation of the exponents is $\mathcal{X}' = a' + {}^{d}\!/_{(x_i^\star + s)}$. Thus, $\mathcal{B}$ can compute $g^{1/(x_i^\star + s)} \leftarrow (\mathsf{wit}^\star_{x_i} \cdot a^{-1})^{d^{-1}}$ and output it as a solution to the t-SDH problem. □

	Scheme	Type D	Type U	Pub. Updates wit	Pub. Updates wit	Pub. Updates acc a	Pub. Updates acc d	ZK	B	$\lvert pk_\Lambda \rvert$	$\lvert wit \rvert$	$\lvert \underline{wit} \rvert$	Und.	Ind.
s-RSA	BDM [BdM93]	-	-	-	-	-	-	-	-	$\mathcal{O}(1)$	$\mathcal{O}(1)$	-	-	cfw‡
s-RSA	BP [BP97]	-	-	-	-	-	-	-	-	$\mathcal{O}(1)$	$\mathcal{O}(1)$	-	-	cfw[dMLPP12]
s-RSA	CL [CL02b]	✓	-	✓	-	✓	-	✓	-	$\mathcal{O}(1)$	$\mathcal{O}(1)$	-	-	cfw‡
s-RSA	LLX [LLX07]	✓	✓	✓	✓	✓	-	✓	-	$\mathcal{O}(1)$	$\mathcal{O}(1)$	$\mathcal{O}(1)$	?	×‡
t-SDH	NY [Ngu05]	✓	-	✓	-	-	-	✓	✓	$\mathcal{O}(t)$	$\mathcal{O}(1)$	-	-	cfw‡
t-SDH	DT [DT08], ATSM [ATSM09]	✓	✓	✓	✓	-	-	✓	✓	$\mathcal{O}(t)$	$\mathcal{O}(1)$	$\mathcal{O}(1)$	✓‡	×‡
t-SDH	This work	✓	-	✓	-	-	-	✓	✓	$\mathcal{O}(1)$	$\mathcal{O}(1)$	-	-	✓‡
t-DHE	CKS [CKS09]	✓	-	✓	-	-	✓	✓	✓	$\mathcal{O}(t)$	$\mathcal{O}(1)$	$\mathcal{O}(1)$	-	×‡
VC	CF (RSA∓, CDH†) [CF13]	✓	✓	✓	✓	✓	✓	-	✓	$\mathcal{O}(t)^{\mp}$, $\mathcal{O}(t^2)^{\dagger}$	$\mathcal{O}(1)$	$\mathcal{O}(1)$	?	×‡
ZKS	This work (Sec. 3.1.4)	-	✓	-	-	-	-	instantiation-dependent					✓‡	✓‡
CRH	BLL [BLL00, BLL02]	-	✓	-	-	-	-	-	-	$\mathcal{O}(1)$	$\mathcal{O}(\log t)$	$\mathcal{O}(\log t)$	✓	×‡
CRH	CHKO [CHKO08]	-	✓	-	-	-	-	-	-	$\mathcal{O}(1)$	$\mathcal{O}(\log t)$	$\mathcal{O}(\log t)$	?	×‡
CRH	BC [BC14]	-	-	-	-	-	-	✓	-	$\mathcal{O}(1)$	$\mathcal{O}(\log t)$	$\mathcal{O}(\log t)$	-	×‡
ROM	NB [Nyb96a, Nyb96b]	-	-	-	-	-	-	-	✓	$\mathcal{O}(1)$	-	-	-	×[dMLPP12]

Table 3.1: Overview of features of existing accumulator schemes. *Legend: D... dynamic, U... universal, Pub. Updates... constant cost for public updates of witnesses and accumulators, a... add, d... delete, ZK... zero-knowledge (non-)membership proofs, B... bounded,* $\lvert pk_\Lambda \rvert$ *... public parameter size,* $\lvert wit \rvert$ *... membership witness size,* $\lvert \underline{wit} \rvert$ *... non-membership witness size, Und... undeniability, Ind...* (cfw) *indistinguishability,* ✓ *... yes,* × *... no, - ... not available, ?... left open,* ‡ *... proven below.*

Indistinguishability of (Dynamic) Accumulators. We investigate the indistinguishability of [BdM93, BP97, CL02b, LLX07, ATSM09, Ngu05, DT08, CF13, BLL00, BLL02, CHKO08]. As we will see, most existing *dynamic* accumulator schemes can be turned into cfw-indistinguishable accumulators under Transformation 3.1, whereas this does not work for existing *universal* accumulators.

Lemma 3.4. *Under Transformation 3.1, the schemes in [BP97], [BdM93] and [CL02b] are* cfw*-indistinguishable.*

Proof. De Meer et al. [dMLPP12] proved a version of [BP97], where Eval randomly chooses a private starting value g, to be indistinguishable in their model.[5] The random choice of g is a way to apply Transformation 3.1. Thus, we can reuse the proof in [dMLPP12] for cfw-indistinguishability.

The schemes in [BP97] and [BdM93] are identical, with the only difference that the accumulation domain in [BP97] is restricted to prime numbers. Since the proof in [dMLPP12] does not require this restriction, indistinguishability also holds for [BdM93]. The difference from [CL02b] to [BdM93, BP97] is that [CL02b] supports dynamic updates. We, thus, need to show that access to $\mathcal{O}^{\mathsf{A}}$ and $\mathcal{O}^{\mathsf{D}}$ does not give the adversary any advantage. The proof in [dMLPP12] shows that the value of the challenge accumulator is random, when x is randomly sampled from the accumulation domain, i.e., it cannot be distinguished from a randomly sampled starting value of an empty accumulator. Taking this, together with the restriction that only elements $a \notin \mathcal{X}_{\cup}$ can be added and only elements $a \in \mathcal{X}_{\cap}$ can be deleted, the adversary does not learn more than when dealing with an empty accumulator. Consequently, it can not win the indistinguishability game with a probability being non-negligibly greater than $1/2$. The same argumentation also holds for [LLX07] under the assumption that the *universal* features, i.e., the non-membership witnesses/proofs, are not being used. □

To achieve indistinguishability one could use the random oracle heuristic and obtain the starting value $g \leftarrow H(r)$ for some uniformly random value r. Observe that this would allow to exclude the case where the adversary chooses the starting value g as g'^{v} so that it easy to come up with a witness for every prime factor of v, even though this value is seemingly not included in the accumulator.

Lemma 3.5. *Under Transformation 3.1, the scheme in [Ngu05] is* cfw*-indistinguishable.*

When applying Transformation 3.1, the accumulator in [Ngu05] is computed as $\Lambda_{\mathcal{X}} \leftarrow g^{u(x+s)\prod_{i=1}^{n}(x_i+s)}$, where $\mathcal{X} = \{x_1, \ldots, x_n\}$, u is a random value in $\mathbb{Z}_p^*$ (chosen during Gen) and x is a random value from the accumulation domain (freshly chosen upon each call to Eval). As discussed in Section 3.1.3, slight changes to the accumulator scheme in [Ngu05] make it indistinguishable. In the following, we show the cfw-indistinguishability under Transformation 3.1.

[5] We note that in their security game an additional random value x is inserted into the accumulator. This value is, however, not required for their indistinguishability proof.

Proof. It is easy to see that for a given accumulator $\Lambda_{\mathcal{X}}$ to set $\mathcal{X} = \{x_1, \ldots, x_n\}$, that is, $\Lambda_{\mathcal{X}} = g^{u(x+s)\prod_{i=1}^{n}(x_i+s)}$ (after applying Transformation 3.1), another value x' exists such that for an arbitrarily chosen set $W = \{w_i, \ldots w_m\}$ it holds that $\Lambda_{\mathcal{X}} = g^{u(x'+s)\prod_{i=1}^{m}(w_i+s)}$. More precisely, this means that the accumulated set is unconditionally hidden within $\Lambda_{\mathcal{X}}$. Moreover, since the witnesses have the same form as the accumulator, they also do not reveal anything beyond the fact that an element has indeed been accumulated. □

Note that it is crucial that the random elements, which are inserted into the accumulator, are not reused to reach this unconditional indistinguishability. Also note that the schemes in [LLX07, ATSM09, DT08], which extend [CL02b] and [Ngu05] by universal features, are cfw-indistinguishable when solely used with membership witnesses. However, as we will see later, non-membership witnesses leak information about the accumulated values.

Lemma 3.6. *Under Transformation 1, the scheme in [CKS09] is distinguishable.*

Proof. Here the accumulator Λ is a product of generators contained in pk_Λ. Thus the number of generators is polynomially bounded in the security parameter. This means that one can choose $\mathcal{X}_0$ and $\mathcal{X}_1$ so that it is possible to efficiently brute force the generators contained in $\Lambda_{\mathcal{X}_b}$. □

Lemma 3.7. *Under Transformation 3.1, the scheme in [BC14] is distinguishable.*

Here, the accumulator is the root of a perfect Merkle hash-tree, where a statistically hiding commitment is used as hash function. This implies that the accumulated values are statistically hidden in the accumulator. A witness for an accumulated value x is the opening of the corresponding commitment together with its authentication path (the authentication path only contains commitments).

Proof. Transformation 3.1 inserts a random value x from the accumulation domain into the accumulator. To win the game, the adversary chooses two arbitrary sets $\mathcal{X}_0, \mathcal{X}_1$ such that $1 \leq |\mathcal{X}_0| + 1 \leq 2^t$ and $2^t < |\mathcal{X}_1| + 1$ for arbitrary integers $t > 0$. According to the game, the adversary can now obtain a membership witness for an element $x' \in \mathcal{X}_0 \cup \mathcal{X}_1$ and decide which set was accumulated by using the length l of the authentication path of the witness, i.e., if $l > t$ the set $\mathcal{X}_1$ was accumulated and $\mathcal{X}_0$ otherwise. □

Since the used commitments are statistically hiding, indistinguishability holds for sets $\mathcal{X}_0, \mathcal{X}_1$ of size $2^t \leq |\mathcal{X}_0|, |\mathcal{X}_1| < 2^{t+1}$ with $t \in \mathbb{N}$, where the hash-tree is filled up with dummy elements if the set size is not a power of two. Thus, for sets of size $|\mathcal{X}_0| = |\mathcal{X}_1| = 2^t$ we have indistinguishability, whereas cfw-indistinguishability holds for arbitrary sets meeting the aforementioned constraints.

Indistinguishability of Universal Dynamic Accumulators. Below, we prove the following lemmas by providing adversaries that succeed with non-negligible probability better than 1/2.

Lemma 3.8. *Under Transformation 3.1, the universal accumulator from [LLX07] is distinguishable.*

Proof. Here, a non-membership witness for a value y_j not contained in $\Lambda_{\mathcal{X}}$ with $\mathcal{X} = \{x_1, \ldots, x_n\}$ is a tuple (a, d), where $a \prod_{i=1}^{n} x_i + by_j = 1$ and $d \equiv g^{-b} \pmod{N}$. From this, an adversary can compute $a^{-1} \bmod y_j$, i.e., $a^{-1} \equiv \prod_{i=1}^{n} x_i \pmod{y_j}$. When $y_j > \prod_{i=1}^{n} x_i$ holds, then obviously also $a^{-1} \bmod y_j = \prod_{i=1}^{n} x_i$.

Transformation 3.1 inserts a random value x from the accumulation domain into the accumulator. Suppose that the adversary chooses two arbitrary sets $\mathcal{X}_0 = \{x_1, \ldots, x_n\}$ and $\mathcal{X}_1 = \{w_1, \ldots, w_m\}$ with $\mathcal{X}_0 \neq \mathcal{X}_1$ and generates k distinct non-membership witnesses $(y_{j_i})_{i=1}^{k} = ((a_{j_i}, d_{j_i}))_{i=1}^{k}$. The adversary can then compute $a_{j_i}^{-1} \bmod y_{j_i}$ for $i = 1, \ldots, k$ such that, depending on whether $\mathcal{X}_0$ or $\mathcal{X}_1$ was accumulated, either $x \prod_{l=1}^{n} x_l \equiv a_{j_i}^{-1} \pmod{y_{j_i}}$ or $x \prod_{l=1}^{m} w_l \equiv a_{j_i}^{-1} \pmod{y_{j_i}}$ holds for $i = 1, \ldots, k$. In the attack, the adversary can compute $a_{\mathsf{CRT}}^{-1} \pmod{\prod_{i=1}^{k} y_{j_i}}$ using $(a_{j_i}^{-1} \bmod y_{j_i})_{i=1}^{k}$ and the Chinese remainder theorem. Now, if $\prod_{i=1}^{k} y_{j_i} > x \prod_{i=1}^{n} x_i$ and $\prod_{i=1}^{k} y_{j_i} > x \prod_{i=1}^{m} w_i$, it either holds that $a_{\mathsf{CRT}}^{-1} = x \prod_{i=1}^{n} x_i$ or $a_{\mathsf{CRT}}^{-1} = x \prod_{i=1}^{m} w_i$. These conditions can always be ensured, since the adversary can generate an arbitrary number of non-membership witnesses y_{j_i}. Next, the adversary can divide a_{CRT}^{-1} by an element being exclusively contained in $\mathcal{X}_0$. If this division leaves a remainder, $\mathcal{X}_1$ was accumulated, and $\mathcal{X}_0$ otherwise. This attack succeeds with overwhelming probability, since only primes are accumulated and the probability that the randomly chosen x is also exclusively contained in one of the sets $\mathcal{X}_0$ or $\mathcal{X}_1$ is negligible. □

Lemma 3.9. *Under Transformation 3.1, the universal accumulator from [DT08] is distinguishable.*

Proof. Again, we show that the proposed non-membership witnesses leak information about the accumulated set. Here, non-membership witnesses are of the form (a, d) with $a = g^{(h(s)-d)/(y_j+s)} = g^{(\prod_{i=1}^{n}(x_i+s)-d)/(y_j+s)}$ and $d = h(-y_j) = \prod_{i=1}^{n}(x_i - y_j)$.

Transformation 3.1 inserts a random value x from the accumulation domain into the accumulator. Suppose w.l.o.g. that the adversary chooses $\mathcal{X}_0 = \{x_1, \ldots, x_n\}$ and $\mathcal{X}_1 = \{w_1, \ldots, w_m\}$ for $m > n$ and obtains $\Lambda_{\mathcal{X}_b} \leftarrow g^{h(s)}$, where $h(s)$ is either equal to $(x + s) \prod_{i=1}^{n}(x_i + s)$ or equal to $(x + s) \prod_{i=1}^{m}(w_i + s)$. The adversary is allowed to generate $n + 2$ distinct non-membership witnesses $((a_i, d_i))_{i=1}^{n+2}$ corresponding to the non-members $(y_i)_{i=1}^{n+2}$. Now, since $d_i = h(-y_i)$, one can simply recover a suspected polynomial of degree $n+1$ by its evaluations $(d_i)_{i=1}^{n+2}$ at $(-y_i)_{i=1}^{n+2}$ using polynomial interpolation. This interpolation yields the polynomial corresponding to the accumulated set in case $\mathcal{X}_0$ was accumulated, and some arbitrary polynomial with the evaluations $(d_i)_{i=1}^{n+2}$ at $(-y_i)_{i=1}^{n+2}$

in case $\mathcal{X}_1$ was accumulated. Given this polynomial and $(g, g^s, \ldots, g^{s^t})$, one can reevaluate the accumulator, i.e., $\prod_{i=0}^{n+1}(g^{s^i})^{a_i}$, where a_i is the i-th coefficient of the expanded polynomial. If $\Lambda_{\mathcal{X}_b} = \prod_{i=0}^{n+1}(g^{s^i})^{a_i}$ holds, then we know with overwhelming probability that $\Lambda_{\mathcal{X}_b}$ accumulates $\mathcal{X}_0$ and $\mathcal{X}_1$ otherwise. Note that this attack succeeds with overwhelming probability, since the probability that the wrong polynomial has the same evaluation at a random, unknown s is negligible. □

We further note that, due to the structure of the values d in the non-membership witnesses, also other attacks could apply.

Lemma 3.10. *Under Transformation 3.1, the universal accumulator presented in [ATSM09] is distinguishable.*

Proof. We show that the proposed non-membership witnesses leak information about the accumulated set. Here, non-membership witnesses are of the form (a, d) such that $a = g^{(\prod_{i=1}^{n}(x_i+s)-d)/(y_j+s)}$ and d is the remainder of the polynomial division $(\prod_{i=1}^{n}(x_i+s))/(y_j+s)$.

Transformation 3.1 inserts a random value x from the accumulation domain into the accumulator. Similar to the attack against [LLX07], we can w.l.o.g. assume that the adversary chooses an arbitrary set $\mathcal{X}$ and two arbitrary distinct elements $x_1, x_2 \in \mathbb{Z}_p$ such that $x_1, x_2 \notin \mathcal{X}$. Then, we have $\mathcal{X}_0 = \mathcal{X} \cup \{x_1\}$ and $\mathcal{X}_1 = \mathcal{X} \cup \{x_2\}$, respectively. According to the game, all values of $\mathcal{X}$ can be deleted. Then, the value of the challenge accumulator is $g^{(x+s)(x_i+s)}$ for an $i \in \{1, 2\}$. Furthermore, the polynomial division $((x+s)(x_i+s))/(y_j+s)$ yields a remainder of the form $xx_i - (x+x_i-y_j)y_j = d \mod p$. Now, the adversary obtains a non-membership witness for some value y_j, yielding the remainder d_j. Given that, the adversary can compute two candidate values x', x'' for x from the equation above – simply by trying both x_1 and x_2. Thus, the adversary knows two potentially accumulated polynomials $(x'+s)(x_1+s)$ and $(x''+s)(x_2+s)$, which can be used to reevaluate the accumulator with respect to these two polynomials in the same way as in the proof for Lemma 3.9. Thus, the adversary can decide which set has been accumulated by comparing the resulting accumulators to the challenge accumulator. This attack succeeds with overwhelming probability, since the probability that the wrong polynomial has the same evaluation at s is negligible for a random, unknown s. □

We note that in [ATSM09] a second method for generating non-membership witnesses (where the knowledge of s is required) is proposed. Here, d is reduced modulo $y_j + s$, which, in turn, means that the attack above only succeeds when $d \bmod p = d \bmod y_j + s$. However, since d leaks information about the set, also other, more efficient attacks might be possible.

Lemma 3.11. *Under Transformation 3.1, the universal accumulators from vector commitments [CF13] are distinguishable.*

Proof. We show that the proposed accumulator schemes from vector commitments in the RSA and CDH setting are distinguishable by presenting an adversary that wins the indistinguishability game with a probability of 1.

In accumulators from vector commitments, the accumulation domain is the set $D = \{1, \ldots, t\}$. In both, the RSA and the CDH instantiation, the accumulator (vector commitment) for a set $\mathcal{X} \subseteq D$ is computed as $\Lambda_{\mathcal{X}} \leftarrow \prod_{i=1}^{t} g_i^{\chi_{\mathcal{X}}(i)}$, where $\chi_{\mathcal{X}}(\cdot)$ is the characteristic function. The values g_i are contained in the public parameters.

The adversary can choose two arbitrary sets $\mathcal{X}_0, \mathcal{X}_1$, such that $|\mathcal{X}_0 \cap \mathcal{X}_1| \leq |\mathcal{X}_0| - 2$ and $|\mathcal{X}_0 \cap \mathcal{X}_1| \leq |\mathcal{X}_1| - 2$, i.e., $\mathcal{X}_0$ and $\mathcal{X}_1$ are different in at least two elements. Transformation 3.1 inserts an additional random value from the accumulation domain into the accumulator. Then, the value of $\Lambda_{\mathcal{X}_b}$ is either equal to $g_r \prod_{i \in \mathcal{X}_0} g_i$ with $r \notin \mathcal{X}_0$, or equal to $g_{r'} \prod_{i \in \mathcal{X}_1} g_i$ with $r' \notin \mathcal{X}_1$. Note that due to the choice of the sets $\mathcal{X}_0$ and $\mathcal{X}_1$ we can guarantee that adding a single element cannot make the sets collide. Consequently, the adversary can w.l.o.g. try if an element from the set $D \setminus \mathcal{X}_0$ was used as randomizer for evaluating $\Lambda_{\mathcal{X}_b}$ with respect to $\mathcal{X}_0$. If such an element is found, $\mathcal{X}_0$ was accumulated and $\mathcal{X}_1$ otherwise. Note that the size of the public parameters (and thus the size of D) is polynomially bounded. Thus, this brute-force strategy can be realized efficiently. □

We note that the same argumentation can be applied if the accumulation domain is an arbitrary set of size n (as proposed in Remark 16 in [CF13]).

Lemma 3.12. *Under Transformation 3.1, the universal accumulators presented in [BLL00], [BLL02] and [CHKO08] are distinguishable.*

Proof. We show that the accumulators from collision-resistant hashing are distinguishable by presenting an adversary that wins the indistinguishability game with probability 1. Here, one can use the fact that the sets are sorted and that non-membership witnesses contain two elements of the respective sets. Again, Transformation 3.1 inserts an additional random element into the set upon Eval.

To break indistinguishability, the adversary chooses two disjoint sets $\mathcal{X}_0, \mathcal{X}_1$. It is allowed to obtain a non-membership witness for a value $y_i \notin \mathcal{X}_0 \cup \mathcal{X}_1$, which contains two consecutive values x_1, x_2 out of the accumulated set such that $x_1 < y_i < x_2$ holds.

Then we distinguish three cases. In the first case x_1 and x_2 are contained in $\mathcal{X}_i$ for an $i \in \{0, 1\}$, meaning that the adversary knows that $\mathcal{X}_i$ has been accumulated (neither x_1 nor x_2 is the random element). In the second case, we can w.l.o.g. assume that $x_2 \notin \mathcal{X}_0 \cup \mathcal{X}_1$ (is the random element which is contained in none of the initially chosen sets). This means that $x_1 \in \mathcal{X}_i$ for an $i \in \{0, 1\}$ and the adversary knows that $\mathcal{X}_i$ was accumulated. In the remaining case, we can w.l.o.g. assume that $x_1 \in \mathcal{X}_0$ and $x_2 \in \mathcal{X}_1$, and, thus, either x_1 is the random element added to $\mathcal{X}_1$ upon Eval or vice versa. Then, the adversary can request another non-membership witness for $y_i' < x_1$ and $y_i' \notin \mathcal{X}_0 \cup \mathcal{X}_1$ and obtains x_3 and x_4 such that $x_3 < y_i' < x_4$. It could be the case that $x_1 = x_4$, but still x_3

is either an element of $\mathcal{X}_0$ or $\mathcal{X}_1$ and, thus, the adversary can decide which set was accumulated.

Finally, we note that the adversary can always choose y_i and y_i' such that obtaining witnesses fulfilling the requirements above is possible, since $\mathcal{X}_0$ and $\mathcal{X}_1$ are also chosen by the adversary. □

3.1.3 Commitments from Indistinguishable Accumulators

In [CF13], it has been shown that universal dynamic accumulators can be black-box constructed from vector commitments. The question arises whether it is also possible to provide black-box constructions for certain types of commitments from indistinguishable accumulators. It is apparent that it is not possible to build vector commitments solely from accumulators in a black-box fashion, since their position binding would at least require some additional encoding. Nevertheless, we will show how to construct non-interactive commitments from indistinguishable 1-bounded accumulators. We start by proposing the first indistinguishable t-bounded accumulator construction (for arbitrary t) and, then, provide the black-box construction based on any such scheme.

An Indistinguishable t-Bounded Dynamic Accumulator. Here, we will build an indistinguishable t-bounded accumulator from the t-SDH based dynamic accumulator in [Ngu05]. This construction already uses a randomizer (denoted as u) that is chosen by Gen and added to pk_Λ. As a consequence, the Eval algorithm is still deterministic. In order to obtain indistinguishability, we modify the way u is used and observe that randomly choosing it on each call to Eval yields an indistinguishable accumulator. Scheme 3.1 states our construction. Subsequently, we prove it to be a secure indistinguishable t-bounded dynamic accumulator. For sake of consistency, the value u used in [Ngu05] will be denoted by r.

Theorem 3.1. *Under the t-SDH assumption, Scheme 3.1 is an indistinguishable t-bounded dynamic accumulator.*

Lemma 3.13. *Scheme 3.1 is correct.*

Correctness is easy to verify by inspection; the proof is omitted.

Lemma 3.14. *If the t-SDH assumption holds, then Scheme 3.1 is collision free.*

Proof. Similar to [ATSM09], we prove the collision freeness for membership witnesses by showing that an efficient adversary $\mathcal{A}$ against the collision freeness can be turned into an efficient adversary $\mathcal{B}$ against the t-SDH assumption. $\mathcal{B}$ gets a t-SDH instance $(g, g^s, \ldots, g^{s^t})$ as input and starts the game by setting $\mathsf{pk}_\Lambda = (g, g^s, \ldots, g^{s^t})$ and handing pk_Λ over to $\mathcal{A}$. Moreover, note that $\mathcal{B}$ can simulate all oracles by simply running the respective algorithms without sk_Λ.

$\underline{\mathsf{Gen}(1^\kappa, t)}$: Run $\mathsf{BG} \leftarrow \mathsf{BGGen}(1^\kappa, 1)$ and $s \xleftarrow{R} \mathbb{Z}_p^*$, and return $(\mathsf{sk}_\Lambda, \mathsf{pk}_\Lambda) \leftarrow (s, (\mathsf{BG}, (g^{s^i})_{i=0}^t))$.

$\underline{\mathsf{Eval}_r((\mathsf{sk}_\Lambda^\sim, \mathsf{pk}_\Lambda), \mathcal{X})}$: Parse $\mathcal{X}$ as $\{x_1, \ldots, x_n\}$ and choose $r \xleftarrow{R} \mathbb{Z}_p^*$. Return $\Lambda_\mathcal{X}$ and $\mathsf{aux} \leftarrow (r, \mathcal{X})$, where

$$\Lambda_\mathcal{X} \leftarrow \begin{cases} g^{r\prod_{i=1}^n (x_i+s)} & \text{if } \mathsf{sk}_\Lambda \neq \emptyset, \text{ and} \\ (\prod_{i=0}^n (g^{s^i})^{a_i})^r & \text{otherwise, where } (a_i)_{0\le i\le n} \leftarrow Exp(\mathcal{X}). \end{cases}$$

$\underline{\mathsf{WitCreate}((\mathsf{sk}_\Lambda^\sim, \mathsf{pk}_\Lambda), \Lambda_\mathcal{X}, \mathsf{aux}, x_i)}$: Parse aux as $(r, \mathcal{X})$. Return $\perp$ if $x_i \notin \mathcal{X}$. Otherwise return wit_{x_i}, where

$$\mathsf{wit}_{x_i} \leftarrow \begin{cases} \Lambda_\mathcal{X}^{(x_i+s)^{-1}} & \text{if } \mathsf{sk}_\Lambda \neq \emptyset, \text{ and} \\ \left(\prod_{i=0}^{n-1} (g^{s^i})^{a_i}\right)^r & \text{otherwise, where } (a_i)_{0\le i\le n-1} \leftarrow Exp(\mathcal{X} \setminus \{x_i\}). \end{cases}$$

$\underline{\mathsf{Verify}(\mathsf{pk}_\Lambda, \Lambda_\mathcal{X}, \mathsf{wit}_{x_i}, x_i)}$: Return 1 if the following holds, and 0 otherwise:

$$e(\Lambda_\mathcal{X}, g) = e(\mathsf{wit}_{x_i}, g^{x_i} g^s).$$

$\underline{\mathsf{Add}((\mathsf{sk}_\Lambda^\sim, \mathsf{pk}_\Lambda), \Lambda_\mathcal{X}, \mathsf{aux}, x_i)}$: Parse aux as $(r, \mathcal{X})$ and return $\perp$ if $x_i \in \mathcal{X}$. Otherwise, return Λ' and $\mathsf{aux}' \leftarrow (r, \mathcal{X} \cup \{x_i\})$, and $\mathsf{aux}_\mathsf{u} \leftarrow (\Lambda', \Lambda_\mathcal{X}, x_i, \mathtt{add})$, where

$$\Lambda' \leftarrow \begin{cases} \Lambda_\mathcal{X}^{(x_i+s)} & \text{if } \mathsf{sk}_\Lambda \neq \emptyset, \text{ and} \\ (\prod_{i=0}^{n+1} (g^{s^i})^{a_i})^r & \text{otherwise, where } (a_i)_{0\le i\le n+1} \leftarrow Exp(\mathcal{X} \cup \{x_i\}). \end{cases}$$

$\underline{\mathsf{Delete}((\mathsf{sk}_\Lambda^\sim, \mathsf{pk}_\Lambda), \Lambda_\mathcal{X}, \mathsf{aux}, x_i)}$: Parse aux as $(r, \mathcal{X})$ and return $\perp$ if $x_i \notin \mathcal{X}$. Otherwise, return Λ', $\mathsf{aux}' \leftarrow (r, \mathcal{X} \setminus \{x_i\})$, and $\mathsf{aux}_\mathsf{u} \leftarrow (\Lambda', \Lambda_\mathcal{X}, x_i, \mathtt{delete})$, where

$$\Lambda' \leftarrow \begin{cases} \Lambda_\mathcal{X}^{(x_i+s)^{-1}} & \text{if } \mathsf{sk}_\Lambda \neq \emptyset, \text{ and} \\ (\prod_{i=0}^{n-1} (g^{s^i})^{a_i})^r & \text{otherwise, where } (a_i)_{0\le i\le n-1} \leftarrow Exp(\mathcal{X} \setminus \{x_i\}). \end{cases}$$

$\underline{\mathsf{WitUpdate}(\mathsf{pk}_\Lambda, \mathsf{wit}_{x_i}, \mathsf{aux}_\mathsf{u}, x_j)}$: Parse aux_u as $(\Lambda', \Lambda, x_i, \mathsf{op})$, where Λ' represents the already updated accumulator, Λ the accumulator before the update. Information op determines whether x_j was added or deleted to/from the accumulator Λ'. It returns the updated witness wit'_{x_i}, where

$$\mathsf{wit}'_{x_i} \leftarrow \begin{cases} \Lambda \cdot \mathsf{wit}_{x_i}^{x_j - x_i} & \text{if } \mathsf{op} = \mathtt{add}, \text{ and} \\ (\Lambda'^{-1} \cdot \mathsf{wit}_{x_i})^{1/(x_j - x_i)} & \text{if } \mathsf{op} = \mathtt{delete}. \end{cases}$$

$\underline{Exp(\mathcal{X})}$: This helper function expands the polynomial $\prod_{x\in\mathcal{X}}(x + X)$ to $\sum_{i=0}^{|\mathcal{X}|} a_i \cdot X^i$ and returns $(a_i)_{0\le i\le|\mathcal{X}|}$.

Scheme 3.1: Indistinguishable t-bounded dynamic accumulator scheme.

Eventually, $\mathcal{A}$ outputs a membership witness $\mathsf{wit}^\star_{x_j}$ for some value $x_j^\star \notin \mathcal{X}^\star$, a set $\mathcal{X}^\star$ and a randomizer $r^\star$ such that the verification relation $e(\mathsf{wit}^\star_{x_j}, g^{x_j^\star} g^s) = e(\Lambda_{\mathcal{X}^\star}, g)$ holds, where $(\Lambda_{\mathcal{X}^\star}, \mathsf{aux}) \leftarrow \mathsf{Eval}_r((\emptyset, \mathsf{pk}_\Lambda), \mathcal{X}^\star)$. Then, $\mathcal{B}$ knows the polynomial $h(X) = \prod_{x\in\mathcal{X}^\star}(x + X)$, the polynomial $\phi(X)$ and d such that $h(X) = \phi(X)(x_j + X) + d$ holds (because $x_j^\star \notin \mathcal{X}^\star$). Then, $\mathcal{B}$ can compute

$g^{r^\star \cdot \phi(s)}$ by expanding the polynomial $\phi(X)$ to $\sum_{i=0}^{|\mathcal{X}^\star|-1} a_i \cdot X^i$ and computing $g^{r^\star \cdot \phi(s)} \leftarrow (\prod_{i=0}^{|\mathcal{X}^\star|-1} (g^{s^i})^{a_i})^{r^\star}$. Thus, $\mathcal{B}$ can output

$$\left(\mathsf{wit}^\star_{x_j} \cdot \left(g^{r^\star \cdot \phi(s)}\right)^{-1}\right)^{\frac{1}{r^\star \cdot d}} \leftarrow \left(g^{\frac{r^\star \cdot h(s)}{x_j^\star + s}} g^{-\frac{r^\star \cdot (h(s)-d)}{x_j^\star + s}}\right)^{\frac{1}{r^\star \cdot d}} = \left(g^{\frac{r^\star \cdot d}{x_j^\star + s}}\right)^{\frac{1}{r^\star \cdot d}} = g^{\frac{1}{x_j^\star + s}}$$

together with $x_j^\star$ as solution to the t-SDH problem. □

Lemma 3.15. *Scheme 3.1 is indistinguishable.*

Proof. Recall that we have $\mathcal{X}_0 \neq \mathcal{X}_1$ and only values from $\mathcal{X}_0 \cap \mathcal{X}_1$ can be deleted via $\mathcal{O}^{\mathsf{D}}$. The adversary sees $\Lambda_{\mathcal{X}_b} = g^{r \prod_{x \in \mathcal{X}_b}(x+s)}$, and for values $x_j \in \mathcal{X}_0 \cap \mathcal{X}_1$ it is possible to obtain witnesses $\mathsf{wit}_{x_j} = g^{r \prod_{x \in \mathcal{X}_b \setminus \{x_j\}}(x+s)}$. Furthermore, it is possible to add values $x \notin \mathcal{X}_0 \cup \mathcal{X}_1$.

Now, we observe that there exist randomizers r_0 and r_1 for both candidate sets $\mathcal{X}_0$ and $\mathcal{X}_1$, respectively, such that $\Lambda_{\mathcal{X}_b} = g^{r_0 \prod_{x \in \mathcal{X}_0}(x+s)} = g^{r_1 \prod_{x \in \mathcal{X}_1}(x+s)}$ (even if one of the sets is empty). Note that this even holds in presence of the deletion oracle as it can only be called for values in $\mathcal{X}_0 \cap \mathcal{X}_1$. Likewise, for all witnesses wit_{x_j}, where $x_j \in \mathcal{X}_0 \cap \mathcal{X}_1$, there are representations consistent with the values r_0 and r_1 such that $\mathsf{wit}_{x_j} = g^{r_0 \prod_{x \in \mathcal{X}_0 \setminus \{x_j\}}(x+s)} = g^{r_1 \prod_{x \in \mathcal{X}_1 \setminus \{x_j\}}(x+s)}$, which means that witnesses do not give an additional advantage.

Summing up, even an unbounded adversary can not decide whether $r = r_0$ or $r = r_1$, meaning that it is not possible to distinguish whether $\mathcal{X}_0$ or $\mathcal{X}_1$ was accumulated. □

Black-Box Construction of Non-Interactive Commitments. We present a black-box construction of commitments from indistinguishable accumulators Scheme 3.2 and prove the so obtained construction secure (Theorem 3.2). Before we continue, we want to recall that in the trusted setup model all algorithms can be correctly executed without sk_Λ.

$\underline{\mathsf{PGen}(1^\kappa)}$: Run $(\mathsf{sk}_\Lambda^\sim, \mathsf{pk}_\Lambda) \leftarrow \mathsf{Acc.Gen}(1^\kappa, 1)$, discard sk_Λ and return $\mathsf{PP} \leftarrow \mathsf{pk}_\Lambda$.

$\underline{\mathsf{Commit}(\mathsf{PP}, m)}$: Choose randomness r, run $(C, \mathsf{aux}) \leftarrow \mathsf{Eval}_r((\emptyset, \mathsf{pk}_\Lambda),\ m)$, set $O \leftarrow (r, m, \mathsf{aux})$ and return (C, O).

$\underline{\mathsf{Open}(\mathsf{PP}, C, O)}$: Compute $\mathsf{wit}_m \leftarrow \mathsf{WitCreate}((\emptyset, \mathsf{pk}_\Lambda), C, \mathsf{aux}, m)$ and check whether $\mathsf{Eval}_r((\emptyset, \mathsf{pk}_\Lambda), m) = (C, \mathsf{aux}) \ \wedge\ \mathsf{Verify}(\mathsf{pk}_\Lambda, C, \mathsf{wit}_m, m) = 1$ and return m on success and $\perp$ otherwise.

Scheme 3.2: Commitment scheme from indistinguishable accumulators.

Theorem 3.2. *If indistinguishable* 1*-bounded accumulators exist, then non-interactive commitments exist as well.*

Proof. We show that an efficient adversary $\mathcal{A}$ against the binding (hiding) property of Scheme 3.2 can be turned into an adversary $\mathcal{B}$ against the collision freeness (indistinguishability) of the indistinguishable 1-bounded accumulator. To do so, we construct a reduction $\mathcal{B}$, which interacts with a challenger $\mathcal{C}$ from the respective accumulator game (and internally simulates the challenger $\mathcal{C}'$ of the respective commitment game for adversary $\mathcal{A}$).

Binding: Let us assume that there exists an efficient adversary $\mathcal{A}$ against the binding property of Scheme 3.2. $\mathcal{C}$ runs $\mathsf{pk}_\Lambda \leftarrow \mathsf{Acc.Gen}(1^\kappa, 1)$ and starts $\mathcal{B}$ on pk_Λ. That is, $\mathcal{C}'$ sets $\mathsf{PP} \leftarrow \mathsf{pk}_\Lambda$ and starts $\mathcal{A}$ on input PP. Eventually, $\mathcal{A}$ outputs $C^\star, O^\star = (r, m, \mathsf{aux}), O'^\star = (r', m', \mathsf{aux}')$ such that $m \leftarrow \mathsf{Open}(\mathsf{PP}, C^\star, O^\star)$ and $m' \leftarrow \mathsf{Open}(\mathsf{PP}, C^\star, O'^\star)$ and $m \neq m' \wedge m \neq \bot \wedge m' \neq \bot$. Note that this means that $\mathsf{Eval}_r((\emptyset, \mathsf{pk}_\Lambda), \{m\}) = (C^\star, \mathsf{aux})$, $\mathsf{Eval}_{r'}((\emptyset, \mathsf{pk}_\Lambda), \{m'\}) = (C^\star, \mathsf{aux}')$, $\mathsf{Verify}(\mathsf{pk}_\Lambda, C^\star, \mathsf{WitCreate}((\emptyset, \mathsf{pk}_\Lambda), C^\star, \mathsf{aux}, m), m) = 1$ and $\mathsf{Verify}(\mathsf{pk}_\Lambda, C^\star, \mathsf{WitCreate}((\emptyset, \mathsf{pk}_\Lambda), C^\star, \mathsf{aux}', m'), m') = 1$. Thus, $\mathcal{B}$ returns $(\mathsf{WitCreate}((\emptyset, \mathsf{pk}_\Lambda), C^\star, \mathsf{aux}, m), m, \{m'\}, r')$ as a collision for the accumulator.

Hiding: Let us assume that there exists an efficient adversary $\mathcal{A}$ against the hiding property of Scheme 3.2. $\mathcal{C}$ runs $\mathsf{pk}_\Lambda \leftarrow \mathsf{Acc.Gen}(1^\kappa, 1)$ and starts $\mathcal{B}$ on pk_Λ, meaning that $\mathcal{C}'$ sets $\mathsf{PP} \leftarrow \mathsf{pk}_\Lambda$, runs $(m_0, m_1, \mathsf{ST}) \leftarrow \mathcal{A}(\mathsf{PP})$ and returns $(\mathcal{X}_0, \mathcal{X}_1, \mathsf{ST}) \leftarrow (\{m_0\}, \{m_1\}, \mathsf{ST})$. $\mathcal{C}$ then computes the challenge accumulator $\Lambda_{\mathcal{X}_b}$ and hands it to $\mathcal{B}$. Given that, $\mathcal{C}'$ starts $\mathcal{A}(\mathsf{PP}, \Lambda_{\mathcal{X}_b}, \mathsf{ST})$ and obtains and outputs $b^\star$. Thus, $\mathcal{B}$ breaks the indistinguishability of the accumulator. □

The black-box construction from Scheme 3.2 can easily be extended to support commitments to sets (where the opening is always with respect to the entire set) by setting the bound t of the bounded accumulator to the desired set size. Furthermore, using sk_Λ as trapdoor, one can also construct trapdoor commitments.

We finally note that cfw-indistinguishable accumulators (and hence also Transformation 3.1) are not useful for constructing commitments. The reason for this is that the accumulation of the additional random value immediately breaks the binding property.

3.1.4 ZK-Sets Imply Indistinguishable Undeniable Accumulators

Zero-knowledge sets (ZK-sets) [MRK03] allow to commit to a set $\mathcal{X}$ and then prove predicates of the form $x_i \in \mathcal{X}$ or $x_i \notin \mathcal{X}$ without revealing anything else about the set. We observe that ZK-sets can be used to model indistinguishable, unbounded, undeniable accumulators. Unfortunately, there is no formal security definition for zero-knowledge sets (in [KZG10] only the algorithms are formalized, while security is stated informally). However, zero-knowledge sets are a special instance of zero-knowledge elementary databases (ZK-EDB) [MRK03]. ZK-EDBs store key-value pairs and when querying the database with a key, the respective value is returned (or $\bot$ if the given key is not contained in the EDB). Thereby, no further information about the remaining EDB leaks. Therefore,

ZK-sets are ZK-EDBs where the values for all contained keys are set to 1 (or the values are omitted at all). We can, thus, define the security on the basis of the models in [MRK03, CHL+13] as follows.

Definition 3.10. *A ZK-set is a tuple of efficient algorithms* $(\mathsf{Gen}, \mathsf{Commit}, \mathsf{Query}, \mathsf{Verify})$*, which are defined as follows:*

$\mathsf{Gen}(1^\kappa)$: *This (probabilistic) algorithm takes input a security parameter* κ *and outputs a public key* pk*.*

$\mathsf{Commit}(\mathsf{pk}, \mathcal{X})$: *This algorithm takes input the public key* pk *and a set* $\mathcal{X}$ *and outputs a commitment* C *to* $\mathcal{X}$*.*

$\mathsf{Query}(\mathsf{pk}, \mathcal{X}, C, x)$: *This algorithm takes input the public key* pk*, a set* $\mathcal{X}$*, a corresponding commitment* C *and value* x*. It outputs a proof* π_x *if* $x \in \mathcal{X}$ *and a proof* $\underline{\pi}_x$ *if* $x \notin \mathcal{X}$*.*

$\mathsf{Verify}(\mathsf{pk}, C, x, \pi_x/\underline{\pi}_x)$: *This algorithm takes input the public key* pk*, a commitment* C *and a value* x*. Furthermore, it either takes a membership proof* π_x *or a non-membership proof* $\underline{\pi}_x$ *(denoted by* $\pi_x/\underline{\pi}_x$*). It outputs* 1 *if the proof can be correctly verified and* 0 *otherwise.*

For security, ZK-sets require *perfect completeness*, *soundness* and *zero-knowledge*. Perfect completeness requires that for every honestly generated key, every honestly computed commitment C, value x and corresponding proof $\pi_x/\underline{\pi}_x$, the Verify algorithm always returns 1. Since this property is straightforward, we do not formally state it here. We formally define the remaining properties:

Definition 3.11 (Soundness)**.** *A ZK-set is* sound*, if for all PPT adversaries* $\mathcal{A}$ *there is a negligible function* $\varepsilon(\cdot)$ *such that*

$$\Pr\left[\begin{array}{l|r} \mathsf{pk} \leftarrow \mathsf{Gen}(1^\kappa), & \mathsf{Verify}(\mathsf{pk}, C^\star, \pi_x^\star, x^\star) = 1 \ \wedge \\ (C^\star, x^\star, \pi_x^\star, \underline{\pi}_x^\star) \leftarrow \mathcal{A}(\mathsf{pk}) & \mathsf{Verify}(\mathsf{pk}, C^\star, \underline{\pi}_x^\star, x^\star) = 1 \end{array}\right] \leq \varepsilon(\kappa).$$

Definition 3.12 (Zero Knowledge)**.** *A ZK-set is* zero-knowledge*, if for all PPT adversaries* $\mathcal{A}$ *there is a negligible function* $\varepsilon(\cdot)$ *such that*

$$\left|\Pr\left[\begin{array}{l} \mathsf{pk} \leftarrow \mathsf{Gen}(1^\kappa), \\ (\mathcal{X}, \mathsf{st}) \leftarrow \mathcal{A}(\mathsf{pk}), \\ C \leftarrow \mathsf{Commit}(\mathsf{pk}, \mathcal{X}) \ : \\ \mathcal{A}^{\mathcal{O}^{\mathsf{Q}}(\cdot)}(\mathsf{st}, \mathsf{pk}, C) = 1 \end{array}\right] - \Pr\left[\begin{array}{l} (\mathsf{pk}, \tau) \leftarrow \mathcal{S}^{\mathsf{G}}(1^\kappa), \\ (\mathcal{X}, \mathsf{st}) \leftarrow \mathcal{A}(\mathsf{pk}), \\ (C, \tau') \leftarrow \mathcal{S}^{\mathsf{E}}(\mathsf{pk}, \tau) \ : \\ \mathcal{A}^{\mathcal{S}^{\mathsf{Q}}(\tau', \cdot)}(\mathsf{st}, \mathsf{pk}, C) = 1 \end{array}\right]\right| \leq \varepsilon(\kappa).$$

Here, $\mathcal{O}^{\mathsf{Q}}(x) \coloneqq \mathsf{Query}(\mathsf{pk}, \mathcal{X}, C, x)$, whereas $\mathcal{S} = (\mathcal{S}^{\mathsf{G}}, \mathcal{S}^{\mathsf{E}}, \mathcal{S}^{\mathsf{Q}})$ denotes a PPT simulator, which allows to execute the simulated Gen, Eval and Query algorithms, respectively. We note that the definition above is tailored to cover computational zero-knowledge. It could, however, easily be modified to also cover statistical or perfect zero knowledge.

In Scheme 3.3 we present a black-box construction of indistinguishable unbounded undeniable accumulators from ZK-sets.

$\underline{\mathsf{Gen}(1^\kappa)}$: Run $\mathsf{pk} \leftarrow \mathsf{ZKS.Gen}(1^\kappa)$ and return $(\mathsf{sk}_\Lambda, \mathsf{pk}_\Lambda) \leftarrow (\emptyset, \mathsf{pk})$.

$\underline{\mathsf{Eval}((\emptyset, \mathsf{pk}_\Lambda), \mathcal{X})}$: Run $\Lambda_\mathcal{X} \leftarrow \mathsf{ZKS.Commit}(\mathsf{pk}_\Lambda, \mathcal{X})$ and return $\Lambda_\mathcal{X}$ and $\mathsf{aux} \leftarrow \mathcal{X}$.

$\underline{\mathsf{WitCreate}((\emptyset, \mathsf{pk}_\Lambda), \Lambda_\mathcal{X}, \mathsf{aux}, x_i, \mathsf{type})}$: Parse aux as $\mathcal{X}$ and run $\pi_{x_i}/\underline{\pi_{x_i}} \leftarrow \mathsf{ZKS.Query}(\mathsf{pk}, \mathcal{X}, \Lambda_\mathcal{X}, x_i)$. If $\pi_{x_i}/\underline{\pi_{x_i}}$ conflicts with the requested witness type, return $\perp$. Otherwise, return $\mathsf{wit}_{x_i} \leftarrow \pi_{x_i}$ or $\underline{\mathsf{wit}_{x_i}} \leftarrow \underline{\pi_{x_i}}$, respectively.

$\underline{\mathsf{Verify}(\mathsf{pk}_\Lambda, \Lambda, \mathsf{wit}_{x_i}, x_i, \mathsf{type})}$: Check whether type conflicts with the type of the supplied witness and return $\perp$ if so. Otherwise, return the result of $\mathsf{ZKS.Verify}(\mathsf{pk}, \Lambda, x_i, \mathsf{wit}_{x_i})$.

Scheme 3.3: Indistinguishable unbounded undeniable accumulator from ZK-sets.

Theorem 3.3. *If ZK-sets exist, then indistinguishable, unbounded, undeniable accumulators exist as well.*

Proof. It is easy to see that the notions perfect completeness and soundness are equivalent to the correctness and undeniability notions of accumulators. However, the zero-knowledge property of ZK-sets is defined in a simulation-based way, whereas the indistinguishability is defined in a game-based way. To show that the zero-knowledge property of ZK-sets implies indistinguishability, we use the following sequence of games where we denote the winning condition of Game i by S_i.

As a ZK-set is zero-knowledge, there is a simulator $\mathcal{S}$ whose output cannot be distinguished from an honest output with non-negligible probability $\varepsilon(\kappa)$. We will now use $\mathcal{S}$ to replace the oracle answers in the indistinguishability game.

Game 0**:** The original accumulator indistinguishability game.

Game 1**:** We change the game such that all relevant steps are executed by the simulator. That is, the setup (Gen) is performed using $\mathcal{S}^\mathsf{G}$ and the computation of $\Lambda_{\mathcal{X}_b}$ is performed by the simulator using $\mathcal{S}^\mathsf{E}$. Furthermore, the outputs of the oracles $\mathcal{O}^\mathsf{W}$ and $\mathcal{O}^{\underline{\mathsf{W}}}$ are replaced by the output of $\mathcal{S}^\mathsf{Q}$. More precisely, if $\mathcal{O}^\mathsf{W}$ has been called and $\mathcal{S}^\mathsf{Q}$ returns a membership proof then $\mathcal{O}^\mathsf{W}$ returns π and $\perp$ otherwise. The oracle $\mathcal{O}^{\underline{\mathsf{W}}}$ works in the same way.

Transition - Game 0 → Game 1: A distinguisher between Game 0 to Game 1 is a zero-knowledge distinguisher, i.e., $|\Pr[S_0] - \Pr[S_1]| \leq \varepsilon(\kappa)$.

In Game 1, the adversary only gets to see simulated values, which contain no information about the respective set. Therefore, the advantage for the adversary to win this game is equal to 0, i.e., $\Pr[S_1] = 1/2$. Taking everything together, we derive that $|\Pr[S_0] - 1/2| \leq \varepsilon(\kappa)$ which shows that every ZK-set fulfilling the zero knowledge property is also an indistinguishable accumulator. □

The above black-box construction yields the first construction of indistinguishable undeniable accumulators. We note that it is, however, questionable whether the two notions of ZK-sets and indistinguishable undeniable accumulators are

equivalent (as the simulation based model of zero-knowledge appears to be stronger than the game based indistinguishability model).[6]

In [KZG10], Kate et al. introduced nearly ZK-sets. The difference to ordinary ZK-sets is that nearly ZK-sets have a public upper bound on the cardinality of set $\mathcal{X}$. It is apparent that these constructions imply indistinguishable t-bounded undeniable accumulators. In further consequence, this means that nearly ZK-sets can also be used to construct commitments (cf. Section 3.1.3).

3.2 Key-Homomorphic Signatures

The design of cryptographic schemes that possess certain homomorphic properties on their message space has witnessed significant research within the last years. In the domain of encryption, the first candidate construction of fully homomorphic encryption (FHE) due to Gentry [Gen09] has initiated a fruitful area of research with important applications to computations on (outsourced) encrypted data. In the domain of signatures, the line of work on homomorphic signatures [JMSW02], i.e., signatures that are homomorphic with respect to the message space, has only quite recently attracted attention. Firstly, due to the introduction of computing on authenticated data [ABC$^+$12]. Secondly, due to the growing interest in the application to verifiable delegation of computations (cf. [Cat14] for a quite recent overview), and, finally, due to the recent construction of fully homomorphic signatures [GVW15, BFS14].

In this section we are interested in another type of homomorphic schemes, so called key-homomorphic schemes. Specifically, we study key-homomorphic signature schemes, that is, signature schemes which are homomorphic with respect to the key space. As we will show, this concept turns out to be a very interesting and versatile tool.

While we are the first to explicitly study key-homomorphic properties of signatures, some other primitives have already been studied with respect to key-homomorphic properties previously. Applebaum et al. in [AHI11] studied key-homomorphic symmetric encryption schemes in context of related key attacks (RKAs). Recently, Dodis et al. [DMS16] have shown that any such key-homomorphic symmetric encryption scheme implies public key encryption. Rothblum [Rot11] implicitly uses key malleability to construct (weakly) homomorphic public key bit-encryption schemes from private key ones. Goldwasser et al. in [GLW12], and subsequently Tessaro and Wilson in [TW14], use public key encryption schemes with linear homomorphisms over their keys (and some related properties) to construct bounded-collusion identity-based encryption (IBE). Recently, Boneh et al. introduced the most general notion of fully key-homomorphic encryption [BGG$^+$14]. In such a scheme, when given a ciphertext under a public key pk, anyone can translate it into a ciphertext to the same plaintext under public key $(f(\mathsf{pk}), f)$ for any efficiently computable function f.

[6] Meanwhile this question has been answered: it was shown that ZK-sets are actually stronger than indistinguishable undeniable accumulators [GOP$^+$16].

Another line of work recently initiated by Boneh et al. [BLMR13] is concerned with key-homomorphic pseudorandom functions (PRFs) and pseudo random generators (PRGs). Loosely speaking, a secure PRF family $F : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$, is key-homomorphic if the keys live in a group $(\mathcal{K}, +)$, and, given two evaluations $F(k_1, x)$ and $F(k_2, x)$ for the same value under two keys, one can efficiently compute $F(k_1 + k_2, x)$. Such PRFs turn out to yield interesting applications such as distributed PRFs, symmetric key proxy re-encryption or updatable encryption. Continuing the work in this direction, alternative constructions [BP14] and extended functionality in the form of constrained key-homomorphic PRFs have been proposed [BFP+15]. We note that the result from Dodis et al. [DMS16], although not mentioned, answers the open question posed by Boneh et al.[BLMR13] "whether key-homomorphic PRFs whose performance is comparable to real-world block ciphers such as AES exist" in a negative way. Finally, Benhamouda et al. use key-homomorphic projective hash functions to construct aggregator oblivious encryption schemes [BJL16] and inner-product functional encryption schemes [BBL17].

When switching to the field of signatures, we can define key-homomorphisms in various different ways, of which we subsequently sketch two to provide a first intuition. One notion is to require that given two signatures for the same message m valid under some pk_1 and pk_2 respectively, one can publicly compute a signature to message m that is valid for a public key pk' that is obtained via some operation on pk_1 and pk_2. Another variant for instance is to require that, given a signature σ to a message m that verifies under pk, σ can be adapted to a signature to m under pk'. Thereby, pk and pk' have a well defined relationship (cf. Section 3.2 for the details).

Although key-homomorphic signatures have never been discussed or studied explicitly, some implicit use of key-homomorphisms can be found. A recent work by Kiltz et al. [KMP16] introduces a property for canonical identification schemes denoted as random self-reducibility. This basically formalizes the re-randomization of key-pairs as well as adapting parts of transcripts of identification protocols consistently. Earlier, Fischlin and Fleischhacker in [FF13] used re-randomization of key-pairs implicitly in their meta reduction technique against Schnorr signatures. This concept has recently been formalized, yielding the notion of signatures with re-randomizable keys [FKM+16]. In such schemes the EUF-CMA security notion is slightly tweaked, by additionally allowing the adversary to see signatures under re-randomized keys. These signatures with re-randomizable keys are then used as basis of an elegant construction of unlinkable sanitizable signatures (cf. [FKM+16]). Allowing the adversary to also access signatures under re-randomized (related) keys, has earlier been studied in context of security of signature schemes against related-key attacks (RKAs) [BCM11, BPT12]. In this context, the goal is to prevent that signature schemes have key-homomorphic properties that allow to adapt signatures under related keys to signatures under the original key (cf. e.g., [MSM+15]).

Concurrent Work. In concurrent and independent work Lai et al. [LTWC16] study different flavours of multi-key homomorphic signatures. They consider

homomorphisms on the message and/or key space and show equivalences of different types of such multi-key homomorphic signatures (which are all implied by zk-SNARKS). What they call multi-key key-message-homomorphic signatures can be seen as related to our notion of key-homomorphisms. Yet, our works target totally different directions. Their approach is top-down, i.e., the focus is on introducing new primitives and showing implications between them. In contrast, our approach is bottom-up, i.e., our focus lies on distilling additional properties of larger classes of existing schemes, to (1) obtain new insights regarding generic construction paradigms involving schemes from those classes, and (2) to obtain new instantiations by solely analyzing schemes with respect to their properties. We also note that another concurrent and independent work [FMNP16] introduces the notion of multi-key homomorphic authenticators. However, their work is only related to the parts of [i5] which are not included in this thesis, and, therefore, we refer the reader to [i5] for a discussion.

Contribution. We initiate the study of key-homomorphic signature schemes. In doing so, we propose various natural definitions of key-homomorphic signatures, generalizing larger classes of existing signature schemes. This generalization makes it possible to infer general statements about signature schemes from those classes by simply making black-box use of the respective properties. Thereby, we rule out certain combinations of key-homomorphism and existing unforgeability notions of signatures. We then employ the formalisms provided by our definitional framework to show various interesting relations and implications. From a theoretical viewpoint our results contribute towards establishing a better understanding of the paradigms which are necessary to construct certain schemes and/or to achieve certain security notions. In particular, we start from very mild security requirements and show how to employ our framework to amplify those to yield relatively strong security guarantees. From a practical viewpoint, our so obtained constructions compare favorably to existing work: they are conceptually extremely easy to understand and therefore less prone to wrong usage. At the same time, our results yield instantiations with no or even reduced overhead when compared to existing work.

More specifically, besides our framework which we see as a contribution on its own, our contributions are as follows.

Generic Compilers. We show that our framework enables various compilers from classes of schemes providing different types of key-homomorphisms to other interesting variants of signature schemes. As a first example, we show that multisignatures are directly implied by signatures with certain key-homomorphic properties in this section. We, however, stress that our framework yields various other (potentially more interesting) compilers which we will present in separate sections. In particular, we present compilers to (universal) designated verifier signatures (Section 3.3), simulation sound extractable arguments of knowledge (Section 3.4), as well as ring signatures (Section 4.1). The so obtained constructions, besides being very efficient, are simple and elegant from a construction and security analysis point of view. Basically, for ring signatures, (universal) des-

ignated verifier signatures and weakly simulation sound extractable argument systems, one computes a signature using any suitable key-homomorphic scheme under a freshly sampled key and then proves a simple relation over public keys *only*. For simulation sound extractable argument systems we additionally require a strong one-time signature scheme (which, however, also exists under standard assumptions [Gro06]).

Tight Key-Prefixed Multi-User Security. We prove a theorem which tightly relates the single-user existential unforgeability under chosen message attacks (EUF-CMA) of a class of schemes admitting a particular key-homomorphism to its key-prefixed multi-user EUF-CMA security. This theorem addresses a frequently occurring question in the context of standardization and generalizes existing theorems [Ber15, Lac16] (where such implications are proven for concrete signature schemes) so that it is applicable to a larger class of signature schemes.

(Standard Model & Standard Assumption) Instantiations. We give examples of existing signature schemes admitting types of key-homomorphisms we define. Using our compilers, this directly yields previously unknown instantiations of all variants of signature schemes mentioned above. Most interestingly, we can show that a variant of Waters' signatures in the SXDH setting is perfectly adaptable. As we will see, this gives us novel and simple constructions of various types of signature schemes without random oracles from standard assumptions (if required, we can use witness-indistinguishable Groth-Sahai [GS08] proofs as argument system). All our instantiations compare favorably to existing constructions regarding conceptual simplicity and come at no or even reduced computational overhead. Likewise, our general theorem for multi-user security attests the multi-user security for schemes whose multi-user security was previously unknown.

A Note on the Security of Multiparty Signatures. In multiparty signature schemes one often relies on the so called knowledge of secret key (KOSK) assumption within security proofs, where the adversary is required to reveal the secret keys it utilizes to the environment. This is important to prevent rogue-key attacks, i.e., attacks where the adversary constructs public keys based on existing public keys in the system so that it is not required to know the secret key corresponding to the resulting public keys.

To prevent such rogue-key attacks, Ristenpart and Yilek [RY07] introduced and formalized an abstract key-registration concept for multiparty signatures. Any such key-registration protocol is represented as a pair of interactive algorithms $(\mathsf{RegP}, \mathsf{RegV})$. A party registering a key runs RegP with inputs public key pk and private key sk. A certifying authority (CA) runs RegV, where the last message is from RegV to RegP and contains either a pk or $\perp$. For instance, in the plain model $\mathsf{RegP}(\mathsf{pk}, \mathsf{sk})$ simply sends pk to the CA and RegV on receiving pk simply returns pk. For the KOSK assumption, $\mathsf{RegP}(\mathsf{pk}, \mathsf{sk})$ simply sends $(\mathsf{pk}, \mathsf{sk})$ to the CA, which checks if $(\mathsf{sk}, \mathsf{pk}) \in \mathsf{KeyGen}(\mathsf{PP})$ and if so replies with pk and $\perp$ otherwise.

To get rid of the KOSK assumption in real protocols without revealing the secret key, one can require the adversary to prove knowledge of its secret key in a way that it can be straight-line extracted by the environment. We assume this to happen for all our mulitparty signature schemes. Yet, we do not make it explicit to avoid complicated models and we simply introduce an RKey oracle that allows the adversary to register key pairs. We stress that our goal is not to study multiparty signatures with respect to real-world key-registration procedures, as done in [RY07].

3.2.1 Formal Definition of the Framework

In this section, we introduce a definitional framework for key-homomorphic signature schemes. In doing so, we propose different natural notions and relate the definitions to previous work that already implicitly used functionality that is related or covered by our definitions.

We focus on signature schemes $\Sigma = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$, where the secret and public key elements live in groups $(\mathbb{H}, +)$ and $(\mathbb{G}, \cdot)$, respectively. We start with the notion of an efficiently computable homomorphism between secret keys and public keys in analogy to the use within IBE in [TW14]. Such a functionality has been implicitly used recently in [FKM+16] to define the notion of signatures with re-randomizable keys.

Definition 3.13 (Secret Key to Public Key Homomorphism)**.** *A signature scheme Σ provides a secret key to public key homomorphism, if there exists an efficiently computable map $\mu : \mathbb{H} \to \mathbb{G}$ such that for all $\mathsf{sk}, \mathsf{sk}' \in \mathbb{H}$ it holds that $\mu(\mathsf{sk} + \mathsf{sk}') = \mu(\mathsf{sk}) \cdot \mu(\mathsf{sk}')$, and for all $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KeyGen}$, it holds that $\mathsf{pk} = \mu(\mathsf{sk})$.*

We stress that secret keys and public keys may be vectors containing elements of $\mathbb{H}$ and $\mathbb{G}$ respectively. Then, the operations $+$, $\cdot$ and the map μ are applied componentwise. To keep the definitions compact, we however do not make this explicit. Also, some schemes require to include a copy of the public key in the secret key. In our definitions those parts are implicitly covered via the constraints on the public keys, and we may therefore simply ignore them in the secret keys.

In the discrete logarithm setting, where we often have $\mathsf{sk} \xleftarrow{R} \mathbb{Z}_p$ and $\mathsf{pk} = g^{\mathsf{sk}}$ with g being the generator of some prime order p group $\mathbb{G}$, it is obvious that there exists $\mu : \mathsf{sk} \mapsto g^{\mathsf{sk}}$ that is efficiently computable.

Now, we can introduce the first flavour of key-homomorphic signatures, where we focus on the class of functions Φ^+ representing linear shifts and note that one could easily adapt our definition to other suitable classes Φ of functions instead of linear shifts. We stress that we consider Φ as a finite set of functions, all with the same domain and range, and they usually depend on the public key of the signature scheme (which we will not make explicit). Moreover, Φ admits an efficient membership test, is efficiently samplable, and, its functions are efficiently computable. Definition 3.14 together with the adaptability of signatures (Definition 3.15) or perfect adaption (Definition 3.16) are inspired by key-homomorphic encryption schemes [AHI11].

Definition 3.14 (Φ^+-Key-Homomorphic Signatures)**.** *A signature scheme is called Φ^+-key-homomorphic, if it provides a secret key to public key homomorphism and an additional PPT algorithm* Adapt*, defined as:*

$\mathsf{Adapt}(\mathsf{pk}, m, \sigma, \Delta)$: *Takes a public key* pk*, a message m, a signature σ, and a function $\Delta \in \Phi^+$ as input, and outputs a public key* pk' *and a signature σ'.*

Additionally, we require that for all $\Delta \in \Phi^+$ and all $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\kappa)$, all messages m and all $\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, m)$ and $(\mathsf{pk}', \sigma') \leftarrow \mathsf{Adapt}(\mathsf{pk}, m, \sigma, \Delta)$ it holds that

$$\Pr[\mathsf{Verify}(\mathsf{pk}', m, \sigma') = 1] = 1 \quad \wedge \quad \mathsf{pk}' = \Delta(\mathsf{pk}).$$

In the remainder of this thesis, we identify a function $\Delta \in \Phi^+$ with its "shift amount" $\Delta \in \mathbb{H}$.

An interesting property in the context of key-homomorphic signatures is whether adapted signatures look like freshly generated signatures. Therefore, we introduce two different flavours of such a notion, inspired by the context hiding notion for P-homomorphic signatures [ABC+12, ALP12] as well as the adaptability notion from [FHS15a] for equivalence class signatures [HS14].

Definition 3.15 (Adaptability of Signatures)**.** *A Φ^+-key-homomorphic signature scheme provides adaptability of signatures, if for every $\kappa \in \mathbb{N}$ and every message m, it holds that $\mathsf{Adapt}(\mathsf{pk}, m, \mathsf{Sign}(\mathsf{sk}, m), \Delta)$ and $(\mathsf{pk} \cdot \mu(\Delta), \mathsf{Sign}(\mathsf{sk} + \Delta, m))$ as well as $(\mathsf{sk}, \mathsf{pk})$ and $(\mathsf{sk}', \mu(\mathsf{sk}'))$ are identically distributed, where $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KeyGen}(1^\kappa)$, $\mathsf{sk}' \xleftarrow{R} \mathbb{H}$, and $\Delta \xleftarrow{R} \Phi^+$.*

Remark 3.2. *Kiltz et al. [KMP16] have recently used a notion related to Definition 3.15 (denoted as random self-reducibility) in the context of canonical identification schemes. We observe that when turning canonical identification schemes into signature schemes in the random oracle model using the Fiat-Shamir heuristic, every random-self-reducible scheme also satisfies adaptability. However, the contrary is not true as our notion covers a broader class of signature schemes, and in particular including schemes in the standard model (cf. Appendix 3.2.2).*

Thus, the examples of random-self-reducible canonical identification schemes given in [KMP16] directly yield examples of adaptable signatures in the ROM. In Section 3.2.2 we explicitly show that the Schnorr signatures [Sch91] scheme, as well as a signature scheme due to Katz and Wang [KW03, GJKW07] are adaptable according to the definition above.

An even stronger notion for the indistinguishability of fresh signatures and adapted signatures on the same message is achieved when requiring the distributions to be indistinguishable *even* when the initial signature used in Adapt is known. All schemes that satisfy this stronger notion (stated below) also satisfy Definition 3.15.

Definition 3.16 (Perfect Adaption)**.** *A Φ^+-key-homomorphic signature scheme provides perfect adaption, if for every $\kappa \in \mathbb{N}$, every message m, and every signature $\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, m)$, it holds that $(\sigma, \mathsf{Adapt}(\mathsf{pk}, m, \sigma, \Delta))$ and $(\sigma, \mathsf{pk} \cdot \mu(\Delta), \mathsf{Sign}($*

$\mathsf{sk} + \Delta, m))$ *as well as* $(\mathsf{sk}, \mathsf{pk})$ *and* $(\mathsf{sk}', \mu(\mathsf{sk}'))$ *are identically distributed, where* $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KeyGen}(1^\kappa)$, $\mathsf{sk}' \xleftarrow{R} \mathbb{H}$, *and* $\Delta \xleftarrow{R} \Phi^+$.

One immediately sees that signatures from random-self-reducible canonical identification schemes, and, thus, Schnorr signatures as well as Katz-Wang signatures, do not satisfy Definition 3.16 as the commitment sent in the first phase remains fixed. However, we note that there are various existing schemes that satisfy Definition 3.16. For example, BLS signatures [BLS04], the recent re-randomizable scheme by Pointcheval and Sanders [PS16], a variant of the well known Waters' signatures [Wat05], or the CL signature variant from [CHP12] to name some (cf. Section 3.2.2 for a formal treatment of these schemes).

When looking at Definition 3.14, one could ask whether it is possible to replace Δ in the Adapt algorithm with its public key $\mu(\Delta)$. However, it is easily seen that the existence of such an algorithm contradicts even the weakest security guarantees the underlying signature scheme would need to provide, i.e., universal unforgeability under no-message attacks ($\mathsf{UUF\text{-}NMA}$ security).

Lemma 3.16. *There cannot be an* $\mathsf{UUF\text{-}NMA}$ *secure* Φ^+*-key-homomorphic signature scheme* Σ *for which there exists a modified PPT algorithm* Adapt' *taking* $\mu(\Delta)$ *instead of* Δ *that still satisfies Definition 3.14.*

Proof. We prove this by showing that any such scheme implies an adversary against $\mathsf{UUF\text{-}NMA}$ security of Σ. Let us assume that an $\mathsf{UUF\text{-}NMA}$ challenger provides a public key $\mathsf{pk}^\star$ and a target message $m^\star$. Run $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KeyGen}(1^\kappa)$ being compatible with public key $\mathsf{pk}^\star$, compute $\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, m^\star)$, then compute $\mathsf{pk}' \leftarrow \mathsf{pk}^\star \cdot \mathsf{pk}^{-1}$ and obtain a forgery $\sigma^\star$ for message $m^\star$ under the target public key $\mathsf{pk}^\star$ by running $(\sigma^\star, \mathsf{pk}^\star) \leftarrow \mathsf{Adapt}(\mathsf{pk}, m^\star, \sigma, \mathsf{pk}')$. □

Now, we move to a definition that covers key-homomorphic signatures where the adaption of a *set of* signatures, each to the same message, to a signature for the same message under a combined public key does not even require the knowledge of the relation between the secret signing keys.

Definition 3.17 (Publicly Key-Homomorphic Signatures)**.** *A signature scheme is called publicly key-homomorphic, if it provides a secret key to public key homomorphism and an additional PPT algorithm* $\mathsf{Combine}$*, defined as:*

$\mathsf{Combine}((\mathsf{pk}_i)_{i=1}^n, m, (\sigma_i)_{i=1}^n)$: *Takes public keys* $(\mathsf{pk}_i)_{i\in[n]}$*, a message* m*, signatures* $(\sigma_i)_{i\in[n]}$ *as input, and outputs a public key* $\hat{\mathsf{pk}}$ *and a signature* $\hat{\sigma}$*,*

such that for all $n > 1$*, all* $((\mathsf{sk}_i, \mathsf{pk}_i) \leftarrow \mathsf{KeyGen}(1^\kappa))_{i=1}^n$*, all messages* m *and all* $(\sigma_i \leftarrow \mathsf{Sign}(\mathsf{sk}_i, m))_{i\in[n]}$ *and* $(\tilde{\mathsf{pk}}, \tilde{\sigma}) \leftarrow \mathsf{Combine}((\mathsf{pk}_i)_{i=1}^n, m, (\sigma_i)_{i=1}^n)$ *it holds that* $\tilde{\mathsf{pk}} = \prod_{i=1}^n \mathsf{pk}_i \quad \wedge \quad \Pr[\mathsf{Verify}(\tilde{\mathsf{pk}}, m, \tilde{\sigma}) = 1] = 1$.

Analogously to Definitions 3.15 and 3.16, one can define indistinguishability of fresh and combined signatures, but we omit it here as it is straight forward. We want to mention that Definition 3.17 is, for instance, satisfied by BLS signatures, Waters' signatures with shared Waters' hash parameters (cf. [LOS+06]), as well

as the scheme with shared parameters assuming synchronized time in [CHP12] being a variant of the CL signature scheme [CL04] (cf. Section 3.2.2 for a more formal treatment of these schemes).

3.2.2 Examples of Key-Homomorphic Signature Schemes

We now give some examples of signature schemes providing key-homomorphic properties. Before we provide the details, we compactly subsume our results in Table 3.2.

Scheme	A	PA	PKH	$\mathsf{Dom}(\Delta)$	$\mathsf{PoK}(\Delta)$
Schnorr [Sch91]	✓	×	×	$\mathbb{Z}_p$	$\mathbb{Z}_p$
BLS [BLS04]	✓	✓	✓	$\mathbb{Z}_p$	$\mathbb{Z}_p$
Katz-Wang [KW03, GJKW07]	✓	×	×	$\mathbb{Z}_p$	$\mathbb{Z}_p$
Waters [Wat05, BFG13]	✓	✓	✓	$\mathbb{G}_1 \times \mathbb{G}_2$	$\mathbb{G}_1$
PS [PS16]	✓	✓	×	$\mathbb{Z}_p \times \mathbb{Z}_p$	$\mathbb{Z}_p \times \mathbb{Z}_p$
CL Variant [CHP12]	✓	✓	✓	$\mathbb{Z}_p$	$\mathbb{Z}_p$

Table 3.2: Overview of key-homomorphic properties of existing signature schemes. All schemes admit a Φ^+ key-homomorphism. Legend: A . . . adaptable, PA . . . perfectly adaptable, PKH . . . publicly key-homomorphic, $\mathsf{Dom}(\Delta)$ . . . domain where the shift amounts live in, $\mathsf{PoK}(\Delta)$ . . . domain of witnesses in proof of knowledge of shift amount.

Schnorr Signatures [Sch91]. In Scheme 3.4 we recall the Schnorr signature scheme.

$\underline{\mathsf{PGen}(1^\kappa)}$: Run $\mathsf{G} \leftarrow \mathsf{GGen}(1^\kappa)$, and choose a hash function $H : \mathbb{G} \times \mathcal{M} \to \{0,1\}^\kappa$ uniformly at random from hash function family $\{H_k\}_k$. Set and return $\mathsf{pp} \leftarrow (\mathsf{G}, H)$.

$\underline{\mathsf{KeyGen}(\mathsf{pp})}$: Parse pp as (G, H), choose $x \xleftarrow{R} \mathbb{Z}_p$, set $\mathsf{pk} \leftarrow (\mathsf{pp}, g^x)$, $\mathsf{sk} \leftarrow (\mathsf{pk}, x)$ and output $(\mathsf{sk}, \mathsf{pk})$.

$\underline{\mathsf{Sign}(\mathsf{sk}, m)}$: Parse sk as x, choose $r \xleftarrow{R} \mathbb{Z}_p$, compute $R \leftarrow g^r$, $c \leftarrow H(R, m)$, $y \leftarrow r + x \cdot c \bmod p$, and output $\sigma \leftarrow (c, y)$

$\underline{\mathsf{Verify}(\mathsf{pk}, m, \sigma)}$: Parse pk as (pp, g^x) and σ as (c, y), verify whether $c = H((g^x)^{-c} g^y, m)$ and output 1 if so and 0 otherwise.

Scheme 3.4: Schnorr signatures.

Lemma 3.17. *Schnorr signatures are adaptable according to Definition 3.15.*

Proof. We prove the lemma above by presenting an Adapt algorithm satisfying the perfect adaptability notion.

$\mathsf{Adapt}(\mathsf{pk}, m, \sigma, \Delta)$: Let $\Delta \in \mathbb{Z}_p$ and $\mathsf{pk} = (\mathsf{pp}, g^x)$. Return (pk', σ'), where $\mathsf{pk}' \leftarrow (\mathsf{pp}, g^x \cdot g^\Delta)$ and $\sigma' \leftarrow (c, y')$ with $y' \leftarrow y + c \cdot \Delta \bmod p$.

It is immediate that adapted signatures are identical to fresh signatures under $\mathsf{pk}' = (\mathsf{PP}, g^{x+\Delta})$ as long as the initial signature is unknown. □

BLS Signatures [BLS04]. In Scheme 3.5 we recall BLS signatures in a Type 3 setting (cf. [CHKM10] for a treatment of security of this BLS variant). We stress that the properties which we discuss below are equally valid for the original BLS scheme in [BLS04] instantiated in a Type 2 setting.

$\underline{\mathsf{PGen}(1^\kappa)}$: Run $\mathsf{BG} \leftarrow \mathsf{BGGen}(1^\kappa, 3)$, choose a hash function $H : \mathcal{M} \to \mathbb{G}_1$ uniformly at random from hash function family $\{H_k\}_k$, set $\mathsf{PP} \leftarrow (\mathsf{BG}, H)$.

$\underline{\mathsf{KeyGen}(\mathsf{PP})}$: Parse PP as (BG, H), choose $x \xleftarrow{R} \mathbb{Z}_p$, set $\mathsf{pk} \leftarrow (\mathsf{PP}, \hat{g}^x)$, $\mathsf{sk} \leftarrow (\mathsf{pk}, x)$, and return $(\mathsf{sk}, \mathsf{pk})$.

$\underline{\mathsf{Sign}(\mathsf{sk}, m)}$: Parse sk as x and return $\sigma \leftarrow H(m)^x$.

$\underline{\mathsf{Verify}(\mathsf{pk}, m, \sigma)}$: Parse pk as $(\mathsf{PP}, \hat{g}^x)$, verify whether $e(H(m), \hat{g}^x) = e(\sigma, \hat{g})$ and return 1 if so and 0 otherwise.

Scheme 3.5: Type 3 BLS signatures.

Lemma 3.18. *BLS signatures are perfectly adaptable according to Definition 3.16.*

Proof. We prove the lemma above by presenting an Adapt algorithm satisfying the perfect adaptability notion.

$\mathsf{Adapt}(\mathsf{pk}, m, \sigma, \Delta)$: Let $\Delta \in \mathbb{Z}_p$ and $\mathsf{pk} = (\mathsf{PP}, \hat{g}^x)$. Return (pk', σ'), where $\mathsf{pk}' \leftarrow (\mathsf{PP}, \hat{g}^x \cdot \hat{g}^\Delta)$ and $\sigma' \leftarrow \sigma \cdot H(m)^\Delta$.

It is immediate that adapted signatures are identical to fresh signatures under $\mathsf{pk}' = (\mathsf{PP}, \hat{g}^{x+\Delta})$. □

Lemma 3.19. *BLS signatures are publicly key-homomorphic according to Definition 3.17.*

Proof. We prove the lemma above by presenting a suitable $\mathsf{Combine}$ algorithm.

$\mathsf{Combine}((\mathsf{pk}_i)_{i=1}^n, m, (\sigma_i)_{i=1}^n)$: Let $\mathsf{pk}_i = (\mathsf{PP}, \hat{g}^{x_i})$. Run $\tilde{\mathsf{pk}} \leftarrow (\mathsf{PP}, \prod_{i=1}^n \hat{g}^{x_i})$, and $\tilde{\sigma} \leftarrow \prod_{i=1}^n \sigma_i$ and return public key $\tilde{\mathsf{pk}}$ and signature $\tilde{\sigma}$. □

Katz-Wang Signatures [KW03, GJKW07]. Katz and Wang in [KW03] presented a signature scheme that enjoys a tight security reduction to the DDH problem. Basically, the public key of the scheme represents a Diffie-Hellman (DH) tuple and the signature is a non-interactive zero-knowledge proof (obtained using the Fiat-Shamir heuristic) that the public key indeed forms a DH tuple. We present the version from [GJKW07] (Section 4) in Scheme 3.6, which in contrast to the original one in [KW03] does not include the statement (public key) in the Fiat-Shamir transform.

$\underline{\mathsf{PGen}(1^\kappa)}$: Run $\mathsf{G} \leftarrow \mathsf{GGen}(1^\kappa)$, $h \xleftarrow{R} \mathbb{G}$, and choose a hash function $H : \mathbb{G} \times \mathbb{G} \times \mathcal{M} \rightarrow \{0,1\}^\kappa$ uniformly at random from hash function family $\{H_k\}_k$. Set and return $\mathsf{PP} \leftarrow (\mathsf{G}, h, H)$.

$\underline{\mathsf{KeyGen}(\mathsf{PP})}$: Parse PP as (G, h, H), choose $x \xleftarrow{R} \mathbb{Z}_p$, set $\mathsf{pk} \leftarrow (\mathsf{PP}, g^x, h^x)$, $\mathsf{sk} \leftarrow (\mathsf{pk}, x)$, and return $(\mathsf{sk}, \mathsf{pk})$.

$\underline{\mathsf{Sign}(\mathsf{sk}, m)}$: Parse sk as x, choose $r \xleftarrow{R} \mathbb{Z}_p$, set $A \leftarrow g^r$, $B \leftarrow h^r$, $c \leftarrow H(A, B, m)$, compute $s \leftarrow cx + r \bmod p$ and return $\sigma \leftarrow (c, s)$.

$\underline{\mathsf{Verify}(\mathsf{pk}, m, \sigma)}$: Parse pk as (PP, y_1, y_2) and σ as (c, s) with $c \in \{0,1\}^n$ and $s \in \mathbb{Z}_p$. Compute $A \leftarrow g^s y_1^{-c}$, $B \leftarrow h^s y_2^{-c}$ and return 1 if $c = H(A, B, m)$ and 0 otherwise.

Scheme 3.6: Katz-Wang signatures.

Lemma 3.20. *Katz-Wang signatures are adaptable according to Definition 3.15.*

Proof. We prove the lemma above by presenting an Adapt algorithm satisfying the adaptability notion.

$\mathsf{Adapt}(\mathsf{pk}, m, \sigma, \Delta)$: Let $\Delta \in \mathbb{Z}_p$, $\mathsf{pk} = (\mathsf{PP}, y_1, y_2)$ and $\sigma = (c, s)$. Return (pk', σ'), where $\mathsf{pk}' \leftarrow (\mathsf{PP}, y_1 \cdot g^\Delta, y_2 \cdot h^\Delta)$ and $\sigma' \leftarrow (c, s + c\Delta \pmod p)$.

It is immediate that adapted signatures are identical to fresh signatures under $\mathsf{pk}' = (\mathsf{PP}, y_1 \cdot g^\Delta, y_2 \cdot h^\Delta)$ as long as the initial signature is unknown. □

Waters' Signatures [Wat05]. Below we recall Waters' signatures with shared hashing parameters in the Type-3 bilinear group setting as used in [BFG13] (a similar variant is presented in [CHKM10]). We note that for Waters' signatures without shared hash parameters [Wat05] it seems to be impossible to define an Adapt algorithm satisfying Definition 3.14.

$\underline{\mathsf{PGen}(1^\kappa)}$: Run $\mathsf{BG} \leftarrow \mathsf{BGGen}(1^\kappa, 3)$, choose $U = (h, u_0, \ldots u_n) \xleftarrow{R} \mathbb{G}_1^{k+1}$, and define $H : \mathcal{M} \rightarrow \mathbb{G}_1$ as $H(m) \coloneqq u_0 \cdot \prod_{i=1}^n u_i^{m_i}$, where $\mathcal{M} = \{0,1\}^n$. Set $\mathsf{PP} \leftarrow (\mathsf{BG}, U, H)$.

$\underline{\mathsf{KeyGen}(\mathsf{PP})}$: Parse PP as (BG, U, H), choose $x \xleftarrow{R} \mathbb{Z}_p$, set $\mathsf{pk} \leftarrow (\mathsf{PP}, \hat{g}^x)$, $\mathsf{sk} \leftarrow (\mathsf{pk}, x)$, and return $(\mathsf{sk}, \mathsf{pk})$.

$\underline{\mathsf{Sign}(\mathsf{sk}, m)}$: Parse sk as (pk, x), choose $r \xleftarrow{R} \mathbb{Z}_p$, set $\alpha \leftarrow h^x \cdot H(m)^r$, $\beta \leftarrow \hat{g}^r, \gamma \leftarrow g^r$ and return $\sigma \leftarrow (\alpha, \beta, \gamma)$.

$\underline{\mathsf{Verify}(\mathsf{pk}, m, \sigma)}$: Parse pk as $(\mathsf{PP}, \hat{g}^x)$ and σ as (α, β, γ). Verify whether $e(\alpha, \hat{g}) = e(h, \hat{g}^x) \cdot e(H(m), \beta) \ \wedge \ e(\gamma, \hat{g}) = e(g, \beta)$ and return 1 if it holds and 0 otherwise.

Scheme 3.7: Waters' signatures with shared hash parameters.

Lemma 3.21. *Waters' signatures with shared hash parameters are perfectly adaptable according to Definition 3.16.*

Proof. We prove the lemma above by presenting an Adapt algorithm satisfying the perfect adaptability notion.

$\mathsf{Adapt}(\mathsf{pk}, m, \sigma, \Delta)$: Let $\Delta \in \mathbb{Z}_p$ and compute $\Delta_1 \leftarrow h^{\Delta}$ and $\Delta_2 \leftarrow \hat{g}^{\Delta}$. Further, let $\sigma = (\alpha, \beta, \gamma)$, and $\mathsf{pk} = (\mathsf{PP}, \hat{g}^x)$. Choose $r' \xleftarrow{R} \mathbb{Z}_p$, compute $\sigma' \leftarrow (\alpha \cdot \Delta_1 \cdot H(m)^{r'}, \beta \cdot \hat{g}^{r'}, \gamma \cdot g^{r'})$ and $\mathsf{pk}' \leftarrow (\mathsf{PP}, \hat{g}^x \cdot \Delta_2)$, and return (pk', σ').

Signatures output by Adapt are identically distributed as fresh signatures under randomness $r + r'$ und key $\mathsf{pk} = (\mathsf{PP}, \hat{g}^x \cdot \Delta_2,)$, which proves the lemma. □

When instantiating our argument system from Section 3.4 with Groth-Sahai proofs, it is beneficial to use $\mathsf{sk} \leftarrow (\mathsf{pk}, h^x, \hat{g}^x)$ as secret key (note that the security reduction still works in the same way when using this modified key). The associated secret key space would then be all tuples $(\Delta_1, \Delta_2) \in \mathbb{H} \subset \mathbb{G}_1 \times \mathbb{G}_2$ where $e(\Delta_1, \hat{g}) = e(h, \Delta_2)$ and also $\Delta \in \mathbb{H}$. This is favourable regarding the extractability properties of the Groth-Sahai proof system. Observe that in this setting Δ_2 is implicitly given by the difference of the public keys and thus one only needs to prove knowledge of a *single* group element.

Lemma 3.22. *Waters' signatures are publicly key-homomorphic according to Definition 3.17.*

Proof. We prove the lemma above by presenting a suitable $\mathsf{Combine}$ algorithm.

$\mathsf{Combine}((\mathsf{pk}_i)_{i=1}^n, m, (\sigma_i)_{i=1}^n)$: Let $\sigma_i = (\alpha_i, \beta_i, \gamma_i)$ and $\mathsf{pk}_i = (\mathsf{PP}, \hat{g}^{x_i})$. Run $\tilde{\mathsf{pk}} \leftarrow (\mathsf{PP}, \prod_{i=1}^n \hat{g}^{x_i}))$ and $\tilde{\sigma} \leftarrow (\prod_{i=1}^n \alpha_i, \prod_{i=1}^n \beta_i, \prod_{i=1}^n \gamma_i)$ and return public key $\tilde{\mathsf{pk}}$ and signature $\tilde{\sigma}$. □

PS Signatures [PS16]. In Scheme 3.8 we recall a recent signature scheme from [PS16], which provides perfect adaption, but is not publicly key-homomorphic.

$\underline{\mathsf{PGen}(1^\kappa)}$: Run $\mathsf{BG} \leftarrow \mathsf{BGGen}(1^\kappa, 3)$ and set $\mathsf{PP} \leftarrow \mathsf{BG}$.

$\underline{\mathsf{KeyGen}(\mathsf{PP})}$: Parse PP as BG, choose $x, y \xleftarrow{R} \mathbb{Z}_p$, compute $\hat{X} \leftarrow \hat{g}^x$, $\hat{Y} \leftarrow \hat{g}^y$ and set $\mathsf{pk} \leftarrow (\mathsf{PP}, \hat{X}, \hat{Y})$, $\mathsf{sk} \leftarrow (\mathsf{pk}, x, y)$, and return $(\mathsf{sk}, \mathsf{pk})$.

$\underline{\mathsf{Sign}(\mathsf{sk}, m)}$: Parse sk as (pk, x, y), choose $h \xleftarrow{R} \mathbb{G}_1^*$ and return $\sigma \leftarrow (h, h^{(x+y \cdot m)})$.

$\underline{\mathsf{Verify}(\mathsf{pk}, m, \sigma)}$: Parse pk as $(\mathsf{PP}, \hat{X}, \hat{Y})$ and σ as (σ_1, σ_2). Check whether $\sigma_1 \neq 1_{\mathbb{G}_1}$ and $e(\sigma_1, \hat{X} \cdot \hat{Y}^m) = e(\sigma_2, \tilde{g})$ holds. If both checks hold return 1 and 0 otherwise.

Scheme 3.8: PS signatures.

Lemma 3.23. *PS signatures are perfectly adaptable according to Definition 3.16.*

Proof. We prove the lemma above by presenting an Adapt algorithm satisfying the perfect adaptability notion.

$\mathsf{Adapt}(\mathsf{pk}, m, \sigma, \Delta)$: Parse pk as $(\mathsf{PP}, \hat{X}, \hat{Y})$, σ as (σ_1, σ_2) and Δ as $(\Delta_1, \Delta_2) \in \mathbb{Z}_p^2$ and choose $r \xleftarrow{R} \mathbb{Z}_p$. Compute $\mathsf{pk}' \leftarrow (\mathsf{PP}, \hat{X} \cdot \hat{g}^{\Delta_1}, \hat{Y} \cdot \hat{g}^{\Delta_2})$ and $\sigma' \leftarrow (\sigma_1^r, (\sigma_2 \cdot \sigma_1^{\Delta_1 + \Delta_2 m})^r)$ and return (pk', σ').

The key $\mathsf{pk}' = (\hat{g}^{x+\Delta_1}, \hat{g}^{y+\Delta_2})$ and $\sigma' = (h^r, (h^r)^{x+\Delta_1+m(y+\Delta_2)})$ output by the Adapt algorithm is identically distributed to a fresh signature under randomness h^r and pk'. □

It is easy to see, that PS signatures are, however, not publicly key-homomorphic as independently generated signatures are computed with respect to different bases h with unknown discrete logarithms. Consequently, there is no efficient means to obtain a succinct representation of $\tilde{\sigma}$ that is suitable for Verify.

CL Signature Variant [CHP12]. While the original pairing-based CL signature scheme [CL04] does not satisfy any of the key-homomorphic properties discussed in this paper, we recall a CL signature variant from [CHP12] in Scheme 3.9 which does.

$\underline{\mathsf{PGen}(1^\kappa)}$: Run $\mathsf{BG} \leftarrow \mathsf{BGGen}(1^\kappa, 1)$, choose some polynomially bound set Ψ and hash functions $H_1 : \Psi \to \mathbb{G}$, $H_2 : \Psi \to \mathbb{G}$, $H_3 : \mathcal{M} \times \Psi \to \mathbb{Z}_p$ uniformly at random from suitable hash function families. Set $\mathsf{PP} \leftarrow (\mathsf{BG}, H_1, H_2, H_3)$.

$\underline{\mathsf{KeyGen}(\mathsf{PP})}$: Parse PP as $(\mathsf{BG}, H_1, H_2, H_3)$, choose $x \xleftarrow{R} \mathbb{Z}_p$ and set $\mathsf{pk} \leftarrow (\mathsf{PP}, g^x)$, $\mathsf{sk} \leftarrow (\mathsf{pk}, x)$, and return $(\mathsf{sk}, \mathsf{pk})$.

$\underline{\mathsf{Sign}(\mathsf{sk}, (m, \psi))}$: If it is the first call to Sign during time period $\psi \in \Psi$, then parse sk as (pk, x), compute $w \leftarrow H_3(m, \psi)$, $a \leftarrow H_1(\psi)$, $b \leftarrow H_2(\psi)$ and return $\sigma \leftarrow a^x b^{xw}$. Otherwise abort.

$\underline{\mathsf{Verify}(\mathsf{pk}, (m, \psi), \sigma)}$: Parse pk as (PP, X) and compute $w \leftarrow H_3(m, \psi)$, $a \leftarrow H_1(\psi)$, $b \leftarrow H_2(\psi)$ and check whether $e(\sigma, g) = e(a, X) \cdot e(b, X)^w$ holds. If so return 1 and 0 otherwise.

Scheme 3.9: CL signature variant.

Lemma 3.24. *Adapted CL signatures are perfectly adaptable according to Definition 3.16.*

Proof. We prove the lemma above by presenting an Adapt algorithm satisfying the perfect adaptability notion.

$\mathsf{Adapt}(\mathsf{pk}, (m, \psi), \sigma, \Delta)$: Parse pk as (PP, X) and compute $w \leftarrow H_3(m, \psi)$, $a \leftarrow H_1(\psi)$, $b \leftarrow H_2(\psi)$. Compute $\mathsf{pk}' \leftarrow (\mathsf{PP}, X \cdot g^\Delta)$ and $\sigma' \leftarrow \sigma \cdot a^\Delta \cdot b^{\Delta \cdot w}$ and return (pk', σ').

It is easy to see that adapted signatures are identical to fresh signatures under $\mathsf{pk}' = (\mathsf{PP}, X \cdot g^\Delta)$. □

Lemma 3.25. *Adapted CL signatures are publicly key-homomorphic according to Definition 3.17.*

Proof. We prove the lemma above by presenting a suitable Combine algorithm.

$\mathsf{Combine}((\mathsf{pk}_i)_{i=1}^n, m, (\sigma_i)_{i=1}^n)$: Let $\mathsf{pk}_i = (\mathsf{PP}, g^{x_i})$. Run $\tilde{\mathsf{pk}} \leftarrow (\mathsf{PP}, \prod_{i=1}^n g^{x_i}$ and $\tilde{\sigma} \leftarrow (\prod_{i=1}^n \sigma_i)$ and return public key $\tilde{\mathsf{pk}}$ and signature $\tilde{\sigma}$. □

3.2.3 Applications

In this section we present first applications of our framework. As already mentioned before, the applications which are closer to the focus of this thesis are presented in separate, subsequent sections.

Multisignatures. A multisignature scheme [IN83] is a signature scheme that allows a group of signers to jointly compute a compact signature for a message. Well known schemes are the BMS [Bol03] and the WMS [LOS+06] schemes that are directly based on the BLS [BLS04] and variants of the Waters' signature scheme [Wat05] respectively. Both of them are secure under the knowledge of secret key (KOSK) assumption, but can be shown to also be secure under (slightly tweaked) real-world proofs of possession protocols [RY07].

Our construction can be seen as a generalization of the paradigm behind all existing multisignature schemes. Making this paradigm explicit eases the search for new schemes, i.e., one can simply check whether a particular signature scheme is publicly key-homomorphic. For instance, as we show in Section 3.2.2, the modified CL signature scheme from [CHP12] provides this key-homomorphism, and, therefore, directly yields a new instantiation of multisignatures.

We now give a formal definition of multisignatures, where we follow Ristenpart and Yilek [RY07]. As already noted before, we use the KOSK modeled via RKey for simplicity. Nevertheless, we stress that we could use any other key-registration that provides extractability or also the extractable key-verification notion by Bagherzandi and Jarecki [BJ08]. This does not make any difference for our subsequent discussion as long as the secret keys are extractable.

Definition 3.18. *A multisignature scheme* MS *is a tuple* (PGen, KeyGen, Sign, Verify) *of PPT algorithms, which are defined as follows:*

$\mathsf{PGen}(1^\kappa)$: *This parameter generation algorithm takes a security parameter* κ *and produces global parameters* PP *(including the security parameters and a description of the message space* $\mathcal{M}$*).*

$\mathsf{KeyGen}(\mathsf{PP})$: *This algorithm takes the global parameters* PP *as input and outputs a secret (signing) key* sk *and a public (verification) key* pk*.*

Sign : *This is an interactive multisignature algorithm executed by a group of signers who intend to sign the same message* m*. Each signer* S_i *executes* Sign *on public inputs* PP*, public key multiset* PK*, message* m *and secret input its secret* sk_i *and outputs a multisignature* σ*.*

$\mathsf{Verify}(\mathsf{PP}, \mathsf{PK}, m, \sigma)$: *This algorithm takes public parameters* PP*, a public key multiset* PK*, a message* m *and a multisignature* σ *as input and outputs a bit* $b \in \{0,1\}$.

The above tuple of algorithms must satisfy correctness, which basically states that $\mathsf{Verify}(\mathsf{PP}, \mathsf{PK}, m, \mathsf{Sign}(\mathsf{PP}, \mathsf{PK}, m, \mathsf{sk})) = 1$ for any m, any honestly generated PP and when every participant correctly follows the algorithms. Besides correctness, we require existential unforgeability under a chosen message attack against a single honest player.

Definition 3.19 (MSEUF-CMA)**.** *A multisignature scheme* MS *is* MSEUF-CMA *secure, if for all PPT adversaries* $\mathcal{A}$ *there is a negligible function* $\varepsilon(\cdot)$ *such that*

$$\Pr\left[\begin{array}{l} \mathsf{PP} \leftarrow \mathsf{PGen}(1^\kappa), \\ (\mathsf{sk}^\star, \mathsf{pk}^\star) \leftarrow \mathsf{KeyGen}(1^\kappa), \\ \mathcal{O} \leftarrow \{\mathsf{Sign}(\cdot,\cdot), \mathsf{RKey}(\cdot,\cdot,\cdot)\}, \\ (\mathsf{PK}^\star, m^\star, \sigma^\star) \leftarrow \mathcal{A}^{\mathcal{O}}(\mathsf{PP}, \mathsf{pk}^\star) \end{array} : \begin{array}{r} \mathsf{Verify}(\mathsf{PP}, \mathsf{PK}^\star, m^\star, \sigma^\star) = 1 \ \wedge \\ \mathsf{pk}^\star \in \mathsf{PK}^\star \ \wedge \ m^\star \notin \mathcal{Q}^{\mathsf{Sign}} \ \wedge \\ (\mathsf{PK}^\star \setminus \{\mathsf{pk}^\star\}) \setminus \mathcal{Q}^{\mathsf{RKey}} = \emptyset \end{array}\right] \leq \varepsilon(\kappa),$$

where the environment keeps track of signing and registration queries via $\mathcal{Q}^{\mathsf{Sign}}$ *and* $\mathcal{Q}^{\mathsf{RKey}}$*, respectively. The adversary has access to the following oracles:*

$\mathsf{Sign}(\mathsf{PK}, m)$: *This oracle obtains a public key set* PK *and returns* $\perp$ *if* $\mathsf{pk}^\star \notin \mathsf{PK}$*. Otherwise it acts on behalf of the honest user in a new instance of* $\mathsf{Sign}(\mathsf{PP}, \mathsf{PK}, m, \mathsf{sk}^\star)$ *forwarding messages to and from* $\mathcal{A}$ *appropriately and sets* $\mathcal{Q}^{\mathsf{Sign}} \xleftarrow{\cup} m$.

$\mathsf{RKey}(\mathsf{sk}, \mathsf{pk})$: *This oracle checks if* $(\mathsf{sk}, \mathsf{pk}) \in \mathsf{KeyGen}(\mathsf{PP})$ *and sets* $\mathcal{Q}^{\mathsf{RKey}} \xleftarrow{\cup} \mathsf{pk}$ *if so.*

Our Construction. We restrict ourselves to non-interactive Sign protocols, which basically means that every signer S_i locally computes a signatures σ_i and then broadcasts it to all other signers in PK. Furthermore, we consider the signature scheme Σ to work with common parameters PP and in Scheme 3.10 let us for the sake of presentation assume that $\mathsf{PK} := (\mathsf{pk}_1, \ldots, \mathsf{pk}_n)$ is an ordered set instead of a multiset.

Theorem 3.4. *If* Σ *is correct,* EUF-CMA *secure, and publicly key-homomorphic, then Scheme 3.10 is* MSEUF-CMA *secure.*

Proof. We show that an efficient adversary $\mathcal{A}$ against MSEUF-CMA can be efficiently turned into an efficient EUF-CMA adversary for Σ. To do so, we simulate the environment for $\mathcal{A}$ by obtaining $\mathsf{pk}^\star$ from an EUF-CMA challenger of Σ, then setting PP accordingly, and starting $\mathcal{A}$ on $(\mathsf{PP}, \mathsf{pk}^\star)$. Additionally, we record the secret keys provided to RKey in a list KEY indexed by the respective public keys, i.e., $\mathtt{KEY}[\mathsf{pk}] \leftarrow \mathsf{sk}$. Whenever a signature with respect to $\mathsf{pk}^\star$ is required we use the Sign oracle provided by the challenger. Eventually, the adversary outputs $(\mathsf{PK}^\star, m^\star, \sigma^\star)$ such that $\Sigma.\mathsf{Verify}(\prod_{\mathsf{pk} \in \mathsf{PK}^\star} \mathsf{pk}, m^\star, \sigma^\star) = 1$, $\mathsf{pk}^\star \in \mathsf{PK}^\star$, all

$\underline{\mathsf{PGen}(1^\kappa)}$: Run $\mathsf{pp} \leftarrow \Sigma.\mathsf{PGen}(1^\kappa)$ and return pp.

$\underline{\mathsf{KeyGen}(\mathsf{pp})}$: Run $(\mathsf{sk}, \mathsf{pk}) \leftarrow \Sigma.\mathsf{KeyGen}(\mathsf{pp})$ and return $(\mathsf{sk}, \mathsf{pk})$.

$\underline{\mathsf{Sign}(\mathsf{pp}, \mathsf{pk}, m, \mathsf{sk})}$: Let $i \in [n]$. Every participating S_i with $\mathsf{pk}_i \in \mathsf{pk}$ proceeds as follows:

- Compute $\sigma_i \leftarrow \Sigma.\mathsf{Sign}(\mathsf{sk}_i, m)$ and broadcast σ_i.
- Receive all signatures σ_j for $j \neq i$.
- Compute $(\mathsf{pk}, \sigma) \leftarrow \mathsf{Combine}(\mathsf{pk}, m, (\sigma_\ell)_{\ell \in [n]})$ and output σ.

$\underline{\mathsf{Verify}(\mathsf{pp}, \mathsf{pk}, m, \sigma)}$: Return 1 if the following holds and 0 otherwise:

$$\Sigma.\mathsf{Verify}\big(\textstyle\prod_{\mathsf{pk} \in \mathsf{pk}} \mathsf{pk}, m, \sigma\big) = 1.$$

Scheme 3.10: Black-box construction of multisignatures.

other keys in $\mathsf{pk}^\star$ were registered, yet $m^\star$ was never queried to the signing oracle. We compute $\mathsf{sk}' \leftarrow \sum_{\mathsf{pk} \in \mathsf{pk}^\star \setminus \{\mathsf{pk}^\star\}} -\mathtt{KEY}[\mathsf{pk}]$, compute $\sigma' \leftarrow \Sigma.\mathsf{Sign}(\mathsf{sk}', m^\star)$, obtain $(\mathsf{pk}^\star, \sigma) \leftarrow \mathsf{Combine}((\prod_{\mathsf{pk} \in \mathsf{pk}^\star} \mathsf{pk}, \prod_{\mathsf{pk} \in \mathsf{pk}^\star \setminus \{\mathsf{pk}^\star\}} \mathsf{pk}^{-1}), m^\star, (\sigma^\star, \sigma'))$ and output $(m^\star, \sigma)$ as a forgery. □

Tight Multi-User Security from Key-Homomorphisms. When using signature schemes in practice, it is often argued that EUF-CMA security does not appropriately capture the requirements appearing in practical settings [GMS02, MS04]. Currently we experience a growing interest in the multi-user setting (e.g., [BJLS16, GHKW16, KMP16]), where an adversary can attack one out of various public keys instead of a single one. This setting is also a frequently discussed topic on the mailing list of the CFRG.[7]

Since many schemes have already been investigated regarding their single-user security, an important question in this context is whether one can infer statements about the multi-user security of a certain scheme based on its single-user security. Without using any further properties of the signature scheme, every naïve reduction looses a factor of N, where N is the number of users in the system [GMS02].[8] Such a reduction is non-tight and drastically reduces the security guarantees a scheme provably provides. Thus, it is important to come up with tight security reductions. This was done in [GMS02], where a tight implication from single-user EUF-CMA to multi-user EUF-CMA for Schnorr signatures was proven. Unfortunately, a flaw in this proof was discovered by Bernstein in [Ber15], where it was also shown that single-user EUF-CMA tightly implies key-prefixed mulit-user EUF-CMA for Schnorr signatures. Recently, Lacharité in

[7] `https://www.ietf.org/mail-archive/web/cfrg/current/maillist.html`

[8] For instance, assuming 2^{30} keys in a system, such a reduction loss requires to significantly increase the parameters.

[Lac16] showed this tight implication under key-prefixing for BLS [BLS04] signatures and BGLS [BGLS03] aggregate signatures. Subsequent to the work in [Ber15], Kiltz et al. [KMP16] studied multi-user security of random self-reducible canonical identification schemes when turned to signatures in the random oracle model using the Fiat-Shamir heuristic. They show that for such schemes single-user security tightly implies multi-user security without key-prefixing. This, in particular, holds for Schnorr signatures.

Our theorem essentially generalizes the work of [Ber15, Lac16] to be applicable to a larger class of signature schemes. For example, using our results from Section 3.2.2, it attests the multi-user EUF-CMA security of various variants of Water's signatures [Wat05], PS signatures [PS16], and the CL signature [CL04] variant from [CHP12], which were previously unknown to provide tight multi-user security. Furthermore, it can be seen as orthogonal to the work of [KMP16], where the requirement of key-prefixing is avoided at the cost of tailoring the results to a class of signature schemes from specific canonical identification schemes in the random oracle model.

Below, we will first recall a definition of multi-user EUF-CMA and then prove Theorem 3.5, which formalizes the main result of this section.

Definition 3.20 (MU-EUF-CMA)**.** *A signature scheme Σ is* MU-EUF-CMA *secure, if for all PPT adversaries $\mathcal{A}$ there is a negligible function $\varepsilon(\cdot)$ such that*

$$\Pr\left[\begin{array}{l|r} \{(\mathsf{sk}_i,\mathsf{pk}_i)\leftarrow\mathsf{KeyGen}(1^\kappa)\}_{i\in[\mathsf{poly}(\kappa)]}, & \\ (i^\star,m^\star,\sigma^\star)\leftarrow\mathcal{A}^{\mathsf{Sign}(\cdot,\cdot)}(\{\mathsf{pk}_i\}_{i\in[\mathsf{poly}(\kappa)]}) & : \\ & \mathsf{Verify}(\mathsf{pk}_{i^\star},m^\star,\sigma^\star)=1\ \wedge \\ & (i^\star,m^\star)\notin\mathcal{Q}^{\mathsf{Sign}} \end{array}\right]\leq\varepsilon(\kappa),$$

where $\mathsf{Sign}(i,m):=\Sigma.\mathsf{Sign}(\mathsf{sk}_i,m)$ *and the environment keeps track of the queries to the signing oracle via* $\mathcal{Q}^{\mathsf{Sign}}$*.*

Theorem 3.5. *Let* $\Sigma=(\mathsf{KeyGen},\mathsf{Sign},\mathsf{Verify})$ *be a signature scheme which provides adaptability of signatures where the success ratio (i.e., the quotient of its success probability and its runtime) of any* EUF-CMA *adversary is ρ. Then the success ratio of any adversary against* MU-EUF-CMA *of* $\Sigma'=(\mathsf{KeyGen}',\mathsf{Sign}',\mathsf{Verify}')$ *is* $\rho'\approx\rho$*, where* $\mathsf{KeyGen}'(1^\kappa):=\mathsf{KeyGen}(1^\kappa)$*,* $\mathsf{Sign}'(\mathsf{sk},m):=\mathsf{Sign}(\mathsf{sk},\mu(\mathsf{sk})||m)$*, and* $\mathsf{Verify}(\mathsf{pk},m,\sigma):=\mathsf{Verify}(\mathsf{pk},\mathsf{pk}||m,\sigma)$*.*

Proof. First, our reduction $\mathcal{R}$ obtains a public key pk_1 from an EUF-CMA challenger $\mathcal{C}$ and initializes an empty list $\mathtt{SK}$. It sets $\mathtt{SK}[1]\leftarrow 0$, and for $2\leq i\leq\mathsf{poly}(\kappa)$, it chooses $\mathtt{SK}[i]\xleftarrow{R}\mathbb{H}$, and sets $\mathsf{pk}_i\leftarrow\mathsf{pk}_1\cdot\mu(\mathtt{SK}[i])$. Then, it starts $\mathcal{A}$ on $\{\mathsf{pk}_i\}_{i\in[\mathsf{poly}(\kappa)]}$ and simulates Sign' inside the $\mathsf{Sign}(\cdot,\cdot)$ oracle as follows (where $\mathcal{C}.\mathsf{Sign}(\cdot)$ denotes the signing oracle provided by $\mathcal{C}$).

$\mathsf{Sign}(i,m)$: Obtain $\sigma\leftarrow\mathcal{C}.\mathsf{Sign}(\mathsf{pk}_i||m)$, compute $(\mathsf{pk}_i,\sigma')\leftarrow\mathsf{Adapt}(\mathsf{pk}_1,\mathsf{pk}_i||m,\sigma,\mathtt{SK}[i])$, and return σ'.

Eventually, $\mathcal{A}$ outputs a forgery $(i^\star, m^\star, \sigma^\star)$, where $(i^\star, m^\star) \notin \mathcal{Q}^{\mathsf{Sign}}$ by definition. Thus, $\mathcal{R}$ has never sent $\mathsf{pk}_{i^\star}||m^\star$ to the sign oracle of $\mathcal{C}$ and can obtain $(\mathsf{pk}_1, \sigma'^\star) \leftarrow \mathsf{Adapt}(\mathsf{pk}_{i^\star}, \mathsf{pk}_{i^\star}||m^\star, \sigma^\star, -\mathtt{SK}[i])$ and output $(\mathsf{pk}_{i^\star}||m^\star, \sigma'^\star)$ as an EUF-CMA forgery. Due to adaptability of signatures the simulation of the oracle is perfect; the running time of $\mathcal{R}$ is approximately the same as the running time of $\mathcal{A}$ which concludes the proof. □

It is quite straight forward to see that such an implication can also be proven for weaker unforgeability notions. Essentially the security proof would be analogous, but without the need to simulate the signing oracle. Furthermore, it is important to note that for key-recovery attacks, where no signatures need to be simulated, a secret key to public key homomorphism would be sufficient to tightly relate the single-user setting to the key-prefixed multi-user setting.

3.3 Universal Designated Verifier Signatures

In designated verifier signatures [JSI96] a signer chooses a designated verifier upon signing a message and, given this signature, only the designated verifier is convinced of its authenticity. The idea behind those constructions is to ensure that the designated verifier can "fake" signatures which are indistinguishable from signatures of the original signer. Universal designated verifier signatures (UDVS) [SBWP03] further extend this concept by introducing an additional party, which performs the designation process by converting a conventional signature to a designated-verifier one. There exists quite a lot of work on UDVS, and, most notably, in [SS08] it was shown how to convert a large class of signature schemes to UDVS. Their approach can be seen as related to our approach, yet they do not rely on key-homomorphisms and they only achieve weaker security guarantees.[9]

While one can interpret designated verifier signatures as a special case of ring signatures where the ring is composed of the public keys of signer and designated verifier (as noted in [RST01, BKM09]), there seems to be no obvious black-box relation turning ring signatures into UDVS. Mainly, since UDVS require the functionality to convert standard signatures to designated verifier ones.[10]

Contribution. Even though we already discussed the contributions presented in this section in the context of our framework for key-homomorphic signatures, we briefly recall the most important aspects here. We present a compiler turning (perfectly) adaptable signature schemes and witness indistinguishable arguments of knowledge into UDVS. Our compiler is extremely simple and can be efficiently

[9] We also note that [SS08] informally mention that their approach is also useful to construct what they call hierarchical ring signatures. However their paradigm is not useful to construct ring signatures as we do in Section 4.1.

[10] We, however, note that an extension of the UDVS model to universal designated verifier *ring* signatures would be straight forward and also our scheme would be straight forwardly extensible using the same techniques as in Scheme 4.1.

instantiated using a wide variety of schemes. This also includes an instantiation using Waters' signatures and Groth-Sahai witness indistinguishable proofs, which constitutes the first instantiation of UDVS in the standard model under standard assumptions.

3.3.1 Formal Security Model

We start by recalling the security model from [SBWP03] including some notational adaptations and a strengthened version of the DV-unforgeability notion which we introduce here.

Definition 3.21. *A universal designated verifier signature scheme* UDVS *extends a conventional signature scheme* $\Sigma = (\mathsf{PGen}, \mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ *by additionally providing the PPT algorithms* $(\mathsf{DVGen}, \mathsf{Desig}, \mathsf{Sim}, \mathsf{DVerify})$*, which are defined as follows.*

$\mathsf{DVGen}(\mathsf{PP})$: *This algorithm takes the public parameters* PP *as input and generates and outputs a designated-verifier key pair* $(\mathsf{vsk}, \mathsf{vpk})$.

$\mathsf{Desig}(\mathsf{pk}, \mathsf{vpk}, m, \sigma)$: *This algorithm takes a signer public key* pk*, a designated-verifier public key* vpk*, a message* m*, and a valid signature* σ *as input, and outputs a designated-verifier signature* δ.

$\mathsf{Sim}(\mathsf{pk}, \mathsf{vsk}, m)$: *This algorithm takes a signer public key* pk*, a designated-verifier secret key* vsk*, and a message* m *as input, and outputs a designated-verifier signature* δ.

$\mathsf{DVerify}(\mathsf{pk}, \mathsf{vsk}, m, \delta)$: *This algorithm takes a signer public key* pk*, a designated-verifier secret key* vsk*, a message* m*, and a designated-verifier signature* δ *as input, and outputs a bit* $b \in \{0, 1\}$.

In the following, we formally recall the security properties, where we omit the obvious correctness notion. For the remaining notions we largely follow [SBWP03, SS08].

DV-unforgeability captures the intuition that it should be infeasible to come up with valid designated verifier signatures where no corresponding original signature exists. We introduce a stronger variant of DV-unforgeability, which we term *simulation-sound DV-unforgeability*. This notion additionally provides the adversary with an oracle to simulate designated-verifier signatures on other messages for the targeted designated verifier. It is easy to see that our notion implies DV-unforgeability in the sense of [SBWP03].

Definition 3.22 (Simulation-Sound DV-Unforgeability)**.** *An* UDVS *provides simulation-sound DV-unforgeability, if for all PPT adversaries* $\mathcal{A}$*, there exists a*

negligible function $\varepsilon(\cdot)$ such that it holds that

$$\Pr\left[\begin{array}{l}\mathsf{PP}\leftarrow\mathsf{PGen}(1^\kappa),\ (\mathsf{sk},\mathsf{pk})\leftarrow\mathsf{KeyGen}(\mathsf{PP}),\\ (\mathsf{vsk},\mathsf{vpk})\leftarrow\mathsf{DVGen}(\mathsf{PP}),\ \mathcal{O}\leftarrow\{\mathsf{Sig}(\mathsf{sk},\cdot),\\ \mathsf{Vrfy}(\mathsf{pk},\mathsf{vsk},\cdot,\cdot),\ \mathsf{S}(\mathsf{pk},\mathsf{vsk},\cdot)\},\ (m^\star,\delta^\star)\leftarrow\mathcal{A}^{\mathcal{O}}(\mathsf{pk},\mathsf{vpk})\end{array}\ :\ \begin{array}{r}\mathsf{DVerify}(\mathsf{pk},\mathsf{vsk},m^\star,\delta^\star)=1\ \wedge\\ m^\star\notin\mathcal{Q}^{\mathsf{Sig}}\ \wedge\ m^\star\notin\mathcal{Q}^{\mathsf{Sim}}\end{array}\right]\leq\varepsilon(\kappa),$$

where $\mathsf{Sig}(\mathsf{sk},m):=\mathsf{Sign}(\mathsf{sk},m)$, $\mathsf{Vrfy}(\mathsf{pk},\mathsf{vsk},m,\delta):=\mathsf{DVerify}(\mathsf{pk},\mathsf{vsk},m,\delta)$, *and* $\mathsf{S}(\mathsf{pk},\mathsf{vsk},m):=\mathsf{Sim}(\mathsf{pk},\mathsf{vsk},m)$. *Furthermore, the environment keeps tracks of the messages queried to* Sig *and* S *via* $\mathcal{Q}^{\mathsf{Sig}}$ *and* $\mathcal{Q}^{\mathsf{Sim}}$, *respectively.*

Non-transferability privacy models the requirement that the designated verifier can simulate signatures which are indistinguishable from honestly designated signatures.

Definition 3.23 (Non-Transferability Privacy)**.** *An* UDVS *provides non-transferability privacy, if for all PPT adversaries $\mathcal{A}$, there exists a negligible function $\varepsilon(\cdot)$ such that it holds that*

$$\Pr\left[\begin{array}{l}\mathsf{PP}\leftarrow\mathsf{PGen}(1^\kappa),\ (\mathsf{sk},\mathsf{pk})\leftarrow\mathsf{KeyGen}(\mathsf{PP}),\\ b\xleftarrow{R}\{0,1\},\ \mathcal{O}\leftarrow\{\mathsf{Sig}(\mathsf{sk},\cdot),\mathsf{RKey}(\cdot,\cdot,\cdot)\},\\ (m^\star,\mathsf{st})\leftarrow\mathcal{A}^{\mathcal{O}}(\mathsf{pk}),\ \sigma\leftarrow\mathsf{Sign}(\mathsf{sk},m^\star),\\ b^\star\leftarrow\mathcal{A}^{\mathcal{O}\cup\{\mathsf{SoD}(\mathsf{pk},\cdot,m^\star,\sigma,b)\}}(\mathsf{st})\end{array}\ :\ \begin{array}{r}b=b^\star\ \wedge\\ m^\star\notin\mathcal{Q}^{\mathsf{Sig}}\end{array}\right]\leq 1/2+\varepsilon(\kappa),$$

where the oracles are defined as follows:

$\mathsf{Sig}(\mathsf{sk},m)$: *This oracle computes* $\sigma\leftarrow\mathsf{Sign}(\mathsf{sk},m)$ *and returns* σ.

$\mathsf{RKey}(i,\mathsf{vsk},\mathsf{vpk})$: *This oracle checks whether* $\mathtt{DVK}[i]\neq\perp$ *and returns* $\perp$ *if so. Otherwise, it checks whether (*vsk, vpk*) is a valid output of* DVGen *and sets* $\mathtt{DVK}[i]\leftarrow(\mathsf{vsk},\mathsf{vpk})$ *if so.*

$\mathsf{SoD}(\mathsf{pk},i,m,\sigma,b)$**:** *This oracle obtains* $(\mathsf{vsk},\mathsf{vpk})\leftarrow\mathtt{DVK}[i]$ *and returns* $\perp$ *if no entry for i exists. Then, if $b=0$, it computes* $\delta\leftarrow\mathsf{Sim}(\mathsf{pk},\mathsf{vsk},m)$, *and, if $b=1$ it computes* $\delta\leftarrow\mathsf{Desig}(\mathsf{pk},\mathsf{vpk},m,\sigma)$. *In the end it returns δ. This oracle can only be called once.*

Further, the environment maintains a list $\mathcal{Q}^{\mathsf{Sig}}$ *keeping track of the* Sig *queries.*

The notion above captures non-transferability privacy in the sense of [SS08]. This notion can be strengthened to what we call *strong non-transferability privacy* which allows multiple calls to SoD (as in [SBWP03]). While non-transferability privacy is often sufficient in practice, we will prove that our construction provides strong non-transferability privacy (clearly implying non-transferability privacy) to obtain the most general result.

3.3.2 Our Construction

In Scheme 3.11, we present our construction of UDVS from any Φ^+-key-homomorphic EUF-CMA secure Σ with perfect adaption of signatures, any witness indistinguishable argument system Π that admits proofs of knowledge, and any one-way function f. Our construction uses the "OR-trick" [JSI96], known from DVS. Upon computing designations and simulations of designated-verifier signatures, we require to prove knowledge of witnesses for the following **NP** relation R:

$$((\mathsf{pk}, \mathsf{vpk}), (\mathsf{sk}, \mathsf{vsk})) \in R \iff \mathsf{pk} = \mu(\mathsf{sk}) \lor \mathsf{vpk} = f(\mathsf{vsk}).$$

For brevity we assume that the parameters pp generated upon setup are implicit in every pk and vpk generated by Gen and DVGen respectively. Furthermore, we assume that R is implicitly defined by the scheme.

$\underline{\mathsf{PGen}(1^\kappa)}$: Run $\mathsf{pp}' \leftarrow \Sigma.\mathsf{PGen}(1^\kappa)$, $\mathsf{crs} \leftarrow \Pi.\mathsf{Setup}(1^\kappa)$, and return $\mathsf{pp} \leftarrow (\mathsf{pp}', \mathsf{crs})$.

$\underline{\mathsf{DVGen}(\mathsf{pp})}$: Run $\mathsf{vsk} \xleftarrow{R} \{0,1\}^\kappa$, set $\mathsf{vpk} \leftarrow f(\mathsf{vsk})$ and return $(\mathsf{vsk}, \mathsf{vpk})$.

$\underline{\mathsf{Desig}(\mathsf{pk}, \mathsf{vpk}, m, \sigma)}$: Output $\delta \leftarrow (\mathsf{pk}', \sigma_\mathsf{R}, \pi)$, where

$$(\mathsf{sk}', \mathsf{pk}') \leftarrow \Sigma.\mathsf{KeyGen}(1^\kappa),\ (\mathsf{pk}_\mathsf{R}, \sigma_\mathsf{R}) \leftarrow \Sigma.\mathsf{Adapt}(\mathsf{pk}, m, \sigma, \mathsf{sk}'),$$

$$\pi \leftarrow \Pi.\mathsf{Proof}(\mathsf{crs}, (\mathsf{pk}', \mathsf{vpk}), (\mathsf{sk}', \bot)).$$

$\underline{\mathsf{Sim}(\mathsf{pk}, \mathsf{vsk}, m)}$: Output $\delta \leftarrow (\mathsf{pk}', \sigma_\mathsf{R}, \pi)$, where

$$(\mathsf{sk}_\mathsf{R}, \mathsf{pk}_\mathsf{R}) \leftarrow \Sigma.\mathsf{KeyGen}(1^\kappa),\ \mathsf{pk}' \leftarrow \mathsf{pk}_\mathsf{R} \cdot \mathsf{pk}^{-1},\ \sigma_\mathsf{R} \leftarrow \Sigma.\mathsf{Sign}(\mathsf{sk}_\mathsf{R}, m),$$

$$\pi \leftarrow \Pi.\mathsf{Proof}(\mathsf{crs}, (\mathsf{pk}', f(\mathsf{vsk})), (\bot, \mathsf{vsk})).$$

$\underline{\mathsf{DVerify}(\mathsf{pk}, \mathsf{vsk}, m, \delta)}$: Parse δ as $(\mathsf{pk}', \sigma_\mathsf{R}, \pi)$ and return 1 if the following holds, and 0 otherwise:

$$\Sigma.\mathsf{Verify}(\mathsf{pk} \cdot \mathsf{pk}', m, \sigma_\mathsf{R}) = 1 \quad \land \quad \Pi.\mathsf{Verify}(\mathsf{crs}, (\mathsf{pk}', f(\mathsf{vsk})), \pi) = 1.$$

Scheme 3.11: Black-box construction of UDVS.

3.3.3 Formal Security Proof

We show that Scheme 3.11 is secure by proving the following theorem.

Theorem 3.6. *If Σ is EUF-CMA secure and perfectly adapts signatures, f is a one-way function, and Π is witness indistinguishable and admits proofs of knowledge, then Scheme 3.11 is correct, simulation-sound DV-unforgeable, and provides strong non-transferability privacy.*

We prove the theorem above by proving the subsequent lemmas, and note that if non-transferability privacy is sufficient, Σ only needs to be adaptable. Then,

besides the candidate schemes presented in Section 3.2.2, one can, e.g., also instantiate Scheme 3.11 with the very efficient Schnorr signature scheme.

Lemma 3.26. *If Σ is correct, and Π is complete, then Scheme 3.11 is correct.*

Lemma 3.26 follows from inspection and the proof is therefore omitted.

Lemma 3.27. *If Σ is $\mathsf{EUF\text{-}CMA}$ secure and adapts signatures, f is a one-way function, and Π is witness indistinguishable and admits proofs of knowledge, then Scheme 3.11 is simulation-sound DV-unforgeable.*

Proof. We bound the success probability of an adversary using a sequence of games, where we let $q_{\mathsf{Sim}} \leq \mathsf{poly}(\kappa)$ be the number Sim queries.

Game 0: The original simulation-sound DV-unforgeability game.

Game 1: As Game 0, but inside the S oracle we execute the following modified Sim algorithm Sim', which additionally takes sk as input.

$\mathsf{Sim}'(\mathsf{pk}, \mathsf{vsk}, m, \boxed{\mathsf{sk}})$: Output $\delta = (\mathsf{pk}', \sigma_{\mathsf{R}}, \pi)$, where

$(\mathsf{sk}_{\mathsf{R}}, \mathsf{pk}_{\mathsf{R}}) \leftarrow \Sigma.\mathsf{KeyGen}(1^\kappa)$, $\mathsf{pk}' \leftarrow \mathsf{pk}_{\mathsf{R}} \cdot \mathsf{pk}^{-1}$, $\sigma_{\mathsf{R}} \leftarrow \Sigma.\mathsf{Sign}(\mathsf{sk}_{\mathsf{R}}, m)$,

$\pi \leftarrow \Pi.\mathsf{Proof}(\mathsf{crs}, (\mathsf{pk}', f(\mathsf{vsk})), \boxed{(\mathsf{sk}_{\mathsf{R}} - \mathsf{sk}, \bot)})$.

Transition - Game 0 → Game 1: A distinguisher between $\mathcal{D}^{0\to 1}$ is a distinguisher for adaptive witness indistinguishability of Π, i.e., $|\Pr[S_0] - \Pr[S_1]| \leq \varepsilon_{\mathsf{wi}}(\kappa)$.

Game 2: As Game 1, but instead of generating crs upon PGen, we obtain $(\mathsf{crs}, \xi) \leftarrow \Pi.\mathsf{E}_1(1^\kappa)$ and store ξ.

Transition - Game 1 → Game 2: A distinguisher between Game 1 and 2 distinguishes an honest crs from an extraction crs, i.e., $|\Pr[S_1] - \Pr[S_2]| \leq \varepsilon_{\mathsf{e1}}(\kappa)$.

Game 3: As Game 2, but whenever the adversary outputs a forgery $(m^\star, \delta^\star)$, where $\delta^\star = (\mathsf{pk}'^\star, \sigma^\star_{\mathsf{R}}, \pi^\star)$ we extract a witness $(\mathsf{sk}'^\star, \mathsf{vsk}'^\star) \leftarrow \Pi.\mathsf{E}_2(\mathsf{crs}, \xi, (\mathsf{pk}'^\star, \mathsf{vpk}^\star), \pi^\star)$ and abort if the extractor fails.

Transition - Game 2 → Game 3: Game 2 and Game 3 proceed identically, unless the extractor fails, i.e., $|\Pr[S_1] - \Pr[S_2]| \leq \varepsilon_{\mathsf{e2}}(\kappa)$.

Game 4: As Game 3, but we further modify Sim' to Sim'' as follows:

$\mathsf{Sim}''(\mathsf{pk}, \mathsf{vsk}, m, \mathsf{sk})$: Output $\delta = (\mathsf{pk}', \sigma_{\mathsf{R}}, \pi)$, where

$\boxed{\sigma \leftarrow \Sigma.\mathsf{Sign}(\mathsf{sk}, m)}$,

$\boxed{(\mathsf{sk}', \mathsf{pk}') \leftarrow \Sigma.\mathsf{KeyGen}(1^\kappa),\ (\mathsf{pk}_{\mathsf{R}}, \sigma_{\mathsf{R}}) \leftarrow \Sigma.\mathsf{Adapt}(\mathsf{pk}, m, \sigma, \mathsf{sk}')}$,

$\pi \leftarrow \Pi.\mathsf{Proof}(\mathsf{crs}, (\mathsf{pk}', f(\mathsf{vsk})), (\boxed{\mathsf{sk}'}, \bot))$.

Transition Game 3 → Game 4: Under adaptability of signatures, this change is conceptual and $\Pr[S_3] = \Pr[S_4]$.

Game 5: As Game 4, but instead of generating $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{Gen}(\mathsf{PP}')$, we obtain pk from an EUF-CMA challenger. Further, whenever a signature under pk is required, we use the Sign oracle provided by the challenger.

Transition - Game 4 → Game 5: This change is conceptual, i.e., $\Pr[S_4] = \Pr[S_5]$.

Game 6: As Game 5, but we obtain vpk from a one-wayness challenger and set $\mathsf{vsk} = \bot$. In addition, we simulate the Vrfy oracle by using vpk instead of $f(\mathsf{vsk})$ inside the $\mathsf{DVerify}$ algorithm.

Transition - Game 5 → Game 6: This change is conceptual, i.e., $\Pr[S_5] = \Pr[S_6]$.

In Game 6, we either have extracted $\mathsf{vsk}^\star$ so that $f(\mathsf{vsk}^\star) = \mathsf{vpk}$ and we can output $\mathsf{vsk}^\star$ to the one-wayness challenger, or we have extracted $\mathsf{sk}'^\star$ such that $\mu(\mathsf{sk}'^\star) = \mathsf{pk}'^\star$ and can obtain $(\mathsf{pk}, \sigma) \leftarrow \Sigma.\mathsf{Adapt}(\mathsf{pk} \cdot \mathsf{pk}'^\star, m^\star, \sigma^\star_{\mathsf{R}}, -\mathsf{sk}'^\star)$ and output $(m^\star, \sigma)$ as a forgery for Σ. Taking the union bound yields $\Pr[S_6] \leq \varepsilon_{\mathsf{f}}(\kappa) + \varepsilon_{\mathsf{ow}}(\kappa)$, and we obtain $\Pr[S_0] \leq \varepsilon_{\mathsf{f}}(\kappa) + \varepsilon_{\mathsf{ow}}(\kappa) + \varepsilon_{\mathsf{wi}}(\kappa) + \varepsilon_{\mathsf{e1}}(\kappa) + \varepsilon_{\mathsf{e2}}(\kappa)$ which is negligible. □

Lemma 3.28. *If Σ perfectly adapts signatures, and Π is witness indistinguishable, then Scheme 3.11 is strongly non-transferable private.*

Proof. We bound the success probability using a sequence of games.

Game 0: The original non-transferability privacy game.

Game 1: As Game 0, but instead of generating crs upon setup, we obtain crs from a witness indistinguishability challenger $\mathcal{C}^{\mathsf{wi}}_\kappa$ upon Setup.

Transition - Game 0 → Game 1: This change is conceptual, i.e., $\Pr[S_0] = \Pr[S_1]$.

Game 2: As Game 1, but inside SoD we execute the following modified the Desig algorithm Desig' which additionally takes vsk as input:

$\mathsf{Desig}'(\mathsf{pk}, \mathsf{vpk}, m, \sigma, \boxed{\mathsf{vsk}})$: Output $\delta \leftarrow (\mathsf{pk}', \sigma_{\mathsf{R}}, \pi)$, where

$(\mathsf{sk}', \mathsf{pk}') \leftarrow \Sigma.\mathsf{KeyGen}(1^\kappa)$, $(\mathsf{pk}_{\mathsf{R}}, \sigma_{\mathsf{R}}) \leftarrow \Sigma.\mathsf{Adapt}(\mathsf{pk}, m, \sigma, \mathsf{sk}')$,

$\pi \leftarrow \Pi.\mathsf{Proof}(\mathsf{crs}, (\mathsf{pk}', \mathsf{vpk}), \boxed{(\bot, \mathsf{vsk})})$.

Transition - Game 1 → Game 2: A distinguisher between $\mathcal{D}^{1\to 2}$ is a distinguisher for adaptive witness indistinguishability of Π, i.e., $|\Pr[S_2] - \Pr[S_1]| \leq \varepsilon_{\mathsf{wi}}(\kappa)$.

Game 3: As Game 2, but we further modify Desig' to Desig'' as follows:

$\mathsf{Desig}''(\mathsf{pk}, \mathsf{vpk}, m, \sigma, \mathsf{vsk})$: Output $\delta \leftarrow (\mathsf{pk}', \sigma_{\mathsf{R}}, \pi)$, where

$$\boxed{(\mathsf{sk}_{\mathsf{R}}, \mathsf{pk}_{\mathsf{R}}) \leftarrow \Sigma.\mathsf{KeyGen}(1^{\kappa}),\ \mathsf{pk}' \leftarrow \mathsf{pk}_{\mathsf{R}} \cdot \mathsf{pk}^{-1},\ \sigma_{\mathsf{R}} \leftarrow \Sigma.\mathsf{Sign}(\mathsf{sk}_{\mathsf{R}}, m)},$$

$$\pi \leftarrow \Pi.\mathsf{Proof}(\mathsf{crs}, (\mathsf{pk}', \mathsf{vpk}), (\bot, \mathsf{vsk})).$$

Transition - Game 2 → Game 3: By the perfect adaption of signatures, this change is conceptual, i.e., $\Pr[S_2] = \Pr[S_3]$.

In Game 3, Desig'' is identical to Sim. This means that SoD is simulated independently of b and $|\Pr[S_3] - \Pr[S_0]| \leq \varepsilon_{\mathsf{wi}}(\kappa)$, which proves the lemma. □

3.4 Simulation Sound Extractable Arguments

The construction of UDVS in the previous sections, as well as the construction of ring signatures in Section 4.1 implicitly use techniques to ensure that even though one needs to simulate proofs within the security reductions, it is still possible to extract the required witness for the forgery. In this section we isolate the essence of this techniques and show that they are generally applicable to extend witness indistinguishable argument systems admitting proofs of knowledge to (weak) simulation sound extractable argument systems using $\mathsf{EUF\text{-}CMA}$ secure signature schemes that adapt signatures. This makes our techniques useful in a broader range of applications.

For the stronger variant of simulation sound extractability we additionally require strong one-time signatures. We start by defining such schemes. Then we proceed in showing that for weak simulation sound extractability, we do not even require strong one-time signatures.

Definition 3.24 (Strong One-Time Signature Scheme)**.** *A strong one-time signature scheme Σ_{ot} provides the same interface as a conventional signature scheme Σ and satisfies the following unforgeability notion: For all PPT adversaries $\mathcal{A}$ there is a negligible function $\varepsilon(\cdot)$ such that*

$$\Pr\left[\begin{array}{l|r} (\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KeyGen}(1^{\kappa}), & \mathsf{Verify}(\mathsf{pk}, m^{\star}, \sigma^{\star}) = 1 \wedge \\ (m^{\star}, \sigma^{\star}) \leftarrow \mathcal{A}^{\mathsf{Sign}(\mathsf{sk}, \cdot)}(\mathsf{pk}) & (m^{\star}, \sigma^{\star}) \notin \mathcal{Q}^{\mathsf{Sign}} \end{array}\right] \leq \varepsilon(\kappa),$$

where the oracle $\mathsf{Sign}(\mathsf{sk}, \mathsf{m}) \coloneqq \Sigma.\mathsf{Sign}(\mathsf{sk}, \mathsf{m})$ can only be called once.

An efficient example of a strong one-time signature scheme can be found in [Gro06].

3.4.1 Our Construction

For our construction, first let L be an arbitrary **NP**-language $L = \{x \mid \exists\ w : R(x, w) = 1\}$, for which we aim to construct a simulation sound extractable argument system, and let L' be defined via the **NP**-relation R':

$$((x, \mathsf{cpk}, \mathsf{pk}), (w, \mathsf{csk} - \mathsf{sk})) \in R' \iff (x, w) \in R \ \vee\ \mathsf{cpk} = \mathsf{pk} \cdot \mu(\mathsf{csk} - \mathsf{sk}).$$

In Scheme 3.12 we present our construction of a simulation sound extractable argument system Π_{sse} for L. Our technique is inspired by [GMY03, GMY06, Gro06] but conceptually simpler. This is mainly due to the fact that the adaptability of the used signature scheme allows us to get rid of the encryption scheme, and, consequently, also the requirement to prove statements about encrypted values.[11]

Essentially, the intuition of our construction is the following. We use a combination of an adaptable EUF-CMA secure signature scheme Σ and a strong one-time signature scheme Σ_{ot} to add the required non-malleability guarantees to the underlying argument system.[12] Upon each proof computation, we use Σ to "certify" the public key of a newly generated key pair of Σ_{ot}. The associated secret key of Σ_{ot} is then used to sign the parts of the proof which must be non-malleable. Adaptability of Σ makes it possible to also use a newly generated key pair of Σ upon each proof computation. In particular, the relation associated to L' is designed so that the second clause in the OR statement is the "shift amount" required to shift such signatures to signatures under a key cpk in the crs. A proof for $x \in L$ is easy to compute when given w such that $(x, w) \in R$. One does not need a satisfying assignment for the second clause in the OR statement, and can thus compute all signatures under newly generated keys. To simulate proofs, however, we can set up crs in a way that we know csk corresponding to cpk, compute the "shift amount" and use it as a satisfying witness for the second clause in the OR statement. Under this strategy, the witness indistinguishability of the underlying argument system for L', the crs indistinguishability provided by the proof of knowledge property, and the secret-key to public-key homomorphism of Σ guarantees the zero-knowledge property of our argument system for L.

What remains is to argue that we can use the extractor of the underlying argument system for L' as an extractor for L in the simulation sound extractability setting. In fact, under the strategy we use, we never have to simulate proofs for statements outside L' which is sufficient for the extractor for L' to work with overwhelming probability. Furthermore, we can show that the probability to extract a valid witness for the second clause in the OR statement is negligible, as this either yields a forgery with respect to Σ_{ot} under some $\mathsf{pk}_{\mathsf{ot}}$ previously obtained from the simulator (if the adversary modified any of the non-malleable parts of a proof previously obtained via the simulator) or for Σ under cpk (if $\mathsf{pk}_{\mathsf{ot}}$ has never been certified). Now we know, however, that the extractor for L' works with overwhelming probability by definition, which means that we will extract a satisfying witness for $x \in L$ with overwhelming probability.

3.4.2 Formal Security Proof

We show that Scheme 3.12 is secure by proving the following theorem.

[11] We however note that the schemes which use encryption can actually achieve a stronger simulation sound extractability notion. For example, in [Gro06] the adversary also receives the extraction trapdoor.

[12] Σ_{ot} is only required as the signatures produced by Σ may be malleable on their own.

$\underline{\mathsf{Setup}(1^\kappa)}$: Run $\mathsf{crs}_\Pi \leftarrow \Pi.\mathsf{Setup}(1^\kappa)$, $(\mathsf{csk}, \mathsf{cpk}) \leftarrow \Sigma.\mathsf{KeyGen}(1^\kappa)$ and return $\mathsf{crs} \leftarrow (\mathsf{crs}_\Pi, \mathsf{cpk})$.

$\underline{\mathsf{Proof}(\mathsf{crs}, x, w)}$: Run $(\mathsf{sk}, \mathsf{pk}) \leftarrow \Sigma.\mathsf{KeyGen}(1^\kappa)$, $(\mathsf{sk}_{\mathsf{ot}}, \mathsf{pk}_{\mathsf{ot}}) \leftarrow \Sigma_{\mathsf{ot}}.\mathsf{KeyGen}(1^\kappa)$, and return $\pi \leftarrow (\pi_\Pi, \mathsf{pk}, \sigma, \mathsf{pk}_{\mathsf{ot}}, \sigma_{\mathsf{ot}})$, where

$$\pi_\Pi \leftarrow \Pi.\mathsf{Proof}(\mathsf{crs}, (x, \mathsf{cpk}, \mathsf{pk}), (w, \perp)),\ \sigma \leftarrow \Sigma.\mathsf{Sign}(\mathsf{sk}, \mathsf{pk}_{\mathsf{ot}}),\ \text{and}$$

$$\sigma_{\mathsf{ot}} \leftarrow \Sigma_{\mathsf{ot}}.\mathsf{Sign}(\mathsf{sk}_{\mathsf{ot}}, \pi_\Pi || x || \mathsf{pk} || \sigma).$$

$\underline{\mathsf{Verify}(\mathsf{crs}, x, \pi)}$: Parse π as $(\pi_\Pi, \mathsf{pk}, \sigma, \mathsf{pk}_{\mathsf{ot}}, \sigma_{\mathsf{ot}})$ and return 1 if the following holds and 0 otherwise:

$$\Pi.\mathsf{Verify}(\mathsf{crs}, (x, \mathsf{cpk}, \mathsf{pk}), \pi_\Pi) = 1 \ \wedge\ \Sigma.\mathsf{Verify}(\mathsf{pk}, \mathsf{pk}_{\mathsf{ot}}) = 1 \ \wedge$$

$$\Sigma_{\mathsf{ot}}.\mathsf{Verify}(\mathsf{pk}_{\mathsf{ot}}, \pi_\Pi || x || \mathsf{pk} || \sigma) = 1.$$

$\underline{\mathsf{S}_1(1^\kappa)}$: Run $(\mathsf{crs}_\Pi, \perp) \leftarrow \Pi.\mathsf{E}_1(1^\kappa)$, $(\mathsf{csk}, \mathsf{cpk}) \leftarrow \Sigma.\mathsf{KeyGen}(1^\kappa)$ and return (crs, τ), where

$$\mathsf{crs} \leftarrow (\mathsf{crs}_\Pi, \mathsf{cpk}) \text{ and } \tau \leftarrow \mathsf{csk}.$$

$\underline{\mathsf{S}_2(\mathsf{crs}, \tau, x)}$: Parse τ as csk, run $(\mathsf{sk}, \mathsf{pk}) \leftarrow \Sigma.\mathsf{KeyGen}(1^\kappa)$, $(\mathsf{sk}_{\mathsf{ot}}, \mathsf{pk}_{\mathsf{ot}}) \leftarrow \Sigma_{\mathsf{ot}}.\mathsf{KeyGen}(1^\kappa)$, and return $\pi \leftarrow (\pi_\Pi, \mathsf{pk}, \sigma, \mathsf{pk}_{\mathsf{ot}}, \sigma_{\mathsf{ot}})$, where

$$\pi_\Pi \leftarrow \Pi.\mathsf{Proof}(\mathsf{crs}, (x, \mathsf{cpk}, \mathsf{pk}), (\perp, \mathsf{csk} - \mathsf{sk})),\ \sigma \leftarrow \Sigma.\mathsf{Sign}(\mathsf{sk}, \mathsf{pk}_{\mathsf{ot}}),\ \text{and}$$

$$\sigma_{\mathsf{ot}} \leftarrow \Sigma_{\mathsf{ot}}.\mathsf{Sign}(\mathsf{sk}_{\mathsf{ot}}, \pi_\Pi || x || \mathsf{pk} || \sigma).$$

$\underline{\mathsf{S}(1^\kappa)}$: Run $(\mathsf{crs}_\Pi, \xi) \leftarrow \Pi.\mathsf{E}_1(1^\kappa)$, $(\mathsf{csk}, \mathsf{cpk}) \leftarrow \Sigma.\mathsf{KeyGen}(1^\kappa)$ and return $(\mathsf{crs}, \tau, \xi)$, where

$$\mathsf{crs} \leftarrow (\mathsf{crs}_\Pi, \mathsf{cpk}) \text{ and } \tau \leftarrow \mathsf{csk}.$$

$\underline{\mathsf{E}(\mathsf{crs}, \xi, x, \pi)}$: Run $(w, \perp) \leftarrow \Pi.\mathsf{E}_2(\mathsf{crs}, \xi, x, \pi)$ and return w.

Scheme 3.12: Simulation sound extractable argument system Π_{sse}.

Theorem 3.7. *Let Π be a complete, witness indistinguishable non-interactive argument system that admits proofs of knowledges for the language L', let Σ be an* EUF-CMA *secure signature scheme that adapts signatures, and let Σ_{ot} be a strong one-time signature scheme, then the argument system Π_{sse} is a complete, simulation sound extractable argument system for language L.*

We show that Theorem 3.7 holds by proving the subsequent lemmas.

Lemma 3.29. *If Π is complete and Σ is correct, Π_{sse} is complete.*

The lemma above follows from inspection and the proof is therefore omitted.

Lemma 3.30. *If Π is witness indistinguishable and admits proofs of knowledge, and Σ provides a secret-key to public-key homomorphism, then Π_{sse} is zero-knowledge.*

Proof. We prove that zero-knowledge follows from witness indistinguishability.

Game 0: The zero-knowledge game, where we use the real $\mathsf{Proof}(\mathsf{crs}, \cdot, \cdot)$ algorithm on witnesses $(w, \bot)$ to reply to queries of the adversary.

Game 1: As Game 0, but we store csk upon Setup.

Transition Game 0 → Game 1: This change is conceptual, i.e., $\Pr[S_0] = \Pr[S_1]$.

Game 2: As Game 1, but use the following modified Proof algorithm Proof' which additionally takes csk as input:

$\mathsf{Proof}'(\mathsf{crs}, x, w, \boxed{\mathsf{csk}})$: Run $(\mathsf{sk}, \mathsf{pk}) \leftarrow \Sigma.\mathsf{KeyGen}(1^\kappa)$, $(\mathsf{sk}_{\mathsf{ot}}, \mathsf{pk}_{\mathsf{ot}}) \leftarrow \Sigma_{\mathsf{ot}}.\mathsf{KeyGen}(1^\kappa)$, and return $\pi \leftarrow (\pi_\Pi, \mathsf{pk}, \sigma, \mathsf{pk}_{\mathsf{ot}}, \sigma_{\mathsf{ot}})$, where

$\pi_\Pi \leftarrow \Pi.\mathsf{Proof}(\mathsf{crs}, (x, \mathsf{cpk}, \mathsf{pk}), \boxed{(\bot, \mathsf{csk} - \mathsf{sk})})$, $\sigma \leftarrow \Sigma.\mathsf{Sign}(\mathsf{sk}, \mathsf{pk}_{\mathsf{ot}})$, and $\sigma_{\mathsf{ot}} \leftarrow \Sigma_{\mathsf{ot}}.\mathsf{Sign}(\mathsf{sk}_{\mathsf{ot}}, \pi_\Pi || x || \mathsf{pk} || \sigma)$.

Transition - Game 1 → Game 2: We present a hybrid game which shows that both games are indistinguishable under the witness indistinguishability of the argument system. First, we conceptually change the Setup algorithm to Setup' which obtains crs_Π from a witness indistinguishability challenger:

$\mathsf{Setup}'(1^\kappa)$: Run $\boxed{\mathsf{crs}_\Pi \leftarrow \mathcal{C}^{\mathsf{wi}}_\kappa}$, $(\mathsf{csk}, \mathsf{cpk}) \leftarrow \Sigma.\mathsf{KeyGen}(1^\kappa)$, return $\mathsf{crs} \leftarrow (\mathsf{crs}_\Pi, \mathsf{cpk})$.

The change above is only conceptual. Furthermore, we use the following Proof'' algorithm instead of Proof':

$\mathsf{Proof}''(\mathsf{crs}, x, w, \boxed{\mathsf{csk}})$: Run $(\mathsf{sk}, \mathsf{pk}) \leftarrow \Sigma.\mathsf{KeyGen}(1^\kappa)$, $(\mathsf{sk}_{\mathsf{ot}}, \mathsf{pk}_{\mathsf{ot}}) \leftarrow \Sigma_{\mathsf{ot}}.\mathsf{KeyGen}(1^\kappa)$, and return $\pi \leftarrow (\pi_\Pi, \mathsf{pk}, \sigma, \mathsf{pk}_{\mathsf{ot}}, \sigma_{\mathsf{ot}})$, where

$\boxed{\pi_\Pi \leftarrow \mathcal{C}^{\mathsf{wi}}_\kappa((x, \mathsf{cpk}, \mathsf{pk}), (w, \bot), (\bot, \mathsf{csk} - \mathsf{sk}))}$, $\sigma \leftarrow \Sigma.\mathsf{Sign}(\mathsf{sk}, \mathsf{pk}_{\mathsf{ot}})$, and $\sigma_{\mathsf{ot}} \leftarrow \Sigma_{\mathsf{ot}}.\mathsf{Sign}(\mathsf{sk}_{\mathsf{ot}}, \pi_\Pi || x || \mathsf{pk} || \sigma)$.

Now depending of whether the challenger uses the first witness ($b = 0$) or the second witness ($b = 1$) we either simulate Game 1 or Game 2. More precisely, Proof'' produces the identical distribution as Proof if $b = 0$ and the identical distribution to Proof' if $b = 1$. That is $|\Pr[S_1] - \Pr[S_2]| \leq \varepsilon_{\mathsf{wi}}(\kappa)$.

Game 3: As Game 2, but we further modify Proof' to Proof''' so that it no longer takes w as input:

$\mathsf{Proof}'''(\mathsf{crs}, x, \mathsf{csk})$: Run $(\mathsf{sk}, \mathsf{pk}) \leftarrow \Sigma.\mathsf{KeyGen}(1^\kappa)$, $(\mathsf{sk}_{\mathsf{ot}}, \mathsf{pk}_{\mathsf{ot}}) \leftarrow \Sigma_{\mathsf{ot}}.\mathsf{KeyGen}(1^\kappa)$, and return $\pi \leftarrow (\pi_\Pi, \mathsf{pk}, \sigma, \mathsf{pk}_{\mathsf{ot}}, \sigma_{\mathsf{ot}})$, where

$\pi_\Pi \leftarrow \Pi.\mathsf{Proof}(\mathsf{crs}, (x, \mathsf{cpk}, \mathsf{pk}), (\bot, \mathsf{csk} - \mathsf{sk}))$, $\sigma \leftarrow \Sigma.\mathsf{Sign}(\mathsf{sk}, \mathsf{pk}_{\mathsf{ot}})$, and $\sigma_{\mathsf{ot}} \leftarrow \Sigma_{\mathsf{ot}}.\mathsf{Sign}(\mathsf{sk}_{\mathsf{ot}}, \pi_\Pi || x || \mathsf{pk} || \sigma)$.

Transition - Game 2 → Game 3: This change is conceptual, i.e., $\Pr[S_2] = \Pr[S_3]$.

Game 4: As Game 3, but instead of obtaining crs using Setup, we obtain $(\mathsf{crs}, \tau) \leftarrow \mathsf{S}_1$ (observe that $\tau = \mathsf{csk}$, so we still know csk). Now the setup is already as in the second distribution of the zero-knowledge game.

Transition - Game 3 → Game 4: A crs output by S_1 is indistinguishable from an honest crs under the CRS indistinguishability provided by the proof of knowledge property (observe that S_1 internally uses E_1 to obtain crs). Thus, $|\Pr[S_3] - \Pr[S_4]| \leq \varepsilon_{\mathsf{pok1}}(\kappa)$.

Game 5: As Game 4, but we further modify Proof''' to Proof'''' as follows:

$\mathsf{Proof}''''(\mathsf{crs}, \boxed{\tau}, x)$: $\boxed{\text{Parse } \tau \text{ as } \mathsf{csk}}$. Run $(\mathsf{sk}, \mathsf{pk}) \leftarrow \Sigma.\mathsf{KeyGen}(1^\kappa)$, $(\mathsf{sk}_{\mathsf{ot}}, \mathsf{pk}_{\mathsf{ot}}) \leftarrow \Sigma_{\mathsf{ot}}.\mathsf{KeyGen}(1^\kappa)$, and return $\pi \leftarrow (\pi_\Pi, \mathsf{pk}, \sigma, \mathsf{pk}_{\mathsf{ot}}, \sigma_{\mathsf{ot}})$, where

$\pi_\Pi \leftarrow \Pi.\mathsf{Proof}(\mathsf{crs}, (x, \mathsf{cpk}, \mathsf{pk}), (\bot, \mathsf{csk} - \mathsf{sk}))$, $\sigma \leftarrow \Sigma.\mathsf{Sign}(\mathsf{sk}, \mathsf{pk}_{\mathsf{ot}})$, and $\sigma_{\mathsf{ot}} \leftarrow \Sigma_{\mathsf{ot}}.\mathsf{Sign}(\mathsf{sk}_{\mathsf{ot}}, \pi_\Pi || x || \mathsf{pk} || \sigma)$.

Now Proof'''' is equivalent to S_2.

Transition - Game 4 → Game 5: This change is conceptual, i.e., $\Pr[S_4] = \Pr[S_5]$.

In Game 0 we simulate the first distribution of the zero-knowledge game whereas in Game 5 we simulate the second distribution. We have that $|\Pr[S_0] - \Pr[S_5]| \leq \varepsilon_{\mathsf{wi}}(\kappa) + \varepsilon_{\mathsf{pok1}}(\kappa)$ which concludes the proof. □

Now, we have already established the existence of a simulator by proving zero-knowledge and can go on by proving simulation sound extractability.

Lemma 3.31. *If Π is witness indistinguishable and admits proof of knowledge and Σ is* EUF-CMA *secure and adapts signatures, then Π_{sse} is simulation sound extractable.*

Proof. We show that even when the adversary sees simulated proofs for arbitrary statements, we are still able to extract a witness w from a proof $\pi^\star$ for a statement $x^\star$ so that $R(x^\star, w) = 1$ as long as $(x^\star, \pi^\star)$ does not correspond to a query-answer pair of the simulation oracle. By Lemma 3.30, we know that $(\mathsf{S}_1, \mathsf{S}_2)$ is a suitable zero-knowledge simulator. In addition, we observe that the output of S is identical to S_1 when restricted to (crs, τ). This completes CRS indistinguishability part of the proof. To prove the second part of simulation sound extractability we proceed using a sequence of games where we let $q \leq \mathsf{poly}(\kappa)$ be the number of queries to the simulator.

Game 0: The original simulation sound extractability game.

Game 1: As Game 0, but we engage with an EUF-CMA challenger within S. That is, we execute the following modified S algorithm S':

$\mathsf{S}'(1^\kappa)$: Run $(\mathsf{crs}_\Pi, \xi) \leftarrow \Pi.\mathsf{E}_1(1^\kappa)$, $\boxed{\mathsf{cpk} \leftarrow \mathcal{C}^{\mathsf{f}}_\kappa}$, and return $(\mathsf{crs}, \tau, \xi)$, where

$$\mathsf{crs} \leftarrow (\mathsf{crs}_\Pi, \mathsf{cpk}) \text{ and } \tau \leftarrow \bot.$$

This also requires us to modify the S_2 algorithm used for simulation to obtain S'_2. Essentially, we leverage the adaptability of signatures to shift signatures obtained from the signing oracle provided by the EUF-CMA challenger under cpk to signatures under a random key. The "shift-amount" is then a valid witness for the relation.

$\mathsf{S}'_2(\mathsf{crs}, x)$: Obtain $\boxed{\mathsf{sk}' \xleftarrow{R} \mathbb{H},\ \mathsf{pk} \leftarrow \mathsf{cpk} \cdot \mu(\mathsf{sk}')}$. Further, run $(\mathsf{sk}_{\mathsf{ot}}, \mathsf{pk}_{\mathsf{ot}}) \leftarrow \Sigma_{\mathsf{ot}}.\mathsf{KeyGen}(1^\kappa)$, and return $\pi \leftarrow (\pi_\Pi, \mathsf{pk}, \sigma, \mathsf{pk}_{\mathsf{ot}}, \sigma_{\mathsf{ot}})$, where

$$\pi_\Pi \leftarrow \Pi.\mathsf{Proof}(\mathsf{crs}, (x, \mathsf{cpk}, \mathsf{pk}), (\bot, \boxed{-\mathsf{sk}'})),\ \boxed{\sigma' \leftarrow \mathcal{C}^{\mathsf{f}}_\kappa.\mathsf{Sign}(\mathsf{pk}_{\mathsf{ot}})},$$
$$\boxed{(\sigma, \bot) \leftarrow \Sigma.\mathsf{Adapt}(\mathsf{cpk}, \mathsf{pk}_{\mathsf{ot}}, \sigma', \mathsf{sk}')}, \text{ and}$$
$$\sigma_{\mathsf{ot}} \leftarrow \Sigma_{\mathsf{ot}}.\mathsf{Sign}(\mathsf{sk}_{\mathsf{ot}}, \pi_\Pi || x || \mathsf{pk} || \sigma).$$

Transition - Game 0 → Game 1: Under adaptability of signatures, this change is only conceptual, i.e., $\Pr[S_0] = \Pr[S_1]$.

Game 2: As Game 1, but we further modify S'_2 to S''_2 as follows: we engage with a strong one-time signature challenger in each call and keep a mapping from challengers to keys.

$\mathsf{S}''_2(\mathsf{crs}, x)$: Obtain $\mathsf{sk}' \xleftarrow{R} \mathbb{H},\ \mathsf{pk} \leftarrow \mathsf{cpk} \cdot \mu(\mathsf{sk}')$. Further, obtain $\boxed{\mathsf{pk}_{\mathsf{ot}} \leftarrow \mathcal{C}^{\mathsf{ot}}_\kappa}$ and return $\pi \leftarrow (\pi_\Pi, \mathsf{pk}, \sigma, \mathsf{pk}_{\mathsf{ot}}, \sigma_{\mathsf{ot}})$, where

$$\pi_\Pi \leftarrow \Pi.\mathsf{Proof}(\mathsf{crs}, (x, \mathsf{cpk}, \mathsf{pk}), (\bot, -\mathsf{sk}')),\ \sigma' \leftarrow \mathcal{C}^{\mathsf{f}}_\kappa.\mathsf{Sign}(\mathsf{pk}_{\mathsf{ot}}),$$
$$(\sigma, \bot) \leftarrow \Sigma.\mathsf{Adapt}(\mathsf{cpk}, \mathsf{pk}_{\mathsf{ot}}, \sigma', \mathsf{sk}'), \text{ and}$$
$$\boxed{\sigma_{\mathsf{ot}} \leftarrow \mathcal{C}^{\mathsf{ot}}_\kappa.\mathsf{Sign}(\pi_\Pi || x || \mathsf{pk} || \sigma)}.$$

Transition - Game 1 → Game 2: This change is conceptual, i.e., $\Pr[S_1] = \Pr[S_2]$.

Game 3: As Game 2, but we assume that E_2 used inside E does not fail to extract a valid witness with respect to L' (i.e., we abort if it fails).

Transition - Game 2 → Game 3: We bound the probability that the adversary outputs a tuple $(x^\star, \pi^\star)$ in Game 3 so that E_2 fails. We refer to this event as F_1. For the sake of contradiction assume that $\Pr[F_1]$ is non-negligible. Then we

could obtain crs_Π from a proof of knowledge challenger and set $\xi \leftarrow \bot$ within S_1'. Whenever the adversary outputs $(x^\star, \pi^\star)$ we output it to the challenger. Now, the probability for our reduction to win the proof of knowledge game is exactly $\Pr[F_1]$. That is, we have that $|\Pr[S_2] - \Pr[S_3]| \leq \varepsilon_{\mathsf{e2}}(\kappa)$.

Game 4: As Game 3, but we assume that for every tuple $(x^\star, \pi^\star)$ output by the adversary, E never fails to output a witness w so that $R(x, w) = 1$ (i.e., we abort if it fails).

Transition Game 3 → Game 4: We bound the probability that the adversary manages to come up with a tuple $(x^\star, \pi^\star)$, where $\pi^\star = (\pi^\star_\Pi, \mathsf{pk}^\star, \sigma^\star, \mathsf{pk}^\star_{\mathsf{ot}}, \sigma^\star_{\mathsf{ot}})$, so that we extract $(\bot, \mathsf{sk}_\mathsf{e}) \leftarrow \mathsf{E}_2(\mathsf{crs}, \xi, x^\star, \pi^\star)$ inside E. We refer to this event as F_2. If F_2 happens, we obtain a signature $(\sigma_\mathsf{f}, \mathsf{cpk}) \leftarrow \Sigma.\mathsf{Adapt}(\mathsf{pk}^\star, \mathsf{pk}^\star_{\mathsf{ot}}, \sigma^\star, \mathsf{sk}_\mathsf{e})$. By definition of the game we know that $(x^\star, \pi^\star)$ is not a query-answer pair of the simulator. Thus, we have two cases: (1) A signature on $\mathsf{pk}^\star_{\mathsf{ot}}$ was never obtained from the $\mathsf{EUF\text{-}CMA}$ challenger and we can output $(\mathsf{pk}^\star_{\mathsf{ot}}, \sigma_\mathsf{f})$ as a valid $\mathsf{EUF\text{-}CMA}$ forgery. (2) A signature on $\mathsf{pk}^\star_{\mathsf{ot}}$ was previously obtained. Then we have by definition that either $\pi^\star_\Pi||x^\star||\mathsf{pk}^\star||\sigma^\star$ or $\sigma^\star_{\mathsf{ot}}$ is different from the tuple signed by the strong one-time signature challenger upon simulation and we can output $(\pi^\star_\Pi||x^\star||\mathsf{pk}^\star||\sigma^\star, \sigma^\star_{\mathsf{ot}})$ as a forgery for the strong one-time signature scheme to the respective challenger. Taking the union bound yields $|\Pr[S_3] - \Pr[S_4]| \leq q \cdot \varepsilon_{\mathsf{ot}}(\kappa) + \varepsilon_\mathsf{f}(\kappa)$.

In Game 4, we always extract a witness w such that $R(x^\star, w) = 1$, i.e., $\Pr[S_4] = 0$; Game 0 and Game 4 are computationally indistinguishable. Overall, we obtain $\Pr[S_0] \leq q \cdot \varepsilon_{\mathsf{ot}}(\kappa) + \varepsilon_\mathsf{f}(\kappa) + \varepsilon_{\mathsf{e2}}(\kappa)$, which completes the proof. □

3.4.3 Weak Simulation Sound Extractability

If one allows the proofs to be malleable and only requires non-malleability with respect to the statements one can omit the strong one-time signature scheme and directly sign $\pi_\Pi||x||\mathsf{pk}$ using Σ. We refer to this modified argument system as Π_{wsse}.

Theorem 3.8. *Let Π be a complete, witness indistinguishable non-interactive argument system that admits proofs of knowledges for the language L', and let Σ be an $\mathsf{EUF\text{-}CMA}$ secure signature scheme that adapts signatures, then the argument system Π_{wsse} is a complete, weakly simulation sound extractable argument system for language L.*

Proof (Sketch). The proof is exactly the same as the one for simulation sound extractability above, except that we do not need to engage with challengers for the one-time signature scheme (i.e., in Game 2 nothing is changed) and $\Pr[F_2]$ is exactly the same as extracting a forgery for Σ in the transition between Game 3 and Game 4. □

3.4.4 Signatures of Knowledge

Also note that using our techniques in a non-black-box way directly yields signatures of knowledge [CL06]. That is, a signature of knowledge on a message m with respect to statement x is simply a proof with respect to x, where m is additionally included upon computing the signature using Σ_{ot}, i.e., one signs $\pi_{\Pi}||x||\mathsf{pk}||\sigma||m$. Then one obtains signatures of knowledge in the strong sense [BCC+15], where even the signature (i.e., the proof) is non-malleable. If security in the original sense—the counterpart of weak simulation-sound extractability where the signature (i.e., the proof π) itself may be malleable—is sufficient, one can even omit the strong one-time signature scheme and directly sign $\pi_{\Pi}||x||\mathsf{pk}||m$ using Σ.

As already mentioned in [CL06], a straight forward application of signatures of knowledge is the construction of ring signatures. Obtaining such a construction based on our techniques presented in this section can thus be seen as an alternative to the direct construction in Section 4.1.

3.4.5 Discussion

Our technique provides nice properties when it comes to converting Groth-Sahai proofs [GS08] over pairing product equations to simulation-sound extractable arguments of knowledge. We can use Waters' signatures as described in Section 3.2.2. Here the secret keys as well as the shift amounts are group elements and the required relations can be proven using a few simple pairing product equations. Thus our technique constitutes an alternative to known techniques and yields conceptually simpler constructions with favorable properties regarding efficiency when, e.g., compared to [Gro06] for applications of SSE NIZKs where the adversary is not required to get the extraction trapdoor, or compared to [BFG13] for signatures of knowledge without random oracles. We also note that achieving SSE NIZK where the adversary also gets the extraction trapdoor seems to crucially require to use encryption. Nevertheless, our techniques might still be useful to simplify the statement which needs to be proven in constructions satisfying the stronger notion, i.e., the statement can be with respect to the encrypted shift amount instead of an encrypted signature.

4

Signatures with Signer Privacy

In this section we present our results on signature schemes which address privacy issues with respect to the signer. We start by presenting a ring signature scheme, which builds upon our results on key-homomorphic signatures, in Section 4.1. After that, in Section 4.2, we present the most efficient dynamic group signature scheme known to date, which provides security in the strong security model by Bellare et al. [BSZ05].

4.1 Ring Signatures

Ring signature schemes [RST01] allow a member of an ad-hoc group $\mathcal{R}$ (the so called ring), defined by the member's public verification keys, to anonymously sign a message on behalf of $\mathcal{R}$. Given a ring signature and all public keys for $\mathcal{R}$, one can verify the validity of such a signature with respect to $\mathcal{R}$, but it is infeasible to identify the actual signer. Due to this anonymity feature ring signatures have proven to be an interesting tool for numerous applications, most notable for whistleblowing. The two main lines of work in the design of ring signatures target reducing the signature size or removing the requirement for random oracles (e.g., [DKNS04, CGS07, GK15]).

Contribution. As already briefly mentioned in Section 3.2, we present another application of our framework for key-homomorphic signatures in this section. In particular, we propose a generic compiler turning any EUF-CMA secure signature scheme which adapts signatures together with a suitable witness indistinguishable non-interactive proof system into a ring signature scheme. Our compiler yields ring signatures with linear signature size. It provides an alternative very

simple generic framework to construct ring signatures in addition to existing ones (cf. [BKM09, BK10]). Using our results from Section 3.2, we obtain various novel instantiations. Most notably, using Waters' signatures and witness indistinguishable Groth-Sahai proofs we obtain an efficient instantiation which does not require random oracles.

4.1.1 Formal Security Model

We now formally define ring signature schemes (adopting [BKM09]) and note that the model implicitly assumes knowledge of secret keys [RY07] as discussed in Section 3.2.

Definition 4.1. *A ring signature scheme* RiS *is a tuple* $\mathsf{RiS} = (\mathsf{Setup}, \mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$ *of PPT algorithms, which are defined as follows.*

$\mathsf{Setup}(1^\kappa)$: *This algorithm takes as input a security parameter* κ *and outputs public parameters* PP.

$\mathsf{Gen}(\mathsf{PP})$: *This algorithm takes as input the public parameters* PP *and outputs a key pair* $(\mathsf{sk}, \mathsf{pk})$.

$\mathsf{Sign}(\mathsf{PP}, \mathsf{sk}_i, m, \mathcal{R})$: *This algorithm takes as input the public parameters* PP, *a secret key* sk_i, *a message* $m \in \mathcal{M}$ *and a ring* $\mathcal{R} = (\mathsf{pk}_j)_{j\in[n]}$ *of* n *public keys such that* $\mathsf{pk}_i \in \mathcal{R}$. *It outputs a signature* σ.

$\mathsf{Verify}(\mathsf{PP}, m, \sigma, \mathcal{R})$: *This algorithm takes as input the public parameters* PP, *a message* $m \in \mathcal{M}$, *a signature* σ *and a ring* $\mathcal{R}$. *It outputs a bit* $b \in \{0, 1\}$.

A secure ring signature scheme needs to be correct, unforgeable, and anonymous. While we omit the obvious correctness definition, we provide formal definitions for the remaining properties following [BKM09]. We note that Bender et al. in [BKM09] have formalized multiple variants of these properties, where we always use the strongest one.

Unforgeability requires that without any secret key sk_i that corresponds to a public key $\mathsf{pk}_i \in \mathcal{R}$, it is infeasible to produce valid signatures with respect to arbitrary such rings $\mathcal{R}$.

Definition 4.2 (Unforgeability)**.** *A ring signature scheme provides unforgeability, if for all PPT adversaries* $\mathcal{A}$, *there exists a negligible function* $\varepsilon(\cdot)$ *such that it holds that*

$$\Pr\left[\begin{array}{l|rl} \{(\mathsf{sk},\mathsf{pk}) \leftarrow \mathsf{Gen}(1^\kappa)\}_{i\in[\mathsf{poly}(\kappa)]}, & \mathsf{Verify}(m^\star,\sigma^\star,\mathcal{R}^\star)=1 \ \wedge \\ \mathcal{O} \leftarrow \{\mathsf{Sig}(\cdot,\cdot,\cdot),\mathsf{Key}(\cdot)\}, & (\cdot,m^\star,\mathcal{R}^\star) \notin \mathcal{Q}^{\mathsf{Sign}} \ \wedge \\ (m^\star,\sigma^\star,\mathcal{R}^\star) \leftarrow \mathcal{A}^{\mathcal{O}}(\{\mathsf{pk}_i\}_{i\in[\mathsf{poly}(\kappa)]}) & \mathcal{R}^\star \subseteq \{\mathsf{pk}_i\}_{i\in[\mathsf{poly}(\kappa)]\setminus\mathcal{Q}^{\mathsf{Key}}} \end{array}\right] \leq \varepsilon(\kappa),$$

where $\mathsf{Sig}(i, m, \mathcal{R}) := \mathsf{Sign}(\mathsf{sk}_i, m, \mathcal{R})$, Sig *returns* $\perp$ *if* $\mathsf{pk}_i \notin \mathcal{R} \ \vee \ i \notin [\mathsf{poly}(\kappa)]$, *and* $\mathcal{Q}^{\mathsf{Sig}}$ *records the queries to* Sig. *Furthermore,* $\mathsf{Key}(i)$ *returns* sk_i *and* $\mathcal{Q}^{\mathsf{Key}}$ *records the queries to* Key.

Anonymity requires that it is infeasible to tell which ring member produced a certain signature as long as there are at least two honest members in the ring.

Definition 4.3 (Anonymity)**.** *A ring signature scheme provides anonymity, if for all PPT adversaries $\mathcal{A}$ there exists a negligible function $\varepsilon(\cdot)$ such that it holds that*

$$\Pr\left[\begin{array}{l}\{(\mathsf{sk}_i,\mathsf{pk}_i)\leftarrow\mathsf{Gen}(1^\kappa)\}_{i\in[\mathsf{poly}(\kappa)]},\\ b\xleftarrow{R}\{0,1\},\ \mathcal{O}\leftarrow\{\mathsf{Sig}(\cdot,\cdot,\cdot)\},\\ (m,j_0,j_1,\mathcal{R},\mathsf{st})\leftarrow\mathcal{A}^{\mathcal{O}}(\{\mathsf{pk}_i\}_{i\in[\mathsf{poly}(\kappa)]}),\\ \sigma\leftarrow\mathsf{Sign}(\mathsf{sk}_{j_b},m,\mathcal{R}),\ b^\star\leftarrow\mathcal{A}^{\mathcal{O}}(\mathsf{st},\sigma,\{\mathsf{sk}_i\}_{i\in[\mathsf{poly}(\kappa)]})\end{array}\ :\ \begin{array}{r}b=b^\star\ \wedge\\ \{\mathsf{pk}_{j_0},\mathsf{pk}_{j_1}\}\subseteq\mathcal{R}\end{array}\right]\leq 1/2+\varepsilon(\kappa),$$

where $\mathsf{Sig}(i,m,\mathcal{R}):=\mathsf{Sign}(\mathsf{sk}_i,m,\mathcal{R})$.

4.1.2 Our Construction

In Scheme 4.1 we present our black-box construction of ring signatures from any Φ^+-key-homomorphic EUF-CMA secure signature scheme Σ with adaptable signatures and any witness indistinguishable argument system Π that admits proofs of knowledge. The idea behind the scheme is as follows. A ring signature for message m with respect to ring $\mathcal{R}$ consists of a signature for $m||\mathcal{R}$ using Σ with a randomly generated key pair together with a proof of knowledge attesting the knowledge of the "shift amount" from the random public key to (at least) one of the public keys in $\mathcal{R}$.[1] Very briefly, unforgeability then holds because—given a valid ring signature—one can always extract a valid signature of one of the ring members. Anonymity holds because the witness indistinguishability of the argument system guarantees that signatures of different ring members are indistinguishable.

Upon signing, we need to prove knowledge of a witness for the following **NP** relation R.

$$((\mathsf{pk},\mathsf{cpk},\mathcal{R}),\mathsf{sk}')\in R\iff\exists\ \mathsf{pk}_i\in\mathcal{R}\cup\{\mathsf{cpk}\}\ :\ \mathsf{pk}_i=\mathsf{pk}\cdot\mu(\mathsf{sk}')$$

For the sake of compactness, we assume that the relation is implicitly defined by the scheme. One can obtain a straight forward instantiation by means of disjunctive proofs of knowledge [CDS94] (similar as it is done in many known constructions). Therefore one could use the following **NP** relation R.

$$((\mathsf{pk},\mathsf{cpk},\mathcal{R}),\mathsf{sk}')\in R\iff\left(\vee_{\mathsf{pk}_i\in\mathcal{R}}\ \mathsf{pk}_i=\mathsf{pk}\cdot\mu(\mathsf{sk}')\right)\ \vee\ \mathsf{cpk}=\mathsf{pk}\cdot\mu(\mathsf{sk}')$$

Using this approach, however, yields signatures of linear size. To reduce the signature size, one could, e.g., follow the approach of [DKNS04].

[1] For technical reasons we need an additional public key cpk in the public parameters.

$\underline{\mathsf{Setup}(1^\kappa)}$: Run $\mathsf{crs} \leftarrow \Pi.\mathsf{Setup}(1^\kappa)$, $(\mathsf{csk}, \mathsf{cpk}) \leftarrow \mathsf{KeyGen}(1^\kappa)$, set $\mathsf{PP} \leftarrow (1^\kappa, \mathsf{crs}, \mathsf{cpk})$ and return PP.

$\underline{\mathsf{Gen}(\mathsf{PP})}$: Run $(\mathsf{sk}_i, \mathsf{pk}_i) \leftarrow \Sigma.\mathsf{KeyGen}(1^\kappa)$ and return $(\mathsf{sk}_i, \mathsf{pk}_i)$.

$\underline{\mathsf{Sign}(\mathsf{PP}, \mathsf{sk}_i, m, \mathcal{R})}$: Parse PP as $(1^\kappa, \mathsf{crs}, \mathsf{cpk})$ and return $\perp$ if $\mu(\mathsf{sk}_i) \notin \mathcal{R}$. Otherwise, return $\sigma \leftarrow (\delta, \mathsf{pk}, \pi)$, where

$$(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KeyGen}(1^\kappa),\ \delta \leftarrow \Sigma.\mathsf{Sign}(\mathsf{sk}, m||\mathcal{R}),\ \text{and}$$
$$\pi \leftarrow \Pi.\mathsf{Proof}(\mathsf{crs}, (\mathsf{pk}, \mathsf{cpk}, \mathcal{R}), (\mathsf{sk}_i - \mathsf{sk})).$$

$\underline{\mathsf{Verify}(\mathsf{PP}, m, \sigma, \mathcal{R})}$: Parse PP as $(1^\kappa, \mathsf{crs}, \mathsf{cpk})$ and σ as $(\delta, \mathsf{pk}, \pi)$ and return 1 if the following holds, and 0 otherwise:

$$\Sigma.\mathsf{Verify}(\mathsf{pk}, m||\mathcal{R}, \delta) = 1 \quad \wedge \quad \Pi.\mathsf{Verify}(\mathsf{crs}, (\mathsf{pk}, \mathsf{cpk}, \mathcal{R}), \pi) = 1.$$

Scheme 4.1: Black-Box construction of ring signatures.

Theorem 4.1. *If Σ is correct, EUF-CMA secure, and provides adaptability of signatures, Π is complete and witness indistinguishable and admits proofs of knowledge, then Scheme 4.1 is correct, unforgeable, and anonymous.*

We show that Theorem 4.1 holds by proving the subsequent lemmas.

Lemma 4.1. *If Σ is correct, and Π is complete, then Scheme 4.1 is correct.*

Lemma 4.1 follows from inspection and the proof is therefore omitted.

Lemma 4.2. *If Σ is EUF-CMA secure, and provides adaptability of signatures, and Π is witness indistinguishable, then Scheme 4.1 is unforgeable.*

Proof. We prove unforgeability using a sequence of games where we let $q_\mathsf{s} \leq \mathsf{poly}(\kappa)$ be the number of Sign queries.

Game 0: The original unforgeability game.

Game 1: As Game 0, but upon setup we store csk and simulate Sign using the following modified algorithm Sign', which additionally takes csk as input:

$\mathsf{Sign}'(\mathsf{PP}, \mathsf{sk}_i, m, \mathcal{R}, \boxed{\mathsf{csk}})$: Parse PP as $(1^\kappa, \mathsf{crs}, \mathsf{cpk})$ and return $\perp$ if $\mu(\mathsf{sk}_i) \notin \mathcal{R}$. Otherwise, return $\sigma \leftarrow (\delta, \mathsf{pk}, \pi)$, where

$$(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KeyGen}(1^\kappa),\ \delta \leftarrow \Sigma.\mathsf{Sign}(\mathsf{sk}, m||\mathcal{R}),\ \text{and}$$
$$\pi \leftarrow \Pi.\mathsf{Proof}(\mathsf{crs}, (\mathsf{pk}, \mathsf{cpk}, \mathcal{R}), (\boxed{\mathsf{csk}} - \mathsf{sk})).$$

Transition - Game 0 $\rightarrow$ Game 1: A distinguisher between $\mathcal{D}^{0\rightarrow 1}$ is a distinguisher for adaptive witness indistinguishability of Π, i.e., $|\Pr[S_0] - \Pr[S_1]| \leq \varepsilon_{\mathsf{wi}}(\kappa)$.

Game 2: As Game 1, but instead of generating crs upon setup, we obtain $(\mathsf{crs}, \xi) \leftarrow \Pi.\mathsf{E}_1(1^\kappa)$ and store ξ.

Transition - Game 1 → Game 2: A distinguisher between Game 1 and 2 distinguishes an honest crs from an extraction crs, i.e., $|\Pr[S_1] - \Pr[S_2]| \leq \varepsilon_{\mathsf{e1}}(\kappa)$.

Game 3: As Game 2, but whenever the adversary outputs a forgery $(m^\star, \sigma^\star, \mathcal{R}^\star)$, where $\sigma^\star = (\delta^\star, \mathsf{pk}^\star, \pi^\star)$ we extract a witness $\mathsf{sk}' \leftarrow \Pi.\mathsf{E}_2(\mathsf{crs}, \xi, (\mathsf{pk}^\star, \mathsf{cpk}, \mathcal{R}^\star), \pi^\star)$ and abort if the extractor fails.

Transition - Game 2 → Game 3: Game 2 and Game 3 proceed identically, unless the extractor fails, i.e., $|\Pr[S_2] - \Pr[S_3]| \leq \varepsilon_{\mathsf{e2}}(\kappa)$.

Game 4: As Game 3, but we further modify Sign' to Sign'' as follows:

$\mathsf{Sign}''(\mathsf{PP}, \boxed{i}, m, \mathcal{R}, \mathsf{csk})$: Parse PP as $(1^\kappa, \mathsf{crs}, \mathsf{cpk})$ and return $\perp$ if $\mathsf{pk}_i \notin \mathcal{R}$. Otherwise, return $\sigma \leftarrow (\delta, \mathsf{pk}, \pi)$, where

$$\boxed{\mathsf{sk} \xleftarrow{R} \mathbb{H},\ \delta' \leftarrow \Sigma.\mathsf{Sign}(\mathsf{csk}, m||\mathcal{R})},$$
$$\boxed{(\mathsf{pk}, \delta) \leftarrow \Sigma.\mathsf{Adapt}(\mathsf{cpk}, m||\mathcal{R}, \delta', -\mathsf{sk})}, \text{ and}$$
$$\pi \leftarrow \Pi.\mathsf{Proof}(\mathsf{crs}, (\mathsf{pk}, \mathsf{cpk}, \mathcal{R}), (\boxed{\mathsf{sk}})).$$

Transition - Game 3 → Game 4: Under adaptability of signatures, this game change is conceptual, i.e., $\Pr[S_3] = \Pr[S_4]$.

Game 5: As Game 4, but we abort whenever we extract an sk' so that $\mathsf{cpk} = \mathsf{pk} \cdot \mu(\mathsf{sk}')$.

Transition - Game 4 → Game 5: Game 4 and Game 5 proceed identical, unless abort event E_1 happens. For the sake of contradiction assume that E_1 occurs with non-negligible probability. Then we can engage with an $\mathsf{EUF\text{-}CMA}$ challenger $\mathcal{C}_\kappa^{\mathsf{f}}$ to obtain cpk upon setup and simulate Sign using the algorithm Sign''' as follows:

$\mathsf{Sign}'''(\mathsf{PP}, i, m, \mathcal{R}, \boxed{\perp})$: Parse PP as $(1^\kappa, \mathsf{crs}, \mathsf{cpk})$ and return $\perp$ if $\mathsf{pk}_i \notin \mathcal{R}$. Otherwise, return $\sigma \leftarrow (\delta, \mathsf{pk}, \pi)$, where

$$\mathsf{sk} \xleftarrow{R} \mathbb{H},\ \boxed{\delta' \leftarrow \mathcal{C}_{\mathsf{f}}^{\kappa}.\mathsf{Sign}(m||\mathcal{R})},$$
$$(\mathsf{pk}, \delta) \leftarrow \Sigma.\mathsf{Adapt}(\mathsf{cpk}, m||\mathcal{R}, \delta', -\mathsf{sk}), \text{ and}$$
$$\pi \leftarrow \Pi.\mathsf{Proof}(\mathsf{crs}, (\mathsf{pk}, \mathsf{cpk}, \mathcal{R}), (\mathsf{sk})).$$

Now, whenever E_1 happens, we use the forgery $(m^\star, \sigma^\star, \mathcal{R}^\star)$, where $\sigma^\star = (\delta^\star, \mathsf{pk}^\star, \pi^\star)$ to obtain $(\mathsf{cpk}, \delta) \leftarrow \mathsf{Adapt}(\mathsf{pk}, m^\star||\mathcal{R}^\star, \delta^\star, \mathsf{sk}')$ and return $(m^\star||\mathcal{R}^\star, \delta)$ as an $\mathsf{EUF\text{-}CMA}$ forgery to $\mathcal{C}_{\mathsf{f}}^{\kappa}$ with probability $\Pr[E_1]$. That is, $|\Pr[S_4] - \Pr[S_5]| \leq \varepsilon_{\mathsf{f}}(\kappa)$.

Game 6: As Game 5, but we guess the index $i^\star$ the adversary will attack at the beginning of the game, and abort if our guess is wrong.

Transition - Game 5 → Game 6: The success probability in Game 5 is the same as in Game 6, unless our guess is wrong, i.e., $\Pr[S_6] = \frac{1}{\mathsf{poly}(\kappa)} \cdot \Pr[S_5]$.

Game 7: As Game 6, but instead of running KeyGen for user $i^\star$, we engage with an $\mathsf{EUF\text{-}CMA}$ challenger of Σ to obtain $\mathsf{pk}_{i^\star}$.

Transition - Game 6 → Game 7: This change is conceptual, i.e., $\Pr[S_6] = \Pr[S_7]$.

If the adversary outputs a forgery $(m^\star, \sigma^\star, \mathcal{R}^\star)$ in Game 7, we compute $(\mathsf{pk}_{i^\star}, \sigma_{i^\star}) \leftarrow \mathsf{Adapt}(\mathsf{pk}^\star, m^\star||\mathcal{R}^\star, \delta^\star, \mathsf{sk}')$ and return $(\sigma_{i^\star}, m^\star||\mathcal{R}^\star)$ as a valid forgery for Σ. That is, $\Pr[S_7] \leq \varepsilon_{\mathsf{f}}(\kappa)$ and we obtain $\Pr[S_0] \leq \mathsf{poly}(\kappa) \cdot \varepsilon_{\mathsf{f}}(\kappa) + \varepsilon_{\mathsf{wi}}(\kappa) + \varepsilon_{\mathsf{e1}}(\kappa) + \varepsilon_{\mathsf{e2}}(\kappa) + \varepsilon_{\mathsf{f}}(\kappa)$ as a bound for the success probability which concludes the proof. □

Lemma 4.3. *If Σ provides adaptability of signatures and Π is witness indistinguishable, then Scheme 4.1 is anonymous.*

Proof. We show that a simulation of the anonymity game for $b = 0$ is indistinguishable from a simulation of the anonymity game with $b = 1$.

Game 0: The anonymity game with $b = 0$.

Game 1: As Game 0, but instead of generating crs upon setup, we obtain crs from a witness indistinguishability challenger $\mathcal{C}_\kappa^{\mathsf{wi}}$ upon Setup.

Transition - Game 0 → Game 1: This change is conceptual, i.e., $\Pr[S_0] = \Pr[S_1]$.

Game 2: As Game 1, but instead of obtaining σ via Sign, we execute the following modified algorithm Sign', which, besides PP, m and $\mathcal{R}$, takes sk_0 and sk_1 as input:

$\mathsf{Sign}'(\mathsf{PP}, \mathsf{sk}_0, \mathsf{sk}_1, m, \mathcal{R})$: Parse PP as $(1^\kappa, \mathsf{crs})$ and return $\perp$ if $\mu(\mathsf{sk}_0) \notin \mathcal{R}$ $\boxed{\vee\ \mu(\mathsf{sk}_1) \notin \mathcal{R}}$. Otherwise, return $\sigma \leftarrow (\delta, \mathsf{pk}, \pi)$, where

$$(\mathsf{sk}, \mathsf{pk}) \leftarrow \Sigma.\mathsf{KeyGen}(1^\kappa),\ \delta \leftarrow \Sigma.\mathsf{Sign}(\mathsf{sk}, m||\mathcal{R}),\ \text{and}$$
$$\pi \leftarrow \Pi.\mathsf{Proof}(\mathsf{crs}, (\mathsf{pk}, \mathcal{R}), (\boxed{\mathsf{sk}_1} - \mathsf{sk})).$$

Transition - Game 1 → Game 2: A distinguisher between $\mathcal{D}^{1\to 2}$ is a distinguisher for adaptive witness indistinguishability of Π, i.e., $|\Pr[S_2] - \Pr[S_1]| \leq \varepsilon_{\mathsf{wi}}(\kappa)$.

In Game 2, we have a simulation for $b = 1$; $|\Pr[S_2] - \Pr[S_0]| \leq \varepsilon_{\mathsf{wi}}(\kappa)$, which proves the lemma. □

4.2 Dynamic Group Signatures

Group signatures, initially introduced by Chaum and van Heyst [CvH91], allow a group manager to set up a group so that every member of this group can later anonymously sign messages on behalf of the group. Thereby, a dedicated authority (called opening authority) can open a given group signature to determine the identity of the actual signer. Group signatures were first rigorously formalized for static groups by Bellare et al. in [BMW03]. In this setting, all members are fixed at setup and also receive their honestly generated keys at setup from the group manager. This model was later extended to the dynamic case by Bellare et al. in [BSZ05] (henceforth denoted by BSZ model), where new group members can be dynamically and concurrently enrolled to the group. Further, it separates the role of the issuer and the opener so that they can operate independently. Moreover, the BSZ model requires a strong anonymity notion, where anonymity of a group signature is preserved even if the adversary can see arbitrary key exposures and arbitrary openings of other group signatures. A slightly weaker model, which is used to prove the security (and in particular anonymity) of the popular BBS group signature scheme was introduced by Boneh et al. [BBS04]. This model is a relaxation of the BSZ model, and in particular weakens anonymity so that the adversary can not request openings for signatures. As it is common, we refer to this anonymity notion as CPA-full anonymity, whereas we use CCA2-full anonymity to refer to anonymity in the sense of BSZ.

Over the years, two main construction paradigms for group signatures have been established. The first one is the widely used sign-encrypt-prove (SEP) paradigm [CS97]. Here, a signature is essentially an encrypted membership certificate together with a signature of knowledge, where the signer demonstrates knowledge of some signed value in the ciphertext [ACJT00, BBS04, NS04, BSZ05, KY05, DP06, BW07, BW06, Gro07, LPY15, LLM+16, LMPY16]. As an alternative to this paradigm, Bichsel et al. in [BCN+10] used an elegant design paradigm for group signatures which does not require to encrypt the membership certificate to produce signatures.[2] Henceforth we call this paradigm sign-randomize-proof (SRP). Essentially, they use a signature scheme which supports (1) randomization of signatures so that multiple randomized versions of the same signature are unlinkable, and (2) efficiently proving knowledge of a signed value. In their construction, on joining the group, the issuer uses such a signature scheme to sign a commitment to the user's secret key. The user can then produce a group signature for a message by randomizing the signature and computing a signature of knowledge on the message, which demonstrates knowledge of the signed secret key. To open signatures, in contrast to constructions following SEP which support constant time opening by means of decrypting the ciphertext in the signature, constructions in this paradigm require a linear scan, i.e., to check a given signature against each potential user. Bichsel et al.

[2] Note that a similar paradigm was earlier also used in [ACHdM05]. We note that their scheme provides the same security guarantees while being less efficient than [BCN+10] which is why we omit it in our comparison later on.

proposed an instantiation based on the randomizable pairing-based Camensich-Lysyanskaya (CL) signature scheme [CL04] (whose EUF-CMA security is based on the interactive LRSW assumption). Recently, Pointcheval and Sanders [PS16] proposed another randomizable signature scheme (whose EUF-CMA security is proven in the generic group model), which allows to instantiate the approach due to Bichsel et al. more efficiently. We note that while these two existing constructions do not explicitly use public key encryption, the required assumptions for the scheme imply public key encryption. Yet, it seems to be beneficial regarding performance to avoid to explicitly use public key encryption.

The main drawback of existing constructions following the SRP paradigm is that they rely on a security model that is weaker than the BSZ model [BSZ05]. In particular, anonymity only holds for users whose keys do not leak. This essentially means that once a user key leaks, all previous signatures of this user can potentially be attributed to this user. Furthermore, the models used for SRP constructions so far assume that the opening authority and the issuing authority are one entity, meaning that the issuer can identify all signers when seeing group signatures. Both aforementioned weakenings can be highly problematic in practical applications of group signatures. It is thus a natural question to ask whether it is possible to prove that constructions following the SRP paradigm provide CPA- or even CCA2-full anonymity. Unfortunately, for existing constructions, we have to answer this negatively. Even when allowing to modify the existing constructions in [ACHdM05, BCN+10, PS16] to allow the explicit use of encryption upon joining the group (which might solve the separability issue regarding issuer and opener), it is easy to see that knowledge of the user secret key breaks CCA2- as well as CPA-full anonymity for both constructions.[3] Since CCA2-full anonymity straight forwardly implies anonymity in the SRP model, this example confirms that CCA2-full anonymity is a strictly stronger notion. The notion of CPA-full anonymity is somewhat orthogonal to the anonymity notion used by the SRP model: it appropriately models the leakage of user secret keys, but restricts the open oracle access. Yet, in practice it seems that the risk that a user secret key leaks is extremely hard to quantify, which is why we deem CPA-full anonymity to be more desirable. This is also underpinned by the fact that—to the best of our knowledge—no attacks arising from the restriction of the open oracle access in CPA-full anonymity are known.

Motivation. Group signatures have received significant attention from the cryptographic community and also gain increasing practical relevance due to technological innovations in intelligent transportation systems (e.g., floating car data, toll systems) as well as public transportation systems (i.e., smart ticketing), where user privacy is considered to play an important role (cf. EU Directive 2010/40/EU). These developments make it important to have particularly efficient group signature candidates at hand. As an illustrative example for the

[3] Each valid group signature contains a valid randomizable signature on the secret key of the user. Being in possession of secret key candidates allows to simply test them using the verification algorithm of the randomizable signature scheme. This clearly provides a distinguisher against CCA2- as well as CPA-full anonymity.

importance of very fast signature generation and verification times, consider public transportation system where every user needs to sign on passing a gate.

Despite their increasing practical importance, no progress has been made with respect to computational efficiency improvements of signature generation (and verification) of group signature schemes providing the more desirable notions of CPA- as well as CCA2-full anonymity within the last decade. The most efficient schemes used today are the BBS group signature scheme [BBS04] (which achieves CPA-full anonymity) and the XSGS group signature scheme [DP06] (which achieves CCA2-full anonymity).

We tackle the following open questions, which are of both theoretical and practical interest:

- *Is it possible to construct schemes providing the more desirable* CPA*-full and* CCA2*-full anonymity notions, where particularly efficient signature generation (and verification) is reached by (1) avoiding the explicit encryption of the membership certificate upon signing, yet (2) allowing to explicitly use encryption during the joining of a group?*
- *Is it possible to further push the computational efficiency limits of group signature schemes providing those more desirable anonymity notions?*

We, henceforth, refer to such schemes as "without encryption".

Contribution. We answer both questions posed above to the affirmative by contributing a novel approach to construct group signatures "without encryption". Our approach is a composition of structure preserving signatures on equivalence classes (SPS-EQ) [HS14, FHS15b], conventional digital signatures, public key encryption, non-interactive zero-knowledge proofs, and signatures of knowledge. Although these tools may sound quite heavy, we obtain surprisingly efficient group signatures, which provably provide CCA2-full anonymity in the strongest model for dynamic group signatures, i.e., the BSZ model. In doing so, we obtain the first construction which achieves this strong security notion without an encrypted membership certificate in the signature. In addition to that, we introduce an even more efficient CPA-fully anonymous variant of our scheme.

We proceed in showing how to instantiate our constructions in the random oracle model (ROM) to obtain particularly efficient schemes. We are thereby able to further push the long standing computational efficiency limits for both CPA- and CCA2-fully anonymous schemes regarding signature generation and verification. When comparing to the popular BBS group signature scheme [BBS04] (which achieves CPA-full anonymity in the ROM), besides being more efficient we surprisingly even obtain shorter signatures. Ultimately, when comparing to instantiations in the vein of Bichsel et al. (which provide a less desirable anonymity notion), our instantiations provide comparable computational efficiency.

4.2.1 Formal Security Model

Below we recall the established model for dynamic group signatures. We follow Bellare et al. [BSZ05] (BSZ model), with the slight difference that we relax the perfect correctness to only require computational correctness. Furthermore, we also present the weaker anonymity notion of CPA-full anonymity from [BBS04] and the notion of opening soundness [SSE+12], which addresses issues regarding hijacking of signatures by malicious group members. In particular, we use the notion of weak opening soundness, where the opening authority is required to be honest, since we believe that this notion provides a good tradeoff between computational efficiency of potential instantiations and expected security guarantees (even the authors of [SSE+12] say that already weak opening soundness addresses the attacks they had in mind).

Definition 4.4. *A dynamic group signature scheme is a tuple* (GKeyGen, UKeyGen, Join, Issue, Sign, Verify, Open, Judge) *of PPT algorithms which are defined as follows:*

$\mathsf{GKeyGen}(1^\kappa)$: *This algorithm takes a security parameter* κ *as input and outputs a triple* $(\mathsf{gpk}, \mathsf{ik}, \mathsf{ok})$ *containing the group public key* gpk*, the issuing key* ik *as well as the opening key* ok*.*

$\mathsf{UKeyGen}(1^\kappa)$: *This algorithm takes a security parameter* κ *as input and outputs a user key pair* $(\mathsf{usk}_i, \mathsf{upk}_i)$*.*

$\mathsf{Join}(\mathsf{gpk}, \mathsf{usk}_i, \mathsf{upk}_i)$: *This algorithm takes the group public key* gpk *and the user's key pair* $(\mathsf{usk}_i, \mathsf{upk}_i)$ *as input. It interacts with the* Issue *algorithm and outputs the group signing key* gsk_i *of user* i *on success.*

$\mathsf{Issue}(\mathsf{gpk}, \mathsf{ik}, i, \mathsf{upk}_i, \mathsf{reg})$: *This algorithm takes the group public key* gpk*, the issuing key* ik*, the index* i *of a user, user* i*'s public key* upk_i*, and the registration table* reg *as input. It interacts with the* Join *algorithm and adds an entry for user* i *in* reg *on success. In the end, it returns* reg*.*

$\mathsf{Sign}(\mathsf{gpk}, \mathsf{gsk}_i, m)$: *This algorithm takes the group public key* gpk*, a group signing key* gsk_i*, and a message* m *as input and outputs a group signature* σ*.*

$\mathsf{Verify}(\mathsf{gpk}, m, \sigma)$: *This algorithm takes the group public key* gpk*, a message* m *and a signature* σ *as input and outputs a bit* $b \in \{0, 1\}$*.*

$\mathsf{Open}(\mathsf{gpk}, \mathsf{ok}, \mathsf{reg}, m, \sigma)$: *This algorithm takes the group public key* gpk*, the opening key* ok*, the registration table* reg*, a message* m*, and a valid signature* σ *on* m *under* gpk *as input. It extracts the identity of the signer and returns a pair* (i, τ)*, where* τ *is a proof.*

$\mathsf{Judge}(\mathsf{gpk}, m, \sigma, i, \mathsf{upk}_i, \tau)$: *This algorithm takes the group public key* gpk*, a message* m*, a valid signature* σ *on* m *under* gpk*, an index* i*, user* i*'s public key* upk_i*, and a proof* τ*. It returns a bit* $b \in \{0, 1\}$*.*

Oracle Definitions. In the following we recall the definitions of the oracles required by the security model. We assume that the keys $(\mathsf{gpk}, \mathsf{ik}, \mathsf{ok})$ created in the experiments are implicitly available to the oracles. Furthermore, the environment maintains the sets $\mathtt{HU}, \mathtt{CU}$ of honest and corrupted users, the set GS of message-signature tuples returned by the challenge oracle, the lists $\mathsf{upk}, \mathsf{usk}, \mathsf{gsk}$ of user public keys, user private keys, and group signing keys. The list upk is publicly readable and the environment also maintains the registration table reg. Finally, $\mathtt{SI}$ represents a list that ensures the consistency of subsequent calls to CrptU and SndToI. All sets are initially empty and all list entries are initially set to $\perp$. In the context of lists, we use $\mathsf{upk}_i, \mathsf{usk}_i$, etc. as shorthand for $\mathsf{upk}[i], \mathsf{usk}[i]$, etc.

$\mathsf{AddU}(i)$: This oracle takes an index i as input. If $i \in \mathtt{CU} \cup \mathtt{HU}$ it returns $\perp$. Otherwise it runs $(\mathsf{usk}_i, \mathsf{upk}_i) \leftarrow \mathsf{UKeyGen}(1^\kappa)$ and

$$(\mathsf{reg}, \mathsf{gsk}_i) \leftarrow \langle \mathsf{Issue}(\mathsf{gpk}, \mathsf{ik}, i, \mathsf{upk}_i, \mathsf{reg}) \leftrightarrow \mathsf{Join}(\mathsf{gpk}, \mathsf{usk}_i, \mathsf{upk}_i) \rangle.$$

Finally, it sets $\mathtt{HU} \leftarrow \mathtt{HU} \cup \{i\}$ and returns upk_i.

$\mathsf{CrptU}(i, \mathsf{upk}_j)$: This oracle takes an index i and user public key upk_j as input. If $i \in \mathtt{CU} \cup \mathtt{HU}$ it returns $\perp$. Otherwise it sets $\mathtt{CU} \leftarrow \mathtt{CU} \cup \{i\}$, $\mathtt{SI}[i] \leftarrow \top$ and $\mathsf{upk}_i \leftarrow \mathsf{upk}_j$.

$\mathsf{SndToI}(i)$: This oracle takes an index i as input. If $\mathtt{SI}[i] \neq \top$ it returns $\perp$. Otherwise, it plays the role of an honest issuer when interacting with the corrupted user i. More precisely, it runs

$$\mathsf{reg} \leftarrow \langle \mathsf{Issue}(\mathsf{gpk}, \mathsf{ik}, i, \mathsf{upk}_i, \mathsf{reg}) \leftrightarrow \mathcal{A} \rangle.$$

In the end it sets $\mathtt{SI}[i] \leftarrow \perp$.

$\mathsf{SndToU}(i)$: This oracle takes an index i as input. If $i \notin \mathtt{HU}$ it sets $\mathtt{HU} \leftarrow \mathtt{HU} \cup \{i\}$, runs $(\mathsf{usk}_i, \mathsf{upk}_i) \leftarrow \mathsf{UKeyGen}(1^\kappa)$. Then it plays the role of the honest user i when interacting with a corrupted issuer. More precisely, it runs

$$\mathsf{gsk}_i \leftarrow \langle \mathcal{A} \leftrightarrow \mathsf{Join}(\mathsf{gpk}, \mathsf{usk}_i, \mathsf{upk}_i) \rangle.$$

$\mathsf{USK}(i)$: This oracle takes an index i as input and returns $(\mathsf{gsk}_i, \mathsf{usk}_i)$.

$\mathsf{RReg}(i)$: This oracle takes an index i as input and returns reg_i.

$\mathsf{WReg}(i, \zeta)$: This oracle takes an index i and a registration table entry ζ as input and sets $\mathsf{reg}_i \leftarrow \zeta$.

$\mathsf{GSig}(i, m)$: This oracle takes an index i and a message m as input. If $i \notin \mathtt{HU}$ or $\mathsf{gsk}_i = \perp$ it returns $\perp$ and $\sigma \leftarrow \mathsf{Sign}(\mathsf{gpk}, \mathsf{gsk}_i, m)$ otherwise.

$\mathsf{Ch}(b, i_0, i_1, m)$: This algorithm takes a bit b, two indexes i_0 and i_1, and a message m as input. If $\{i_0, i_1\} \not\subseteq \mathtt{HU} \ \vee \ \mathsf{gsk}_{i_0} = \perp \ \vee \ \mathsf{gsk}_{i_1} = \perp$ it returns $\perp$. Otherwise, it computes $\sigma \leftarrow \mathsf{Sign}(\mathsf{gpk}, \mathsf{gsk}_{i_b}, m)$, sets $\mathsf{GS} \leftarrow \mathsf{GS} \cup \{(m, \sigma)\}$ and returns σ.

$\mathsf{Open}(m, \sigma)$: This oracle takes a message m and a signature σ as input. If $(m, \sigma) \in \mathsf{GS}$ or $\mathsf{Verify}(\mathsf{gpk}, m, \sigma) = 0$ it returns $\perp$. Otherwise, it returns $(i, \tau) \leftarrow \mathsf{Open}(\mathsf{gpk}, \mathsf{ok}, \mathsf{reg}, m, \sigma)$.

Definitions of the Security Notions. We require dynamic group signatures to be correct, anonymous, traceable, non-frameable, and weakly opening sound. We recall the formal definitions below.

Correctness, requires that everything works correctly if everyone behaves honestly. Note that we relax perfect correctness to computational correctness.

Definition 4.5 (Correctness)**.** *A* GSS *is correct, if for all PPT adversaries* $\mathcal{A}$ *there is a negligible function* $\varepsilon(\cdot)$ *such that*

$$\Pr\left[\begin{array}{l}(\mathsf{gpk}, \mathsf{ik}, \mathsf{ok}) \leftarrow \mathsf{GKeyGen}(1^\kappa),\ \mathcal{O} \leftarrow \{\mathsf{AddU}(\cdot), \mathsf{RReg}(\cdot)\},\\ (i^\star, m^\star) \leftarrow \mathcal{A}^{\mathcal{O}}(\mathsf{gpk}),\ \sigma \leftarrow \mathsf{Sign}(\mathsf{gpk}, \mathsf{gsk}_{i^\star}, m^\star), \quad :\\ (j, \tau) \leftarrow \mathsf{Open}(\mathsf{gpk}, \mathsf{ok}, \mathsf{reg}, m^\star, \sigma)\end{array} \begin{array}{r}\mathsf{Verify}(\mathsf{gpk}, m^\star, \sigma) = 1 \ \wedge\\ i^\star \in \mathtt{HU} \ \wedge \ \mathsf{gsk}_{i^\star} \neq \perp \ \wedge \ i^\star = j \ \wedge\\ \mathsf{Judge}(\mathsf{gpk}, m^\star, \sigma, i^\star, \mathsf{upk}_{i^\star}, \tau) = 1\end{array}\right] \geq 1 - \varepsilon(\kappa).$$

Anonymity captures the intuition that group signers remain anonymous for everyone except the opening authority. Thereby, the adversary can see arbitrary key exposures. Furthermore, in the CCA2 case the adversary can even request arbitrary openings of other group signatures.

Definition 4.6 (T-Full Anonymity)**.** *Let* $\mathsf{T} \in \{\mathsf{CPA}, \mathsf{CCA2}\}$. *A* GSS *is* T*-fully anonymous, if for all PPT adversaries* $\mathcal{A}$ *there is a negligible function* $\varepsilon(\cdot)$ *such that*

$$\Pr\left[\begin{array}{l}(\mathsf{gpk}, \mathsf{ik}, \mathsf{ok}) \leftarrow \mathsf{GKeyGen}(1^\kappa),\ b \xleftarrow{R} \{0,1\},\\ b^\star \leftarrow \mathcal{A}^{\mathcal{O}_\mathsf{T}}(\mathsf{gpk}, \mathsf{ik})\end{array} \quad : \quad b = b^\star \right] \leq 1/2 + \varepsilon(\kappa),$$

where

$$\mathcal{O}_\mathsf{T} \leftarrow \begin{cases} \left\{\begin{array}{l}\mathsf{Ch}(b, \cdot, \cdot, \cdot),\ \mathsf{SndToU}(\cdot), \mathsf{WReg}(\cdot, \cdot),\\ \mathsf{USK}(\cdot),\ \mathsf{CrptU}(\cdot, \cdot)\end{array}\right\} & \textit{if} \quad \mathsf{T} = \mathsf{CPA}, \ \textit{and}\\ \left\{\begin{array}{l}\mathsf{Ch}(b, \cdot, \cdot, \cdot),\ \mathsf{Open}(\cdot, \cdot),\ \mathsf{SndToU}(\cdot),\\ \mathsf{WReg}(\cdot, \cdot),\ \mathsf{USK}(\cdot),\ \mathsf{CrptU}(\cdot, \cdot)\end{array}\right\} & \textit{if} \quad \mathsf{T} = \mathsf{CCA2}. \end{cases}$$

Traceability models the requirement that, as long as the issuer behaves honestly and its secret key remains secret, every valid signature can be traced back to a user. This must even hold if the opening authority colludes with malicious users.

Definition 4.7 (Traceability)**.** *A* GSS *is traceable, if for all PPT adversaries* $\mathcal{A}$

there is a negligible function $\varepsilon(\cdot)$ such that

$$\Pr\left[\begin{array}{l}(\mathsf{gpk},\mathsf{ik},\mathsf{ok}) \leftarrow \mathsf{GKeyGen}(1^\kappa),\ \mathcal{O} \leftarrow \{\mathsf{SndToI}(\cdot),\ \mathsf{AddU}(\cdot),\\ \mathsf{RReg}(\cdot),\ \mathsf{USK}(\cdot),\ \mathsf{CrptU}(\cdot)\},\ (m^\star,\sigma^\star) \leftarrow \mathcal{A}^{\mathcal{O}}(\mathsf{gpk},\mathsf{ok}), \quad :\\ (i,\tau) \leftarrow \mathsf{Open}(\mathsf{gpk},\mathsf{ok},\mathsf{reg},m^\star,\sigma^\star)\end{array}\right.$$
$$\left.\begin{array}{r}\mathsf{Verify}(\mathsf{gpk},m^\star,\sigma^\star)=1 \wedge\\ (\mathsf{Judge}(\mathsf{gpk},m^\star,\sigma^\star,i,\mathsf{upk}_i,\tau)=0 \vee\\ i=\bot)\end{array}\right] \leq \varepsilon(\kappa).$$

Non-frameability requires that no one can forge signatures for honest users. This must even hold if the issuing authority, the opening authority, and, other malicious users collude.

Definition 4.8 (Non-Frameability)**.** *A* GSS *is non-frameable, if for all PPT adversaries $\mathcal{A}$ there is a negligible function $\varepsilon(\cdot)$ such that*

$$\Pr\left[\begin{array}{l}(\mathsf{gpk},\mathsf{ik},\mathsf{ok}) \leftarrow \mathsf{GKeyGen}(1^\kappa),\ \mathcal{O} \leftarrow \{\mathsf{SndToU}(\cdot),\ \mathsf{WReg}(\cdot,\cdot),\\ \mathsf{GSig}(\cdot,\cdot),\ \mathsf{USK}(\cdot),\ \mathsf{CrptU}(\cdot)\},\ (m^\star,\sigma^\star,i^\star,\tau^\star) \leftarrow \mathcal{A}^{\mathcal{O}}(\mathsf{gpk},\mathsf{ok},\mathsf{ik}) \quad :\end{array}\right.$$
$$\left.\begin{array}{r}\mathsf{Verify}(\mathsf{gpk},m^\star,\sigma^\star)=1 \ \wedge\ i^\star \in \mathtt{HU} \ \wedge\ \mathsf{gsk}_{i^\star} \neq \bot \wedge\\ i^\star \notin \mathtt{USK} \ \wedge\ (i^\star,m^\star) \notin \mathtt{SIG} \ \wedge\ \mathsf{Judge}(\mathsf{gpk},m^\star,\sigma^\star,i^\star,\mathsf{upk}_{i^\star},\tau^\star)=1\end{array}\right] \leq \varepsilon(\kappa),$$

where USK *and* SIG *denote the queries to the oracles* USK *and* Sign*, respectively.*

Weak opening soundness [SSE+12] essentially requires that no malicious user can claim ownership of a signature issued by an honest user, as long as the opening authority behaves honestly.

Definition 4.9 (Weak Opening Soundness)**.** *A* GSS *is weakly opening sound, if for all PPT adversaries $\mathcal{A}$ there is a negligible function $\varepsilon(\cdot)$ such that*

$$\Pr\left[\begin{array}{l}(\mathsf{gpk},\mathsf{ik},\mathsf{ok}) \leftarrow \mathsf{GKeyGen}(1^\kappa),\ \mathcal{O} \leftarrow \{\mathsf{AddU}(\cdot)\},\\ (m^\star,i^\star,j^\star,\mathsf{st}) \leftarrow \mathcal{A}^{\mathcal{O}}(\mathsf{gpk}),\ \sigma \leftarrow \mathsf{Sign}(\mathsf{gpk},\mathsf{gsk}_{i^\star},m^\star), \quad :\\ \tau \leftarrow \mathcal{A}^{\mathcal{O}}(\mathsf{st},\sigma,\mathsf{gsk}_{j^\star})\end{array}\right.$$
$$\left.\begin{array}{r}i^\star \neq j^\star \ \wedge\\ \{i^\star,j^\star\} \subseteq \mathtt{HU} \ \wedge\\ \mathsf{Judge}(\mathsf{gpk},m^\star,\sigma,j^\star,\mathsf{upk}_{j^\star},\tau)=1\end{array}\right] \leq \varepsilon(\kappa).$$

4.2.2 Our Construction

Our construction idea is inspired by [HS14], who use the "unlinkability" feature of SPS-EQ signatures to construct anonymous credentials. Essentially, a credential in their approach represents a signature for an equivalence class and to show a credential they always present a newly re-randomized signature to a random representative of this class. While, due to the intuitive relation of anonymous credentials and group signatures, it might seem straightforward to map this idea to group signatures, it turns out that there are various subtle, yet challenging issues which we need to solve.

First, the anonymity notion is much stronger than the one of anonymous credentials in that it does not put many restrictions on the Ch and the USK oracles. In particular, Ch can be called an arbitrary number of times and USK can be called for all users. Thus, the user secret keys must be of a form so that it is possible to embed decision problem instances into them upon simulation, while not influencing their distribution (as the adversary sees those keys and would be able to detect the simulation otherwise). More precisely, anonymity in our paradigm seems to require that the user keys contain no $\mathbb{Z}_p$ elements, which, in turn, renders the non-frameability proof more difficult. Second, if CCA2-full anonymity is required, the simulatability of the open oracle needs to be ensured, while the reduction must not be aware of the opening information (as otherwise the reduction could trivially break anonymity on its own and would be meaningless). This seems to crucially require a proof system providing rather strong extractability properties. To maintain efficiency, it is important to find the mildest possible requirement which still allows the security proofs to work out. Third, the non-frameability adversary is given the issuing key as well as the opening key. Thus, the reduction must be able to simulate the whole join process without knowledge of a user secret key in a way that the distribution change is not even detectable with the knowledge of these keys.

Now, before we present our full construction, we briefly revisit our basic idea. In our scheme, each group member chooses a secret vector $(w, g) \in (\mathbb{G}_1^*)^2$ representing an equivalence class where the second component g is identical for all users. When joining the group, a blinded version $(w, g)^q$ with $q \xleftarrow{R} \mathbb{Z}_p^*$ of this vector, i.e., another representative of the class, is signed by the issuer using an SPS-EQ, and, by the re-randomization property of SPS-EQ and the feature to publicly change representatives of classes, the user thus obtains a signature on the unblinded key (w, g) using $\mathsf{ChgRep}_{\mathcal{R}}$ with q^{-1}. To provide a means to open signatures, a user additionally has to provide an encryption of a value $\hat{w} \in \mathbb{G}_2$ such that $e(w, \hat{g}) = e(g, \hat{w})$ on joining (and has to sign the ciphertext as an identity proof). The group signing key of the user is then the pair consisting of the vector (w, g) and the SPS-EQ signature on this vector. A group member can sign a message m on behalf of the group by randomizing its group signing key and computing a signature of knowledge (SoK) to the message m proving knowledge of the used randomizer.[4] The group signature is then the randomized group signing key and the SoK.

Very roughly, a signer remains anonymous since it is infeasible to distinguish two randomized user secret keys under DDH in $\mathbb{G}_1$. The unforgeability of SPS-EQ ensures that each valid signature can be opened. Furthermore, it is hard to forge signatures of honest group members since it is hard to unblind a user secret key under co-CDHI and the SoK essentially ensures that we can extract such an unblinded user secret key from a successful adversary.

Detailed Construction. We now present the details of our construction

[4] For technical reasons and in particular for extractability, we actually require a signature of knowledge for message $m' = \sigma_1 || m$, where σ_1 contains the re-randomized user secret key and SPS-EQ signature.

in Scheme 4.2-4.5. In doing so, we also state the NP-relations for the zero-knowledge proofs used upon Join and Open, and for the signature of knowledge used upon Sign. For the sake of compactness, we assume that the languages defined those relations are implicit in the CRSs crs_J, crs_O, and crs_S, respectively.

We start by presenting the key generation algorithms in Scheme 4.2.

$\underline{\mathsf{GKeyGen}(1^\kappa)}$: Run $\mathsf{BG} \leftarrow \mathsf{BGGen}_\mathcal{R}(1^\kappa, 3)$, $(\mathsf{sk}_\mathcal{R}, \mathsf{pk}_\mathcal{R}) \leftarrow \mathsf{KeyGen}_\mathcal{R}(\mathsf{BG}, 2)$, $(\mathsf{sk}_\mathsf{O}, \mathsf{pk}_\mathsf{O}) \leftarrow \Omega.\mathsf{KeyGen}(1^\kappa)$, $\mathsf{crs}_\mathsf{J} \leftarrow \Pi.\mathsf{Setup}(1^\kappa)$, $\mathsf{crs}_\mathsf{O} \leftarrow \Pi.\mathsf{Setup}(1^\kappa)$, $\mathsf{crs}_\mathsf{S} \leftarrow \mathsf{SoK.Setup}(1^\kappa)$, set $\mathsf{gpk} \leftarrow (\mathsf{pk}_\mathcal{R}, \mathsf{pk}_\mathsf{O}, \mathsf{crs}_\mathsf{J}, \mathsf{crs}_\mathsf{O}, \mathsf{crs}_\mathsf{S})$, $\mathsf{ik} \leftarrow \mathsf{sk}_\mathcal{R}$, $\mathsf{ok} \leftarrow \mathsf{sk}_\mathsf{O}$ and return $(\mathsf{gpk}, \mathsf{ik}, \mathsf{ok})$.

$\underline{\mathsf{UKeyGen}(1^\kappa)}$: Return $(\mathsf{usk}_i, \mathsf{upk}_i) \leftarrow \Sigma.\mathsf{KeyGen}(1^\kappa)$.

Scheme 4.2: Key generation procedures.

On joining the group, a proof for the following **NP** relation R_J is carried out

$$((u_i, v, \hat{C}_{\mathsf{J}_i}, \mathsf{pk}_\mathsf{O}), (r, \omega)) \in R_\mathsf{J} \iff \hat{C}_{\mathsf{J}_i} = \Omega.\mathsf{Enc}(\mathsf{pk}_\mathsf{O}, \hat{g}^r;\ \omega) \ \wedge\ u_i = v^r.$$

In Scheme 4.3 we state the joining procedure.

$\underline{\mathsf{Join}^{(1)}(\mathsf{gpk}, \mathsf{usk}_i, \mathsf{upk}_i)}$: Choose $q, r \xleftarrow{R} \mathbb{Z}_p^*$, set $(u_i, v) \leftarrow (g^{r\cdot q}, g^q)$, and output $M_\mathsf{J} \leftarrow ((u_i, v), \hat{C}_{\mathsf{J}_i}, \sigma_{\mathsf{J}_i}, \pi_{\mathsf{J}_i})$ and $\mathsf{st} \leftarrow (\mathsf{gpk}, q, u_i, v)$, where

$$\hat{C}_{\mathsf{J}_i} \leftarrow \Omega.\mathsf{Enc}(\mathsf{pk}_\mathsf{O}, \hat{g}^r;\ \omega),\ \sigma_{\mathsf{J}_i} \leftarrow \Sigma.\mathsf{Sign}(\mathsf{usk}_i, \hat{C}_{\mathsf{J}_i}),$$

$$\pi_{\mathsf{J}_i} \leftarrow \Pi.\mathsf{Proof}(\mathsf{crs}_\mathsf{J}, (u_i, v, \hat{C}_{\mathsf{J}_i}, \mathsf{pk}_\mathsf{O}), (r, \omega)).$$

$\underline{\mathsf{Issue}(\mathsf{gpk}, \mathsf{ik}, i, \mathsf{upk}_i, \mathsf{reg})}$: Receive $M_\mathsf{J} = ((u_i, v), \hat{C}_{\mathsf{J}_i}, \sigma_{\mathsf{J}_i}, \pi_{\mathsf{J}_i})$, return reg and send σ' to user i, where

$$\mathsf{reg}_i \leftarrow (\hat{C}_{\mathsf{J}_i}, \sigma_{\mathsf{J}_i}),\ \sigma' \leftarrow \mathsf{Sign}_\mathcal{R}((u_i, v), \mathsf{sk}_\mathcal{R}),$$

if $\Pi.\mathsf{Verify}(\mathsf{crs}_\mathsf{J}, (u_i, v, \hat{C}_{\mathsf{J}_i}, \mathsf{pk}_\mathsf{O}), \pi_{\mathsf{J}_i}) = 1 \ \wedge\ \Sigma.\mathsf{Verify}(\mathsf{upk}_i, \hat{C}_{\mathsf{J}_i}, \sigma_{\mathsf{J}_i}) = 1$, and return $\perp$ otherwise.

$\underline{\mathsf{Join}^{(2)}(\mathsf{st}, \sigma')}$: Parse st as $(\mathsf{gpk}, q, u_i, v)$ and return gsk_i, where

$$\mathsf{gsk}_i = ((g^r, g), \sigma) \leftarrow \mathsf{ChgRep}_\mathcal{R}((u_i, v), \sigma', q^{-1}, \mathsf{pk}_\mathcal{R}),$$

if $\mathsf{Verify}_\mathcal{R}((u_i, v), \sigma', \mathsf{pk}_\mathcal{R}) = 1$, and return $\perp$ otherwise.

Scheme 4.3: Join procedure.

Upon signature generation and verification, we require a signature of knowledge which is with respect to the following **NP** relation R_S.

$$((g, h), \rho) \in R_\mathsf{S} \iff h = g^\rho.$$

In Scheme 4.4 we present the signing and verification procedure.

$\underline{\mathsf{Sign}(\mathsf{gpk}, \mathsf{gsk}_i, m)}$: Choose $\rho \xleftarrow{R} \mathbb{Z}_p^*$, and return $\sigma \leftarrow (\sigma_1, \sigma_2)$, where

$$\sigma_1 \leftarrow \mathsf{ChgRep}_{\mathcal{R}}(\mathsf{gsk}_i, \rho, \mathsf{pk}_{\mathcal{R}}), \ddagger\ \sigma_2 \leftarrow \mathsf{SoK.Sign}(\mathsf{crs}_{\mathsf{S}}, (g, \sigma_1[1][2]), \rho, \sigma_1 || m).$$

$\underline{\mathsf{Verify}(\mathsf{gpk}, m, \sigma)}$: Return 1 if the following holds, and 0 otherwise:

$$\mathsf{Verify}_{\mathcal{R}}(\sigma_1, \mathsf{pk}_{\mathcal{R}}) = 1 \quad \wedge \quad \mathsf{SoK.Verify}(\mathsf{crs}_{\mathsf{S}}, (g, \sigma_1[1][2]), \sigma_1 || m, \sigma_2) = 1.$$

‡ Note that gsk_i is of the form $((w, g), \sigma)$ and σ_1 is a randomization of gsk_i. We slightly abuse the notation of $\mathsf{Verify}_{\mathcal{R}}$ and $\mathsf{ChgRep}_{\mathcal{R}}$ and input message-signature tuples instead of separately inputting messages and signatures.

Scheme 4.4: Signing and verification procedures.

Finally, Scheme 4.5 shows the opening procedure. The **NP** relation R_{O} corresponding to the proof carried out upon Open is

$$((\hat{C}_{\mathsf{J}_i}, \mathsf{pk}_{\mathsf{O}}, \sigma), (\mathsf{sk}_{\mathsf{O}}, \hat{w})) \in R_{\mathsf{O}} \iff \hat{w} = \Omega.\mathsf{Dec}(\mathsf{sk}_{\mathsf{O}}, \hat{C}_{\mathsf{J}_i}) \wedge \mathsf{pk}_{\mathsf{O}} \equiv \mathsf{sk}_{\mathsf{O}} \wedge e(\sigma_1[1][1], \hat{g}) = e(\sigma_1[1][2], \hat{w}).$$

Thereby, $\mathsf{pk} \equiv \mathsf{sk}$ denotes the consistency of pk and sk. Note that σ_1 represents the randomized user secret key, i.e., is of the form $((w^\rho, g^\rho), \sigma')$ and consists of a randomized message vector and a corresponding randomized SPS-EQ signature. We use $\sigma_1[1][j]$ to refer to the jth element in the (randomized) message vector.

$\underline{\mathsf{Open}(\mathsf{gpk}, \mathsf{ok}, \mathsf{reg}, m, \sigma)}$: Parse σ as (σ_1, σ_2), and ok as sk_{O}. Obtain the lowest index i,§ so that it holds for $(\hat{C}_{\mathsf{J}_i}, \sigma_{\mathsf{J}_i}) \leftarrow \mathsf{reg}_i$ that $\hat{w} \leftarrow \Omega.\mathsf{Dec}(\mathsf{sk}_{\mathsf{O}}, \hat{C}_{\mathsf{J}_i})$ and $e(\sigma_1[1][1], \hat{g}) = e(\sigma_1[1][2], \hat{w})$. Return (i, τ) and $\perp$ if no such entry exists, where

$$\tau \leftarrow (\pi_{\mathsf{O}}, \hat{C}_{\mathsf{J}_i}, \sigma_{\mathsf{J}_i}), \text{ and } \pi_{\mathsf{O}} \leftarrow \Pi.\mathsf{Proof}(\mathsf{crs}_{\mathsf{O}}, (\hat{C}_{\mathsf{J}_i}, \mathsf{pk}_{\mathsf{O}}, \sigma), (\mathsf{sk}_{\mathsf{O}}, \hat{w})).$$

$\underline{\mathsf{Judge}(\mathsf{gpk}, m, \sigma, i, \mathsf{upk}_i, \tau)}$: Parse τ as $(\pi_{\mathsf{O}}, \hat{C}_{\mathsf{J}_i}, \sigma_{\mathsf{J}_i})$, and return 1 if the following holds and 0 otherwise:

$$\Sigma.\mathsf{Verify}(\mathsf{upk}_i, \hat{C}_{\mathsf{J}_i}, \sigma_{\mathsf{J}_i}) = 1 \quad \wedge \quad \Pi.\mathsf{Verify}(\mathsf{crs}_{\mathsf{O}}, (\hat{C}_{\mathsf{J}_i}, \mathsf{pk}_{\mathsf{O}}, \sigma), \pi_{\mathsf{O}}) = 1.$$

§ We assume that the indexes are in ascending order w.r.t. the time of registration.

Scheme 4.5: Opening procedure.

Note that if multiple users collude and use the same value r upon $\mathsf{Join}^{(1)}$, we always return the first user who registered with this particular value r in Open. Then, Open always returns the signer who initiated the collusion by sharing the r value, which, we think, is the most reasonable choice. Note that this is in line with the BSZ model: traceability only requires that every valid signature can be opened, while not requiring that it opens to one particular user out of the set of colluding users; correctness and non-frameability are defined with respect to honest users and are therefore clearly not influenced.

Additional Hardness Assumption. For our construction, we require a novel assumption which we call computational co-Diffie-Hellman-Inversion (co-CDHI)

assumption. This assumption constitutes a natural assumption in the Type-3 bilinear group setting, which is implied by an assumption from the Uber assumption family [Boy08].

Definition 4.10 (co-CDHI)**.** *The co-CDHI assumption holds relative to* BGGen*, if for all PPT adversaries* $\mathcal{A}$ *there exists a negligible function* $\varepsilon(\cdot)$ *such that:*

$$\Pr\left[\mathsf{BG} \leftarrow \mathsf{BGGen}(1^\kappa, 3),\ a \xleftarrow{R} \mathbb{Z}_p,\ c^\star \leftarrow \mathcal{A}(\mathsf{BG}, g^a, \hat{g}^{1/a})\ :\ c^\star = g^{1/a}\right] \leq \varepsilon(\kappa).$$

To justify its hardness, we state the following additional assumption in the Type-3 bilinear group setting, which falls into the Uber assumption family [Boy08] with $R = \langle 1, 1/b \rangle$, $S = \langle 1, b \rangle$, $T = \langle 1 \rangle$, and $f = b^2$.

Definition 4.11. *Relative to* BGGen *we have that for all PPT adversaries* $\mathcal{A}$ *there exists a negligible function* $\varepsilon(\cdot)$ *such that:*

$$\Pr\left[\mathsf{BG} \leftarrow \mathsf{BGGen}(1^\kappa, 3),\ b \xleftarrow{R} \mathbb{Z}_p,\ \mathbf{c}^\star \leftarrow \mathcal{A}(\mathsf{BG}, g^{1/b}, \hat{g}^b)\ :\ \mathbf{c}^\star = e(g, \hat{g})^{b^2}\right] \leq \varepsilon(\kappa).$$

Lemma 4.4. *If the assumption in Definition 4.11 holds, then also the co-CDHI assumption holds.*

Proof. Assume a co-CDHI adversary $\mathcal{B}$. We obtain a problem instance $g^{1/b}, \hat{g}^b$ relative to BG for the problem underlying the assumption in Definition 4.11, start $\mathcal{B}(\mathsf{BG}, g^{1/b}, \hat{g}^b)$ to obtain $c^\star = g^b$, and output $e(g, \hat{g})^{b^2} \leftarrow e(c^\star, \hat{g}^b)$ with the same probability as $\mathcal{B}$ outputs g^b, i.e., breaks co-CDHI.[5] □

Formal Security Proof of Our Scheme. First, note that our Join ↔ Issue protocol is inherently and trivially concurrently secure: we only have two moves and non-interactive proofs which means that interleaving different Join ↔ Issue instances is impossible. Thus concurrency issues are implicitly covered by our further analysis.

Theorem 4.2. *If* SPS-EQ *is correct,* SoK *is correct, and* Π *is sound, then the group signature scheme described in Scheme 4.2-4.5 is correct.*

Proof (Sketch)*.* Correctness is straight forward to verify by inspection. We only have to take care of one detail: There is the possibility that two honest executions of AddU yield the same value r (which is chosen uniformly at random upon $\mathsf{Join}^{(1)}$). Thus, the probability of two colliding r is negligible. □

Theorem 4.3. *If* Π *is adaptively zero-knowledge,* SoK *is simulatable,* Ω *is* IND-CPA *secure,* SPS-EQ *perfectly adapts signatures, and the DDH assumption holds in* $\mathbb{G}_1$*, then the group signature scheme described in Scheme 4.2-4.5 is* CPA*-full anonymous.*

[5] Note that when writing b as $1/a$ it is immediate that the input distributions the adversary sees in both assumptions are the same.

Theorem 4.4. *If* Π *is adaptively zero-knowledge,* SoK *is simulatable and straight-line* f*-extractable, where* $f : \mathbb{Z}_p \to \mathbb{G}_2$ *is defined as* $r \mapsto \hat{g}^r$*,* Ω *is* $\mathsf{IND\text{-}CCA2}$ *secure,* $\mathsf{SPS\text{-}EQ}$ *perfectly adapts signatures, and the DDH assumption holds in* $\mathbb{G}_1$*, then the group signature scheme described in Scheme 4.2-4.5 is* $\mathsf{CCA2}$*-full anonymous.*

Proof (Anonymity). We prove Theorem 4.3 and 4.4 by showing that the output distributions of the Ch oracle are (computationally) independent of the bit b, where we highlight the parts of the proof which are specific to Theorem 4.4 and can be omitted to prove Theorem 4.3. Therefore, let $q_{\mathsf{Ch}} \leq \mathsf{poly}(\kappa)$ be the number of queries to Ch, $q_{\mathsf{O}} \leq \mathsf{poly}(\kappa)$ be the number of queries to Open, and $q_{\mathsf{SndToU}} \leq \mathsf{poly}(\kappa)$ be the number of queries to SndToU.

Game 0: The original anonymity game.

Game 1: As Game 0, but we run $(\mathsf{crs}_{\mathsf{J}}, \tau_{\mathsf{J}}) \leftarrow \Pi.\mathsf{S}_1(1^\kappa)$ instead of $\mathsf{crs}_{\mathsf{J}} \leftarrow \Pi.\mathsf{Setup}(1^\kappa)$ upon running $\mathsf{GKeyGen}$ and store the trapdoor τ_{J}. Then, we simulate all calls to $\Pi.\mathsf{Proof}$ executed in Join using the simulator (without a witness).

Transition - Game 0 → Game 1: A distinguisher $\mathcal{D}^{0\to1}$ is an adversary against adaptive zero-knowledge of Π, and, therefore, the probability to distinguish Game 0 and Game 1 is negligible, i.e., $|\Pr[S_1] - \Pr[S_0]| \leq \varepsilon_{\mathsf{ZK_J}}(\kappa)$.

Game 2: As Game 1, but we run $(\mathsf{crs}_{\mathsf{O}}, \tau_{\mathsf{O}}) \leftarrow \Pi.\mathsf{S}_1(1^\kappa)$ instead of $\mathsf{crs}_{\mathsf{O}} \leftarrow \Pi.\mathsf{Setup}(1^\kappa)$ upon running $\mathsf{GKeyGen}$ and store the trapdoor τ_{O}. Then, we simulate all calls to $\Pi.\mathsf{Proof}$ in Open using the simulator (without a witness).

Transition - Game 1 → Game 2: A distinguisher $\mathcal{D}^{1\to2}$ is an adversary against adaptive zero-knowledge of Π, and, therefore, the probability to distinguish Game 1 and Game 2 is negligible, i.e., $|\Pr[S_2] - \Pr[S_1]| \leq \varepsilon_{\mathsf{ZK_O}}(\kappa)$.

Game 3: As Game 2, but we run $(\mathsf{crs}_{\mathsf{S}}, \tau_{\mathsf{S}}) \leftarrow \mathsf{SoK.SimSetup}(1^\kappa)$ instead of $\mathsf{crs}_{\mathsf{S}} \leftarrow \mathsf{SoK.Setup}(1^\kappa)$ upon running $\mathsf{GKeyGen}$ and store the trapdoor τ_{S}. Then we simulate all calls to $\mathsf{SoK.Sign}$ using the simulator (without a witness).

Transition - Game 2 → Game 3: A distinguisher $\mathcal{D}^{2\to3}$ is an adversary against simulatability of SoK. Therefore, the distinguishing probability is negligible, i.e., $|\Pr[S_3] - \Pr[S_2]| \leq \varepsilon_{\mathsf{SIM}}(\kappa)$.

Game 4: As Game 3, but instead of setting $(\mathsf{sk}_{\mathsf{O}}, \mathsf{pk}_{\mathsf{O}}) \leftarrow \Omega.\mathsf{KeyGen}(1^\kappa)$ in $\mathsf{GKeyGen}$, we obtain pk_{O} from an $\mathsf{IND\text{-}CPA}$ (resp. $\mathsf{IND\text{-}CCA2}$) challenger and set $\mathsf{sk}_{\mathsf{O}} \leftarrow \perp$.

In the CCA2 case, we additionally maintain secret lists $\mathtt{AU}$ and $\mathtt{OI}$, and upon each call to the SndToU oracle we store $\mathtt{AU}[i] \leftarrow (\mathsf{gsk}_i, \hat{C}_{\mathsf{J}_i}) = (((w, g), \sigma), \hat{C}_{\mathsf{J}_i})$. Then, we simulate the WReg oracle as follows

$\mathsf{WReg}(i, \zeta)$: As the original oracle, but we additionally parse ζ as $(\hat{C}_{\mathsf{J}_i}, \sigma_{\mathsf{J}_i})$. If there exists an index j so that $\mathtt{AU}[j][2] = \hat{C}_{\mathsf{J}_i}$, we parse $\mathtt{AU}[j][1]$ as $((w, g), \sigma)$ and set $\mathtt{OI}[i] \leftarrow (w, \bot)$. If there exists no such index, we obtain $\hat{w}$ using the decryption oracle and set $\mathtt{OI}[i] \leftarrow (\bot, \hat{w})$.

Furthermore, we simulate the Open algorithm within the Open oracle as follows.

$\mathsf{Open}(\mathsf{gpk}, \mathsf{ok}, \mathsf{reg}, m, \sigma)$: First, obtain $\hat{\Psi} = \hat{g}^{\rho}$ using the straight-line f-extractor. Then, obtain the lowest index i where either $e(\sigma_1[1][1], \hat{g}) = e(\sigma_1[1][2], \mathtt{OI}[i][2])$ holds, or $e(\mathtt{OI}[i][1], \hat{\Psi}) = e(\sigma_1[1][1], \hat{g})$ holds. Compute a simulated proof τ and return (i, τ) and $\bot$ if no such index exists.

If the extractor fails at some point, we choose $b \xleftarrow{R} \{0, 1\}$ and return b.

*Transition - Game 3 → Game 4 (*CPA*):* In the CPA case, we do not have to simulate the open oracle, and we only obtain the opening key from an IND-CPA challenger. Thus, this change is conceptual, i.e., $\Pr[S_3] = \Pr[S_4]$.

*Transition - Game 3 → Game 4 (*CCA2*):* By the straight-line f-extractability of the SoK, one can extract a witness ρ in every call to Open with overwhelming probability $1 - \varepsilon_{\mathsf{EXT}}(\kappa)$. Thus, both games proceed identically unless the extraction fails, i.e., $|\Pr[S_3] - \Pr[S_4]| \leq q_{\mathsf{O}} \cdot \varepsilon_{\mathsf{EXT}}(\kappa)$.

Game 5: As Game 4, but we compute the ciphertext $\hat{C}_{\mathsf{J}_i}$ in the Join algorithm (executed within the SndToU oracle) as $\hat{C}_{\mathsf{J}_i} \leftarrow \Omega.\mathsf{Enc}(\mathsf{pk}, \hat{g})$, i.e., with a constant message that is independent of the user.

Transition - Game 4 → Game 5: A distinguisher $\mathcal{D}^{4 \to 5}$ is a distinguisher for the IND-CPA (resp. IND-CCA2) game of Ω, i.e., $|\Pr[S_5] - \Pr[S_4]| \leq q_{\mathsf{SndToU}} \cdot \varepsilon_{\mathsf{CPA}}(\kappa)$ (resp. $|\Pr[S_5] - \Pr[S_4]| \leq q_{\mathsf{SndToU}} \cdot \varepsilon_{\mathsf{CCA2}}(\kappa)$).[6]

Game 6: As Game 5, but we re-add sk_{O}, i.e., we again obtain $(\mathsf{sk}_{\mathsf{O}}, \mathsf{pk}_{\mathsf{O}}) \leftarrow \Omega.\mathsf{KeyGen}(1^{\kappa})$. In the CCA2 case, we again decrypt ourselves within the WReg simulation instead of using the decryption oracle.

Transition - Game 5 → Game 6: This change is conceptual, i.e., $\Pr[S_5] = \Pr[S_6]$.

Game 7: As Game 6, but all calls to $\mathsf{ChgRep}_{\mathcal{R}}(M, \rho, \mathsf{pk}_{\mathcal{R}})$ are replaced by $\mathsf{Sign}_{\mathcal{R}}(M^{\rho}, \mathsf{sk}_{\mathcal{R}})$.

[6] For compactness, we collapsed the q_{SndToU} game changes into a single game change and note that one can straight forwardly unroll this to q_{SndToU} game changes where a single ciphertext is exchanged in each game.

Transition - Game 6 → Game 7: Under perfect adaption of signatures, the output distributions in Game 6 and Game 7 are identical, i.e., $\Pr[S_7] = \Pr[S_6]$.

Game 8: As Game 7, but we modify the Ch oracle as follows. Instead of running $\sigma_1 \leftarrow \mathsf{Sign}_{\mathcal{R}}(\mathsf{gsk}_{i_b}[1]^{\rho}, \mathsf{sk}_{\mathcal{R}})$, we choose $s, t \xleftarrow{R} \mathbb{G}_1$, and compute $\sigma_1 \leftarrow \mathsf{Sign}_{\mathcal{R}}((t, s), \mathsf{sk}_{\mathcal{R}})$.

Transition - Game 7 → Game 8: We claim that $|\Pr[S_7] - \Pr[S_8]| \leq q_{\mathsf{Ch}} \cdot \varepsilon_{\mathsf{DDH}}(\kappa)$. We will below proof this claim separately.

In Game 8, the simulation is independent of the bit b, i.e., $\Pr[S_8] = 1/2$; what remains is to obtain a bound on the success probability in Game 0. In the CPA case, we have that $\Pr[S_0] \leq 1/2 + q_{\mathsf{SndToU}} \cdot \varepsilon_{\mathsf{CPA}}(\kappa) + q_{\mathsf{Ch}} \cdot \varepsilon_{\mathsf{DDH}}(\kappa) + \varepsilon_{\mathsf{ZK_J}}(\kappa) + \varepsilon_{\mathsf{ZK_O}}(\kappa) + \varepsilon_{\mathsf{SIM}}(\kappa)$, which proves Theorem 4.3. In the $\boxed{\mathsf{CCA2}}$ case, we have that $\Pr[S_0] \leq 1/2 + q_{\mathsf{SndToU}} \cdot \varepsilon_{\mathsf{CCA2}}(\kappa) + q_{\mathsf{Ch}} \cdot \varepsilon_{\mathsf{DDH}}(\kappa) + \varepsilon_{\mathsf{ZK_J}}(\kappa) + \varepsilon_{\mathsf{ZK_O}}(\kappa) + \varepsilon_{\mathsf{SIM}}(\kappa) + q_{\mathsf{O}} \cdot \varepsilon_{\mathsf{EXT}}(\kappa)$, which proves Theorem 4.4. □

Proof (of Claim). Below we will show that Game 7 and Game 8 are indistinguishable by introducing further intermediate hybrid games.

Game 7_1: As Game 7, but we introduce a conceptual change which will make the subsequent distribution changes easier to follow. In particular upon each SndToU, we modify the simulation of Join so that we no longer choose $r \xleftarrow{R} \mathbb{Z}_p$ to obtain $(u_i, v) \leftarrow (g^{r \cdot q}, g^q)$, but choose $\zeta \xleftarrow{R} \mathbb{G}_1$ and obtain $(u_i, v) \leftarrow (\zeta^q, g^q)$.

Transition - Game 7 → Game 7_1: This is a conceptual change, i.e., $\Pr[S_7] = \Pr[S_{7_1}]$. Observe that we do not need to know r, as the proofs upon Join are simulated without a witness. Also the user secret keys $\mathsf{gsk}_i = ((w, g), \sigma)$ are exactly the same as honest secret keys.

Game 7_j ($2 \leq j \leq q_{\mathsf{Ch}} + 1$): As Game 7_1, but we modify the Ch oracle as follows. For the first $j-1$ queries, instead of running $\sigma_1 \leftarrow \mathsf{Sign}_{\mathcal{R}}(\mathsf{gsk}_{i_b}[1]^{\rho}, \mathsf{sk}_{\mathcal{R}})$, we choose $s, t \xleftarrow{R} \mathbb{G}_1$, and compute $\sigma_1 \leftarrow \mathsf{Sign}_{\mathcal{R}}((t, s), \mathsf{sk}_{\mathcal{R}})$.

Transition - Game 7_j → Game 7_{j+1} ($1 \leq j \leq q_{\mathsf{Ch}} + 1$): For each transition, we present a hybrid game, which uses a DDH challenger to interpolate between Game 7_j and Game 7_{j+1}. First, we obtain a DDH instance $(g^a, g^b, g^c) \in \mathbb{G}_1^3$ relative to BG. Then we proceed as follows:

- Upon each SndToU, we modify the simulation of Join as follows. Let i be the index of the user to join. We use the random self reducibility of DDH to obtain an independent DDH instance $(r_i, s_i, t_i) \xleftarrow{RSR} (g^a, g^b, g^c)$ and set $\mathtt{CH}[i] \leftarrow (r_i, s_i, t_i)$. Then, we let $(u_i, v) \leftarrow (r_i^q, g^q)$.
- Up to the $j-1$st query to Ch (i.e., for all queries where the answers are already random in Game 7_j), we compute σ_1 by choosing $s, t \xleftarrow{R} \mathbb{G}_1$, and compute $\sigma_1 \leftarrow \mathsf{Sign}_{\mathcal{R}}((t, s), \mathsf{sk}_{\mathcal{R}})$.
- Upon the jth query to Ch, we obtain $(\cdot, s_{i_b}, t_{i_b}) \leftarrow \mathtt{CH}[i_b]$ and set $\sigma_1 \leftarrow \mathsf{Sign}_{\mathcal{R}}((t_{i_b}, s_{i_b}), \mathsf{sk}_{\mathcal{R}})$.

– Starting from the $j+1$st query to Ch (i.e., for all queries where the answers are still honest in Game 7_j), we obtain $(r_{i_b}, \cdot, \cdot) \leftarrow \mathtt{CH}[i_b]$, choose $\rho \xleftarrow{R} \mathbb{Z}_p$ and set $\sigma_1 \leftarrow \mathsf{Sign}_{\mathcal{R}}((r_{i_b}^{\rho}, g^{\rho}), \mathsf{sk}_{\mathcal{R}})$.

In Game 7_j the first $j-1$ answers are already random due to the previous switches. Furthermore, the validity of the DDH instance (g^a, g^b, g^c) provided by the challenger determines whether the answer of Ch for the jth query are for user i_b or random, i.e., if we are in Game j or in Game $j+1$. That is, $|\Pr[S_j] - \Pr[S_{j+1}]| \leq \varepsilon_{\mathsf{DDH}}(\kappa)$.

In Game $7_{q_{\mathsf{Ch}}+1}$ all answers of Ch are random, i.e., this Game is equal to Game 8, i.e., $\Pr[S_8] = \Pr[7_{q_{\mathsf{Ch}}+1}]$. We can conclude the proof by summing over the distinguishing probabilities of all game changes which yields $|\Pr[S_7] - \Pr[S_8]| \leq q_{\mathsf{Ch}} \cdot \varepsilon_{\mathsf{DDH}}(\kappa)$. □

Theorem 4.5. *If* SPS-EQ *is* EUF-CMA *secure, and* Π *is sound, then the group signature scheme described in Scheme 4.2-4.5 is traceable.*

Proof (Traceability). We show that traceability holds using a sequence of games, where we let $q \leq \mathsf{poly}(\kappa)$ be the number of queries to the SndTol oracle.

Game 0: The original traceability game.

Game 1: As Game 0, but we obtain $\mathsf{crs}_{\mathsf{J}}$ from a soundness challenger of Π.

Transition - Game 0 → Game 1: This change is conceptual, i.e., $\Pr[S_0] = \Pr[S_1]$.

Game 2: As Game 1, but after every successful execution of SndTol, we obtain $\hat{w} \leftarrow \Omega.\mathsf{Dec}(\mathsf{sk}_{\mathsf{O}}, C_{\mathsf{J}_i})$ and abort if $e(u_i, \hat{g}) \neq e(v, \hat{w})$.

Transition - Game 0 → Game 1: If we abort we have a valid proof π_{J_i} attesting that $(u_i, v, \hat{C}_{\mathsf{J}_i}, \mathsf{pk}_{\mathsf{O}}) \in L_{\mathsf{R}_{\mathsf{J}}}$, but by the perfect correctness of Ω there exists no ω such that $C_{\mathsf{J}_i} = \Omega.\mathsf{Enc}(\mathsf{pk}_{\mathsf{O}}, \hat{g}^r; \omega) \wedge u_i = v^r$, i.e., $(u_i, v, \hat{C}_{\mathsf{J}_i}, \mathsf{pk}_{\mathsf{O}})$ is actually not in $L_{R_{\mathsf{J}}}$. Thus, both games proceed identically unless the adversary breaks the soundness of Π in one oracle query, i.e., $|\Pr[S_1] = \Pr[S_2]| \leq q \cdot \varepsilon_{\mathsf{S}}(\kappa)$.

Game 3: As Game 2, but we obtain BG and a public key $\mathsf{pk}_{\mathcal{R}}$ from an EUF-CMA challenger of the SPS-EQ. Whenever an SPS-EQ signature is required, the message to be signed is forwarded to the signing oracle provided by the EUF-CMA challenger.

Transition - Game 2 → Game 3: This change is conceptual, i.e., $\Pr[S_2] = \Pr[S_3]$.

If the adversary eventually outputs a valid forgery (m, σ), we know that σ contains an SPS-EQ signature σ_1 for some (g^r, g) such that we have never seen a corresponding $\hat{g}^r$, i.e., there is no entry i in the registration table where $\hat{C}_{\mathsf{J}_i}$ contains $\hat{g}^r$ s.t. $e(\sigma_1[1][1], \hat{g}) = e(\sigma_1[1][2], \hat{g}^r)$ holds. Consequently, σ_1 is a valid SPS-EQ signature for an unqueried equivalence class and we have that $\Pr[S_3] \leq \varepsilon_{\mathsf{F}}(\kappa)$. This yields $\Pr[S_0] \leq \varepsilon_{\mathsf{F}}(\kappa) + q \cdot \varepsilon_{\mathsf{S}}(\kappa)$, which proves the theorem. □

Theorem 4.6. *If Π is sound and adaptively zero-knowledge, SoK is simulatable and extractable, Σ is $\mathsf{EUF\text{-}CMA}$ secure, Ω is perfectly correct, and the co-CDHI assumption holds, then the group signature scheme described in Scheme 4.2-4.5 is non-frameable.*

Proof (Non-frameability). We prove non-frameability using a sequence of games. Thereby we let the number of users in the system be $q_{\mathsf{u}} \leq \mathsf{poly}(\kappa)$.

Game 0: The original non-frameability game.

Game 1: As Game 0, but we guess the index $i^\star$ that will be attacked by the adversary. If the adversary attacks another index, we abort.

Transition - Game 0 → Game 1: The winning probability in Game 1 is the same as in Game 0, unless an abort event happens, i.e., $\Pr[S_1] = \Pr[S_0] \cdot 1/q_{\mathsf{u}}$.

Game 2: As Game 1, but we run $(\mathsf{crs}_{\mathsf{J}}, \tau_{\mathsf{J}}) \leftarrow \Pi.\mathcal{S}_1(1^\kappa)$ instead of $\mathsf{crs}_{\mathsf{J}} \leftarrow \Pi.\mathsf{Setup}(1^\kappa)$ upon running $\mathsf{GKeyGen}$ and store the trapdoor τ_{J}. Then, we simulate all calls to $\Pi.\mathsf{Proof}$ in Join using the simulator (without a witness).

Transition - Game 1 → Game 2: A distinguisher $\mathcal{D}^{1\to 2}$ is an adversary against adaptive zero-knowledge of Π, and, therefore, the probability to distinguish Game 1 and Game 2 is negligible, i.e., $|\Pr[S_2] - \Pr[S_1]| \leq \varepsilon_{\mathsf{ZK}_{\mathsf{J}}}(\kappa)$.

Game 3: As Game 2, but we obtain $\mathsf{crs}_{\mathsf{O}}$ from a soundness challenger upon running $\mathsf{GKeyGen}$.

Transition - Game 2 → Game 3: This change is conceptual, i.e., $\Pr[S_3] = \Pr[S_2]$.

Game 4: As Game 3, but we setup the SoK in simulation mode, i.e., we run $(\mathsf{crs}_{\mathsf{S}}, \tau_{\mathsf{S}}) \leftarrow \mathsf{SoK.SimSetup}(1^\kappa)$ instead of $\mathsf{crs}_{\mathsf{S}} \leftarrow \mathsf{SoK.Setup}(1^\kappa)$ upon running $\mathsf{GKeyGen}$ and store the trapdoor τ_{S}. Then, we simulate all calls to $\mathsf{SoK.Sign}$ using the simulator, i.e., without a witness.

Transition - Game 3 → Game 4: A distinguisher $\mathcal{D}^{3\to 4}$ is an adversary against simulatability of SoK. Therefore, the distinguishing probability is negligible, i.e., $|\Pr[S_4] - \Pr[S_3]| \leq \varepsilon_{\mathsf{SIM}}(\kappa)$.

Game 5: As Game 4, but we choose the values $r, q \xleftarrow{R} \mathbb{Z}_p$ used in the Join algorithm (executed within the SndToU oracle) when queried for user with index $i^\star$ beforehand and let $(u_{i^\star}, v_{i^\star})$ denote (g^{rq}, g^q). Then, on every Join (within SndToU) for a user $i \neq i^\star$ we check whether we have incidentally chosen the same class as for user $i^\star$. This check is implemented as follows: with r_i being the value for r chosen upon Join for user i, we check whether $u_{i^\star} = v_{i^\star}^{r_i}$ (note that this check does not require to know the discrete logarithms q and r for user $i^\star$).

Transition - Game 4 → Game 5: Both games proceed identically unless we have to abort. An abort happens with probability $\varepsilon_{\mathsf{guess}}(\kappa) = q_{\mathsf{u}}/p-1$ and we have that $|\Pr[S_4] - \Pr[S_5]| \leq \varepsilon_{\mathsf{guess}}(\kappa)$.

Game 6: As Game 5, but we obtain a co-CDHI instance $(g^a, \hat{g}^{1/a})$ relative to BG and choose $\tau \xleftarrow{R} \mathbb{Z}_p$. Then, we modify the Join algorithm (executed within the SndToU oracle) when queried for user with index $i^\star$ as follows. We set $(u_{i^\star}, v_{i^\star}) \leftarrow (g^\tau, g^a)$, and compute $\hat{C}_{\mathsf{J}_{i^\star}} \leftarrow \Omega.\mathsf{Enc}(\mathsf{pk}_{\mathsf{O}}, (\hat{g}^{1/a})^\tau)$ and store τ. On successful execution we set $\mathsf{gsk}_{i^\star} \leftarrow ((u_{i^\star}, v_{i^\star}), \sigma')$. Note that $\pi_{\mathsf{J}_{i^\star}}$ as well as the signatures in the GSig oracle are already simulated, i.e., the discrete log of no v_i value is required to be known to the environment.

Transition - Game 5 → Game 6: Since τ is uniformly random, we can write it as $\tau = ra$ for some $r \in \mathbb{Z}_p$. Then it is easy to see that the game change is conceptual, i.e., $\Pr[S_5] = \Pr[S_4]$.

Game 7: As Game 6, but for every forgery output by the $\mathcal{A}$, we extract $\rho \leftarrow \mathsf{SoK.Extract}(\mathsf{crs}_{\mathsf{S}}, \tau_{\mathsf{S}}, (g, \sigma_1[1][2]), \sigma_1 || m, \sigma_2)$ and abort if the extraction fails.

Transition - Game 6 → Game 7: By the extractability of the SoK, one can extract a witness ρ with overwhelming probability $1-\varepsilon_{\mathsf{EXT}}(\kappa)$. Thus, both games proceed identically unless the extractor fails $|\Pr[S_6] - \Pr[S_7]| \leq \varepsilon_{\mathsf{EXT}}(\kappa)$.

Game 8: As Game 7, but we further modify the Join algorithm when queried for user with index $i^\star$ (executed within the SndToU oracle) as follows. Instead of choosing $(\mathsf{usk}_{i^\star}, \mathsf{upk}_{i^\star}) \leftarrow \mathsf{UKeyGen}(1^\kappa)$, we engage with an $\mathsf{EUF\text{-}CMA}$ challenger, obtain $\mathsf{upk}_{i^\star}$ and set $\mathsf{usk}_{i^\star} \leftarrow \emptyset$. If any signature is required, we obtain it using the oracle provided by the $\mathsf{EUF\text{-}CMA}$ challenger.

Transition Game 7 → Game 8: This change is conceptual, i.e., $\Pr[S_7] = \Pr[S_6]$.

At this point we have three possibilities if $\mathcal{A}$ outputs a valid forgery.

1. If a signature for $\hat{C}_{\mathsf{J}_{i^\star}}$ was never requested, $\mathcal{A}$ is an $\mathsf{EUF\text{-}CMA}$ forger for Σ and the forgery is $(\hat{C}_{\mathsf{J}_{i^\star}}, \sigma_{\mathsf{J}_{i^\star}})$. The probability for this to happen is upper bounded by $\varepsilon_{\mathsf{f}}(\kappa)$.

2. Otherwise, we know that $\hat{C}_{\mathsf{J}_{i^\star}}$ is honestly computed by the environment and—by the perfect correctness of Ω—thus contains $\hat{g}^{\tau/a}$, which leaves us two possibilities:

   (a) If $e(\sigma[1][1], \hat{g}) = e(\sigma[1][2], \hat{g}^{\tau/a})$, $\mathcal{A}$ is an adversary against co-CDHI, since we can obtain $(((g^{\tau \cdot 1/a}, g), \sigma')) \leftarrow \mathsf{ChgRep}_{\mathcal{R}}(\sigma_1, \rho^{-1}, \mathsf{pk}_{\mathcal{R}})$ and use τ to output $(g^{\tau \cdot 1/a})^{\tau^{-1}} = g^{1/a}$. The probability for this to happen is upper bounded by $\varepsilon_{\mathsf{co\text{-}CDHI}}(\kappa)$.

   (b) If $e(\sigma[1][1], \hat{g}) \neq e(\sigma[1][2], \hat{g}^{\tau/a})$, $\mathcal{A}$ has produced an opening proof for a statement which is actually not in $L_{R_{\mathsf{O}}}$. The probability for this to happen is upper bounded by $\varepsilon_{\mathsf{S}}(\kappa)$.

Taking the union bound we obtain $\varepsilon_{\mathsf{nf8}}(\kappa) \leq \varepsilon_{\mathsf{f}}(\kappa) + \varepsilon_{\mathsf{co\text{-}CDHI}}(\kappa) + \varepsilon_{\mathsf{S}}(\kappa)$, which yields the following bound for the success probability in Game 1: $\Pr[S_0] \leq$

$q \cdot (\varepsilon_{\mathsf{nf8}}(\kappa) + \varepsilon_{\mathsf{ZK_J}}(\kappa) + \varepsilon_{\mathsf{SIM}}(\kappa) + \varepsilon_{\mathsf{guess}}(\kappa) + \varepsilon_{\mathsf{EXT}}(\kappa))$, which is negligible.[7] □

Theorem 4.7. *If Ω is perfectly correct, and Σ is* EUF-CMA *secure, then the group signature scheme described in Scheme 4.2-4.5 is weakly opening sound.*

Proof (Sketch). Upon honestly executing Join for users i and j, the probability that their r (resp. $\hat{w}$) values collide is negligible. The perfect correctness of Ω and the EUF-CMA security of Σ thus uniquely determine user i as the signer of σ with overwhelming probability. Then, it is easy to see that an adversary against weak opening soundness is an adversary against soundness of Π. □

4.2.3 Instantiation in the ROM

To compare our approach to existing schemes regarding signature size and computational effort upon signature generation and verification, we present the sign and verification algorithms for an instantiation of our scheme with the SPS-EQ from [HS14, FHS15a, FHS15b], whose security is shown to hold in the generic group model. We instantiate SoKs in the ROM by applying the transformation from [FKMV12] to Fiat-Shamir (FS) transformed Σ-protocols.

Before we introduce the approaches to obtain CPA-fully (resp. CCA2-fully) anonymous instantiations, we recall that the group signing key gsk_i consists of a vector of two group elements $(w, g) \in (\mathbb{G}_1^*)^2$ and an SPS-EQ signature $\sigma \in \mathbb{G}_1 \times \mathbb{G}_1^* \times \mathbb{G}_2^*$ on this vector. Randomization of a gsk_i with a random value $\rho \in \mathbb{Z}_p^*$, i.e., $\mathsf{ChgRep}_{\mathcal{R}}$, requires 4 multiplications in $\mathbb{G}_1$ and 1 multiplication in $\mathbb{G}_2$. Verification of an SPS-EQ signature on gsk_i requires 5 pairings.

We note that the proofs performed within Join and Open can straight forwardly be instantiated using standard techniques. Therefore, and since they are neither required within Sign nor Verify, we do not discuss instantiations here.

CPA-Full Anonymity. In the following, we show how Sign and Verify are instantiated in the CPA-full anonymity setting. Therefore, let $H : \{0,1\}^* \to \mathbb{Z}_p$ be a random oracle and let x be the proven statement (which is implicitly defined by the scheme):

$\mathsf{Sign}(\mathsf{gpk}, \mathsf{gsk}_i, m)$: Parse gsk_i as $((w, g), \sigma)$, choose $\rho \xleftarrow{R} \mathbb{Z}_p$, compute $\sigma_1 = ((w', g'), \sigma') \leftarrow \mathsf{ChgRep}_{\mathcal{R}}(\mathsf{gsk}_i, \rho, \mathsf{pk}_{\mathcal{R}})$. Choose $\nu \xleftarrow{R} \mathbb{Z}_p$, compute $n \leftarrow g^{\nu}$, $c \leftarrow H(n||\sigma_1||m||x)$, $z \leftarrow \nu + c \cdot \rho$, set $\sigma_2 \leftarrow (c, z)$, and return $\sigma \leftarrow (\sigma_1, \sigma_2)$.

$\mathsf{Verify}(\mathsf{gpk}, m, \sigma)$: Parse σ as $(\sigma_1, \sigma_2) = (((w', g'), \sigma), (c, z))$, return 0 if $\mathsf{Verify}_{\mathcal{R}}(\sigma_1, \mathsf{pk}_{\mathcal{R}}) = 0$. Otherwise compute $n \leftarrow {}^{g^z}/_{g'^c}$ and check whether $c = H(n||\sigma_1||m||x)$ holds. If so return 1 and 0 otherwise.

[7] We note that we could also write the three cases in the final step as three additional game changes where we abort upon the respective forgeries. However, we opted for this more compact presentation, which also gives us the same bound.

Since the used Σ-protocol is a standard proof of knowledge of the discrete logarithm $\log_g g'$, it is easy to see that applying the transformations from [FKMV12] yields a SoK in the ROM with the properties we require. All in all, group signatures contain 4 elements in $\mathbb{G}_1$, 1 element in $\mathbb{G}_2$ and 2 elements in $\mathbb{Z}_p$. Counting only the expensive operations, signing costs 5 multiplications in $\mathbb{G}_1$ and 1 multiplication in $\mathbb{G}_2$, and verification costs 2 multiplications in $\mathbb{G}_1$ and 5 pairings.

CCA2-Full Anonymity. CCA2-full anonymity requires straight-line extractable SoKs, as standard rewinding would lead to an exponential blowup in the reduction (cf. [BFW15]). One possibility would be to rely on the rather inefficient approach to straight-line extraction due to Fischlin [Fis05]. However, as we do not need to straight-line extract the full witness w, but it is sufficient to straight-line extract an image of w under a one-way function $f : \rho \mapsto \hat{g}^{\rho}$, we can use the notion of straight-line f-extractable SoKs as recently proposed by Cerulli et al. [BCC+15]. This allows us to still use the FS paradigm with good efficiency. The construction uses the generic conversion in [FKMV12, BPW12]. The generic trick in [BCC+15] to obtain straight-line f-extractability is by computing an extractable commitment to the image of the witness w under a function f with respect to an extraction key in the CRS and proving consistency with the witness.[8]

For straight-line extractability, we let $\hat{y}$ be a public key for the ElGamal variant in $\mathbb{G}_2$ from [BCC+15], which is generated upon SoK.Setup and represents the CRS of SoK. SoK.SimSetup additionally returns τ such that $\hat{y} = \hat{g}^{\tau}$. Furthermore, let x be the proven statement (implicitly defined by the scheme and the generic compiler). Then Sign and Verify are instantiated as follows, where $H : \{0,1\}^* \to \mathbb{Z}_p$ is modelled as a random oracle:

$\mathsf{Sign}(\mathsf{gpk}, \mathsf{gsk}_i, m)$: Parse gsk_i as $((w, g), \sigma)$, choose $\rho \xleftarrow{R} \mathbb{Z}_p$, compute $\sigma_1 = ((w', g'), \sigma') \leftarrow \mathsf{ChgRep}_{\mathcal{R}}(\mathsf{gsk}_i, \rho, \mathsf{pk}_{\mathcal{R}})$. Choose $u, \nu, \eta \xleftarrow{R} \mathbb{Z}_p$, compute $(\hat{C}_1, \hat{C}_2) = (\hat{y}^u, \hat{g}^{\rho}\hat{y}^u)$, $n \leftarrow g^{\nu}$, $\hat{m}_1 \leftarrow \hat{y}^{\eta}$, $\hat{m}_2 \leftarrow \hat{g}^{(\nu+\eta)}$, $c \leftarrow H(n||\hat{m}_1||\hat{m}_2||\sigma_1||m||x)$, $z_1 \leftarrow \nu + c \cdot \rho$, $z_2 \leftarrow \eta + c \cdot u$, set $\sigma_2 \leftarrow (\hat{C}_1, \hat{C}_2, c, z_1, z_2)$, and return $\sigma \leftarrow (\sigma_1, \sigma_2)$.

$\mathsf{Verify}(\mathsf{gpk}, m, \sigma)$: Parse σ as $(\sigma_1, \sigma_2) = (((w', g'), \sigma), (c, z_1, z_2))$, return 0 if $\mathsf{Verify}_{\mathcal{R}}(\sigma_1, \mathsf{pk}_{\mathcal{R}}) = 0$. Otherwise compute $n \leftarrow g_1^{z}/g'^{c}$, $\hat{m}_1 \leftarrow \hat{y}^{z_2}/\hat{C}_1^{c}$, $\hat{m}_2 \leftarrow \hat{g}^{(z_1+z_2)}/\hat{C}_2^{c}$, and check whether $c = H(n||\hat{m}_1||\hat{m}_2||\sigma_1||m||x)$ holds. If so return 1 and 0 otherwise.

Note that we additionally require the Σ-protocol to provide quasi-unique responses [Fis05], i.e., given an accepting proof it should be computationally infeasible to find a new valid response for that proof, in order for the compiler in [BCC+15] to apply.

Lemma 4.5. *The above Σ-protocol is perfectly complete, SHVZK, 2-special-sound and has quasi-unique responses.*

[8] Note that one can still obtain the full witness w using a rewinding extractor.

Proof. We investigate all the properties below.

Perfect Completeness. Is straight forward to verify and omitted.

SHVZK. We describe a simulator which outputs transcripts being indistinguishable from real transcripts. First, it chooses $g' \xleftarrow{R} \mathbb{G}_1, \hat{C}_1 \xleftarrow{R} \mathbb{G}_2, \hat{C}_2 \xleftarrow{R} \mathbb{G}_2$. While g' and $\hat{C}_1$ are identically distributed as in a real transcript, the random choice of $\hat{C}_2$ is not detectable under DDH in $\mathbb{G}_2$ which holds in the SXDH setting (more generally under IND-CPA of the used encryption scheme). Then, the simulator chooses $z_1, z_2, c \xleftarrow{R} \mathbb{Z}_p$ and computes $n \leftarrow {g^{z_1}}/{g'^c}$, $\hat{m}_1 \leftarrow {\hat{y}^{z_2}}/{\hat{C}_1^c}$, $\hat{m}_2 \leftarrow {\hat{g}^{(z_1+z_2)}}/{\hat{C}_2^c}$. It is easy to see that the transcript $(g', \hat{C}_1, \hat{C}_2, n, \hat{m}_1, \hat{m}_2, z_1, z_2, c)$ represents a valid transcript and its distribution is computationally indistinguishable from a real transcript.

2-*Special Soundness.* Let us consider that we have two accepting answers (z_1, z_2, c) and (z_1', z_2', c') from the prover for distinct challenges $c \neq c'$. Then we have that

$$z_1 - c \cdot \rho = z_1' - c' \cdot \rho \text{ and } z_2 - c \cdot u = z_2' - c' \cdot u,$$

and extract a witness as $\rho \leftarrow \frac{z_1 - z_1'}{c - c'}$, $u \leftarrow \frac{z_2 - z_2'}{c - c'}$.

Quasi-Unique Responses. The answers z_1 and z_2 are uniquely determined by the word $\hat{y}$, g', $\hat{C}_1$, $\hat{C}_2$, the commitments n, $\hat{m}_1$, $\hat{m}_2$ as well as the challenge c (and thus the verification equation). □

Lemma 4.6. *Applying the generic conversions from [FKMV12] to the Fiat-Shamir transformed version of the above Σ-protocol with the setup* SoK.Setup *as described in Section 4.2.3 produces a signature of knowledge in the random oracle model, that is extractable and straight-line f-extractable.*

The proof is analogous to [BCC+15], but we restate it for completeness.

Proof. For simulatability, we observe that the CRS output by SoK.SimSetup is identical to the CRS output by SoK.Setup and SoK.SimSign programs the random oracle to simulate proofs. Simulatability then follows from SHVZK. For extractability we rely on rewinding, 2-special soundness and quasi-unique responses, using the results from [FKMV12]. For straight-line f-extractability, we use the trapdoor τ to decrypt $(\hat{C}_1, \hat{C}_2)$ in the proof transcript and obtain $\hat{g}^\rho = f(\rho)$. □

Switching Groups. The protocol presented above requires more operations in the more expensive group $\mathbb{G}_2$ than in $\mathbb{G}_1$. As we work in the SXDH setting, we can simply switch the roles of $\mathbb{G}_1$ and $\mathbb{G}_2$ and thus all elements in $\mathbb{G}_1$ to $\mathbb{G}_2$ and vice versa. This allows us to trade computational efficiency for signature size.

4.2.4 Evaluation and Discussion

Finally, we discuss our work in the light of some recent concurrent and independent work and provide a performance evaluation.

The [BCC+16] Model. In independent and concurrent work, a new model for fully-dynamic group signatures was proposed by Bootle et al. in [BCC+16]. Bootle et al. address maliciously generated issuer and opener keys, include the notion of opening soundness from [SSE+12] and formally model revocation by means of epochs. Although we target security in a different model, we want to briefly put our construction in context of their recent model.

In our scheme, one can straight forwardly incorporate the requirement to support maliciously generated keys in the fashion of [BCC+16] by extending the actual public keys of issuer and opener by a (straight-line extractable) zero-knowledge proof of knowledge of the respective secret issuer and opener key.

For a practical revocation approach, it seems to be reasonable to choose a re-issuing based approach, i.e., to set up a new group after every epoch, as also used in [BCC+16]. Their group signature construction being secure in their model builds upon accountable ring signatures [BCC+15]. It comes at the cost of a group public key size linear in the number of group members as well as a signature size logarithmic in the number of group members,[9] and the revocation related re-issuing requires every group member to obtain the new group public key. Applying the same revocation approach to our scheme yields public keys as well as signatures of constant size, and re-issuing requires each group member which is still active to re-join the new group.

While our scheme provides weak opening soundness, achieving the stronger notion for our scheme (where the opening authority may be malicious) would require the opening authority to additionally prove that the opened index i corresponds to the lowest index in reg so that the respective entry together with the signature in question satisfies the relation R_{O}. Such a proof could efficiently be instantiated using non-interactive plaintext in-equality proofs as in [c4]. Nevertheless, we opted to stick with weak opening soundness because: (1) The only benefit of strong opening soundness would be to also cover dishonest opening authorities, while we believe that assuming the opening authority's honesty—given its power to deanonymize every user—is a crucial and very reasonable assumption. (2) Even [SSE+12], who introduced opening soundness emphasize that already weak opening soundness addresses all the attacks that motivated opening soundness in the first place. (3) Strong opening soundness would unnecessarily degrade the simplicity of our scheme.

Performance Evaluation and Comparison. To underline the practical efficiency of our approach, we provide a comparison of our ROM instantiation with other schemes in the ROM. In particular we use two schemes who follow the approach of Bichsel et al., i.e., [BCN+10, PS16], which provide less desirable anonymity guarantees (denoted CCA^-), and the well known BBS scheme [BBS04] (with and without precomputations) providing CPA-full anonymity. We note that we use the plain BBS scheme for comparison, which does not even provide non-frameability and the non-frameable version would be even more

[9] We note that there is a recent accountable ring signature scheme [LZCS16], which enables constant size signatures.

expensive. Moreover, we use the group signature scheme with the shortest known signatures [DP06] (with and without precomputations) being secure in the strong BSZ model and thus providing CCA2-full anonymity. Finally, we also compare our scheme to the recent CCA2-fully anonymous scheme by Libert et al. [LMPY16] which is secure in the ROM under SXDH.[10]

In Table 4.1 we provide a comparison of the estimated efficiency in a 254bit BN-pairing setting, where we highlight the values where our scheme is currently the best known scheme among other existing schemes providing the same security guarantees. Our estimations are based on performance values on an ARM-Cortex-M0+ with drop-in hardware accelerator [UW14]. This processor is small enough to be suited for smart cards or wireless sensor nodes [UW14]. Table 4.2 provides an abstract comparison regarding signature size, computational costs, and the type of the underlying hardness assumption.

Scheme	Anon.	Signature Size	Sign	Verify
[BCN+10]	CCA$^-$	1273bit	351ms	1105ms
[PS16]	CCA$^-$	1018bit	318ms	777ms
[BBS04]	CPA	2289bit	1545ms	2092ms
[BBS04] (prec.)	CPA	2289bit	1053ms	1600ms
This paper	CPA	**2037bit**	**266ms**	**886ms**
This paper	CCA2	3309bit	**771ms**	**1290ms**
This paper (switch)	CCA2	3563bit	**703ms**	**1154ms**
[DP06]	CCA2	2290bit	1380ms	2059ms
[DP06] (prec.)	CCA2	2290bit	1020ms	1353ms
[LMPY16]	CCA2	2547bit	1688ms	2299ms

Table 4.1: Estimations based on a BN-pairing implementation on an ARM-Cortex-M0+ with drop-in hardware accelerator, operating at 48MHz [UW14]. The performance figures using 254-bit curves are 33ms-101ms-252ms-164ms ($\mathbb{G}_1$-$\mathbb{G}_2$-$\mathbb{G}_T$-pairing). For the estimation of signature sizes, we use 255bit for elements in $\mathbb{G}_1$, 509bit for elements in $\mathbb{G}_2$ and 254bit for elements in $\mathbb{Z}_p$. We note that [BBS04] is defined for a Type-2 pairing setting, which means that our performance estimation for this scheme is rather optimistic and likely to be worse in practice. The bold values highlight where our schemes are currently the fastest and have the shortest signatures.

Computational Efficiency. When comparing our CPA-fully anonymous scheme as well as our CCA2-fully anonymous scheme to other schemes providing the same anonymity guarantees, ours are the *by now fastest ones regarding signature generation and verification costs.* While some of the schemes used for comparison use slightly less progressive assumptions, it seems that very good performance

[10] Camenisch and Groth [CG04] report that there are implementation results in a Master's thesis by Hansen and Pagels, which show that their strong RSA based group signature scheme slightly outperforms [BBS04]. Since we do not have timing values for the strong RSA setting on this particular platform, we can not include their scheme in the comparison. However, since there is a large gap between the efficiency of [BBS04] and our scheme, we also expect to achieve better efficiency than [CG04].

Scheme	Anon.	Signature Size	Sign	Verify	Assumption Type
[BCN+10]	CCA$^-$	$3\mathbb{G}_1 + 2\mathbb{Z}_p$	$1\mathbb{G}_T + 3\mathbb{G}_1$	$5\mathrm{P} + 1\mathbb{G}_T + 1\mathbb{G}_1$	Interactive
[PS16]	CCA$^-$	$2\mathbb{G}_1 + 2\mathbb{Z}_p$	$1\mathbb{G}_T + 2\mathbb{G}_1$	$3\mathrm{P} + 1\mathbb{G}_T + 1\mathbb{G}_1$	GGM
[BBS04]	CPA	$3\mathbb{G}_1 + 6\mathbb{Z}_p$	$3\mathrm{P} + 3\mathbb{G}_T + 9\mathbb{G}_1$	$5\mathrm{P} + 4\mathbb{G}_T + 8\mathbb{G}_1$	q-Type (non-static)
[BBS04] (prec.)	CPA	$3\mathbb{G}_1 + 6\mathbb{Z}_p$	$3\mathbb{G}_T + 9\mathbb{G}_1$	$4\mathbb{G}_T + 8\mathbb{G}_1$	q-Type (non-static)
This paper	CPA	$1\mathbb{G}_2 + 4\mathbb{G}_1 + 2\mathbb{Z}_p$	$1\mathbb{G}_2 + 5\mathbb{G}_1$	$5\mathrm{P} + 2\mathbb{G}_1$	GGM
This paper	CCA2	$3\mathbb{G}_2 + 4\mathbb{G}_1 + 3\mathbb{Z}_p$	$6\mathbb{G}_2 + 5\mathbb{G}_1$	$5\mathrm{P} + 4\mathbb{G}_2 + 2\mathbb{G}_1$	GGM
This paper (switch)	CCA2	$4\mathbb{G}_2 + 3\mathbb{G}_1 + 3\mathbb{Z}_p$	$5\mathbb{G}_2 + 6\mathbb{G}_1$	$5\mathrm{P} + 2\mathbb{G}_2 + 4\mathbb{G}_1$	GGM
[DP06]	CCA2	$4\mathbb{G}_1 + 5\mathbb{Z}_p$	$3\mathrm{P} + 3\mathbb{G}_T + 4\mathbb{G}_1$	$5\mathrm{P} + 4\mathbb{G}_T + 7\ \mathbb{G}_1$	q-Type (non-static)
[DP06] (prec.)	CCA2	$4\mathbb{G}_1 + 5\mathbb{Z}_p$	$3\mathbb{G}_T + 8\mathbb{G}_1$	$1\mathrm{P} + 3\mathbb{G}_T + 2\mathbb{G}_2 + 7\mathbb{G}_1$	q-Type (non-static)
[LMPY16]	CCA2	$7\mathbb{G}_1 + 3\mathbb{Z}_p$	$4\mathrm{P} + 2\mathbb{G}_T + 16\mathbb{G}_1$	$8\mathrm{P} + 3\mathbb{G}_T + 7\mathbb{G}_1$	Standard

Table 4.2: Comparison of related group signature schemes in the ROM regarding signature size, signing and verification cost, and required hardness assumptions, where, in terms of computational costs, we only count the expensive operations in $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$ as well as the pairings. The values for [BCN+10] and [PS16] are taken from [PS16]. We use 'CCA^-' to denote anonymity in the sense of [BCN+10] and note that precomputation in [BBS04, DP06] requires to store extra elements in $\mathbb{G}_T$.

requires more progressive assumptions. When looking for instance at the most compact CCA2-fully anonymous group signatures in the standard model under standard assumptions (SXDH and a generalization of DLIN to groups with an asymmetric pairing) by Libert et al. [LPY15], signature sizes in the best case will have 30 $\mathbb{G}_1$ and 14 $\mathbb{G}_2$ elements ($\approx$ 15000 bit when taking the setting in Table 4.1), large public keys and computation times that are far from being feasible for resource constrained devices.

Regarding signature generation, we want to emphasize that our CPA-fully anonymous instantiation is the fastest among all schemes used for comparison (even among the ones providing CCA^- anonymity), and, to the best of our knowledge, *the fastest among all existing schemes*. This is of particular importance since signature generation is most likely to be executed on a constrained device. Regarding signature verification our CPA-fully anonymous instantiation is only outperformed by the CCA^- anonymous instantiation in [PS16].

Signature Size. Comparing schemes providing the same anonymity guarantees, our CPA-fully anonymous instantiation even provides shorter signature sizes than the popular BBS scheme [BBS04] and, to the best of our knowledge, the *shortest signature sizes among all* CPA*-fully anonymous schemes*. Regarding CCA2-fully anonymous schemes, it seems that gained efficiency in the "without encryption" paradigm comes at the cost of larger signatures compared to instantiations following the SEP paradigm. It is interesting to note that the schemes in the vein of Bichsel et al. providing only CCA^- anonymity have the smallest signatures among all schemes.

5

Signatures with Data Privacy

Our results on signatures providing privacy features with respect to the signed data are manifold and include various variants of malleable signatures. In Section 5.1, we present our unified model for redactable signatures together with three generic constructions of redactable signature schemes. Our constructions use indistinguishable accumulators (Section 3.1) as a central tool. Then, in Section 5.2 we introduce a framework which extends the privacy features of redactable signatures in various directions. We present two instantiations: the first one is a black-box construction and is more of theoretical interest, while the second achieves very good practical efficiency by building upon one of the generic constructions of redactable signatures from Section 5.1 and exploiting a particular type of key-homomorphism (Section 3.2). In Section 5.3, we switch our focus from redactable signatures to the related concept of sanitizable signatures. In particular, we strengthen the privacy notion for extended sanitizable signature schemes [CJ10] and provide a black-box construction of such schemes from standard sanitizable signature schemes [BFF+09] and any indistinguishable accumulator scheme (Section 3.1). Finally, in Section 5.4 we tackle more expressive ways of modifying signed data in a controlled way. In particular, we present a framework for multi-source data aggregation scenarios, where a semi-trusted aggregator can collect signed data items from different sources, and then report authentic evaluations of linear functions on those signed data items to a receiver.

5.1 Generalizing Redactable Signatures

A redactable signature scheme (RS) allows any party to *remove* parts of a signed message such that the corresponding signature σ can be updated without the signers' secret key sk. The so derived signature $\mathring{\sigma}$ then still verifies under the signer's public key pk. This separates RSs from standard digital signatures, which prohibit *any* alteration of signed messages. Such a primitive comes in handy in use cases where only parts of the signed data are required, but initial origin authentication must still hold and re-signing is not possible or too expensive. One real-world application scenario is privacy-preserving handling of patient data [BBM09, BB12, SR10, WHT+12]. For instance, identifying information in a patient's record can be redacted for processing during accounting.

Related Work. RSs have been introduced in [JMSW02, SBZ01]. Their ideas have been extended to address special data-structures such as trees [BBD+10, SPB+12a] and graphs [KB13]. While the initial idea was that redactions are public, the notion of accountable RSs appeared recently [PS15]. Here, the redactor becomes a designated party which can be held accountable for redactions. Further, RSs with dependencies between elements have been introduced and discussed in [BBM09]. Unfortunately, their work neither introduces a formal security model nor provides a security analysis for their construction. Consecutive redaction control allows intermediate redactors to prohibit further redactions by subsequent ones [MHI06, MIM+05, SPB+12b].

Much more work on RSs exists. However, they do not use a common security model and most of the presented schemes do not provide the important security property denoted as transparency [BBD+10]. As an example, [HHH+08, KB13, WHT+12] are not transparent in our model. In such non-transparent constructions, a third party can potentially deduce statements about the original message from a redacted message-signature pair. In particular, their schemes allow to see where a redaction took place which might be unwanted in certain applications.

Ahn et al. [ABC+12] introduced the notion of statistically unlinkable RSs as a stronger privacy notion. Their scheme only allows for quoting instead of arbitrary redactions, i.e., redactions are limited to the beginning and the end of an ordered list. Moreover, [ABC+12] only achieves the weaker and less common notion of selective unforgeability. Lately, even stronger privacy notions have been proposed in [ALP12, ALP13] in the context of the framework of $\mathcal{P}$-homomorphic signatures. There also exists a huge amount of related yet different signature primitives, where we refer the reader to [d2] for a comprehensive overview of the state of the art.

Motivation. RSs have many applications. In particular, minimizing signed data before passing it to other parties makes RSs an easy to comprehend privacy enhancing tool. However, the need for different security models and different data structures prohibits an easy integration into applications that require such privacy features, as RSs do not offer a flexible, widely applicable framework. While the model of RSs for sets (e.g. [MHI06]) can protect unstructured data such as votes, it is, e.g., unclear if it can be used for multi-sets. For ordered lists (such as

a text) this already becomes more difficult: should one only allow quoting (i.e., redactions at the beginning and/or the end of a text) or general redactions? For trees (such as data-bases, XML or JSON), we have even more possibilities: only allow leaf-redactions [BBD+10], or leaves and inner nodes [KB13], or even allow to alter the structure [PSdMP12]. Furthermore, over the years more sophisticated features such as dependencies, fixed elements and redactable structure appeared. They complicate the specialized models even more.

We want to abandon the necessity to invent specialized security models tailored to specific use cases and data-structures. Namely, we aim for a framework that generalizes away details and covers existing approaches. At the same time we want to keep the model compact and understandable. We aim at RSs to become generally applicable to the whole spectrum of existing use cases. In addition, we explicitly want to support the trend to allow the signer to limit the power of redactors [KL06, CJ10, c10]. To demonstrate the applicability of our framework, we present three new constructions which hide the length of the original message, the positions of redactions, and the fact that a redaction has even happened.

Contribution. Our contribution is manifold.

- Existing work focuses on messages' representations in only a specific data-structure, whereas our model is generally applicable (even for data-structures not yet considered for RSs in the literature). Our general framework also captures more sophisticated redaction possibilities such as dependencies between redactable parts, fixed parts and consecutive redaction control.
- We introduce the notion of designated redactors. While this concept might seem similar to the concept of accountable RSs [PS15], we are not interested in accountability, but only want to allow to hand an extra piece of information to the redactor(s). This often allows to increase the efficiency of the respective scheme.
- We present two RSs, one for sets and one for lists, constructed in a black-box way from digital signatures and indistinguishable cryptographic accumulators. We show that existing constructions of RSs are instantiations of our generic constructions but tailored to specific instantiations of accumulators (often this allows to optimize some of the parameters of the schemes).
- We present a black-box construction of RSs with designated redactors for lists from RSs for sets and non-interactive zero-knowledge proof systems. We stress that all three proposed constructions provide transparency, which is an important property, but quite hard to achieve.

5.1.1 Our Generalized Security Model

We use the formalization by *Brzuska* et al. [BBD+10] as a starting point. In contrast to their model, however, ours is not specifically tailored to trees, but is generally applicable to all kinds of data. The resulting model is more restrictive than the ones introduced in the original works [JMSW02, SBZ01], while it is not as restrictive as [ABC+15, ALP12, ALP13, BFLS10, BPS13, CDHK15]. We

think that the security model introduced in [BBD+10] is sufficient for most use cases, while the ones introduced in [ABC+15, ALP12, ALP13, BFLS10, BPS13, CDHK15] seem to be overly strong for real world applications. Namely, we do not require unlinkability and its derivatives (constituting even stronger privacy notations), as almost all messages (documents) occurring in real world applications contain data usable to link them, e.g., unique identifiers.[1] Moreover, we do not formalize accountability, as this notion can easily be achieved by applying the generic transformation presented in [PS15] to constructions being secure in our model.[2]

In the following, we assume that a message m is some arbitrarily structured piece of data and for the general framework we use the following notation. ADM is an abstract data structure which describes the admissible redactions and may contain descriptions of dependencies, fixed elements or relations between elements. MOD is used to actually describe how a message m is redacted. Next, we define how ADM, MOD and the message m are tangled, for which we introduce the following notation: We use $\mathsf{ADM} \preceq \mathsf{m}$ to denote that ADM matches m and $\mathsf{MOD} \preceq \mathsf{ADM}$ to denote that MOD matches ADM. By $\mathring{\mathsf{m}} \overset{\mathsf{MOD}}{\longleftarrow} \mathsf{m}$, we denote the derivation of $\mathring{\mathsf{m}}$ from m with respect to MOD and $\mathring{\mathsf{ADM}} \overset{\mathsf{MOD}}{\longleftarrow} \mathsf{ADM}$ to denote the derivation of $\mathring{\mathsf{ADM}}$ from ADM with respect to MOD. Clearly, how MOD, ADM, $\overset{\mathsf{MOD}}{\longleftarrow}$ and $\preceq$ are implemented depends on the data structure in question and on the features of the concrete RS. Let us give a simple example for sets without using dependencies or other advanced features: MOD and ADM, as well as m, are sets. A redaction $\mathring{\mathsf{m}} \overset{\mathsf{MOD}}{\longleftarrow} \mathsf{m}$ simply would be $\mathring{\mathsf{m}} \leftarrow \mathsf{m} \setminus \mathsf{MOD}$. This further means that $\mathsf{MOD} \preceq \mathsf{ADM}$ holds if $\mathsf{MOD} \subseteq \mathsf{ADM}$, while $\mathsf{ADM} \preceq \mathsf{m}$ holds if $\mathsf{ADM} \subseteq \mathsf{m}$. We want to stress that the definitions of these operators also define how a redaction is actually performed, e.g., if a redacted block leaves a visible special symbol $\perp$ or not. Furthermore, if the message m is a list, i.e., $\mathsf{m} = (m_1, \ldots, m_{|\mathsf{m}|})$, we call m_i a block and use $|\mathsf{m}| \in \mathbb{N}$ to denote the number of blocks in the message m.

Now, we formally define an RS within our general framework.

Definition 5.1. *An* RS *is a tuple of four efficient algorithms* $(\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify}, \mathsf{Redact})$*, which are defined as follows:*

$\mathsf{KeyGen}(1^\kappa)$: *Takes a security parameter* κ *as input and outputs a key pair* $(\mathsf{sk}, \mathsf{pk})$*.*

$\mathsf{Sign}(\mathsf{sk}, \mathsf{m}, \mathsf{ADM})$: *Takes a secret key* sk*, a message* m *and* ADM *as input and outputs a message-signature pair* (m, σ) *together with some auxiliary redaction information* RED*.*[3]

$\mathsf{Verify}(\mathsf{pk}, \sigma, \mathsf{m})$: *Takes a public key* pk*, a signature* σ *and a message* m *as input and outputs a bit* $b \in \{0, 1\}$*.*

$\mathsf{Redact}(\mathsf{pk}, \sigma, \mathsf{m}, \mathsf{MOD}, \mathsf{RED})$: *Takes a public key* pk*, a valid signature* σ *for a message* m*, modification instructions* MOD*, and auxiliary redaction information*

[1] However, we stress that our model can be extended in a straightforward way.
[2] Our model could also be extended to cover accountability in a straightforward way.
[3] We assume that ADM can always be recovered from (m, σ).

RED *as input, and outputs a redacted message-signature pair* $(\mathring{\mathsf{m}}, \mathring{\sigma})$ *and an updated auxiliary redaction information* $\mathring{\mathsf{RED}}$.[4]

We also require that Sign returns $\perp$, if $\mathsf{ADM} \not\preceq \mathsf{m}$, while Redact also returns $\perp$, if $\mathsf{MOD} \not\preceq \mathsf{ADM}$. We will omit this explicit check in our constructions. Note that RED can also be $\emptyset$ if no auxiliary redaction information is required.

Security Properties. The security properties for RSs have already been formally treated for tree data-structures in [BBD+10]. We adapt them to our general framework.

Correctness. Correctness requires that all honestly computed/redacted signatures verify correctly. More formally this means that $\forall\ \kappa \in \mathbb{N},\ \forall\ n \in \mathbb{N},\ \forall\ \mathsf{m},\ \forall\ \mathsf{ADM} \preceq \mathsf{m},\ \forall\ (\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KeyGen}(1^\kappa),\ \forall\ ((\mathsf{m}_0, \sigma_0), \mathsf{RED}_0) \leftarrow \mathsf{Sign}(\mathsf{sk}, \mathsf{m}, \mathsf{ADM}),\ (\forall\ \mathsf{MOD}_i \overset{\mathsf{ADM}}{\preceq} \mathsf{m}_i,\ \forall\ ((\mathsf{m}_{i+1}, \sigma_{i+1}), \mathsf{RED}_{i+1}) \leftarrow \mathsf{Redact}(\mathsf{pk}, \sigma_i, \mathsf{m}_i, \mathsf{MOD}_i, \mathsf{RED}_i))_{0 \le i < n}$ it holds that for $0 \le i \le n : \mathsf{Verify}(\mathsf{pk}, \sigma_i, \mathsf{m}_i) = 1$, where $(S_i)_{0 \le i < n}$ is shorthand for $S_0, \ldots, S_{n-1}$.

Unforgeability. Unforgeability requires that, without a signing key sk, it is infeasible to compute a valid signature σ on a message m, which is not a valid redaction of any message obtained by adaptive signature queries.

Definition 5.2 (Unforgeability)**.** *An* RS *is unforgeable, if for all PPT adversaries* $\mathcal{A}$ *there exists a negligible function* $\varepsilon(\cdot)$ *such that*

$$\Pr\left[\begin{array}{l|r} (\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KeyGen}(1^\kappa), & \mathsf{Verify}(\mathsf{pk}, \mathsf{m}^\star, \sigma^\star) = 1 \wedge \\ (\mathsf{m}^\star, \sigma^\star) \leftarrow \mathcal{A}^{\mathsf{Sign}(\mathsf{sk}, \cdot, \cdot)}(\mathsf{pk}) & \nexists\ (\mathsf{m}, \mathsf{ADM}) \in \mathcal{Q}^{\mathsf{Sign}} : \mathsf{m}^\star \overset{\mathsf{ADM}}{\preceq} \mathsf{m} \end{array}\right] \le \varepsilon(\kappa),$$

where the environment keeps track of the queries to the signing oracle via $\mathcal{Q}^{\mathsf{Sign}}$.

Note that the adversary can perform redactions on its own (also transitively).

Privacy. For anyone except the involved signers and redactors, it should be infeasible to derive information on redacted message parts.

Definition 5.3 (Privacy)**.** *An* RS *is private, if for all PPT adversaries* $\mathcal{A}$ *there exists a negligible function* $\varepsilon(\cdot)$ *such that*

$$\Pr\left[\begin{array}{l|r} (\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KeyGen}(1^\kappa),\ b \xleftarrow{R} \{0,1\}, & \\ \mathcal{O} \leftarrow \{\mathsf{Sign}(\mathsf{sk}, \cdot, \cdot), \mathsf{LoRRedact}((\mathsf{sk}, \mathsf{pk}), \cdot, \cdot, b)\}, & b = b^\star \\ b^\star \leftarrow \mathcal{A}^{\mathcal{O}}(\mathsf{pk}) & \end{array}\right] \le 1/2 + \varepsilon(\kappa),$$

where Sign *denotes a signing oracle and* LoRRedact *is defined as follows:*

$\mathsf{LoRRedact}((\mathsf{sk}, \mathsf{pk}), (\mathsf{m}_0, \mathsf{ADM}_0, \mathsf{MOD}_0), (\mathsf{m}_1, \mathsf{ADM}_1, \mathsf{MOD}_1), b)$:
1: *Compute* $((\mathsf{m}_c, \sigma_c), \mathsf{RED}_c) \leftarrow \mathsf{Sign}(\mathsf{sk}, \mathsf{m}_c, \mathsf{ADM}_c)$ *for* $c \in \{0,1\}$.
2: *Let* $((\mathring{\mathsf{m}}_c, \mathring{\sigma}_c), \mathring{\mathsf{RED}}_c) \leftarrow \mathsf{Redact}(\mathsf{pk}, \sigma_c, \mathsf{m}_c, \mathsf{MOD}_c, \mathsf{RED}_c)$ *for* $c \in \{0,1\}$.
3: *If* $\mathring{\mathsf{m}}_0 \neq \mathring{\mathsf{m}}_1 \ \vee\ \mathring{\mathsf{ADM}}_0 \neq \mathring{\mathsf{ADM}}_1$, *return* $\perp$.
4: *Return* $(\mathring{\mathsf{m}}_b, \mathring{\sigma}_b)$.

[4] This algorithm may also alter ADM.

Here, the admissible modifications $\mathring{\mathsf{ADM}}_0$ *and* $\mathring{\mathsf{ADM}}_1$ *corresponding to the redacted messages are implicitly defined by (and recoverable from) the tuples* $(\mathring{\mathsf{m}}_0, \mathring{\sigma}_0)$ *and* $(\mathring{\mathsf{m}}_1, \mathring{\sigma}_1)$ *and the oracle returns* $\perp$ *if any of the algorithms returns* $\perp$*.*

In our privacy definition, we allow the adversary to provide distinct values for ADM_0 and ADM_1 to the signing oracle. While this guarantees the required flexibility to support arbitrary data structures, it yields a rather strong definition of privacy.

Transparency. It should be infeasible to decide whether a signature directly comes from the signer (i.e., is a fresh signature) or has been generated using the Redact algorithm, for anyone except the signer and the possibly involved redactor(s). More formally, this means:

Definition 5.4 (Transparency)**.** *An* RS *is transparent, if for all PPT adversaries* $\mathcal{A}$ *there exists a negligible function* $\varepsilon(\cdot)$ *such that*

$$\Pr\left[\begin{array}{l}(\mathsf{sk},\mathsf{pk}) \leftarrow \mathsf{KeyGen}(1^\kappa),\ b \xleftarrow{R} \{0,1\},\\ \mathcal{O} \leftarrow \{\mathsf{Sign}(\mathsf{sk},\cdot,\cdot),\ \mathsf{SoR}((\mathsf{sk},\mathsf{pk}),\cdot,\cdot,\cdot,b)\},\ :\ b = b^\star\\ b^\star \leftarrow \mathcal{A}^{\mathcal{O}}(\mathsf{pk})\end{array}\right] \leq \frac{1}{2} + \varepsilon(\kappa),$$

where Sign *denotes a signing oracle and* SoR *is defined as follows:*

$\mathcal{O}^{\mathsf{SoR}}((\mathsf{sk},\mathsf{pk}),\mathsf{m},\mathsf{MOD},\mathsf{ADM},b)$:
1: *Compute* $((\mathsf{m},\sigma),\mathsf{RED}) \leftarrow \mathsf{Sign}(\mathsf{sk},\mathsf{m},\mathsf{ADM})$.
2: *Compute* $((\mathring{\mathsf{m}},\sigma_0),\cdot) \leftarrow \mathsf{Redact}(\mathsf{pk},\sigma,\mathsf{m},\mathsf{MOD},\mathsf{RED})$.
3: *Compute* $((\mathring{\mathsf{m}},\sigma_1),\cdot) \leftarrow \mathsf{Sign}(\mathsf{sk},\mathring{\mathsf{m}},\mathring{\mathsf{ADM}})$*, where* $\mathring{\mathsf{ADM}} \xleftarrow{\mathsf{MOD}} \mathsf{ADM}$.
4: *Return* $(\mathring{\mathsf{m}}_b,\mathring{\sigma}_b)$.

The oracle returns $\perp$ *if any of the algorithms returns* $\perp$*.*

We call an RS secure, if it is correct, unforgeable, private, and transparent.

We want to emphasize that additionally returning auxiliary redaction information RED in Sign and Redact does not contradict transparency or privacy, as the "final" verifier never sees any RED (which is why the privacy and transparency games do not return RED for the challenge message-signature pair). Intuitively, only if an intermediate redactor exists, RED is given away by the signer to selected designated entities that become redactors.[5]

Relations between Security Properties. The relations between the different security properties do not change compared to the work done in [BBD+10]. Namely, transparency implies privacy, while privacy does not imply transparency. Furthermore, unforgeability is independent of privacy and transparency. The formal proofs for these statements are analogous to the ones in [BBD+10]. To this end, we do not include them here and note that they can be found in the full paper corresponding to this section [c9].

[5] This also distinguishes designated redactors from accountable redactable signatures [PS15]. Namely, the additional information RED can be given to any redactor, while the redactor is a fixed entity in accountable RSs. Hence, in our notion, the redactors can even form a chain, and can be pinpointed in an ad-hoc manner.

Notes on our Model. In a nutshell, our generalized framework leaves the concrete data-structure—and, thus, also the definition of ADM, MOD, and RED—open to the instantiation. For clarity, let us match our framework to already existing definitions. In particular, consider the model of [BBD+10]. It does not explicitly define ADM, but implicitly assumes that only leaves of a given tree are redactable, i.e., MOD may only contain changes which are possible with recursive leaf-redaction. *Pöhls* et al. [PSdMP12] explicitly define ADM as the edges between different nodes in their model for RSS for trees, while allowing arbitrary redactions, i.e., MOD may contain any set of nodes in the tree (including the tree's root), as well as edges.

Finally, we note that our model also covers consecutive redaction control [MHI06, MIM+05, SPB+12b] via ADM. Recall that ADM is contained in all signatures and Redact may also change ADM.

5.1.2 Redactable Signatures for Sets

For our RSS for sets (cf. Scheme 5.1), we compute an accumulator representing the set to be signed and then sign the accumulator using any EUF-CMA secure digital signature scheme. For verification, one simply provides witnesses for each element in the set and it is verified whether the digital signature on the accumulator as well as the witnesses are valid. Redaction amounts to simply throwing away witnesses corresponding to redacted elements. To maintain transparency, while still allowing the signer to determine which blocks (i.e., elements) of the message (i.e., the set) are redactable, we model ADM as a set containing all blocks which must not be redacted. We also parametrize the scheme by an operator ord$(\cdot)$, which allows to uniquely encode ADM as a sequence. MOD is modeled as a set containing all blocks of the message to be redacted. We note that one can straightforwardly extend Scheme 5.1 to support multi-sets by concatenating a unique identifier to each set element. Below, we prove the following:

Theorem 5.1. *If* A *is correct, collision free, and indistinguishable, and* Σ *is correct and* EUF-CMA *secure, then Scheme 5.1 is secure.*

We prove Theorem 5.1 by proving Lemma 5.1-5.3 and deriving Corollary 5.1.

Lemma 5.1. *If* A *is correct and* Σ *is correct, the construction in Scheme 5.1 is correct.*

The lemma above follows from inspection.

Lemma 5.2. *If* A *is collision free and* Σ *is* EUF-CMA *secure, then Scheme 5.1 is unforgeable.*

Proof. Assume an efficient adversary $\mathcal{A}^{\mathsf{uf}}$ against unforgeability. We show how $\mathcal{A}^{\mathsf{uf}}$ can be used to construct (1) an efficient adversary $\mathcal{A}^{\mathsf{cf}}$ against the collision freeness of the accumulator or (2) an efficient adversary $\mathcal{A}^{\mathsf{f}}$ against the EUF-CMA security of the signature scheme. To do so, we describe efficient reductions $\mathcal{R}^{\mathsf{cf}}$ and $\mathcal{R}^{\mathsf{f}}$, respectively.

$\underline{\mathsf{KeyGen}(1^\kappa)}$: Fix a digital signature scheme Σ and an indistinguishable accumulator scheme A and return $(\mathsf{sk}, \mathsf{pk}) \leftarrow ((\mathsf{sk}_\Sigma, \mathsf{sk}_\Lambda, \mathsf{pk}_\Lambda), (\mathsf{pk}_\Sigma, \mathsf{pk}_\Lambda))$, where

$$(\mathsf{sk}_\Sigma, \mathsf{pk}_\Sigma) \leftarrow \Sigma.\mathsf{KeyGen}(1^\kappa),\ (\mathsf{sk}_\Lambda, \mathsf{pk}_\Lambda) \leftarrow \mathsf{A}.\mathsf{Gen}(1^\kappa, \infty).$$

$\underline{\mathsf{Sign}(\mathsf{sk}, \mathsf{m}, \textsc{adm})}$: Compute $(\Lambda, \mathsf{aux}) \leftarrow \mathsf{A}.\mathsf{Eval}((\mathsf{sk}_\Lambda, \mathsf{pk}_\Lambda), \mathsf{m})$, $\sigma_\Sigma \leftarrow \Sigma.\mathsf{Sign}(\mathsf{sk}_\Sigma, \Lambda || \operatorname{ord}(\textsc{adm}))$ and return (m, σ) and $\textsc{red}$, where

$$\sigma \leftarrow (\sigma_\Sigma, \Lambda, \{\mathsf{wit}_{m_i}\}_{m_i \in \mathsf{m}}, \textsc{adm}), \textsc{red} \leftarrow \emptyset, \text{ and}$$
$$\{\mathsf{wit}_{m_i} \leftarrow \mathsf{A}.\mathsf{WitCreate}((\mathsf{sk}_\Lambda, \mathsf{pk}_\Lambda), \Lambda, \mathsf{aux}, m_i)\}_{m_i \in \mathsf{m}}.$$

$\underline{\mathsf{Verify}(\mathsf{pk}, \sigma, \mathsf{m})}$: Return 1 if the following holds and 0 otherwise:

$$\Sigma.\mathsf{Verify}(\mathsf{pk}_\Sigma, \Lambda || \operatorname{ord}(\textsc{adm}), \sigma_\Sigma) = 1 \ \wedge\ \textsc{adm} \cap \mathsf{m} = \textsc{adm} \wedge$$
$$\forall\ m_i \in \mathsf{m} : \mathsf{A}.\mathsf{Verify}(\mathsf{pk}_\Lambda, \Lambda, \mathsf{wit}_{m_i}, m_i) = 1.$$

$\underline{\mathsf{Redact}(\mathsf{pk}, \sigma, \mathsf{m}, \textsc{mod}, \textsc{red})}$: Parse σ as $(\sigma_\Sigma, \Lambda, \textsc{wit}, \textsc{adm})$ and return $(\mathring{\mathsf{m}}, \mathring{\sigma})$ and $\mathring{\textsc{red}}$, where

$$\mathring{\mathsf{m}} \leftarrow \mathsf{m} \setminus \textsc{mod},\ \mathring{\textsc{wit}} \leftarrow \textsc{wit} \setminus \{\mathsf{wit}_{m_i}\}_{m_i \in \textsc{mod}},\ \mathring{\sigma} \leftarrow (\sigma_\Sigma, \Lambda, \mathring{\textsc{wit}}, \textsc{adm}),\ \mathring{\textsc{red}} \leftarrow \textsc{red}.$$

$\underline{\operatorname{ord}(\textsc{adm})}$: This operator takes a set $\textsc{adm}$, applies some unique ordering (e.g., lexicographic) to the elements in $\textsc{adm}$ and returns the corresponding sequence.

Scheme 5.1: A RS for sets.

$\underline{\mathcal{R}^{\mathsf{cf}}}$: Here, $\mathcal{R}^{\mathsf{cf}}$ obtains the accumulator public key pk_Λ from the challenger $\mathcal{C}^{\mathsf{cf}}$ of the collision freeness game of the used accumulator scheme and completes the setup by running $(\mathsf{sk}_\Sigma, \mathsf{pk}_\Sigma) \leftarrow \Sigma.\mathsf{KeyGen}(1^\kappa)$ and handing $(\mathsf{pk}_\Sigma, \mathsf{pk}_\Lambda)$ to $\mathcal{A}^{\mathsf{uf}}$. It is easy to see that $\mathcal{R}^{\mathsf{cf}}$ can simulate all oracles for $\mathcal{A}^{\mathsf{uf}}$ by forwarding the respective calls to $\mathcal{O}^{\mathsf{E}}$ and $\mathcal{O}^{\mathsf{W}}$ provided by $\mathcal{C}^{\mathsf{cf}}$. Furthermore, $\mathcal{R}^{\mathsf{cf}}$ can choose the randomness used in the calls to $\mathcal{O}^{\mathsf{E}}$ and keeps a mapping of accumulators and corresponding randomizers. Eventually, $\mathcal{A}^{\mathsf{uf}}$ outputs a tuple $(\mathsf{m}^\star, \sigma^\star)$, where $\sigma^\star = (\sigma^\star_\Sigma, \Lambda^\star, \{\mathsf{wit}_{m_i}\}_{m_i \in \mathsf{m}^\star}, \textsc{adm}^\star)$ such that $\nexists(\mathsf{m}, \textsc{adm}) \in \mathcal{Q}_{\mathsf{Sign}}$, $\nexists \textsc{mod} \preceq \textsc{adm} : \mathsf{m}^\star \overset{\textsc{mod}}{\longleftarrow} \mathsf{m}$. If $\Lambda^\star || \operatorname{ord}(\textsc{adm}^\star)$ was never signed using Σ, it aborts. Otherwise, we have at least one m_i with a corresponding witness wit_{m_i} such that $\mathsf{A}.\mathsf{Verify}(\mathsf{pk}_\Lambda, \Lambda^\star, \mathsf{wit}_{m_i}, m_i) = 1$ but $m_i \notin \mathsf{m}^\star$. Consequently, $\mathcal{R}^{\mathsf{cf}}$ can look up the randomness r used to compute Λ and output $(\mathsf{wit}_{m_i}, m_i, \mathsf{m}^\star, r)$ as a collision for the accumulator.

$\underline{\mathcal{R}^{\mathsf{f}}}$: Here, $\mathcal{R}^{\mathsf{f}}$ obtains the Σ public key pk_Σ from the challenger $\mathcal{C}^{\mathsf{f}}$ of the EUF-CMA game of the used signature scheme and completes the setup by running $(\mathsf{sk}_\Lambda, \mathsf{pk}_\Lambda) \leftarrow \mathsf{A}.\mathsf{Gen}(1^\kappa, \infty)$ and handing $(\mathsf{pk}_\Sigma, \mathsf{pk}_\Lambda)$ to $\mathcal{A}^{\mathsf{uf}}$. It is easy to see that $\mathcal{R}^{\mathsf{cf}}$ can simulate all oracles for $\mathcal{A}^{\mathsf{uf}}$ by forwarding the respective calls to Sign to the $\Sigma.\mathsf{Sign}$ oracle provided by $\mathcal{C}^{\mathsf{f}}$. Eventually, $\mathcal{A}^{\mathsf{uf}}$ outputs a tuple $(\mathsf{m}^\star, \sigma^\star)$, where $\sigma^\star = (\sigma^\star_\Sigma, \Lambda^\star, \{\mathsf{wit}_{m_i}\}_{m_i \in \mathsf{m}^\star}, \textsc{adm}^\star)$ such that $\nexists(\mathsf{m}, \textsc{adm}) \in \mathcal{Q}_{\mathsf{Sign}}$, $\nexists \textsc{mod} \preceq \textsc{adm} : \mathsf{m}^\star \overset{\textsc{mod}}{\longleftarrow} \mathsf{m}$. If $\Lambda^\star || \operatorname{ord}(\textsc{adm}^\star)$ was signed using the signing oracle provided by $\mathcal{C}^{\mathsf{f}}$ it aborts. Otherwise, we can output $(\sigma^\star_\Sigma, \Lambda^\star || \operatorname{ord}(\textsc{adm}^\star))$ as a forgery.

Overall Bound. The simulations in both reductions are indistinguishable from a real game. In front of an adversary we randomly guess the adversary's strategy, inducing a loss of $1/2$. Our reductions for both forger types always succeed if the forger succeeds. This means that the probability to break unforgeability is upper-bounded by $2 \cdot \max\{\varepsilon_{\mathsf{cf}}(\kappa), \varepsilon_{\mathsf{f}}(\kappa)\}$. □

Lemma 5.3. *If* A *is indistinguishable, then Scheme 5.1 is transparent.*

Proof. We will bound the probability to win the transparency game by using a sequence of games. Thereby, we denote the event that the adversary wins Game i by S_i and let k be the number of queries to the SoR oracle.

Game 0: The original transparency game.

Game 1: As the original game, but all calls to $\mathsf{A.Eval}$ in SoR are performed with respect to the originally submitted message m.

Transition - Game 0 → Game 1: A distinguisher between Game 0 and Game 1 is an indistinguishability distinguisher for one of the accumulators, i.e., $|\Pr[S_0] - \Pr[S_1]| \leq k \cdot \varepsilon_{\mathsf{ind}}(\kappa)$.[6]

In Game 1, the value of the accumulator is independent of the bit b. The remaining signature components are independent of b anyway. This means that $\Pr[S_1] = 1/2$, and, in further consequence, $\Pr[S_0] \leq 1/2 + k \cdot \varepsilon_{\mathsf{ind}}(\kappa)$. This concludes the proof. □

Since transparency implies privacy, the lemma above yields the following corollary.

Corollary 5.1. *If* A *is indistinguishable, then Scheme 5.1 is private.*

Observations and Optimizations. Depending on the properties of the used accumulator scheme, one can reduce the signature size from $\mathcal{O}(n)$ to $\mathcal{O}(1)$. In particular, it turns out that quasi-commutativity as discussed in Section 3.1.1 is sufficient to achieve this. For our illustrations, we recall that the quasi-commutative accumulator function $f : \mathcal{Z}_A^\kappa \times \mathcal{Z}_I^\kappa \to \mathcal{Z}_A^\kappa$. Using this function the accumulator for a set $\mathcal{X} = (x_1, \ldots, x_n)$ can be recursively computed via $\Lambda_{\mathcal{X}} \leftarrow f(f(\ldots f(\Lambda_\emptyset, x_1), \ldots), x_n)$, where $\Lambda_\emptyset$ denotes the "empty accumulator" which is fixed by the key generation algorithm. Witnesses are defined as $\mathsf{wit}_x \leftarrow \Lambda_{\mathcal{X} \setminus \{x\}}$

Now we can use f as a means to achieve batch-membership verification, i.e., to verify that a set $\mathcal{Y}$ is a subset of the accumulated set $\mathcal{X}$ using only a single succinct witness $\mathsf{wit}_{\mathcal{Y}}$. Formally, this means that there are two additional algorithms $\mathsf{WitCreateB}$ and $\mathsf{VerifyB}$, which we define below. Note that we can without loss of generality assume that aux is of the form $(\mathsf{aux}', \mathcal{X})$, i.e., contains the accumulated set.

[6] For compactness, we collapse the exchange of the accumulators to one single game change, which can straightforwardly be unrolled to k game changes.

$\mathsf{WitCreateB}((\mathsf{sk}_\Lambda^\sim, \mathsf{pk}_\Lambda), \Lambda_\mathcal{X}, \mathsf{aux}, \mathcal{Y})$: Parse aux as $(\mathsf{aux}', \mathcal{X})$ and output $\mathsf{wit}_\mathcal{Y} \leftarrow \Lambda_{\mathcal{X}\setminus\mathcal{Y}}$.

$\mathsf{VerifyB}(\mathsf{pk}_\Lambda, \Lambda_\mathcal{X}, \mathsf{wit}_\mathcal{Y}, \mathcal{Y})$: Parse $\mathcal{Y}$ as $(y_1, \ldots, y_n)$ and return 1 if $\Lambda_\mathcal{X} = f(\ldots f(\mathsf{wit}_\mathcal{Y}, y_1), \ldots), y_n)$ and 0 otherwise.

Also observe that a collision with respect to a batch witness implies a collision in the underlying accumulator: for a valid collision we have that there must be at least one $y \in \mathcal{Y}$ where we have that $y \notin \mathcal{X}$, even though $\mathsf{wit}_\mathcal{Y}$ attests that $\mathcal{Y} \subseteq \mathcal{X}$. Now one can simply compute wit_y from $\mathsf{wit}_\mathcal{Y}$ using the quasicommutativity and the elements $\mathcal{Y} \setminus \{y\}$.

From this, it is straightforward to derive the following corollary:

Corollary 5.2. *For quasi-commutative schemes it holds that $\forall\{x, y\} \subseteq \mathcal{X}$, one can use $\mathsf{wit}_{\{x\}\cup\{y\}}$ and $\Lambda_\mathcal{X}$ to attest that x is a member of $\Lambda_{\mathcal{X}\setminus\{y\}}$. Furthermore, one can efficiently compute $\mathsf{wit}_{\{x\}\cup\{y\}}$ from $\mathsf{wit}_{\{x\}}$ and y.*

Then, only a single witness needs to be stored and verification is performed with respect to this witness. Redaction is performed by publicly updating the witness (can be interpreted as removing elements from the accumulator). Such a scheme generalizes the RS for sets from [PSPdM12], which builds upon the RSA accumulator.

Our construction may look similar to the one in [PS14]. However, in contrast to our construction, they require a rather specific definition of accumulators, which they call trapdoor accumulators. Trapdoor accumulators differ from conventional accumulators regarding their features and security properties. In particular, they need to support updates of the accumulated set without modifying the accumulator itself. Further, they require a non-standard property denoted as strong collision resistance, which can be seen as a combination of conventional collision resistance and indistinguishability. Clearly, such a specific accumulator model limits the general applicability.

5.1.3 Redactable Signatures for Linear Documents

We build our RS for linear documents upon the RS for sets presented in the previous section. From an abstract point of view, moving from sets to linear documents means to move from an unordered message to an ordered one. A naïve approach to assign an ordering to the message blocks would be to concatenate each message block with its position in the message and insert these extended tuples into the accumulator. However, such an approach trivially contradicts transparency, since the positions of the messages would reveal if redactions have taken place. Thus, inspired by [CLX09], we choose some indistinguishable accumulator scheme and use accumulators to encode the positions. More precisely, with n being the number of message blocks, we draw a sequence of n uniformly random numbers $(r_j)_{j=1}^n$ from the accumulation domain. Then, for each message block m_i, $1 \leq i \leq n$, an accumulator Λ_i containing $(r_j)_{j=1}^i$ is computed (i.e., Λ_i contains i randomizers). Finally, for each m_i, one appends $\Lambda_i || r_i$ and signs

the so obtained set $\bigcup_{j=1}^{n}\{(m_i||\Lambda_i||r_i)\}$ using the RS for sets. Upon verification, one simply verifies the signature on the set and checks for each i whether one can provide i valid witnesses for $(r_j)_{j=1}^{i}$ with respect to Λ_i. Redaction again amounts to throwing away witnesses corresponding to redacted message blocks.

Here, $\mathsf{m} = (m_i)_{i=1}^{n}$ is a sequence of message blocks m_i, ADM is the corresponding sequence of fixed message blocks, and the operator ord$(\cdot)$ for the underlying RS for sets simply returns ADM without modification. All possible valid redactions, forming the transitive closure of a message m, with respect to Redact, are denoted as $\mathrm{span}_{\vdash}(M)$, following [CLX09] and [SPB+12b]. Note that for ADM it must hold that $\text{ADM} \in \mathrm{span}_{\vdash}(\mathsf{m})$. MOD is modeled as a sequence of message blocks to be redacted and we assume an encoding that allows to uniquely match a message block with its corresponding message block in the original message.

KeyGen(1^κ) : Fix a redactable signature scheme $\mathbf{RS}(\text{ord}) = \{\mathbf{KeyGen}, \mathbf{Sign}, \mathbf{Verify}, \mathbf{Redact}\}$ for sets (with ord$(\cdot)$ as defined below) and an indistinguishable accumulator scheme $\mathsf{A} = \{\mathsf{Gen}, \mathsf{Eval}, \mathsf{WitCreate}, \mathsf{Verify}\}$, run $(\mathsf{sk}_\Lambda, \mathsf{pk}_\Lambda) \leftarrow \mathsf{A.Gen}(1^\kappa, \infty)$, $(\mathbf{sk}, \mathbf{pk}) \leftarrow \mathbf{KeyGen}(1^\kappa)$ and return $(\mathsf{sk}, \mathsf{pk}) \leftarrow ((\mathbf{sk}, \mathsf{sk}_\Lambda, \mathsf{pk}_\Lambda), (\mathbf{pk}, \mathsf{pk}_\Lambda))$.

Sign$(\mathsf{sk}, \mathsf{m}, \text{ADM})$: Chooses $(r_i)_{i=1}^{|\mathsf{m}|} \xleftarrow{R} \mathsf{Dom}(\Lambda)^{|\mathsf{m}|}$, set $\mathsf{m}' \leftarrow \emptyset$ and for $1 \le i \le |\mathsf{m}|$:

$$(\Lambda_i, \mathsf{aux}) \leftarrow \mathsf{A.Eval}((\mathsf{sk}_\Lambda, \mathsf{pk}_\Lambda), \cup_{j=1}^{i}\{r_j\}), \mathsf{WIT}_i \leftarrow (\mathsf{wit}_{i_j})_{j=1}^{i}, \text{ where}$$
$$\mathsf{wit}_{i_j} \leftarrow \mathsf{A.WitCreate}((\mathsf{sk}_\Lambda, \mathsf{pk}_\Lambda), \Lambda_i, \mathsf{aux}, r_j) \text{ for } 1 \le j \le i.$$

Finally, return (m, σ) and RED, where

$$\sigma \leftarrow (\delta, (\Lambda_i)_{i=1}^{|\mathsf{m}|}, (\mathsf{WIT}_i)_{i=1}^{|\mathsf{m}|}, (r_i)_{i=1}^{|\mathsf{m}|}), \ \text{RED} \leftarrow \emptyset, \ \text{and}$$
$$\delta \leftarrow \mathbf{Sign}(\mathbf{sk}, \textstyle\bigcup_{i=1}^{|\mathsf{m}|}\{(m_i||\Lambda_i||r_i)\}, \text{ADM}).$$

Verify$(\mathsf{pk}, \sigma, \mathsf{m})$: Parse σ as $(\delta, (\Lambda_i)_{i=1}^{|\mathsf{m}|}, (\mathsf{WIT}_i)_{i=1}^{|\mathsf{m}|}, (r_i)_{i=1}^{|\mathsf{m}|})$ and return 1 if the following holds, and 0 otherwise:

$$\mathbf{Verify}(\mathbf{pk}, \delta, \textstyle\bigcup_{i=1}^{|\mathsf{m}|}\{(m_i||\Lambda_i||r_i)\}) = 1 \ \wedge \ \text{ADM} \in \mathrm{span}_{\vdash}(\mathsf{m}) \ \wedge$$
$$\forall\ 1 \le i \le |\mathsf{m}|,\ \forall\ 1 \le j \le i : \mathsf{A.Verify}(\mathsf{pk}_\Lambda, \Lambda_i, \mathsf{wit}_{i_j}, r_j) = 1$$

Redact$(\mathsf{pk}, \sigma, \mathsf{m}, \text{MOD}, \text{RED})$: Parse σ as $(\delta, (\Lambda_i)_{i=1}^{|\mathsf{m}|}, (\mathsf{WIT}_i)_{i=1}^{|\mathsf{m}|}, (r_i)_{i=1}^{|\mathsf{m}|})$, set $\text{MOD}' \leftarrow \bigcup_{m_i \in \text{MOD}}\{(m_i||\Lambda_i||r_i)\}$, run $(\cdot, \mathring{\delta}) \leftarrow \mathbf{Redact}(\mathbf{pk}, \hat{\sigma}, \bigcup_{i=1}^{|\mathsf{m}|}\{(m_i||\Lambda_i||r_i)\}, \text{MOD}')$ and for all $m_i \in \text{MOD}$, remove the corresponding entries from m, $(\Lambda_i)_{i=1}^{|\mathsf{m}|}$, $(\mathsf{WIT}_i)_{i=1}^{|\mathsf{m}|}$ and $(r_i)_{i=1}^{|\mathsf{m}|}$ to obtain $\mathring{\mathsf{m}}$, $(\Lambda_i)_{i=1}^{|\mathring{\mathsf{m}}|}$, $(\mathsf{WIT}_i)_{i=1}^{|\mathring{\mathsf{m}}|}$ and $(r_i)_{i=1}^{|\mathring{\mathsf{m}}|}$. Finally, return $(\mathring{\mathsf{m}}, \mathring{\sigma})$ and $\mathring{\text{RED}}$, where

$$\mathring{\sigma} \leftarrow (\mathring{\delta}, (\Lambda_i)_{i=1}^{|\mathring{\mathsf{m}}|}, (\mathsf{WIT}_i)_{i=1}^{|\mathring{\mathsf{m}}|}, (r_i)_{i=1}^{|\mathring{\mathsf{m}}|}), \text{ and } \mathring{\text{RED}} \leftarrow \emptyset.$$

---

ord(ADM) : This operator returns ADM.

Scheme 5.2: A RS for linear documents.

Theorem 5.2. *If* A *is correct, collision-free, and indistinguishable and* **RS** *is is correct, unforgeable and transparent, then Scheme 5.2 is correct, unforgeable, private and transparent.*

We prove Theorem 5.2 by proving Lemma 5.4-5.6 and deriving Corollary 5.3.

Lemma 5.4. *If* A *is correct and* **RS** *is correct, then Scheme 5.2 is correct as well.*

The lemma above follows from inspection.

Lemma 5.5. *If* A *is collision free and* **RS** *is unforgeable, then Scheme 5.2 is unforgeable.*

Proof. Assume an efficient adversary $\mathcal{A}^{\mathsf{uf}}$ against unforgeability. We show how $\mathcal{A}^{\mathsf{uf}}$ can be used to construct (1) an efficient adversary $\mathcal{A}^{\mathsf{cf}}$ against the collision freeness of the accumulator or (2) an efficient adversary $\mathcal{A}^{\mathsf{uf}}$ against the unforgeability of the underlying RS for sets **RS**. To do so, we describe efficient reductions $\mathcal{R}^{\mathsf{cf}}$ and $\mathcal{R}^{\mathsf{uf}}$, respectively.

$\underline{\mathcal{R}^{\mathsf{cf}}}$: Here, $\mathcal{R}^{\mathsf{cf}}$ obtains the accumulator public key pk_Λ from the challenger $\mathcal{C}^{\mathsf{cf}}$ of the collision freeness game of the used accumulator scheme and completes the setup by running $(\mathbf{sk}, \mathbf{pk}) \leftarrow \mathbf{KeyGen}(1^\kappa)$ and handing $(\mathbf{pk}, \mathsf{pk}_\Lambda)$ to $\mathcal{A}^{\mathsf{uf}}$. It is easy to see that $\mathcal{R}^{\mathsf{cf}}$ can simulate all oracles for $\mathcal{A}^{\mathsf{uf}}$ by forwarding the respective calls to $\mathcal{O}^{\mathsf{E}}$ and $\mathcal{O}^{\mathsf{W}}$ provided by $\mathcal{C}^{\mathsf{cf}}$. Furthermore, $\mathcal{R}^{\mathsf{cf}}$ can choose the randomness used in the calls to $\mathcal{O}^{\mathsf{E}}$ and keeps a mapping of accumulators and corresponding randomizers. Eventually, $\mathcal{A}^{\mathsf{uf}}$ outputs a tuple $(\mathsf{m}^\star, \sigma^\star)$, where $\sigma^\star = (\delta^\star, (\Lambda_i)_{i=1}^{|\mathsf{m}^\star|}, (\mathsf{WIT}_i)_{i=1}^{|\mathsf{m}^\star|}, (r_i)_{i=1}^{|\mathsf{m}^\star|})$ and $\delta^\star$ contains $\mathsf{ADM}^\star$ such that $\nexists\ (\mathsf{m}, \mathsf{ADM}^\star) \in \mathcal{Q}_{\mathsf{Sign}}$, $\nexists\ \mathsf{MOD} \preceq \mathsf{ADM}^\star : \mathsf{m}^\star \overset{\mathsf{MOD}}{\leftarrow\!\!-} \mathsf{m}$. If there was no signing query for a superset of $\bigcup_{i=1}^{|\mathsf{m}^\star|}\{(m_i||\Lambda_i||r_i)\}$ and $\mathsf{ADM}^\star$, it aborts. Otherwise, we have at least one accumulator Λ_i, corresponding set $R_i = \{r_j\}_{j=1}^{i}$, witness wit_{r_k} and randomizer r_k such that $r_k \notin R_i$ but $\mathsf{A.Verify}(\mathsf{pk}_\Lambda, \Lambda_i, \mathsf{wit}_{r_k}, r_k) = 1$. Then, $\mathcal{R}^{\mathsf{cf}}$ can look up the randomizer r corresponding to Λ_i and output $(\mathsf{wit}_{r_k}, r_k, R_i, r)$ as a collision for the accumulator.

$\underline{\mathcal{R}^{\mathsf{uf}}}$: Here, $\mathcal{R}^{\mathsf{uf}}$ obtains the RS public key $\mathbf{pk}$ from the challenger $\mathcal{C}^{\mathsf{uf}}$ of the unforgeability game of the used redactable signature scheme for sets and completes the setup by running $(\mathsf{sk}_\Lambda, \mathsf{pk}_\Lambda) \leftarrow \mathsf{A.Gen}(1^\kappa, \infty)$ and handing $(\mathbf{pk}, \mathsf{pk}_\Lambda)$ to $\mathcal{A}^{\mathsf{uf}}$. It is easy to see that $\mathcal{R}^{\mathsf{cf}}$ can simulate all oracles for $\mathcal{A}^{\mathsf{uf}}$ by forwarding the respective calls to **Sign** to the oracles provided by $\mathcal{C}^{\mathsf{uf}}$. Eventually, $\mathcal{A}^{\mathsf{uf}}$ outputs a tuple $(\mathsf{m}^\star, \sigma^\star)$, where $\sigma^\star = (\delta^\star, (\Lambda_i)_{i=1}^{|\mathsf{m}^\star|}, (\mathsf{WIT}_i)_{i=1}^{|\mathsf{m}^\star|}, (r_i)_{i=1}^{|\mathsf{m}^\star|})$ and $\delta^\star$ contains $\mathsf{ADM}^\star$ such that $\nexists(\mathsf{m}, \mathsf{ADM}^\star) \in \mathcal{Q}_{\mathsf{Sign}}$, $\nexists\mathsf{MOD} \preceq \mathsf{ADM}^\star : \mathsf{m}^\star \overset{\mathsf{MOD}}{\leftarrow\!\!-} \mathsf{m}$. If a superset of $\bigcup_{i=1}^{|\mathsf{m}^\star|}\{(m_i||\Lambda_i||r_i)\}$ and $\mathsf{ADM}^\star$ was signed using the oracle provided by $\mathcal{C}^{\mathsf{uf}}$ it aborts. Otherwise, it outputs the tuple $(\delta^\star, \bigcup_{i=1}^{|\mathsf{m}^\star|}\{(m_i||\Lambda_i||r_i))$ as a forgery for the underlying RS for sets.

Overall Bound. The simulations in both reductions are indistinguishable from a real game. In front of an adversary we randomly guess the adversary's strategy, inducing a loss of $1/2$. Our reductions for both forger types always succeed if the forger succeeds. This means that the probability to break group unforgeability is upper-bounded by $2 \cdot \max\{\varepsilon_{\mathsf{cf}}(\kappa), \varepsilon_{\mathsf{uf}}(\kappa)\}$. □

Lemma 5.6. *If* A *is indistinguishable and* **RS** *is transparent, then Scheme 5.2 is transparent.*

Proof. We will bound the probability to win the transparency game by using a sequence of games. Thereby, we denote the event that the adversary wins Game i by S_i, where we let k be the overall number of required accumulators.

Game 0: The original transparency game.

Game 1: As the original game, but all accumulators Λ_i in $\mathcal{O}^{\mathsf{SoR}}$ are computed with respect to the initial set of randomizers $\{r_i\}_{i=1}^{|\mathsf{m}|}$.

Transition - Game 0 → Game 1: A distinguisher between Game 0 and Game 1 is an indistinguishability distinguisher for one of the accumulators, i.e., $|\Pr[S_0] - \Pr[S_1]| \leq k \cdot \varepsilon_{\mathsf{ind}}(\kappa)$.[7]

In Game 1, the accumulators Λ_i are independent of the bit b. In this game the adversary can only win the game by breaking the transparency of the underlying RS, i.e., $\Pr[S_1] \leq 1/2 + \varepsilon_{\mathsf{RS}}(\kappa)$. All in all, we have that $|\Pr[S_0] - \Pr[S_1]| \leq k \cdot \varepsilon_{\mathsf{ind}}(\kappa)$, meaning that the probability to win the transparency game is $\Pr[S_0] \leq 1/2 + \varepsilon_{\mathsf{RS}}(\kappa) + k \cdot \varepsilon_{\mathsf{ind}}(\kappa)$, which concludes the proof. □

The lemma above yields the following corollary.

Corollary 5.3. *If* A *is indistinguishable and* RS *is transparent, construction in Scheme 5.2 is private.*

Observations and Optimizations. Depending on the used accumulator scheme, it is possible to reduce the signature size from $\mathcal{O}(n^2)$ to $\mathcal{O}(n)$. Let us assume a quasi-commutative accumulator, which means that also Corollary 5.2 holds. For our following explanations let $R_i := \bigcup_{k=1}^{i} \{r_k\}$. Then for each message block m_i one only needs to store one batch witness wit_{R_i}. Furthermore, upon redaction of message block m_i with corresponding randomizer r_i, one can update the witnesses wit_{R_j} for all $i > j \leq |\mathsf{m}|$ by computing $\mathsf{wit}'_{R_j} \leftarrow \mathsf{wit}_{R_j \cup \{r_i\}}$ and removing witness wit_{R_i} and randomizer r_i from the signature, which can be interpreted as "removing" r_i from all accumulators. The so-obtained construction then essentially generalizes the approach of [CLX09], which make (white-box) use of the RSA accumulator.

5.1.4 Designated Redactor RSS for Linear Documents

The signature size and computational complexity of RSs can often be improved by explicitly considering the possibility to allow RED to be non-empty. In Scheme 5.3 we follow this approach and present such a generic construction of RSs for linear documents. Basically, the idea is to compute commitments to the

[7] As in the proof of Lemma 5.3, we collapse the exchange of the accumulators to one single game change for compactness.

positions of the messages blocks and concatenate them to the respective message blocks. Then, one signs the so obtained set of concatenated messages and commitments using an RS for sets. Additionally, one includes a non-interactive zero-knowledge proof of an order relation on the committed positions for attesting the correct order of the message blocks. The information RED then represents the randomness used to compute the single commitments. Redacting message blocks simply amounts to removing the single blocks from the signature of the RS for sets and recomputing a non-interactive proof for the ordering on the remaining commitments. Since redaction control via ADM can straightforwardly be achieved as in Scheme 5.2, we omit it here for simplicity, i.e., we assume $\mathsf{ADM} = \infty$. Also note that without ADM the operator $\mathrm{ord}(\cdot)$ is not required. MOD is defined as in Scheme 5.2. We emphasize that one can easily obtain constant size RED by pseudorandomly generating the randomizers $(r_i)_{i=1}^{|\mathsf{m}|}$ and storing the seed for the PRG in RED instead of the actual randomizers.

Before we define the scheme, we define the required **NP** relation of commitments and respective openings. Therefore, we may assume without loss of generality that the openings O of the commitment C are of the form (x, r) where x is the committed value and r is the randomness used in the commitment:

$$((C_1, C_2), (O_1, O_2)) \in R_{\mathsf{ord}} \iff C_1 = \mathsf{Commit}(x_1;\ r_1) \ \wedge \\ C_2 = \mathsf{Commit}(x_2;\ r_2) \ \wedge \ x_1 \leq x_2.$$

Instantiating the proof system Π for R_{ord} can be done straightforwardly by using zero-knowledge set membership proofs. Below, we briefly discuss the efficiency of the instantiations of Scheme 5.3, when based on three common techniques. We note that the below Σ-protocols can all easily be made non-interactive (having all the required properties) using the Fiat-Shamir transform and the results from [FKMV12].

Square Decomposition. An efficient building block for range proofs in hidden order groups is a proof that a secret integer x is positive [Bou00, Lip03], which is sufficient for our instantiation. Technically, therefore we need a homomorphic integer commitment scheme and R_{ord} for Π is as follows:

$$((C_1, C_2), (x, r)) \in R_{\mathsf{ord}} \iff C_2 - C_1 = \mathsf{Commit}(x;\ r) \ \wedge \ x \geq 0.$$

This approach yields $O(n)$ signature generation cost, signature size and verification cost and has a constant size public key. It, however, only works in a hidden order group setting.

For the subsequent two approaches we need to introduce an upper bound k on the number of message blocks which will be a parameter of Scheme 5.3.

Multi-Base Decomposition. This technique for range proofs works by decomposing the secret integer $x = \sum_{i=1}^{n} G_i \cdot b_i$ with $b_i \in [0, u-1]$ into a (multi)-base representation and then proving that every b_i belongs to the respective small

$\underline{\mathsf{KeyGen}(1^\kappa)}$: Fix a redactable signature scheme for sets $\mathbf{RS} = \{\mathbf{KeyGen}, \mathbf{Sign}, \mathbf{Verify}, \mathbf{Redact}\}$, a commitment scheme $\mathsf{C} = (\mathsf{PGen}, \mathsf{Commit}, \mathsf{Open})$ as well as a non-interactive zero-knowledge proof system $\Pi = (\mathsf{Setup}, \mathsf{Proof}, \mathsf{Verify})$ for R_{ord}. Run $(\mathbf{sk}, \mathbf{pk}) \leftarrow \mathbf{KeyGen}(1^\kappa)$, $\mathsf{PP} \leftarrow \mathsf{C.PGen}(1^\kappa)$, $\mathsf{crs} \leftarrow \Pi.\mathsf{Setup}(1^\kappa)$, sets $(\mathsf{sk}, \mathsf{pk}) \leftarrow ((\mathbf{sk}, \mathsf{PP}, \mathsf{crs}), (\mathbf{pk}, \mathsf{PP}, \mathsf{crs}))$ and returns $(\mathsf{sk}, \mathsf{pk})$.

$\underline{\mathsf{Sign}(\mathsf{sk}, \mathsf{m}, \mathsf{ADM})}$: Return $\perp$ if $\mathsf{ADM} \neq \infty$. Otherwise, for $i \in [|\mathsf{m}|]$ compute $(C_i, O_i) \leftarrow \mathsf{Commit}(\mathsf{PP}, i)$, and for $i \in [|\mathsf{m}| - 1]$ compute $\pi_i \leftarrow \mathsf{Proof}(\mathsf{crs}, (C_i, C_{i+1}), (O_i, O_{i+1}))$. Finally, return (m, σ) and RED, where

$$\sigma \leftarrow (\delta, (C_i)_{i=1}^{|\mathsf{m}|}, (\pi_i)_{i=1}^{|\mathsf{m}|-1}),\ \mathsf{RED} \leftarrow (O_i)_{i=1}^{|\mathsf{m}|},\ \text{and}$$
$$\delta \leftarrow \mathbf{Sign}(\mathbf{sk}, \textstyle\bigcup_{i\in[|\mathsf{m}|]}\{C_i||m_i\}, \infty).^\ddagger$$

$\underline{\mathsf{Verify}(\mathsf{pk}, \sigma, \mathsf{m})}$: Parse σ as $(\delta, (C_i)_{i=1}^{|\mathsf{m}|}, (\pi_i)_{i=1}^{|\mathsf{m}|-1})$ and return 1 if the following holds and 0 otherwise:

$$\mathbf{Verify}(\mathbf{pk}, \delta, \textstyle\bigcup_{i\in[|\mathsf{m}|]}\{C_i||m_i\}) = 1 \wedge$$
$$\forall\, i \in [|\mathsf{m}|] : \Pi.\mathsf{Verify}(\mathsf{crs}, (C_i, C_{i+1}), \pi_i) = 1$$

$\underline{\mathsf{Redact}(\mathsf{pk}, \sigma, \mathsf{m}, \mathsf{MOD}, \mathsf{RED})}$: Parse σ as $(\delta, (C_i)_{i=1}^{|\mathsf{m}|}, (\pi_i)_{i=1}^{|\mathsf{m}|-1})$, and RED as $(O_i)_{i=1}^{|\mathsf{m}|}$. Set $\mathsf{MOD}' \leftarrow \bigcup_{m_i\in\mathsf{MOD}}\{C_i||m_i\}$, and run $(\cdot, \mathring{\delta}) \leftarrow \mathbf{Redact}(\mathbf{pk}, \delta, \bigcup_{i=1}^{|\mathsf{m}|}\{(C_i||m_i)\}, \mathsf{MOD}')$. For $m_i \in \mathsf{MOD}$ remove the corresponding entries from m, $(C_i)_{i=1}^{|\mathsf{m}|}$ and $(O_i)_{i=1}^{|\mathsf{m}|}$ to obtain $\mathring{\mathsf{m}}$, $(C_i)_{i=1}^{|\mathring{\mathsf{m}}|}$, and $(O_i)_{i=1}^{|\mathring{\mathsf{m}}|}$. Finally, for $i \in [|\mathring{\mathsf{m}}| - 1]$ compute

$$\pi_i \leftarrow \mathsf{Proof}(\mathsf{crs}, (C_i, C_{i+1}), (O_i, O_{i+1})),$$

set $\mathring{\sigma} \leftarrow (\mathring{\delta}, (C_i)_{i=1}^{|\mathring{\mathsf{m}}|}, (\pi_i)_{i=1}^{|\mathring{\mathsf{m}}|-1})$, $\mathring{\mathsf{RED}} \leftarrow (O_i)_{i=1}^{|\mathring{\mathsf{m}}|}$ and return $(\mathring{\mathsf{m}}, \mathring{\sigma})$ and RED'.

‡ We note that we set $\mathsf{RED} \leftarrow (O_i)_{i=1}^{|\mathsf{m}|} = (m_i, r_i)_{i=1}^{|\mathsf{m}|}$ for notational convenience, while one would only require $\mathsf{RED} \leftarrow (r_i)_{i=1}^{|\mathsf{m}|}$.

Scheme 5.3: A designated redactor RS for linear documents.

set ([LAN02], cf. [CCJT13] for an overview). It also works in the prime order group setting. Here, the relation R_{ord} for Π is as follows:

$$((C_1, C_2), (x, r)) \in R_{\mathsf{ord}} \iff C_2 - C_1 = \mathsf{Commit}(x;\ r) \ \wedge\ 0 \leq x < k.$$

This approach yields $O(n \log k)$ signature generation costs, signature size and verification costs and a constant size public key.

Signature-Based Approach. This technique [CCS08] pursues the idea of signing every element in the interval[8] using a suitable signature scheme (KeyGen, $\mathsf{Sign}, \mathsf{Verify}$). In our application let the interval be $[0, k[$ and let us denote the corresponding public signatures by $\sigma = (\sigma_0, \sigma_1, \ldots, \sigma_{k-1})$. Now, proving membership of x in $[0, k[$ amounts to the relation R_{ord} under crs being σ and the

[8] Actually, [CCS08] also propose a combination of this approach with a (multi)-base decomposition, which we do not consider here for brevity.

respective public key pk_σ (public parameters):

$$((C_1, C_2), (x, r)) \in R_{\mathsf{ord}} \iff C_2 - C_1 = \mathsf{Commit}(x;\ r) \ \wedge \\ \exists\ i \in [0, k[\ : \mathsf{Verify}(\mathsf{pk}_\sigma, x, \sigma_i) = 1.$$

This approach yields $O(n)$ signature generation cost, signature size and verification cost. The crs representing the public signatures and the verification key may be included into the public key of **RS**, yielding a public key of size $O(k)$.

Theorem 5.3. *If* C *is correct, perfectly binding and hiding,* Π *is complete and adaptively zero-knowledge, and* **RS** *is correct, unforgeable and transparent, then Scheme 5.3 is secure.*

We prove Theorem 5.3 by proving Lemma 5.7-5.9 and deriving Corollary 5.4.

Lemma 5.7. *If* **RS** *is correct,* C *is correct, and* Π *is complete, then Scheme 5.3 is correct.*

The lemma above follows from inspection.

Lemma 5.8. *If* **RS** *is unforgeable,* C *is perfectly binding, and* Π *is sound, then Scheme 5.3 is unforgeable.*

Proof. To prove unforgeability, we show how an efficient adversary against unforgeability $\mathcal{A}^{\mathsf{uf}}$ can be used to construct (1) an efficient adversary $\mathcal{A}^{\mathsf{uf}}$ against the unforgeability of the underlying RS for sets **RS** or (2) an efficient adversary $\mathcal{A}^{\mathsf{so}}$ against the soundness of the underlying proof system. Below we describe efficient reductions $\mathcal{R}^{\mathsf{uf}}$ and $\mathcal{R}^{\mathsf{so}}$, respectively.

$\underline{\mathcal{R}^{\mathsf{uf}}}$: $\mathcal{R}^{\mathsf{uf}}$ obtains the RS public key **pk** from the challenger $\mathcal{C}^{\mathsf{uf}}$ of the unforgeability game of the used redactable signature scheme for sets and completes the setup by running $\mathsf{pp} \leftarrow \mathsf{C.PGen}(1^\kappa)$, $\mathsf{crs} \leftarrow \Pi.\mathsf{Setup}(1^\kappa)$, setting $(\mathsf{sk}, \mathsf{pk}) \leftarrow ((\perp, \mathsf{pp}, \mathsf{crs}), (\mathbf{pk}, \mathsf{pp}, \mathsf{crs}))$ and handing pk to $\mathcal{A}^{\mathsf{uf}}$. It is easy to see that $\mathcal{R}^{\mathsf{cf}}$ can simulate all oracles for $\mathcal{A}^{\mathsf{uf}}$ by forwarding the respective calls to **Sign** to the oracles provided by $\mathcal{C}^{\mathsf{uf}}$. Eventually, $\mathcal{A}^{\mathsf{uf}}$ outputs a valid tuple $(\mathsf{m}^\star, \sigma^\star)$, where $\sigma^\star = (\delta^\star, (C_i)_{i=1}^{|\mathsf{m}^\star|}, (\pi_i)_{i=1}^{|\mathsf{m}^\star|-1})$ such that $\nexists\ (\mathsf{m}, \infty) \in \mathcal{Q}_{\mathsf{Sign}} : \mathsf{m}^\star \in \mathrm{span}_\vdash(\mathsf{m})$. If a superset of $\bigcup_{i=1}^{|\mathsf{m}^\star|}\{(C_i||m_i)\}$ was signed using the oracle provided by $\mathcal{C}^{\mathsf{uf}}$ it aborts. Otherwise, it outputs $(\delta^\star, \bigcup_{i=1}^{|\mathsf{m}|}\{(C_i||m_i)\})$ as a forgery for the RS for sets.

$\underline{\mathcal{R}^{\mathsf{so}}}$: $\mathcal{R}^{\mathsf{so}}$ obtains crs from the challenger $\mathcal{C}^{\mathsf{so}}$ of the soundness game of the underlying non-interactive proof system and completes the setup by running $(\mathbf{sk}, \mathbf{pk}) \leftarrow \mathbf{KeyGen}(1^\kappa)$, $\mathsf{pp} \leftarrow \mathsf{C.PGen}(1^\kappa)$, setting $(\mathsf{sk}, \mathsf{pk}) \leftarrow ((\mathbf{sk}, \mathsf{pp}, \mathsf{crs}), (\mathbf{pk}, \mathsf{pp}, \mathsf{crs}))$ and handing pk to $\mathcal{A}^{\mathsf{uf}}$. It is easy to see that the reduction can simulate all oracles as in the real game. Eventually, $\mathcal{A}^{\mathsf{uf}}$ outputs a valid tuple $(\mathsf{m}^\star, \sigma^\star)$, where $\sigma^\star = (\delta^\star, (C_i)_{i=1}^{|\mathsf{m}^\star|}, (\pi_i)_{i=1}^{|\mathsf{m}^\star|-1})$ such that $\nexists\ (\mathsf{m}, \infty) \in \mathcal{Q}_{\mathsf{Sign}} : \mathsf{m}^\star \in \mathrm{span}_\vdash(\mathsf{m})$. If no superset of $\bigcup_{i=1}^{|\mathsf{m}^\star|}\{(C_i||m_i)\}$ was ever signed it aborts.

Otherwise, there is at least one i for $1 \leq i < |\mathsf{m}^*|$ such that $\Pi.\mathsf{Verify}(\mathsf{crs}, (C_i, C_{i+1}), \pi_i) = 1$ but $(C_i, C_{i+1}) \notin L$, which means that $\mathcal{R}^{\mathsf{so}}$ can output $((C_i, C_{i+1}), \pi_i)$ to win the soundness game of the non-interactive proof system.

Overall Bound. The simulations in both reductions are indistinguishable from a real game. In front of an adversary we randomly guess the adversary's strategy, inducing a loss of $1/2$. Our reductions for both forger types always succeed if the forger succeeds. This means that the probability to break group unforgeability is upper-bounded by $2 \cdot \max\{\varepsilon_{\mathsf{uf}}(\kappa), \varepsilon_{\mathsf{so}}(\kappa)\}$. □

Lemma 5.9. *If* **RS** *is transparent,* C *is hiding, and* Π *is adaptively zero-knowledge, the construction in Scheme 5.3 is transparent.*

Proof. We will bound the probability to win the transparency game by using a sequence of games. Thereby, we denote the event that the adversary wins Game i by S_i and let k be the overall number of required commitments.

Game 0: The original transparency game.

Game 1: As the original game, but the environment sets up the crs for the non-interactive proof system using the simulator S_1, i.e., $(\mathsf{crs}, \tau) \leftarrow \mathsf{S}_1(1^\kappa)$ and followingly simulates all proofs using $\mathsf{S}_2(\mathsf{crs}, \tau, \cdot, \cdot)$.

Transition Game 0 → Game 1: A distinguisher between Game 0 and Game 1 is an adaptive zero-knowledge distinguisher for Π, i.e., $|\Pr[S_0] - \Pr[S_1]| \leq \varepsilon_{\mathsf{zk}}(\kappa)$.

Game 2: As Game 1, but all commitments inside $\mathcal{O}^{\mathsf{SoR}}$ are replaced by commitments to 0.

Transition Game 1 → Game 2: A distinguisher between Game 1 and Game 2 is a distinguisher for the hiding game of C, i.e., the distinguishing probability $|\Pr[S_1] - \Pr[S_2]| \leq k \cdot \varepsilon_{\mathsf{hd}}(\kappa)$.[9]

In Game 2, all values except δ are independent of the bit b, meaning that the adversary has the same advantage as in the transparency game of the underlying RS for sets, i.e., $\Pr[S_2] = 1/2 + \varepsilon_{\mathsf{RS}}(\kappa)$. Taking all together, we have $\Pr[S_0] \leq 1/2 + \varepsilon_{\mathsf{RS}}(\kappa) + \varepsilon_{\mathsf{zk}}(\kappa) + k \cdot \varepsilon_{\mathsf{hd}}(\kappa)$, which concludes the proof. □

The lemma above yields the following corollary.

Corollary 5.4. *If* **RS** *is transparent,* C *is hiding, and* Π *is adaptively zero-knowledge, then Scheme 5.3 is private.*

[9] For compactness, we combine the exchange of the commitments in one game change and note that it is straightforward to unroll the exchange of the commitments in k game changes.

5.2 Extended Privacy for Redactable Signatures

The work presented in this section is dedicated to the development of a cryptographically enhanced solution for a real world hospital, which is currently planning to complement its existing information sharing system for electronic patient data with additional privacy features. The overall idea of the system is to grant patients access to all their medical records via a cloud-based platform. The patients are then able to use this as a central hub to distribute their documents to different stakeholders, e.g., to request reimbursement by the insurance, or to forward (parts of) the documents to the family doctor for further treatment. While means for access control and data confidentiality are already in place, the system should be complemented by strong authenticity guarantees. At the same time a high degree of privacy should be maintained, i.e., by allowing the patients, on a fine-granular basis, to decide which parts of which document should be visible to which party. For instance, the family doctor might *not* need to learn the precise costs of a treatment; similarly a medical research laboratory should *not* learn the patients' identities.

From a research point of view, one motivation behind this work is to show how rather complex real world scenarios with conflicting interests and strong security and privacy requirements can be elegantly and securely realized by means of rigorous cryptographic design and analysis. More importantly, we can indeed come up with provably secure and practical solutions being well suited for real world use. We also note that our results are not limited to the proposed application, but can also be directly applied to various other contexts such as notary authorities or e-government services. Now, we discuss the motivation for our design.

Redactable Signatures. A trivial solution for the above problem would be to let the hospital cloud create a fresh signature on the information to be revealed every time the user wishes to forward authentic subsets of a document to other parties. However, this is not satisfactory as it would require strong trust assumptions into the cloud: one could not efficiently guarantee that the signed data has not been altered over time by the cloud or by a malicious intruder. It is therefore preferable to use *redactable signatures* (RS). Then it is not necessary to let the cloud attest the authenticity of the forwarded data, as the signature on the redacted document can be extracted from the doctor's signature on the original document without requiring the doctor's secret signing key or further interaction with the doctor.

Designated Verifiers. Unfortunately, using redactable signatures in their vanilla form in our scenario would lead to severe privacy problems, i.e., everyone getting hold of a signed document would be convinced of its authenticity. In such a case, for instance, if someone leaks a signed health record of an employee to an employer, the employer might reliably learn the employee's disease, and in further consequence dismiss the employee. What is therefore needed is a *designated verifier* for each redacted version of a document. That is, when redacting

a document, the patient should be able to define the intended receiver. Then, while everybody can check the validity of a leaked document, *only* the designated verifier is convinced about its authenticity. This can be achieved by constructing the schemes in a way that the designated verifier can fake indistinguishable signatures on its own. Thereby, one may observe that the property that signatures can be publicly verified might as well be a motivation for designated verifiers to not leak/sell documents, as this reduces the circle of possible suspects to the data owner and the designated verifier.

Group Signatures. Another problem of RS is that they only support a single signer. However, a hospital potentially employing hundreds of doctors will not use a single signing key that is shared by all its employees. By doing so, the identity of the signing doctor could not be revealed in case of a dispute, e.g., after a malpractice. However, using different keys for different doctors poses a privacy risk again. For instance, if the document was signed using an oncologist's key, one could infer sensitive information about the disease—even though the diagnosis was blacked out. What is therefore needed are features known from *group signatures*, where towards the verifier the doctor's identity remains hidden within the set of doctors in the hospital, while re-identification is still possible by a dedicated entity.

Contribution. The properties we need for our scenario are contributed by three distinct cryptographic concepts and what we actually need can be considered as a *signer-anonymous designated-verifier redactable signature scheme*. However, while a lot of existing work studies the different concepts in isolation, there is no work which aims at combining them in a way to profit from a combination of their individual properties. Trying to obtain this by simply combining them in an ad-hoc fashion, however, is dangerous. It is well known that the ad-hoc combination of cryptographic primitives to larger systems is often problematic (as subtle issues often remain hidden when omitting formal analysis) and security breaches resulting from such approaches are often seen in practice. Unlike following such an ad-hoc approach, we follow a rigorous approach and formally model what is required by the use case, introduce a comprehensive security model and propose two (semi-)black-box constructions that are provably secure within our model. While such a (semi-)black-box construction is naturally interesting from a theoretical point of view, our second construction is also entirely practical and thus also well suited to be used within the planned system. Finally, as a contribution which may be of independent interest, we also obtain the first *group redactable signatures* as a byproduct of our definitional framework.

Technical Overview. Our constructions provably achieve the desired functionality by means of a two-tier signature approach: a message is signed using a freshly generated RS key pair where the corresponding public key of this "one-time RS" is certified using a group signature. For the designated verifier feature, we follow two different approaches. Firstly, we follow the naïve approach and use a disjunctive non-interactive proof of knowledge which either demonstrates knowledge of a valid RS signature on the message, *or* it demonstrates knowledge

of a valid signature of the designated verifier on the same message. While this approach is very generic, its efficiency largely depends on the complexity to prove knowledge of an RS signature. To this end, we apply our framework for key-homomorphic signatures from Section 3.2 to one of our generic RS constructions from the previous section. In particular, we use the observation that a large class of RS can easily be turned into RS admitting the required key-homomorphism, to obtain a practical construction. More precisely, besides conventional group signatures and conventional redactable signatures, our approach only requires to prove a single statement demonstrating knowledge of the relation between two RS keys *or* demonstrating knowledge of the designated verifier's secret key. For instance, in the discrete logarithm setting when instantiating this proof using Fiat-Shamir [FS86] transformed Σ-protocols, they are highly efficient as they only require two group exponentiations.

Related Work. Although redactable signatures suffer from the aforementioned problems, we can use them as important building blocks. In particular, we will rely on the general framework for redactable signatures introduced in the previous section. We refer the reader to this section for a discussion of related work.

Besides that, there is a large body of work on signatures with designated verifiers. However, none of the approaches considers selective disclosure via redaction or a group signing feature. In designated verifier (DV) signatures (or proofs) [JSI96], a signature produced by a signer can only be validated by a single user who is designated by the signer during the signing process (cf. [LWB05] for a refined security model). Designation can only be performed by the signer and verification requires the designated verifier's secret. Thus, this concept is not directly applicable to our setting. In [JSI96] also the by now well known "*OR trick*" was introduced as a DV construction paradigm.

Undeniable signatures [CA89] are signatures that can not be verified without the signer's cooperation and the signer can either prove that a signature is valid or invalid. This is not suitable for us as this is an interactive process.

Designated confirmer signatures [Cha94] introduce a third entity besides the signer and the verifier called designated confirmer. This party, given a signature, has the ability to privately verify it as well as to convince anyone of its validity or invalidity. Additionally, the designated confirmer can convert a designated confirmer signature into an ordinary signature that is then publicly verifiable. This is not suitable for our scenario, as it is exactly the opposite of what we require, i.e., here the signature for the confirmer is not publicly verifiable, but the confirmer can always output publicly verifiable versions of this signature.

Another concept, which is closer to the designation functionality that we require, are universal designated verifier (UDV) signatures introduced in [SBWP03]. They are similar to designated verifier signatures, but universal in the sense that any party who is given a publicly verifiable signature from the signer can designate the signature to any designated verifier by using the verifier's public key. Then, the designated verifier can verify that the message was signed by the signer, but is unable to convince anyone else of this fact. Like with ordinary

DV signatures, UDV signatures also require the designated verifier's secret key for verification. There are some generic results for UDV signatures. In [Ver06] it was shown how to convert various pairing-based signature schemes into UDV signatures. In [SS08] it was shown how to convert a large class of signature schemes into UDV signatures. Some ideas in our second construction are conceptually related to this generic approach. However, as we only require to prove relations among public keys, our approach is conceptually simpler and often more efficient.

5.2.1 Formal Security Model

Now we formally define signer-anonymous designated-verifier redactable signature schemes (AD-RS). To obtain the most general result, we follow our generic model for redactable signatures and do not make the structure of the messages to be signed explicit. Inspired by [MPV09], we view signatures output by Sign as being of the form $\sigma = (\underline{\sigma}, \overline{\sigma})$. That is, signatures are composed of a public signature component $\underline{\sigma}$ and a private signature component $\overline{\sigma}$, where $\underline{\sigma}$ may also be empty. Intuitively, this allows for stronger security definitions: while our signatures are malleable by definition we may still require that the public components are non-malleable. For the sake of simple presentation we model our system for static groups, since an extension to dynamic groups [BSZ05] is straight forward.

Definition 5.5. *An* AD-RS *is a tuple* (Setup, DVGen, Sign, GVerify, Open, Redact, Verify, Sim) *of PPT algorithms, which are defined as follows.*

$\mathsf{Setup}(1^\kappa, n)$: *Takes a security parameter* κ *and the group size* $n \leq \mathsf{poly}(\kappa)$ *as input. It generates and outputs a group public key* gpk*, a group opening key* gok*, and a list of group signing keys* $\mathsf{gsk} = \{\mathsf{gsk}_i\}_{i\in[n]}$.

$\mathsf{DVGen}(1^\kappa)$: *Takes a security parameter* κ *as input and outputs a designated verifier key pair* $(\mathsf{vsk}_j, \mathsf{vpk}_j)$.

$\mathsf{Sign}(\mathsf{gsk}_i, \mathsf{m}, \mathsf{ADM})$: *Takes a group signing key* gsk_i*, a message* m*, and admissible modifications* ADM *as input, and outputs a signature* σ.

$\mathsf{GVerify}(\mathsf{gpk}, \mathsf{m}, \sigma)$: *Takes a group public key* gpk*, a message* m*, and a signature* σ *as input, and outputs a bit* b.

$\mathsf{Open}(\mathsf{gok}, \mathsf{m}, \sigma)$: *Takes a group opening key* gok*, a message* m*, and a valid signature* σ *as input, and outputs an identity* i.

$\mathsf{Redact}(\mathsf{gpk}, \mathsf{vpk}_j, \mathsf{m}, \sigma, \mathsf{MOD})$: *Takes a group public key* gpk*, a designated-verifier public key* vpk_j*, a message* m*, a valid signature* σ*, and modification instructions* MOD *as input, and returns a designated-verifier message-signature pair* $(\mathring{\mathsf{m}}, \rho)$.

$\mathsf{Verify}(\mathsf{gpk}, \mathsf{vpk}_j, \mathsf{m}, \rho)$: *Takes a group public key* gpk*, a designated-verifier public key* vpk_j*, a message* m*, and a designated-verifier signature* ρ*. It returns a bit* b.

$\mathsf{Sim}(\mathsf{gpk}, \mathsf{vsk}_j, \mathsf{m}, \mathsf{ADM}, \mathsf{MOD}, \underline{\sigma})$**:** *Takes a group public key* gpk*, a designated-verifier secret key* vsk_j*, a message* m*, admissible modifications* ADM*, modification instructions* MOD*, and a valid public signature component* $\underline{\sigma}$ *as input and outputs a designated-verifier message signature pair* $(\mathring{\mathsf{m}}, \rho)$*.*

Oracles. We base our security notions on the following oracles and assume that $(\mathsf{gpk}, \mathsf{gok}, \mathsf{gsk})$ generated in the experiments are implicitly available to them. The environment stores a list $\mathtt{DVK}$ of designated-verifier key pairs, and a set of public signature components $\mathtt{SIG}$. Each list entry and each set is initially set to $\perp$.

$\mathsf{Key}(i)$: This oracle returns gsk_i.

$\mathsf{DVGen}(j)$: If $\mathtt{DVK}[j] \neq \perp$ this oracle returns $\perp$. Otherwise, it runs $(\mathsf{vsk}_j, \mathsf{vpk}_j) \leftarrow \mathsf{DVGen}(1^\kappa)$, sets $\mathtt{DVK}[j] \leftarrow (\mathsf{vsk}_j, \mathsf{vpk}_j)$, and returns vpk_j.

$\mathsf{DVKey}(j)$: This oracle returns vsk_j.

$\mathsf{Sig}(i, \mathsf{m}, \mathsf{ADM})$: This oracle runs $\sigma = (\underline{\sigma}, \overline{\sigma}) \leftarrow \mathsf{Sign}(\mathsf{gsk}_i, \mathsf{m}, \mathsf{ADM})$, sets $\mathtt{SIG} \leftarrow \mathtt{SIG} \cup \{\underline{\sigma}\}$ and returns σ.

$\mathsf{Open}(\mathsf{m}, \sigma)$: This oracle runs $i \leftarrow \mathsf{Open}(\mathsf{gok}, \mathsf{m}, \sigma)$ and returns i.

$\mathsf{Sim}(j, \mathsf{m}, \mathsf{ADM}, \mathsf{MOD}, \underline{\sigma})$: If $\underline{\sigma} \notin \mathtt{SIG}$, this oracle returns $\perp$. Otherwise, it runs $(\mathring{\mathsf{m}}, \rho) \leftarrow \mathsf{Sim}(\mathsf{gpk}, \mathsf{vsk}_j, \mathsf{m}, \mathsf{ADM}, \mathsf{MOD}, \underline{\sigma})$ and returns $(\mathring{\mathsf{m}}, \rho)$.

$\mathsf{RoS}(b, j, \mathsf{m}, \mathsf{ADM}, \mathsf{MOD}, \sigma)$: If $b = 0$, this oracle runs $(\mathring{\mathsf{m}}, \rho) \leftarrow \mathsf{Redact}(\mathsf{gpk}, \mathsf{vpk}_j, \mathsf{m}, \sigma, \mathsf{MOD})$ and returns $(\mathring{\mathsf{m}}, \rho)$. Otherwise, it uses the Sim oracle to obtain $(\mathring{\mathsf{m}}, \rho) \leftarrow \mathsf{Sim}(j, \mathsf{m}, \mathsf{ADM}, \mathsf{MOD}, \underline{\sigma})$ and returns $(\mathring{\mathsf{m}}, \rho)$.

$\mathsf{Ch}(i, j, (\mathsf{m}_0, \mathsf{ADM}_0, \mathsf{MOD}_0), (\mathsf{m}_1, \mathsf{ADM}_1, \mathsf{MOD}_1), b)$: This oracle runs $\sigma_c \leftarrow \mathsf{Sign}(\mathsf{gsk}_i, \mathsf{m}_c, \mathsf{ADM}_c)$, $(\mathring{\mathsf{m}}_c, \rho_c) \leftarrow \mathsf{Redact}(\mathsf{vpk}_j, \mathsf{m}_c, \sigma_c, \mathsf{MOD}_c)$, for $c \in \{0, 1\}$. If $\mathring{\mathsf{m}}_0 \neq \mathring{\mathsf{m}}_1 \vee \mathring{\mathsf{ADM}}_0 \neq \mathring{\mathsf{ADM}}_1$, it returns $\perp$ and $(\mathring{\mathsf{m}}_b, \underline{\sigma}_b, \rho_b)$ otherwise.[10]

The environment stores the oracle queries in lists. In analogy to the oracle labels, we use $\mathcal{Q}^{\mathsf{Key}}$, $\mathcal{Q}^{\mathsf{DVGen}}$, $\mathcal{Q}^{\mathsf{DVKey}}$, $\mathcal{Q}^{\mathsf{Sig}}$, $\mathcal{Q}^{\mathsf{Open}}$, $\mathcal{Q}^{\mathsf{Sim}}$, $\mathcal{Q}^{\mathsf{RoS}}$, and $\mathcal{Q}^{\mathsf{Ch}}$ to denote them.

Security Notions. We require AD-RS to be correct, group unforgeable, designated-verifier unforgeable, simulatable, signer anonymous, and private.

Correctness guarantees that all honestly computed signatures verify correctly.

Formally, we require that for all $n \in \mathbb{N}$, for all $(\mathsf{gpk}, \mathsf{gok}, \mathsf{gsk}) \leftarrow \mathsf{Setup}(1^\kappa, n)$, for all $(\mathsf{vsk}_j, \mathsf{vpk}_j) \leftarrow \mathsf{DVGen}(1^\kappa)$, for all $(\mathsf{vsk}_\ell, \mathsf{vpk}_\ell) \leftarrow \mathsf{DVGen}(1^\kappa)$, for all $(\mathsf{m}, \mathsf{ADM}, \mathsf{MOD})$ where $\mathsf{MOD} \preceq \mathsf{ADM} \wedge \mathsf{ADM} \preceq \mathsf{m}$, for all $(\mathsf{m}', \mathsf{ADM}', \mathsf{MOD}')$ where $\mathsf{MOD}' \preceq \mathsf{ADM}' \wedge \mathsf{ADM}' \preceq \mathsf{m}'$ for all $i \in [n]$, for all $\sigma = (\underline{\sigma}, \overline{\sigma}) \leftarrow \mathsf{Sign}(\mathsf{gsk}_i, \mathsf{m}, \mathsf{ADM})$, for all $u \leftarrow \mathsf{Open}(\mathsf{gok}, \mathsf{m}, \sigma)$, for all $(\mathring{\mathsf{m}}, \rho) \leftarrow \mathsf{Redact}(\mathsf{gpk}, \mathsf{vpk}_j, \mathsf{m}, \sigma, \mathsf{MOD})$, for all $(\mathring{\mathsf{m}}', \rho') \leftarrow \mathsf{Sim}(\mathsf{gpk}, \mathsf{vsk}_\ell, \mathsf{m}', \mathsf{ADM}', \mathsf{MOD}', \underline{\sigma})$, it holds with overwhelming probability in the security parameter κ that $\mathsf{GVerify}(\mathsf{gpk}, \mathsf{m}, \sigma) = 1 \wedge i = u \wedge \mathsf{Verify}(\mathsf{gpk}, \mathsf{vpk}_j, \mathring{\mathsf{m}}, \rho) = 1 \wedge \mathsf{Verify}(\mathsf{gpk}, \mathsf{vpk}_\ell, \mathring{\mathsf{m}}', \rho') = 1$ and that $\mathring{\mathsf{m}} \overset{\mathsf{MOD}}{\longleftarrow} \mathsf{m} \wedge \mathring{\mathsf{m}}' \overset{\mathsf{MOD}'}{\longleftarrow} \mathsf{m}'$.

[10] Here $\mathring{\mathsf{ADM}}_0$ and $\mathring{\mathsf{ADM}}_1$ are derived from ADM_0 and ADM_1 with respect to MOD_0 and MOD_1.

Group unforgeability captures the intuition that the only way of obtaining valid signatures on messages is by applying "allowed" modifications to messages which were initially signed by a group member. Moreover, this property guarantees that every valid signature can be linked to the original signer by some authority.

Technically, the definition captures the traceability property of group signatures while simultaneously taking the malleability of RS into account.

Definition 5.6. *An* AD-RS *is group unforgeable, if for all PPT adversaries* $\mathcal{A}$ *there is a negligible function* $\varepsilon(\cdot)$ *such that*

$$\Pr\left[\begin{array}{l}(\mathsf{gpk},\mathsf{gok},\mathsf{gsk})\leftarrow\mathsf{Setup}(1^\kappa,n),\\ \mathcal{O}\leftarrow\{\mathsf{Sig}(\cdot,\cdot,\cdot),\mathsf{Key}(\cdot)\},\\ (\mathsf{m}^\star,\sigma^\star)\leftarrow\mathcal{A}^{\mathcal{O}}(\mathsf{gpk},\mathsf{gok}),\\ u\leftarrow\mathsf{Open}(\mathsf{gok},\mathsf{m}^\star,\sigma^\star)\end{array}\;:\;\begin{array}{r}\mathsf{GVerify}(\mathsf{gpk},\mathsf{m}^\star,\sigma^\star)=1\;\wedge\\ (u=\bot\;\vee\;(u\notin\mathcal{Q}^{\mathsf{Key}}\;\wedge\\ \nexists(u,\mathsf{m},\mathsf{ADM})\in\mathcal{Q}^{\mathsf{Sig}}:\mathsf{m}^\star\overset{\mathsf{ADM}}{\preceq}\mathsf{m}))\end{array}\right]\leq\varepsilon(\kappa).$$

Designated-verifier unforgeability models the requirement that a designated-verifier signature can only be obtained in two ways: either by corretly redacting a signature (which can be done by everybody having access to the latter), or by having access to the secret key of the designated verifier. The former option would be chosen whenever a signature is to be legitimately forwarded to a receiver, while the latter enables the designated verifier to fake signatures.

Together with the previous definition, designated-verifier unforgeability guarantees that no adversary can come up with a designated-verifier signature for a foreign public key: by Definition 5.6 it is infeasible to forge a signature—and Definition 5.7 states that the only way of generating a designated-verifier signature for somebody else is to know a valid signature to start from.

Definition 5.7. *An* AD-RS *is designated-verifier unforgeable, if there exists a PPT opener* $\mathsf{O}=(O_1,O_2)$ *such that for every PPT adversary* $\mathcal{A}$ *there is a negligible function* $\varepsilon_1(\cdot)$ *such that*

$$\left|\begin{array}{l}\Pr\left[(\mathsf{gpk},\mathsf{gok},\mathsf{gsk})\leftarrow\mathsf{Setup}(1^\kappa,n):\mathcal{A}(\mathsf{gpk},\mathsf{gok},\mathsf{gsk})=1\right]\;-\\ \Pr\left[(\mathsf{gpk},\mathsf{gok},\mathsf{gsk},\tau)\leftarrow O_1(1^\kappa,n):\mathcal{A}(\mathsf{gpk},\mathsf{gok},\mathsf{gsk})=1\right]\end{array}\right|\leq\varepsilon_1(\kappa),$$

and for every PPT adversary $\mathcal{A}$ *there is a negligible function* $\varepsilon_2(\cdot)$ *such that*

$$\Pr\left[\begin{array}{l}(\mathsf{gpk},\mathsf{gok},\mathsf{gsk},\tau)\leftarrow O_1(1^\kappa,n),\mathcal{O}\leftarrow\{\mathsf{Sig}(\cdot,\cdot,\cdot),\mathsf{Key}(\cdot),\\ \mathsf{DVGen}(\cdot),\mathsf{DVKey}(\cdot),\mathsf{Sim}(\cdot,\cdot,\cdot,\cdot,\cdot)\},\\ (\mathsf{m}^\star,\rho^\star,v^\star)\leftarrow\mathcal{A}^{\mathcal{O}}(\mathsf{gpk},\mathsf{gok}),\;u\leftarrow O_2(\tau,\mathtt{DVK},\mathsf{m}^\star,\rho^\star,v^\star)\end{array}\;:\;\begin{array}{r}\mathsf{Verify}(\mathsf{gpk},\mathsf{vpk}_{v^\star},\mathsf{m}^\star,\rho^\star)=1\;\wedge\;v^\star\notin\mathcal{Q}^{\mathsf{DVKey}}\;\wedge\\ (u=\bot\;\vee\;(u\notin\mathcal{Q}^{\mathsf{Key}}\;\wedge\;\nexists(u,\mathsf{m},\mathsf{ADM})\in\mathcal{Q}^{\mathsf{Sig}}:\mathsf{m}^\star\overset{\mathsf{ADM}}{\preceq}\mathsf{m}))\;\wedge\\ \nexists(v^\star,\mathsf{m},\mathsf{ADM},\cdot,\cdot)\in\mathcal{Q}^{\mathsf{Sim}}:\mathsf{m}^\star\overset{\mathsf{ADM}}{\preceq}\mathsf{m})\end{array}\right]\leq\varepsilon_2(\kappa).$$

In our definition, we assume a simple key registration for designated verifiers to ensure that all designated-verifier key pairs have been honestly created and

thus an adversary is not able to mount rogue key attacks. In practice, this requirement can often be alleviated by introducing an option to check the honest generation of the keys (cf. [RY07]), which we omit for simplicity.

Simulatability captures that designated verifiers can simulate signatures on arbitrary messages which are indistinguishable from honestly computed signatures.

Definition 5.8. *An* AD-RS *satisfies the simulatability property, if for all PPT adversaries $\mathcal{A}$ there is a negligible function $\varepsilon(\cdot)$ such that it holds that*

$$\Pr\left[\begin{array}{l} (\mathsf{gpk},\mathsf{gok},\mathsf{gsk}) \leftarrow \mathsf{Setup}(1^\kappa, n),\ b \xleftarrow{R} \{0,1\}, \\ \mathcal{O} \leftarrow \{\mathsf{DVGen}(\cdot), \mathsf{DVKey}(\cdot)\}, \\ ((\mathsf{m}_0, \mathsf{ADM}_0, \mathsf{MOD}_0), (\mathsf{m}_1, \mathsf{ADM}_1), \\ i^\star, j^\star, \mathsf{ST}) \leftarrow \mathcal{A}^{\mathcal{O}}(\mathsf{gpk},\mathsf{gok},\mathsf{gsk}), \\ \sigma = (\underline{\sigma}, \overline{\sigma}) \leftarrow \mathsf{Sign}(\mathsf{gsk}_{i^\star}, \mathsf{m}_b, \mathsf{ADM}_b), \\ (\mathring{\mathsf{m}}_0, \rho) \leftarrow \mathsf{RoS}(b, j^\star, \mathsf{m}_0, \mathsf{ADM}_0, \mathsf{MOD}_0, \sigma), \\ b^\star \leftarrow \mathcal{A}^{\mathcal{O}}(\underline{\sigma}, \mathring{\mathsf{m}}_0, \rho, \mathsf{ST}) \end{array} : \begin{array}{r} \\ \\ b = b^\star \ \wedge \\ \\ \mathsf{ADM}_0 \preceq \mathsf{m}_0 \ \wedge \\ \mathsf{ADM}_1 \preceq \mathsf{m}_1 \\ \\ \\ \end{array}\right] \leq 1/2 + \varepsilon(\kappa).$$

As mentioned earlier, we assume that signatures consist of a private and a public component (the latter being denoted by $\underline{\sigma}$). To eliminate potential privacy issues associated with a public $\underline{\sigma}$, we also give $\underline{\sigma}$ as input to the simulator and the adversary, and require that the adversary cannot tell real and faked signatures apart *even when knowing* $\underline{\sigma}$. This way, our definitional framework guarantees that these parts do not contain any sensitive information.

In a realization of the system, the public parts of all signatures issued by the hospital would be made publicly available (without further meta-information).

Signer anonymity requires that only the opening authority can determine the identity of a signer.

Definition 5.9. *An* AD-RS *is signer anonymous, if for all PPT adversaries $\mathcal{A}$ there is a negligible function $\varepsilon(\cdot)$ such that*

$$\Pr\left[\begin{array}{l} (\mathsf{gpk},\mathsf{gok},\mathsf{gsk}) \leftarrow \mathsf{Setup}(1^\kappa, n),\ b \xleftarrow{R} \{0,1\}, \\ \mathcal{O} \leftarrow \{\mathsf{Open}(\cdot,\cdot)\},\ (i_0^\star, i_1^\star, \mathsf{m}^\star, \mathsf{ADM}^\star, \mathsf{ST}) \leftarrow \mathcal{A}^{\mathcal{O}}(\mathsf{gpk},\mathsf{gsk}), \\ \sigma \leftarrow \mathsf{Sign}(\mathsf{gsk}_{i_b^\star}, \mathsf{m}^\star, \mathsf{ADM}^\star),\ b^\star \leftarrow \mathcal{A}^{\mathcal{O}}(\sigma, \mathsf{ST}) \end{array} : \begin{array}{r} b = b^\star \ \wedge \\ \nexists (\mathsf{m}, (\underline{\sigma}, \cdot)) \in \mathcal{Q}_2^{\mathsf{Open}} : \\ \mathsf{m} \overset{\mathsf{ADM}}{\preceq} \mathsf{m}^\star \end{array}\right] \leq 1/2 + \varepsilon(\kappa),$$

and $\mathcal{A}$ runs in two stages and $\mathcal{Q}_2^{\mathsf{Open}}$ records queries to oracle Open *in stage two.*

The definition guarantees that—no matter how many signatures already have been opened—the signers' identities for all other signatures remain secret. The formulation is, up to the last clause of the winning condition, similar to the anonymity definition of group signature schemes (cf. Definition 2.25). We, however, need to adapt the last clause because Definition 2.25 requires signatures to be non-malleable. In contrast, our signatures are malleable by definition. However, we can still require parts of the signature, and in particular the public part,

to be non-malleable. By doing so, we can achieve a strong notion that resembles anonymity in the sense of group signatures whenever honestly generated signatures have different public components with overwhelming probability. This is in particular the case for our instantiations provided in the next sections.

Privacy guarantees that a redacted designated-verifier signature does not leak anything about the blacked-out parts of the original message.

Definition 5.10. *An* AD-RS *is private, if for all PPT adversaries* $\mathcal{A}$ *there is a negligible function* $\varepsilon(\cdot)$ *such that*

$$\Pr\left[\begin{array}{l|l} (\mathsf{gpk},\mathsf{gok},\mathsf{gsk}) \leftarrow \mathsf{Setup}(1^\kappa, n),\ b \xleftarrow{R} \{0,1\}, & \\ \mathcal{O} \leftarrow \{\mathsf{Sig}(\cdot,\cdot,\cdot), \mathsf{Ch}(\cdot,\cdot,\cdot,\cdot,b)\}, & b = b^\star \\ b^\star \leftarrow \mathcal{A}^{\mathcal{O}}(\mathsf{gpk},\mathsf{gok},\mathsf{gsk}) & \end{array}\right] \leq 1/2 + \varepsilon(\kappa).$$

We call an AD-RS *secure*, if it is correct, group unforgeable, designated-verifier unforgeable, simulatable, signer anonymous, and private.

Group Redactable Signatures. When omitting the DV-related notions and oracles, one directly obtains a definition of group redactable signatures, which may also be useful for applications that require revocable signer-anonymity.

5.2.2 A Generic Construction

Now we present a simple generic construction which can be built by combining any GS, any RS, and any Π that admits proofs of knowledge in a black-box way. In Scheme 5.4 we present our construction which follows the intuition given in the introduction. We use Π to prove knowledge of a witness for the following **NP** relation R required by the verification of designated-verifier signatures.

$$((\mathsf{m},\mathsf{pk},\mathsf{vpk}_j),(\sigma_\mathsf{R},\sigma_\mathsf{v})) \in R \iff \mathsf{RS.Verify}(\mathsf{pk},\mathsf{m},\sigma_\mathsf{R}) = 1 \ \vee\ \Sigma.\mathsf{Verify}(\mathsf{vpk}_j,\mathsf{m},\sigma_\mathsf{v}) = 1.$$

The rationale behind choosing R in this way is that this yields the most general result. That is, no further assumptions on RS or Σ are required. For an instantiation of our construction we can use standard GS and standard RS, where multiple practically efficient instantiations exist. Thus, the time required for signature creation/verification is mainly determined by the cost of the proof of knowledge of the RS signature σ_R. We, however, want to emphasize that—depending on the concrete RS—this proof can usually be instantiated by means of relatively cheap Σ-protocols. Ultimately, as we will show below, we can replace this proof with a much cheaper proof by exploiting properties of the used RS.

Theorem 5.4. *If* GS, RS, *and* Σ *are secure and* Π *is witness indistinguishable and admits proofs of knowledge, then Scheme 5.4 is secure.*

We show that Theorem 5.4 holds by proving Lemma 5.10-5.15.

Lemma 5.10. *If* GS *is correct,* RS *is correct,* Σ *is correct, and* Π *is complete, then Scheme 5.4 is also correct.*

$\underline{\mathsf{Setup}(1^\kappa, n)}$: Run $(\mathsf{gpk}, \mathsf{gok}, \mathsf{gsk}) \leftarrow \mathsf{GS.KeyGen}(1^\kappa, n)$, $\mathsf{crs} \leftarrow \Pi.\mathsf{Setup}(1^\kappa)$, set $\mathsf{gpk}' \leftarrow (\mathsf{gpk}, \mathsf{crs})$ and return $(\mathsf{gpk}', \mathsf{gok}, \mathsf{gsk})$.

$\underline{\mathsf{DVGen}(1^\kappa)}$: Run $(\mathsf{vsk}_j, \mathsf{vpk}_j) \leftarrow \Sigma.\mathsf{KeyGen}(1^\kappa)$ and return $(\mathsf{vsk}_j, \mathsf{vpk}_j)$.

$\underline{\mathsf{Sign}(\mathsf{gsk}_i, \mathsf{m}, \textsc{adm})}$: Run $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{RS.KeyGen}(1^\kappa)$ and return $\sigma = (\underline{\sigma}, \overline{\sigma}) \leftarrow ((\mathsf{pk}, \sigma_\mathsf{G}), (\sigma_\mathsf{R}, \textsc{red}))$, with

$$\sigma_\mathsf{G} \leftarrow \mathsf{GS.Sign}(\mathsf{gsk}_i, \mathsf{pk}), \text{ and } ((\mathsf{m}, \sigma_\mathsf{R}), \textsc{red}) \leftarrow \mathsf{RS.Sign}(\mathsf{sk}, \mathsf{m}, \textsc{adm}).$$

$\underline{\mathsf{GVerify}(\mathsf{gpk}, \mathsf{m}, \sigma)}$: Parse σ as $((\mathsf{pk}, \sigma_\mathsf{G}), (\sigma_\mathsf{R}, \cdot))$ and return 1 if the following holds and 0 otherwise:

$$\mathsf{GS.Verify}(\mathsf{gpk}, \mathsf{pk}, \sigma_\mathsf{G}) = 1 \quad \wedge \quad \mathsf{RS.Verify}(\mathsf{pk}, \mathsf{m}, \sigma_\mathsf{R}) = 1.$$

$\underline{\mathsf{Open}(\mathsf{gok}, \mathsf{m}, \sigma)}$: Parse σ as $((\mathsf{pk}, \sigma_\mathsf{G}), \overline{\sigma})$ and return $\mathsf{GS.Open}(\mathsf{gok}, \mathsf{pk}, \sigma_\mathsf{G})$.

$\underline{\mathsf{Redact}(\mathsf{gpk}, \mathsf{vpk}_j, \mathsf{m}, \sigma, \textsc{mod})}$: Parse σ as $((\mathsf{pk}, \sigma_\mathsf{G}), (\sigma_\mathsf{R}, \textsc{red}))$ and return $(\mathring{\mathsf{m}}, \rho)$, where

$$\begin{aligned}
&((\mathring{\mathsf{m}}, \mathring{\sigma}_\mathsf{R}), \cdot) \leftarrow \mathsf{RS.Redact}(\mathsf{pk}, \mathsf{m}, \sigma_\mathsf{R}, \textsc{mod}, \textsc{red}),\\
&\pi \leftarrow \Pi.\mathsf{Proof}(\mathsf{crs}, (\mathring{\mathsf{m}}, \mathsf{pk}, \mathsf{vpk}_j), (\mathring{\sigma}_\mathsf{R}, \bot)), \text{ and}\\
&\rho \leftarrow ((\mathsf{pk}, \sigma_\mathsf{G}), \pi).
\end{aligned}$$

$\underline{\mathsf{Verify}(\mathsf{gpk}, \mathsf{vpk}_j, \mathsf{m}, \rho)}$: Parse ρ as $((\mathsf{pk}, \sigma_\mathsf{G}), \pi)$ and return 1 if the following holds, and 0 otherwise:

$$\mathsf{GS.Verify}(\mathsf{gpk}, \mathsf{pk}, \sigma_\mathsf{G}) = 1 \quad \wedge \quad \Pi.\mathsf{Verify}(\mathsf{crs}, (\mathsf{m}, \mathsf{pk}, \mathsf{vpk}_j), \pi) = 1.$$

$\underline{\mathsf{Sim}(\mathsf{gpk}, \mathsf{vsk}_j, \mathsf{m}, \textsc{adm}, \textsc{mod}, \underline{\sigma})}$: If $\textsc{mod} \preceq \textsc{adm} \wedge \textsc{adm} \preceq \mathsf{m}$, parse $\underline{\sigma}$ as $(\mathsf{pk}, \sigma_\mathsf{G})$, run $\mathring{\mathsf{m}} \xleftarrow{\textsc{mod}} \mathsf{m}$, and return $(\mathring{\mathsf{m}}, \rho)$, where

$$\begin{aligned}
&\sigma_\mathsf{V} \leftarrow \Sigma.\mathsf{Sign}(\mathsf{vsk}_j, \mathring{\mathsf{m}}),\\
&\pi \leftarrow \Pi.\mathsf{Proof}(\mathsf{crs}, (\mathring{\mathsf{m}}, \mathsf{pk}, \mathsf{vpk}_j), (\bot, \sigma_\mathsf{V})), \text{ and}\\
&\rho \leftarrow (\underline{\sigma}, \pi).
\end{aligned}$$

Otherwise, return $\bot$.

Scheme 5.4: Black-box AD-RS.

Lemma 5.10 straight-forwardly follows from inspection; the proof is omitted.

Lemma 5.11. *If* GS *is traceable and* RS *is unforgeable, then Scheme 5.4 is group unforgeable.*

Proof. We construct efficient reductions $\mathcal{R}^\mathsf{t}$ and $\mathcal{R}^\mathsf{u}$ turning an efficient group unforgeability adversary $\mathcal{A}^{\mathsf{gu}}$, into an efficient adversary (1) $\mathcal{A}^\mathsf{t}$ against traceability of GS, or (2) $\mathcal{A}^\mathsf{u}$ against unforgeability of the RS.

(1) $\mathcal{R}^{\mathsf{t}}$ obtains $(\mathsf{gpk}, \mathsf{gok})$ from a GS traceability challenger $\mathcal{C}^{\mathsf{t}}_{\kappa}$, completes the setup as in the real game, and starts $\mathcal{A}^{\mathsf{gu}}$ on $(\mathsf{gpk}', \mathsf{gok})$. Sig queries are simulated by obtaining the group signature σ_{G} using the Sig oracle provided by $\mathcal{C}^{\mathsf{t}}_{\kappa}$ and running the remaining Sign algorithm as in the original protocol. Key queries are simply forwarded to $\mathcal{C}^{\mathsf{t}}_{\kappa}$. Eventually, $\mathcal{A}^{\mathsf{gu}}$ outputs a forgery $(\mathsf{m}^\star, \sigma^\star)$ which is opened to signer index u (recall that $\sigma^\star = ((\mathsf{pk}, \sigma_{\mathsf{G}}), (\sigma_{\mathsf{R}}, \mathsf{RED}))$). If u exists ($u \neq \bot$), and $\mathcal{A}^{\mathsf{gu}}$ either requested gsk_u or a group signature on pk for u (i.e., $u \in \mathcal{Q}^{\mathsf{Key}} \ \vee \ (u, \mathsf{pk}) \in \mathcal{Q}^{\mathsf{Sign}}$), we abort as we are in the other case. Otherwise, we output $(\mathsf{pk}, \sigma_{\mathsf{G}})$ as a valid forgery for traceability of GS.

(2) $\mathcal{R}^{\mathsf{u}}$ runs the setup as in the real game and starts $\mathcal{A}^{\mathsf{gu}}$ on $(\mathsf{gpk}', \mathsf{gok})$. On each Sig query, $\mathcal{R}^{\mathsf{u}}$ engages with an RS unforgeability challenger $\mathcal{C}^{\mathsf{u}}_{\kappa}$, obtains pk and computes the RS signature using the Sign oracle provided by $\mathcal{C}^{\mathsf{u}}_{\kappa}$. The remaining simulation is performed as in the original scheme. Eventually, $\mathcal{A}^{\mathsf{gu}}$ outputs a forgery $(\mathsf{m}^\star, \sigma^\star)$ which is opened to signer index u (recall that $\sigma^\star = ((\mathsf{pk}, \sigma_{\mathsf{G}}), (\sigma_{\mathsf{R}}, \mathsf{RED}))$. If u does not exist ($u = \bot$), or u exists and $\mathcal{A}^{\mathsf{gu}}$ neither requested gsk_u nor a group signature on pk for u (i.e., $u \notin \mathcal{Q}^{\mathsf{Key}} \wedge (u, \mathsf{pk}) \notin \mathcal{Q}^{\mathsf{Sign}}$) we abort as we are in the other case. Otherwise, we know that we have a valid RS signature on $\mathsf{m}^\star$ under pk which is not derivable from any queried message (i.e., $\nexists\, (u, \mathsf{m}, \mathsf{ADM}) \in \mathcal{Q}^{\mathsf{Sig}} : \mathsf{m}^\star \preceq^{\mathsf{ADM}} \mathsf{m}$) and we can output $(\mathsf{m}^\star, \sigma_{\mathsf{R}})$ as an RS forgery.

Overall Bound. The simulations in both reductions are indistinguishable from a real game. In front of an adversary we randomly guess the adversary's strategy, inducing a loss of $1/2$. Our reduction for a Type 1 forger always succeeds if the adversary succeeds, whereas our reduction for Type 2 succeeds with a probability of $1/q$, where $q \leq \mathsf{poly}(\kappa)$ is the number of queries to the Sig oracle. Overall, this means that the probability to break group unforgeability is upper-bounded by $2 \cdot \max\{\varepsilon_{\mathsf{gu}}(\kappa), q \cdot \varepsilon_{\mathsf{p}}(\kappa)\}$. □

Lemma 5.12. *If* Π *admits proofs of knowledge,* Σ *is* EUF-CMA *secure, and Scheme 5.4 is group unforgeable, then Scheme 5.4 is also designated-verifier unforgeable.*

Proof. We bound the probability to break designated-verifier unforgeability. We start by defining our opener $\mathsf{O} = (O_1, O_2)$.

$O_1(1^\kappa, n)$: Run $(\mathsf{gpk}, \mathsf{gok}, \mathsf{gsk}) \leftarrow \mathsf{GS.KeyGen}(1^\kappa, n)$, $(\mathsf{crs}, \tau) \leftarrow \Pi.\mathsf{E}_1(1^\kappa)$, set $\mathsf{gpk}' \leftarrow (\mathsf{gpk}, \mathsf{crs})$, $\tau' \leftarrow (\mathsf{gpk}', \mathsf{gok}, \mathsf{gsk}, \tau)$ and return $(\mathsf{gpk}', \mathsf{gok}, \mathsf{gsk}, \tau')$.

$O_2(\tau, \mathtt{DVK}, \mathsf{m}^\star, \rho^\star)$: Parse σ as $((\mathsf{pk}, \sigma_{\mathsf{G}}), \overline{\sigma})$ and return $u \leftarrow \mathsf{GS.Open}(\mathsf{gok}, \mathsf{pk}, \sigma_{\mathsf{G}})$.

The tuple $(\mathsf{gpk}', \mathsf{gok}, \mathsf{gsk})$ contained in the output of O_1 is computationally indistinguishable from the output of Setup under the extraction-CRS indistinguishability of the proof system.

What remains is to show that—using O—the success probability of every PPT adversary in the designated-verifier unforgeability game is negligible in the security parameter. We do so by using a sequence of games, where we let $q \leq \mathsf{poly}(\kappa)$ be the number of queries to the DVGen oracle.

Game 0: The original designated-verifier-unforgeability game.

Game 1: As Game 0, but we modify O_1 as follows. We use $\mathcal{C}_\kappa^{\mathsf{gu}}$ to denote a group unforgeability challenger.

$O_1(1^\kappa, n)$: Run $\boxed{(\mathsf{gpk}, \mathsf{gok}) \leftarrow \mathcal{C}_\kappa^{\mathsf{gu}}}$, set $\boxed{\mathsf{gsk} \leftarrow \bot}$, $(\mathsf{crs}, \tau) \leftarrow \Pi.\mathsf{E}_1(1^\kappa)$, set $\mathsf{gpk}' \leftarrow (\mathsf{gpk}, \mathsf{crs})$, $\tau' \leftarrow (\mathsf{gpk}', \mathsf{gok}, \mathsf{gsk}, \tau)$ and return $(\mathsf{gpk}', \mathsf{gok}, \mathsf{gsk}, \tau')$.

Then, we simulate all queries to Sig and Key by forwarding them to $\mathcal{C}_\kappa^{\mathsf{gu}}$.

Transition - Game 0 → Game 1: This change is conceptual, i.e., $\Pr[S_0] = \Pr[S_1]$.

Game 2: As Game 1, but we guess the index $v^\star$ that the adversary will attack.

Transition - Game 1 → Game 2: The success probability in Game 2 is the same as in Game 1, unless our guess is wrong. That is $\Pr[S_2] = \Pr[S_1] \cdot 1/q$.

Game 3: As Game 2, but in the query to DVGen for user $v^\star$ we engage with an $\mathsf{EUF\text{-}CMA}$ challenger $\mathcal{C}_\kappa^{\mathsf{f}}$, obtain a public key pk and return $\mathsf{vpk}_{v^\star} \leftarrow \mathsf{pk}$. Furthermore, the queries to Sim for user $v^\star$ are simulated without $\mathsf{vsk}_{v^\star}$ by using the Sign oracle provided by $\mathcal{C}_\kappa^{\mathsf{f}}$.

Transition - Game 2 → Game 3: This change is conceptual, i.e., $\Pr[S_3] = \Pr[S_2]$.

Game 4: As Game 3, but for every output of the adversary, we obtain $(\sigma_{\mathsf{R}}, \sigma_{\mathsf{V}}) \leftarrow \Pi.\mathsf{E}_2(\mathsf{crs}, \tau, (\mathsf{m}^\star, \mathsf{pk}, \mathsf{vpk}_{v^\star}), \pi)$. If the extractor fails, we abort.

Transition - Game 3 → Game 4: The success probability in Game 2 is the same as in Game 1, unless the extractor fails, i.e., $|\Pr[S_3] - \Pr[S_4]| \leq \varepsilon_{\mathsf{ext2}}(\kappa)$.

In Game 4 we have two possibilities if $\mathcal{A}$ outputs a valid forgery.

1. We extract a signature σ_{R} such that $\mathsf{RS.Verify}(\mathsf{pk}, \mathsf{m}^\star, \sigma_{\mathsf{R}}) = 1$. Since, our implementation of O_1 does the same as what is done in Open and we have that $(u = \bot \vee (u \notin \mathcal{Q}^{\mathsf{Key}} \wedge \nexists(u, \mathsf{m}, \mathsf{ADM}) \in \mathcal{Q}^{\mathsf{Sig}} : \mathsf{m}^\star \preceq^{\mathsf{ADM}} \mathsf{m}))$ by definition, we can compose $\sigma \leftarrow ((\mathsf{pk}, \sigma_{\mathsf{G}}), (\sigma_{\mathsf{R}}, \bot))$ and return $(\mathsf{m}^\star, \sigma)$ to $\mathcal{C}_\kappa^{\mathsf{gu}}$ as a forgery for the group unforgeability game.

2. We extract a signature σ_{V} such that $\Sigma.\mathsf{Verify}(\mathsf{vpk}_{v^\star}, \mathsf{m}^\star, \sigma_{\mathsf{V}}) = 1$. By definition, we have that $v^\star \notin \mathcal{Q}^{\mathsf{DVKey}} \wedge \nexists(v^\star, \mathsf{m}, \mathsf{ADM}, \cdot, \cdot) \in \mathcal{Q}^{\mathsf{Sim}} : \mathsf{m}^\star \preceq^{\mathsf{ADM}} \mathsf{m})$. Since $\mathsf{m}^\star \preceq^{\mathsf{ADM}} \mathsf{m}$ also includes the identity, i.e., the case where $\mathsf{m}^\star = \mathsf{m}$, we know that $\mathsf{m}^\star$ was never queried to the signing oracle provided by $\mathcal{C}_\kappa^{\mathsf{f}}$ and we can output $(\mathsf{m}^\star, \sigma_{\mathsf{V}})$ as a valid $\mathsf{EUF\text{-}CMA}$ forgery.

The union bound yields $\Pr[S_4] \leq \varepsilon_{\mathsf{gu}}(\kappa) + \varepsilon_{\mathsf{f}}(\kappa)$. Furthermore, we have that $\Pr[S_4] = \Pr[S_3] \cdot (1 - \varepsilon_{\mathsf{ext2}}(\kappa))$, that $\Pr[S_3] = \Pr[S_2] = \Pr[S_1] \cdot 1/q$, and that $\Pr[S_0] = \Pr[S_1]$. All in all this yields $\Pr[S_0] \leq q \cdot (\varepsilon_{\mathsf{gu}}(\kappa) + \varepsilon_{\mathsf{f}}(\kappa) + \varepsilon_{\mathsf{ext2}}(\kappa))$, which proves the lemma. □

Lemma 5.13. *If* Π *is witness indistinguishable, then Scheme 5.4 is simulatable.*

Proof. We show that the output in the simulatability game is (computationally) independent of the bit b.

Game 0: The original simulatability game ($\underline{\sigma}$ is already independent of b).

Game 1: As Game 0, but we obtain crs for Π upon Setup from a witness indistinguishability challenger $\mathcal{C}_\kappa^{\mathsf{wi}}$ instead of internally generating it.

Transition - Game 0 → Game 1: This change is conceptual, i.e., $\Pr[S_0] = \Pr[S_1]$.

Game 2: As Game 1, but instead of executing Redact inside RoS, we execute the modified algorithm Redact' with additional input vsk_j, which additionally computes $\boxed{\sigma_{\mathsf{v}} \leftarrow \Sigma.\mathsf{Sign}(\mathsf{vsk}_j, \dot{\mathsf{m}})}$ and then computes π as $\pi \leftarrow \Pi.\mathsf{Proof}(\mathsf{crs}, (\dot{\mathsf{m}}, \mathsf{pk}, \mathsf{vpk}_j), \boxed{(\bot, \sigma_{\mathsf{v}})})$.

Transition - Game 1 → Game 2: A distinguisher $\mathcal{D}^{1\to 2}$ is a distinguisher for adaptive witness indistinguishability of the Π, i.e., $|\Pr[S_3] - \Pr[S_2]| \leq \varepsilon_{\mathsf{wi}}(\kappa)$.

In Game 2, Redact' and Sim are identical, i.e., RoS is independent of b. Thus, the adversary has no advantage in winning the game, i.e., $\Pr[S_2] = 1/2$, which yields $\Pr[S_0] \leq 1/2 + \varepsilon_{\mathsf{wi}}(\kappa)$. □

Lemma 5.14. *If* GS *is anonymous and* RS *is unforgeable, then Scheme 5.4 is signer anonymous.*

Proof. We construct an efficient reduction $\mathcal{R}$ which turns an efficient signer-anonymity adversary $\mathcal{A}^{\mathsf{sa}}$ into an efficient adversary $\mathcal{A}$ against anonymity of the underlying GS. $\mathcal{R}$ obtains $(\mathsf{gpk}, \mathsf{gsk})$ from the challenger $\mathcal{C}_\kappa^{\mathsf{a}}$ of the anonymity game of GS and completes the setup as in the original scheme. $\mathcal{R}$ simulates the Open oracle by using the Open oracle provided by $\mathcal{C}_\kappa^{\mathsf{a}}$ and starts $\mathcal{A}^{\mathsf{sa}}$ on $(\mathsf{gpk}', \mathsf{gsk})$. If $\mathcal{A}^{\mathsf{sa}}$ eventually outputs $b^\star$, then $\mathcal{R}$ outputs $b^\star$ to $\mathcal{C}_\kappa^{\mathsf{a}}$. By the RS unforgeability, the simulation of the Open oracle is computationally indistinguishable from a real game. The reduction succeeds with non-negligible probability whenever $\mathcal{A}^{\mathsf{sa}}$ succeeds with non-negligible probability. □

Lemma 5.15. *If* RS *is private, then Scheme 5.4 is private.*

Proof. We prove privacy using a sequence of games, where we let $q \leq \mathsf{poly}(\kappa)$ be the number of queries to the Ch oracle.

Game 0: The privacy game with bit $b = 0$.

Game 1_ℓ ($1 \leq \ell \leq q$): As Game 0, but we set $b = 1$ for the first ℓ queries to Ch.

Transition - Game 0 $\rightarrow$ Game 1_1: A distinguisher between Game 0 and Game 1_1 is a distinguisher for the RS privacy game. To show this, we engage with an RS privacy challenger $\mathcal{C}_\kappa^{\mathsf{p}}$ in the first call to Ch, obtain pk, compute $\sigma_{\mathsf{G}} \leftarrow \mathsf{GS.Sign}(\mathsf{gsk}_i, \mathsf{pk})$, $(\mathring{\mathsf{m}}, \mathring{\sigma}_{\mathsf{R}}) \leftarrow \mathcal{C}_\kappa^{\mathsf{p}}.\mathsf{LoRRedact}((\mathsf{m}_0, \mathsf{MOD}_0, \mathsf{ADM}_0), (\mathsf{m}_1, \mathsf{MOD}_1, \mathsf{ADM}_1))$, as well as $\pi \leftarrow \Pi.\mathsf{Proof}(\mathsf{crs}, (\mathring{\mathsf{m}}, \mathsf{pk}, \mathsf{vpk}_j), (\mathring{\sigma}_{\mathsf{R}}, \perp))$, and return $(\mathring{\mathsf{m}}, \underline{\sigma}, \rho) = (\mathsf{m}, (\mathsf{pk}, \sigma_{\mathsf{G}}), ((\mathsf{pk}, \sigma_{\mathsf{G}}), \pi))$. Depending on the bit chosen by $\mathcal{C}_\kappa^{\mathsf{p}}$, we either simulate Game 0 or Game 1_1.

Transition - Game $1_\ell \rightarrow 1_{\ell+1}$ ($1 \leq \ell < q$) : The answers of the Ch oracle for the first ℓ queries are already simulated for $b = 1$. As above, a distinguisher between Game 1_ℓ and Game $1_{\ell+1}$ is a RS privacy distinguisher.

In Game 1_q we have a simulation for bit $b = 1$. We can bound probability to distinguish the simulations for $b = 0$ and $b = 1$ by $|\Pr[S_{1_q}] - \Pr[S_0]| \leq q \cdot \varepsilon_{\mathsf{p}}(\kappa)$, which shows that the advantage to win the privacy game is bounded by $1/2 + q \cdot \varepsilon_{\mathsf{p}}(\kappa)$. □

5.2.3 Boosting Efficiency via Key-Homomorphisms

In Section 5.1 we have shown that RS can be generically constructed from any $\mathsf{EUF\text{-}CMA}$ secure signature scheme and indistinguishable accumulators (cf. Section 3.1). In our setting it is most reasonable to consider messages as an (ordered) sequence of message blocks. A straight forward solution would thus be to build upon Scheme 5.2, which is tailored to signing ordered sequences of messages $\mathsf{m} = (m_1, \ldots, m_n)$. Unfortunately, this construction aims to conceal the number of message blocks in the original message, and the positions of the redactions. This can be dangerous in our setting, since it might allow to completely change the document semantics. Besides that, it inherently requires a more complex construction.

To this end, we pursue a different direction and require another message representation: we make the position i of the message blocks m_i in the message explicit and represent messages as sets $\mathsf{m} = \{1||m_1, \ldots, n||m_n\}$. Besides solving the aforementioned issues, it also allows us to build upon the (simpler) RS paradigm for sets presented in Scheme 5.1. This paradigm subsumes the essence of many existing RSs. For convenience of the reader we will briefly recall it now: secret keys, public keys, and signatures are split into two parts each. One corresponds to the signature scheme Σ, and one corresponds to the accumulator Λ. Then, Λ is used to encode the message, whereas Σ is used to sign the encoded message. Consequently, we can look at RS key pairs and signatures as being of the form $(\mathsf{sk}, \mathsf{pk}) = ((\mathsf{sk}_\Sigma, \mathsf{sk}_\Lambda, \mathsf{pk}_\Lambda), (\mathsf{pk}_\Sigma, \mathsf{pk}_\Lambda))$ and $\sigma_{\mathsf{R}} = (\sigma_\Sigma, \sigma_\Lambda)$ where the indexes denote their respective types. We emphasize that for accumulators it holds by definition that sk_Λ is an optional trapdoor which may enable more

efficient computations, but all algorithms also run without sk_Λ and the output distribution of the algorithms does not depend on whether the algorithms are executed with or without sk_Λ. We require this property to be able to create designated verifier signatures (cf. Sim) and use $(\mathsf{sk}_\Sigma, \bot, \mathsf{pk}_\Lambda)$ to denote an RS secret key without sk_Λ.

RS following this paradigm only require Σ (besides correctness) to be $\mathsf{EUF\text{-}CMA}$ secure. We observe that additional constraints on Σ—and in particular a Φ^+-key-homomorphic property with adaptability of signatures as defined in Section 3.2—does not influence RS security, while it enables us to design the relation R such that it admits very efficient proofs.

Φ^+-Key-Homomorphic Redactable Signature Schemes. When instantiating our RS construction paradigm (as outlined above) with a Φ^+-key-homomorphic signature scheme, the key homomorphism of the signature scheme straight-forwardly carries over to the RS and we can define Adapt as follows, where we use $\Lambda(\mathsf{m})$ to denote the encoding of message m using Λ.

$\mathsf{Adapt}(\mathsf{pk}, \mathsf{m}, \sigma, \Delta)$: Parse pk as $(\mathsf{pk}_\Sigma, \mathsf{pk}_\Lambda)$ and σ as $(\sigma_\Sigma, \sigma_\Lambda)$, run $(\mathsf{pk}'_\Sigma, \sigma'_\Sigma) \leftarrow \mathsf{Adapt}(\mathsf{pk}_\Sigma, \Lambda(\mathsf{m}), \sigma_\Sigma, \Delta)$ and return $(\mathsf{pk}', \sigma') \leftarrow ((\mathsf{pk}'_\Sigma, \mathsf{pk}_\Lambda), (\sigma'_\Sigma, \sigma_\Lambda))$.

This allows us to concisely present our construction in Scheme 5.5. The **NP** relation, which needs to be satisfied by valid designated-verifier signatures is as follows.

$$((\mathsf{pk}, \mathsf{vpk}_j), (\mathsf{sk}, \mathsf{vsk}_j)) \in R \iff \mathsf{pk} = \mu(\mathsf{sk}) \ \vee \ \Sigma.\mathsf{VKey}(\mathsf{vsk}_j, \mathsf{vpk}_j) = 1.$$

In the discrete logarithm setting such a proof requires an OR-Schnorr proof of two discrete logs, i.e., only requires *two group exponentiations.*

Theorem 5.5. *If* GS *is secure,* RS *is an adaptable* RS *following Scheme 5.1,* Σ *is secure, and* Π *is weakly simulation sound extractable, then Scheme 5.5 is also secure.*

We show that Theorem 5.5 holds by proving Lemma 5.16-5.21. We note that we could also perform this proof under witness indistinguishability instead of weak simulation sound extractability (analogous as in the proof of Theorem 3.6). However, the instantiation of Π we target provides weak simulation sound extractability anyway, which is why we opt for the simpler proof.

Lemma 5.16. *If* GS *is correct,* RS *is correct and adapts signatures,* Σ *is correct, and* Π *is complete, then Scheme 5.5 is also correct.*

Lemma 5.16 straight-forwardly follows from inspection; the proof is omitted.

Lemma 5.17. *If* GS *is traceable and* RS *is unforgeable, then Scheme 5.5 is group unforgeable.*

Lemma 5.17 can be proven identically as group unforgeability is proven in the previous section and is therefore omitted.

$\underline{\mathsf{Redact}(\mathsf{gpk}, \mathsf{vpk}_j, \mathsf{m}, \sigma, \textsc{mod})}$: Parse σ as $((\mathsf{pk}, \sigma_\mathsf{G}), (\sigma_\mathsf{R}, \textsc{red}))$ and return $(\mathring{\mathsf{m}}, \rho)$, where

$$\mathsf{sk}' \xleftarrow{R} \mathbb{H},\ \mathsf{pk}' \leftarrow \mu(\mathsf{sk}'),\ (\mathsf{pk}_\mathsf{R}, \sigma'_\mathsf{R}) \leftarrow \mathsf{Adapt}(\mathsf{pk}, \mathsf{m}, \sigma_\mathsf{R}, \mathsf{sk}'),$$
$$((\mathring{\mathsf{m}}, \mathring{\sigma}'_\mathsf{R}), \cdot) \leftarrow \mathsf{RS.Redact}(\mathsf{pk}_\mathsf{R}, \mathsf{m}, \sigma'_\mathsf{R}, \textsc{mod}, \textsc{red}),$$
$$\pi \leftarrow \Pi.\mathsf{Proof}(\mathsf{crs}, (\mathsf{pk}', \mathsf{vpk}_j), (\mathsf{sk}', \bot)),\ \text{and}\ \rho \leftarrow ((\mathsf{pk}, \sigma_\mathsf{G}), \mathsf{pk}', \mathring{\sigma}'_\mathsf{R}, \pi).$$

$\underline{\mathsf{Verify}(\mathsf{gpk}, \mathsf{vpk}_j, \mathsf{m}, \rho)}$: Parse ρ as $((\mathsf{pk}, \sigma_\mathsf{G}), \mathsf{pk}', \mathring{\sigma}'_\mathsf{R}, \pi)$, let $\mathsf{pk} = (\mathsf{pk}_\Sigma, \mathsf{pk}_\Lambda)$, compute $\mathsf{pk}_\mathsf{R} \leftarrow (\mathsf{pk}_\Sigma \cdot \mathsf{pk}', \mathsf{pk}_\Lambda)$ and return 1 if the following holds, and 0 otherwise:

$$\mathsf{GS.Verify}(\mathsf{gpk}, \mathsf{pk}, \sigma_\mathsf{G}) = 1 \quad \wedge \quad \Pi.\mathsf{Verify}(\mathsf{crs}, (\mathsf{pk}', \mathsf{vpk}_j), \pi) = 1$$
$$\wedge \quad \mathsf{RS.Verify}(\mathsf{pk}_\mathsf{R}, \mathsf{m}, \mathring{\sigma}'_\mathsf{R}) = 1.$$

$\underline{\mathsf{Sim}(\mathsf{gpk}, \mathsf{vsk}_j, \mathsf{m}, \textsc{adm}, \textsc{mod}, \underline{\sigma})}$: If $\textsc{mod} \preceq \textsc{adm} \wedge \textsc{adm} \preceq \mathsf{m}$, parse $\underline{\sigma}$ as $((\mathsf{pk}_\Sigma, \mathsf{pk}_\Lambda), \sigma_\mathsf{G})$ and return $(\mathring{\mathsf{m}}, \rho)$, where

$$\mathsf{sk}^\Sigma_\mathsf{R} \xleftarrow{R} \mathbb{H},\ \mathsf{pk}^\Sigma_\mathsf{R} \leftarrow \mu(\mathsf{sk}^\Sigma_\mathsf{R}),\ \mathsf{pk}' \leftarrow \mathsf{pk}_\Sigma^{-1} \cdot \mathsf{pk}^\Sigma_\mathsf{R},$$
$$((\mathsf{m}, \sigma'_\mathsf{R}), \textsc{red}) \leftarrow \mathsf{RS.Sign}((\mathsf{sk}^\Sigma_\mathsf{R}, \bot, \mathsf{pk}_\Lambda), \mathsf{m}, \textsc{adm}),$$
$$((\mathring{\mathsf{m}}, \mathring{\sigma}'_\mathsf{R}), \cdot) \leftarrow \mathsf{RS.Redact}((\mathsf{pk}^\Sigma_\mathsf{R}, \mathsf{pk}_\Lambda), \mathsf{m}, \sigma'_\mathsf{R}, \textsc{mod}, \textsc{red}),$$
$$\pi \leftarrow \Pi.\mathsf{Proof}(\mathsf{crs}, (\mathsf{pk}', \mathsf{vpk}_j), (\bot, \mathsf{vsk}_j)),\ \text{and}\ \rho \leftarrow (\underline{\sigma}, \mathsf{pk}', \mathring{\sigma}'_\mathsf{R}, \pi).$$

Scheme 5.5: Semi-black-box AD-RS where Setup, DVGen, Sign, GVerify, and Open are as in Scheme 5.4.

Lemma 5.18. *If Π is weakly simulation sound extractable, Σ is* EUF-CMA *secure,* RS *adapts signatures, Scheme 5.5 is group unforgeable, and the DL assumption holds in $\mathbb{G}$, then Scheme 5.5 is also designated-verifier unforgeable.*

Proof. We prove designated-verifier unforgeability using a sequence of games. First, we define the opener $\mathsf{O} = (O_1, O_2)$ as follows.

$O_1(1^\kappa, n)$: Run $(\mathsf{gpk}, \mathsf{gok}, \mathsf{gsk}) \leftarrow \mathsf{GS.KeyGen}(1^\kappa, n)$, $(\mathsf{crs}, \tau) \leftarrow \Pi.\mathsf{S}_1(1^\kappa)$, set $\mathsf{gpk}' \leftarrow (\mathsf{gpk}, \mathsf{crs})$, $\tau' \leftarrow (\mathsf{gpk}, \mathsf{gok}, \mathsf{gsk}, \tau)$, and return $(\mathsf{gpk}', \mathsf{gok}, \mathsf{gsk}, \tau')$.

$O_2(\tau, \textsc{dvk}, \mathsf{m}^\star, \rho^\star)$: Parse σ as $((\mathsf{pk}, \sigma_\mathsf{G}), \overline{\sigma})$ and return $u \leftarrow \mathsf{GS.Open}(\mathsf{gok}, \mathsf{pk}, \sigma_\mathsf{G})$.

The tuple $(\mathsf{gpk}', \mathsf{gok}, \mathsf{gsk})$ contained in the output of O_1 is computationally indistinguishable from the output of Setup under the simulation-CRS indistinguishability of the proof system. From now on we will simulate all proofs, i.e., replace all calls to $\Pi.\mathsf{Proof}(\mathsf{crs}, x, w)$ by $\Pi.\mathsf{S}_2(\mathsf{crs}, \tau, x)$.

What remains is to show that—using O—the success probability of every PPT adversary in the designated-verifier unforgeability game is negligible in the security parameter. We do so by using a sequence of games where we let $q_{\mathsf{Sim}} \leq \mathsf{poly}(\kappa)$ be the number of queries to the Sim oracle and $q \leq \mathsf{poly}(\kappa)$ the number of users in the system.

Game 0: The original designated-verifier unforgeability game.

Game 1: As Game 0, but we modify O_1 as follows, where $\mathcal{C}_\kappa^{\mathsf{gu}}$ denotes a group-unforgeability challenger (note that we can assume that all required accumulator public keys are obtained from collision freeness challengers as all algorithms also run without secret keys without affecting the output distribution of the algorithms):

$$O_1(1^\kappa, n): \text{Run } \boxed{(\mathsf{gpk}, \mathsf{gok}) \leftarrow \mathcal{C}_\kappa^{\mathsf{gu}}}, \text{ set } \boxed{\mathsf{gsk} \leftarrow \bot}, \ (\mathsf{crs}, \tau) \leftarrow \Pi.\mathsf{S}_1(1^\kappa), \\ \mathsf{gpk}' \leftarrow (\mathsf{gpk}, \mathsf{crs}), \tau' \leftarrow (\mathsf{gpk}, \mathsf{gok}, \mathsf{gsk}, \tau), \text{ and return } (\mathsf{gpk}', \mathsf{gok}, \mathsf{gsk}, \tau').$$

Furthermore, whenever the adversary queries Sig or Key, we use the oracles provided by $\mathcal{C}_\kappa^{\mathsf{gu}}$ to obtain the required group signatures and keys, respectively.

Transition - Game 0 → Game 1: This game change is conceptual and $\Pr[S_1] = \Pr[S_0]$.

Game 2: As Game 1, but whenever the adversary outputs a forgery so that $\mathsf{pk}_\Sigma \cdot \mathsf{pk}'$ corresponds to a key $\mathsf{pk}_\mathsf{R}^\Sigma$ used in a Sim oracle call we check whether the signed accumulator value (used to encode the message) is still the same as the one signed in Sim and abort if so.

Transition - Game 1 → Game 2: If we abort, we have a collision for one of the accumulators. That is, $|\Pr[S_1] - \Pr[S_2]| \leq q_{\mathsf{Sim}} \cdot \varepsilon_{\mathsf{cf}}(\kappa)$.

Game 3: As Game 2, but inside Sim we obtain $\mathsf{pk}_\mathsf{R}^\Sigma$ from an $\mathsf{EUF\text{-}CMA}$ challenger of Σ and obtain the required signatures inside $\mathsf{RS.Sign}$ using the Sign oracle provided by the challenger.

Transition - Game 2 → Game 3: This change is conceptual: $\Pr[S_2] = \Pr[S_3]$.[11]

Game 4: As Game 3, but we guess the index $v^\star$ that the adversary will attack. If our guess is wrong, we abort.

Transition - Game 3 → Game 4: The success probability in Game 4 is the same as in Game 3, unless our guess is wrong. That is $\Pr[S_4] = \Pr[S_3] \cdot 1/q$.

Game 5: As Game 4, but in the query to DVGen for user $v^\star$ we engage with an $\mathsf{EUF\text{-}CMA}$ challenger $\mathcal{C}_\kappa^{\mathsf{f}}$, obtain a public key pk and return $\mathsf{vpk}_{v^\star} \leftarrow \mathsf{pk}$.

Transition - Game 4 → Game 5: This change is conceptual, i.e., $\Pr[S_4] = \Pr[S_5]$.

Game 6: As Game 5, but we further modify O_1 as follows, where $\mathcal{C}_\kappa^{\mathsf{gu}}$ denotes a group-unforgeability challenger:

[11] Note that the changes in Game 2 and Game 3 resemble the unforgeability proof strategy of Scheme 5.1. For further details see Section 5.1.

$O_1(1^\kappa, n)$: Run $(\mathsf{gpk}, \mathsf{gok}) \leftarrow \mathcal{C}_\kappa^{\mathsf{gu}}$, set $\mathsf{gsk} \leftarrow \bot$, $\boxed{(\mathsf{crs}, \tau, \xi) \leftarrow \Pi.\mathsf{S}(1^\kappa)}$, $\mathsf{gpk}' \leftarrow (\mathsf{gpk}, \mathsf{crs})$, $\tau' \leftarrow (\mathsf{gpk}, \mathsf{gok}, \mathsf{gsk}, \tau, \boxed{\xi})$, and return $(\mathsf{gpk}', \mathsf{gok}, \mathsf{gsk}, \tau')$.

Transition - Game 5 → Game 6: This change is conceptual, i.e., $\Pr[S_5] = \Pr[S_6]$.

Game 7: As Game 6, but for every output of the adversary, we check whether $\mathsf{pk}_\mathsf{R}^\Sigma$ corresponds to a key obtained from a challenger in Sim and continue if so. Otherwise, we obtain $(\mathsf{sk}', \mathsf{vsk}_{v^\star}) \leftarrow \Pi.\mathsf{E}(\mathsf{crs}, \xi, (\mathsf{pk}', \mathsf{vpk}_{v^\star}), \pi)$. If the extractor fails, we abort.

Transition - Game 6 → Game 7: Both games proceed identically, unless the extractor fails, i.e., $|\Pr[S_6] - \Pr[S_7]| \leq \varepsilon_{\mathsf{ext}}(\kappa)$.

If the adversary outputs a forgery $(\mathsf{m}^\star, \rho^\star, v^\star)$, where $\rho^\star = (((\mathsf{pk}_\Sigma, \mathsf{pk}_\Lambda), \sigma_\mathsf{G}), \mathsf{pk}', \mathring{\sigma}'_\mathsf{R}, \pi)$, we check if $\mathsf{pk}_\Sigma \cdot \mathsf{pk}'$ corresponds to a key $\mathsf{pk}_\mathsf{R}^\Sigma$ used in Sim. Then, we can output the Σ-signature σ on the accumulator together with the accumulator as an $\mathsf{EUF\text{-}CMA}$ forgery to one of the challengers from Sim. Otherwise, we check whether we have extracted $\mathsf{vsk}_{v^\star}$ such that $\Sigma.\mathsf{VKey}(\mathsf{vpk}_{v^\star}, \mathsf{vsk}_{v^\star}) = 1$. If so, we choose a random message m in the message space of Σ, compute $\sigma \leftarrow \Sigma.\mathsf{Sign}(\mathsf{vsk}_{v^\star}, m)$ and output (m, σ) as an $\mathsf{EUF\text{-}CMA}$ forgery for Σ. If not, we must have extracted a secret key sk' such that $\mathsf{RS.VKey}(\mathsf{pk}', \mathsf{sk}') = 1$. Then, we have that $\mathsf{Verify}(\mathsf{gpk}, \mathsf{vpk}_{v^\star}, \mathsf{m}^\star, \rho^\star) = 1$ by definition and can obtain $(\mathsf{pk}, \mathring{\sigma}''_\mathsf{R}) \leftarrow \mathsf{RS.Adapt}(\mathsf{pk}_\Sigma \cdot \mathsf{pk}', \mathsf{m}^\star, \mathring{\sigma}'_\mathsf{R}, -\mathsf{sk}')$ and output $(\mathsf{m}^\star, \sigma) = (\mathsf{m}^\star, (((\mathsf{pk}_\Sigma, \mathsf{pk}_\Lambda), \sigma_\mathsf{G}), (\mathring{\sigma}''_\mathsf{R}, \bot)))$ to break group unforgeability. Note that our implementation of O_2 does the same as what is done in Open and we have that $(u = \bot \ \vee \ (u \notin \mathcal{Q}^{\mathsf{Key}} \ \wedge \ \nexists(u, \mathsf{m}, \mathsf{ADM}) \in \mathcal{Q}^{\mathsf{Sig}} : \mathsf{m}^\star \overset{\mathsf{ADM}}{\preceq} \mathsf{m}))$ by definition. Taking the union bound, the success probability in Game 7 is bounded by $\Pr[S_7] \leq \varepsilon_{\mathsf{gu}}(\kappa) + (1 + q_{\mathsf{Sim}}) \cdot \varepsilon_{\mathsf{f}}(\kappa)$. Thus, we have that $\Pr[S_0] \leq q \cdot (\varepsilon_{\mathsf{gu}}(\kappa) + (1 + q_{\mathsf{Sim}}) \cdot \varepsilon_{\mathsf{f}}(\kappa) + \varepsilon_{\mathsf{ext}}(\kappa)) + q_{\mathsf{Sim}} \cdot \varepsilon_{\mathsf{cf}}(\kappa)$ which concludes the proof. □

Lemma 5.19. *If Π is witness indistinguishable, and RS adapts signatures, then Scheme 5.5 is simulatable.*

Proof. We prove that the output in the simulatability game is (computationally) independent of the bit b.

Game 0: The original simulatability game ($\underline{\sigma}$ is already independent of b).

Game 1: As Game 0, but we obtain crs for the Π upon Setup from a witness indistinguishability challenger $\mathcal{C}_\kappa^{\mathsf{wi}}$ instead of internally generating it.

Transition - Game 0 → Game 1: This change is conceptual, i.e., $\Pr[S_0] = \Pr[S_1]$.

Game 2: As Game 1, but instead of Redact inside RoS we execute the modified algorithm Redact' which runs on additional input vsk_j and computes π as $\pi \leftarrow \Pi.\mathsf{Proof}(\mathsf{crs}, (\mathsf{pk}', \mathsf{vpk}_j), \boxed{(\bot, \mathsf{vsk}_j)})$.

Transition - Game 1 → Game 2: A distinguisher $\mathcal{D}^{1\to 2}$ is a distinguisher for adaptive witness indistinguishability of Π, i.e., $|\Pr[S_2] - \Pr[S_1]| \leq \varepsilon_{\mathsf{wi}}(\kappa)$.

Game 3: As Game 2, but we further modify Redact' to Redact'' so that it additionally takes ADM as input and works as follows.

$\mathsf{Redact}''(\mathsf{gpk}, \mathsf{vpk}_j, \mathsf{m}, \sigma, \mathsf{MOD}, \mathsf{vsk}_j, \mathsf{ADM})$: Parse σ as $((\mathsf{pk}, \sigma_{\mathsf{G}}), (\sigma_{\mathsf{R}}, \mathsf{RED}))$ and return $(\mathring{\mathsf{m}}, \rho)$, where

$\boxed{\mathsf{sk}_{\mathsf{R}}} \xleftarrow{R} \mathbb{H}, \boxed{\mathsf{pk}_{\mathsf{R}}} \leftarrow \mu(\mathsf{sk}_{\mathsf{R}}), \ \boxed{\mathsf{pk}' \leftarrow \mathsf{pk}^{-1} \cdot \mu(\mathsf{sk}_{\mathsf{R}})}$

$\boxed{((\mathsf{m}, \sigma'_{\mathsf{R}}), \mathsf{RED}) \leftarrow \mathsf{RS.Sign}((\mathsf{sk}_{\mathsf{R}}, \bot, \mathsf{pk}_\Lambda), \mathsf{m}, \mathsf{ADM})},$

$((\mathring{\mathsf{m}}, \mathring{\sigma}'_{\mathsf{R}}), \cdot) \leftarrow \mathsf{RS.Redact}(\mathsf{pk}_{\mathsf{R}}, \mathsf{m}, \sigma'_{\mathsf{R}}, \mathsf{MOD}, \mathsf{RED}),$

$\pi \leftarrow \Pi.\mathsf{Proof}(\mathsf{crs}, (\mathsf{pk}', \mathsf{vpk}_j), (\bot, \mathsf{vsk}_j))$, and $\rho \leftarrow (\underline{\sigma}, \mathsf{pk}', \mathring{\sigma}'_{\mathsf{R}}, \pi)$.

Transition - Game 2 → Game 3: Under adaptability of the RS, Game 2 and Game 3 are perfectly indistinguishable, i.e., $\Pr[S_3] = \Pr[S_2]$.

In Game 3, Redact'' and Sim are identical; RoS is thus independent of b. Thus, the adversary has no advantage in winning the game, i.e., $\Pr[S_3] = 1/2$. Further, we have that $\Pr[S_0] = \Pr[S_1] \leq \Pr[S_2] + \varepsilon_{\mathsf{wi}}(\kappa)$, and that $\Pr[S_3] = \Pr[S_2]$, which yields $\Pr[S_0] \leq 1/2 + \varepsilon_{\mathsf{wi}}(\kappa)$. □

Lemma 5.20. *If* GS *is anonymous, then Scheme 5.5 is signer anonymous.*

The proof is identical to the proof of Lemma 5.14 and therefore not restated here.

Lemma 5.21. *If* RS *is private and adapts signatures, then Scheme 5.5 is private.*

Proof. The proof strategy is identical to the privacy proof in the previous section. We however, use the following hybrid to interpolate between the games: We engage with an RS privacy challenger $\mathcal{C}^{\mathsf{p}}_\kappa$ in the $\ell + 1$st call to Ch, obtain pk, compute $\sigma_{\mathsf{G}} \leftarrow \mathsf{GS.Sign}(\mathsf{gsk}_i, \mathsf{pk})$, $(\mathring{\mathsf{m}}, \mathring{\sigma}_{\mathsf{R}}) \leftarrow \mathcal{C}^{\mathsf{p}}_\kappa.\mathsf{LoRRedact}((\mathsf{m}_0, \mathsf{MOD}_0, \mathsf{ADM}_0), (\mathsf{m}_1, \mathsf{MOD}_1, \mathsf{ADM}_1))$, $\mathsf{sk}' \xleftarrow{R} \mathbb{H}$, $\mathsf{pk}' \leftarrow \mu(\mathsf{sk}')$, $(\mathsf{pk}_{\mathsf{R}}, \mathring{\sigma}'_{\mathsf{R}}) \leftarrow \mathsf{RS.Adapt}(\mathsf{pk}, \mathring{\mathsf{m}}, \mathring{\sigma}_{\mathsf{R}}, \mathsf{sk}')$, as well as $\pi \leftarrow \Pi.\mathsf{Proof}(\mathsf{crs}, (\mathsf{pk}', \mathsf{vpk}_j), (\mathsf{sk}', \bot))$, and return $(\mathring{\mathsf{m}}, \underline{\sigma}, \rho) = (\mathring{\mathsf{m}}, (\mathsf{pk}, \sigma_{\mathsf{G}}), ((\mathsf{pk}, \sigma_{\mathsf{G}}), \mathsf{pk}', \mathring{\sigma}'_{\mathsf{R}}, \pi))$ Then, depending on the bit chosen by $\mathcal{C}^{\mathsf{p}}_\kappa$, we either simulate Game 0 or Game 1_1 (resp. 1_ℓ or Game $1_{\ell+1}$). □

5.2.4 Performance Overview

In this section we evaluate the practical efficiency of Scheme 5.5. We first assess the practicality of the underlying components and then analyze the overhead imposed by the provably provided strong security guarantees.

Group Signatures. It is well known that there exist multiple practically efficient group signature schemes for non-constrained devices such as standard PCs or even more powerful machines in the cloud. Yet, to adequately protect the doctor's group signing key—which is the only key that persists over multiple signing operations—it might make sense to compute the doctor's group signature σ_{G} on the one-time RS public key pk upon Sign on some dedicated signature token such as a smart card or smart phone. Using the estimations in Section 4.2, such a signature can be computed in < 1s on an ARM Cortex-M0+, a processor that is small enough to be employed in smart cards. While this is already acceptable, the performance on smart phones will even be significantly better. For instance, [CDDT12] report execution times of approximately 150ms for the computation of a group signature with the well-known BBS [BBS04] scheme on a by now rather outdated smart phone.

Key-Homomorphic Redactable Signatures. We first note that the RS keys are freshly generated and the secret keys can be deleted after each signing operation. The respective operations can therefore be directly executed on the doctor's PC, potentially even in parallel to the computation of the group signature. Since we are not aware of any performance evaluation of RS on standard PCs, we implemented one possible instantiation of Scheme 5.1. In our RS implementation we use Schnorr signatures and the indistinguishable t-SDH accumulator presented in Scheme 3.1. In Table 5.1, we present our performance results on an Intel Core i7-4790 @ 3.60GHz with 8GB of RAM, running Java 1.8.0_91 on top of Ubuntu 16.04. Each value represents the mean of 100 consecutive executions. These results confirm that the required RS paradigm is perfectly suited for our application.

Sign	Verify	Redact	Verify (after Redact)
73.1ms	886.3ms	0.1ms	450.3ms

Table 5.1: RS timings in milliseconds, with a number of $n = 100$ message blocks, 50% admissibly redactable blocks and 25% of the blocks being redacted upon Redact.

Additional Computations. Using Schnorr signatures, one only needs two group exponentiations for the proof of knowledge[12]; the adaption of the signature only requires a $\mathbb{Z}_p$ operation, which, compared to the group exponentiations, can be neglected. All in all, the additional computations can thus be ignored compared to those of GS and RS[13], even on very constrained devices such as [UW14].

[12] The results of [FKMV12] confirm that one can use Fiat-Shamir transformed Σ-protocols in the discrete log setting as simulation sound extractable (and therefore weakly simulation sound extractable) proof system when including the statement x upon computing the hash for the challenge.

[13] This is underpinned by the results in Table 5.1, where $\mathcal{O}(n)$ exponentiations happen.

Signature Size. Regarding signature size, the dominant part is the size of the RS public key and signature, respectively, which is in turn determined by the choice of the accumulator. In particular, when instantiating the RS with an accumulator having constant key size and supporting batch verification, one can even obtain constant size signatures. We refer the reader to Section 5.1 for a discussion on RS signature sizes and to Section 3.1 for an overview of suitable accumulators.

5.3 Revisiting Extended Sanitizable Signatures

Sanitizable signatures were initially introduced in [ACdMT05] and later formalized in [BFF+09]. They allow to sign messages, where some dedicated party (the *sanitizer*) may later change (*sanitize*) certain predefined parts of the signed message without invalidating the signature and without signer interaction.

To realize a controlled and limited sanitization of digitally signed content without signer-interaction, various approaches to so called sanitizable signatures have been introduced and refined over the years. Today, there are essentially two flavors of sanitizable signatures. The first one focuses on removal (blacking-out) of designated parts not necessarily conducted by a designated party (could be everyone) and it covers redactable signatures as discussed in the previous sections and the sanitizable signatures in [MIM+05]. The second one focuses on replacement of designated parts conducted only by a designated party (the sanitizer) and covers sanitizable signatures as defined in [ACdMT05] and follow up work [BFF+09, BFLS09, BFLS10, BPS12, BPS13, PSP11]. For a separation of these flavors we refer the reader to [dMPPS14b].

In addition to the motivating examples in the beginning, sanitizable signatures have shown to be a useful tool in various scenarios. Their applications include customizing authenticated multicast transmissions, database outsourcing (combating software piracy and unauthorized content distribution), remote integrity checking of outsourced data [CX08] and secure routing [ACdMT05]. Moreover, they find applications in the context of public sector (open government) data [SKZ13], DRM licensing for digital content protection [CLM08, YSL10], privacy protection in smart grids [PK14], privacy-aware management of audit-log data [HHH+08], health record disclosure [BBM09] and anonymization [SR10], as well as identity management [SSZ14, ZS13]. On the more theoretical side, it has been shown how to build attribute-based anonymous credential systems from sanitizable signatures in a black-box fashion [CL13].

In this paper, we focus on sanitizable signatures in the vein of Ateniese et al. [ACdMT05]. The basic idea behind such a scheme is that a message is split into fixed and modifiable (admissible) blocks, where each admissible block is replaced by a chameleon hash (a trapdoor collision resistant hash) of this block, and the concatenation of all blocks is then signed. A sanitizer being in possession of the trapdoor, can then change each admissible block arbitrarily by computing collisions. Such a sanitizable signature scheme needs to satisfy (1) *unforgeability*, which says that no one except the honest signer and sanitizer can create valid

signatures and sanitizations respectively, (2) *immutability*, which says that a malicious sanitizer must not be able to modify any part of the message which has not been specified as admissible by the signer, (3) *privacy*, which says that all sanitized information is unrecoverable for anyone except signer and sanitizer, (4) *transparency*, which says that signatures created by the signer or the sanitizer are indistinguishable, and (5) *accountability*, which requires that a malicious signer or sanitizer is not able to deny authorship. These security properties have later been rigorously defined in [BFF+09], where it is also shown that accountability implies unforgeability, transparency implies privacy[14] and all other properties are independent. Later, the property of (strong) unlinkability [BFLS10, BPS13] as an even stronger privacy property has been introduced. Additionally, other properties such as (blockwise) non-interactive public accountability [BPS12] have been proposed and the model has also been extended to cover several signers and sanitizers simultaneously [CJL12].

Motivation for this Work. With sanitizable signatures, admissible blocks can be replaced *arbitrarily* by the sanitizer. However, this often makes sanitizers *too powerful* and thus may limit their applicability in various scenarios severely. To reduce the sanitizers' power, Klonowski and Lauks [KL06] introduced several extensions for sanitizable signatures, which allow to limit the power of a sanitizer in several ways and thus eliminate the aforementioned concerns. In particular, they have introduced extensions (1) limiting the set of possible modifications for an admissible block (`LimitSet`), (2) forcing the sanitizer to make the same changes in logically linked admissible blocks (`EnforceModif`), (3) limiting the sanitizer to modify at most k out of n admissible blocks (`LimitNbModif`) and (4) forcing the sanitizer to construct less than ℓ versions of a message (`LimitNbSanit`). Later, Canard and Jambert [CJ10] extended the security model of Brzuska et al. [BFF+09] to cover the aforementioned extensions (as [KL06] did not provide any model or proofs).

The `LimitSet` *Extension.* Although all of the aforementioned features improve the applicability of sanitizable signatures, we deem the `LimitSet` extension to be the generally most useful one (besides, it is the only extension that is related to the privacy property). Thus, in the remainder of this paper, we only consider the `LimitSet` extension and refer to schemes that implement this extension as *extended sanitizable signature schemes* (ESS). In existing constructions, `LimitSet` is realized by using cryptographic accumulators, a primitive that allows to succinctly represent a set (as a so called accumulator) and to compute witnesses certifying membership for elements in the set. Basically, the set of admissible changes for such a block is accumulated and the admissible block is replaced by the respective accumulator. Loosely speaking, the signer initially provides an element together with the witness and sanitizing simply requires the sanitizer to exchange this element and the witness.

[14] We note that the implication of privacy by transparency [BFLS10] only holds in the proof-restricted case (cf. Section 5.3.1).

How to define Privacy? Recall that for sanitizable signatures without extensions, privacy means that it should not be possible to recover the original message from a sanitized version. Now, what is the most reasonable definition for privacy given the `LimitSet` extension? It seems to be most natural to require that, given a (sanitized) signature, a `LimitSet` block does not leak any information about the remaining elements in the respective set (and thus no information about the original message). By carefully inspecting the security model for ESS in [CJ10], we, however, observe that their privacy definition *does not* capture this. In fact, an ESS that reveals *all* elements of the sets corresponding to `LimitSet` blocks *will be private* in their model. One motivation for a weak definition of privacy in [CJ10] might have been to preserve the implication from (proof-restricted) transparency (as in the original model from [BFF+09]). However, as it totally neglects any privacy guarantees for the `LimitSet` extension, a stronger privacy notion seems advantageous and often even required. In [BFLS10, BPS13] a stronger notion of privacy for sanitizable signatures—called (strong) unlinkability—has been introduced. This notion, when adapted to ESS, indeed guarantees what we want to achieve. Yet, unlinkability induces a significant overhead for constructions supporting the `LimitSet` extension. As we will see later, the only unlinkable construction that supports the `LimitSet` extension [CL13] is rather inefficient and is only proven secure in a customized model which *does not* consider all security requirements of sanitizable signatures and thus does not represent an ESS. In general, as we will discuss later, efficient unlinkable constructions of ESS seem hard to achieve. Taking all together we conclude that, while the notion of privacy in [CJ10] seems to be too weak, unlinkability seems to be too strong. In the following, we motivate why a stronger privacy notion (inbetween these two notions) that still allows to obtain efficient instantiations is however important for practical applications.

Motivating Applications. We consider use cases where it is required to limit the sanitizers abilities, while at the same time providing privacy with respect to verifiers. For instance, consider authenticity preserving workflows that span multiple enterprises. Using ESS they can be modeled as illustrated in Figure 5.1, with a signer and a sanitizer per enterprise. Then, employees can—within some well defined boundaries—act (in the role of the sanitizer) on behalf of their company, while also being accountable for their actions. However, companies do not disclose sensitive business internals. As a concrete example for such a

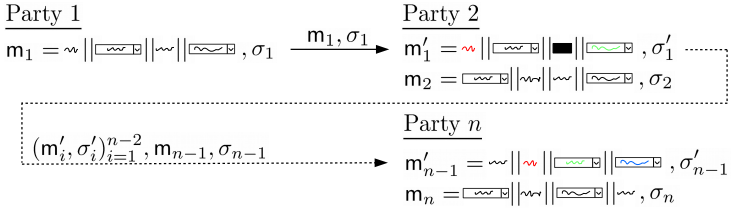

Figure 5.1: Modeling a workflow using ESS.

workflow, envision that a bank signs a document containing a `LimitSet` block with authorized financial transactions for some company once every day. An employee of this company is then able to demonstrate the authorization of single transactions to subsequent enterprises via sanitization, while not being able to maliciously introduce new transactions. The company will definitely want that employees can be held accountable for revealing certain transactions and that transactions which were never revealed by sanitized versions of the orignal document remain concealed. Observe, that an ESS being private according to [CJ10] could reveal sensitive business internals upon signature verification (i.e., the unused transaction information). Another use case is the anonymization of (medical) data before publishing it, e.g., instead of removing the entire address information of some individual, one can replace the precise address with some larger region. To do so, one could define an admissible set with two elements being the precise address and the region. This would greatly help to automate the sanitization and to reduce errors, which, in turn, improves the quality of sanitized documents.[15] Likewise to the previous example, an ESS which is private according to the definition in [CJ10] would allow to reconstruct the precise address from a sanitized document.

Contribution. We take a closer look at the privacy definition for ESS in [CJ10] as well as the unlinkability definitions in [BFLS10, BPS13] when applied to the security model for ESS. We conclude that these notions are either not strict enough to cover the requirements outlined in the previous section or too strict to obtain practical schemes. To this end, we introduce a stronger notion of privacy—denoted *strong privacy*—which explicitly considers privacy issues related to the `LimitSet` extension. More precisely, our strengthened notion guarantees that the sets of allowed modifications remain concealed, while still allowing efficient instantiations. We show that *privacy* is strictly weaker than *strong privacy* and that *unlinkability* is strictly stronger than *strong privacy*. Most importantly, we show that efficient and secure ESS providing strong privacy can be constructed in a *black-box way* from *any* sanitizable signature scheme that is secure in the models of [BFF+09, GQZ10]. We do so by proposing (1) a generic conversion of sanitizable signatures to ESS which support the `LimitSet` extension and (2) showing that instantiating the `LimitSet` extension in this generic conversion with indistinguishable accumulators (cf. Section 3.1) yields constructions that provide strong privacy.

5.3.1 Formalizing Extended Sanitizable Signatures

In this section, we present a formal model for ESS. Our model can thereby be seen as a rigorous formalization of the model for ESS presented in [CJ10]. Additionally, we include the suggestions from [GQZ10], i.e., additionally consider forgeries where one only tampers with ADM. We stress that, when omitting the extensions regarding `LimitSet` and ADM, it is equivalent to the model of

[15] Such sets could be obtained and standardized by using concepts from k-anonymity [Swe02] or t-plausibility [ACJ+12] with the help of domain expert knowledge.

[BFF+09], which is generally considered as the standard model for sanitizable signature schemes.

Definition 5.11 (Message)**.** *A message* $\mathsf{m} = (\mathsf{m}_i)_{i=1}^n$ *is a sequence of* n *bitstrings (message blocks).*

Henceforth, we use ℓ_i to refer to the (maximum) length of message block m_i and assume an encoding that allows to derive $(\ell_i)_{i=1}^n$ from m.

Definition 5.12 (Admissible Modifications)**.** *Admissible modifications* ADM *with respect to a message* $\mathsf{m} = (\mathsf{m}_i)_{i=1}^n$ *are represented as a sequence* $\mathsf{ADM} = (\mathsf{B}_i)_{i=1}^n$*, with* $\mathsf{B}_i \in \{\mathtt{fix}, \mathtt{var}, \mathtt{lim}\}$.

Here $\mathsf{B}_i = \mathtt{fix}$ indicates that no changes are allowed, $\mathsf{B}_i = \mathtt{var}$ indicates that arbitrary replacements are allowed, and $\mathsf{B}_i = \mathtt{lim}$ indicates that the replacements are limited to a predefined set (`LimitSet`).

Definition 5.13 (Set Limitations)**.** *Set limitations* v *with respect to a message* $\mathsf{m} = (\mathsf{m}_i)_{i=1}^n$ *and admissible modifications* $\mathsf{ADM} = (\mathsf{B}_i)_{i=1}^n$ *are represented by a set* $\mathsf{v} = \{(i, \mathsf{M}_i) : \mathsf{B}_i = \mathtt{lim} \ \wedge \ \mathsf{M}_i \subset \bigcup_{j=0}^{\ell_i} \{0,1\}^j\}$.

We use $\mathring{\mathsf{m}} \overset{(\mathsf{ADM},\, \mathsf{v})}{\preceq} \mathsf{m}$ to denote that m' can be derived from m under ADM and v.

Definition 5.14 (Witnesses)**.** *Witnesses* $\mathsf{WIT} = \{(i, \mathsf{wit}_i)\}_{i=1}^t$*, with* $\mathsf{WIT}_i = \{(m_{i_1}, \mathsf{wit}_{i_1}), \ldots, (m_{i_k}, \mathsf{wit}_{i_k})\}$*, are derived from set limitations* $\mathsf{v} = \{(i, \mathsf{M}_i)\}_{i=1}^t$*, with* $\mathsf{M}_i = \{m_{i_1}, \ldots, m_{i_k}\}$*. Thereby,* wit_{i_j} *attests that its corresponding message block* m_{i_j} *is contained in the set* M_i*.*

With $\nu \xleftarrow{(\mathsf{m},\, \mathsf{ADM})} \mathsf{WIT}$, we denote the extraction of the set of witnesses ν corresponding to a message m from the set WIT.

Definition 5.15 (Modification Instructions)**.** *Modification instructions* MOD*, with respect to a message* $\mathsf{m} = (\mathsf{m}_i)_{i=1}^n$*, admissible modifications* ADM *and set limitations* v *are represented by a set* $\mathsf{MOD} = \{(i, \mathring{\mathsf{m}}_i)\}_{i=1}^t$ *with* $t \leq n$*, where* i *refers to the position of the message block in* m*, and* $\mathring{\mathsf{m}}_i$ *is the new content for message block* m_i*.*

With $\mathsf{MOD} \preceq (\mathsf{ADM}, \mathsf{v})$, we denote that the modification instructions in MOD are compatible with ADM and v. Furthermore, with $(\mathsf{m}_0, \mathsf{MOD}_0, \mathsf{ADM}, \mathsf{v}) \equiv (\mathsf{m}_1, \mathsf{MOD}_1, \mathsf{ADM}, \mathsf{v})$, we denote that after applying the changes in MOD_0 and MOD_1 to m_0 and m_1 respectively, the resulting messages $\mathring{\mathsf{m}}_0$ and $\mathring{\mathsf{m}}_1$ are identical.

The Model. Below we formally describe the security model.

Definition 5.16. *An* ESS *is a tuple of PPT algorithms* $(\mathsf{KeyGen}, \mathsf{KeyGen_s}, \mathsf{Sign}, \mathsf{Sanit}, \mathsf{Verify}, \mathsf{Proof}, \mathsf{Judge})$ *which are defined as follows:*

$\mathsf{KeyGen}(1^\kappa)$: *Takes a security parameter* κ *as input and outputs a key pair* $(\mathsf{sk}, \mathsf{pk})$ *for the signer.*

$\mathsf{KeyGen_s}(1^\kappa)$: *Takes a security parameter* κ *as input and outputs a key pair* $(\mathsf{sk_s}, \mathsf{pk_s})$ *for the sanitizer.*

$\mathsf{Sign}(\mathsf{m}, \mathsf{ADM}, \mathsf{V}, (\mathsf{sk}, \mathsf{pk}), \mathsf{pk_s})$: *Takes a message* m*, admissible modifications* ADM*, set limitations* V*, as well as the key pair* $(\mathsf{sk}, \mathsf{pk})$ *of the signer and the verification key* $\mathsf{pk_s}$ *of the sanitizer as input. It computes the set* WIT *from* V*, obtains* $\nu \xleftarrow{\text{MOD}} \mathsf{WIT}$ *and outputs a signature* $\sigma = (\delta, \nu)$ *together with some auxiliary sanitization information* $\mathsf{SAN} = (\mathsf{aux}, \mathsf{WIT})$.[16] *As in [BFF+09], we assume that* ADM *can be recovered from a signature* σ*.*

$\mathsf{Sanit}((\mathsf{m}, \sigma), \mathsf{MOD}, \mathsf{SAN}, \mathsf{pk}, \mathsf{sk_s})$: *Takes a valid message-signature pair* (m, σ)*, modification instructions* MOD*, auxiliary sanitization information* SAN*, the verification key* pk *of the signer, and the signing key* $\mathsf{sk_s}$ *of the sanitizer as input. It modifies* m *and* σ *according to* MOD *and outputs an updated message-signature pair* $(\mathring{\mathsf{m}}, \mathring{\sigma})$*. We assume that* V *can be reconstructed from* SAN*.*

$\mathsf{Verify}((\mathsf{m}, \sigma), \mathsf{pk}, \mathsf{pk_s})$: *Takes a message-signature pair* (m, σ) *and the public verification keys of the signer* pk *and the sanitizer* $\mathsf{pk_s}$ *as input and returns a bit* b*.*

$\mathsf{Proof}((\mathsf{m}, \sigma), \{(\mathsf{m}_j, \sigma_j)\}_{j=1}^q, (\mathsf{sk}, \mathsf{pk}), \mathsf{pk_s})$: *Takes a message-signature pair* (m, σ)*,* q *message-signature pairs* $\{(\mathsf{m}_j, \sigma_j)\}_{j=1}^q$ *created by the signer, the key pair* $(\mathsf{sk}, \mathsf{pk})$ *of the signer and the public key* $\mathsf{pk_s}$ *of the sanitizer as input and outputs a proof* π*.*

$\mathsf{Judge}((\mathsf{m}, \sigma), \mathsf{pk}, \mathsf{pk_s}, \pi)$: *Takes a message-signature pair* (m, σ)*, the verification keys of the signer* pk *and the sanitizer* $\mathsf{pk_s}$ *and a proof* π *as input and outputs a decision* $d \in \{\mathtt{sig}, \mathtt{san}\}$*.*

Security Properties. An ESS is required to fulfill the following properties.

Definition 5.17 (Correctness)**.** *An* ESS *is correct, if*

$$
\begin{aligned}
&\forall\ \kappa, \forall\ q \leq \mathsf{poly}(\kappa), \forall\ \mathsf{m}, \forall\ \mathsf{ADM}, \forall\ \mathsf{V}, \forall\ \mathsf{MOD} \preceq (\mathsf{ADM}, \mathsf{V}),\\
&\forall\ (\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KeyGen}(1^\kappa),\ \forall\ (\mathsf{sk_s}, \mathsf{pk_s}) \leftarrow \mathsf{KeyGen_s}(1^\kappa),\\
&\forall\ (\sigma, \mathsf{SAN}) \leftarrow \mathsf{Sign}(\mathsf{m}, \mathsf{ADM}, \mathsf{V}, (\mathsf{sk}, \mathsf{pk}), \mathsf{pk_s}),\\
&\forall\ (\mathring{\mathsf{m}}, \mathring{\sigma}) \leftarrow \mathsf{Sanit}((\mathsf{m}, \sigma), \mathsf{MOD}, \mathsf{SAN}, \mathsf{pk}, \mathsf{sk_s}),\\
&\forall\ \{(\mathsf{m}_1, \mathsf{ADM}_1, \mathsf{V}_1), \ldots, (\mathsf{m}_q, \mathsf{ADM}_q, \mathsf{V}_q)\},\\
&\forall\ ((\sigma_j, \cdot) \leftarrow \mathsf{Sign}(\mathsf{m}_j, \mathsf{ADM}_j, \mathsf{V}_j, (\mathsf{sk}, \mathsf{pk}), \mathsf{pk_s}))_{j=1}^q,\\
&\forall\ \pi \leftarrow \mathsf{Proof}((\mathring{\mathsf{m}}, \mathring{\sigma}), \{(\mathsf{m}, \sigma)\} \cup \{(\mathsf{m}_j, \sigma_j)\}_{j=1}^q, (\mathsf{sk}, \mathsf{pk}), \mathsf{pk_s})\ :\\
&\mathsf{Verify}((\mathsf{m}, \sigma), \mathsf{pk}, \mathsf{pk_s}) = 1\ \wedge\ \mathsf{Verify}((\mathring{\mathsf{m}}, \mathring{\sigma}), \mathsf{pk}, \mathsf{pk_s}) = 1\ \wedge\\
&\mathsf{Judge}((\mathring{\mathsf{m}}, \mathring{\sigma}), \mathsf{pk}, \mathsf{pk_s}, \pi) = \mathtt{san},
\end{aligned}
$$

where we omit to make the domains of the values over which we quantify explicit for brevity.

[16] While SAN is not required for plain sanitizable signature schemes, ESS additionally return SAN to pass auxiliary information, which is only relevant for the sanitizer.

Definition 5.18 (Unforgeability)**.** *An* ESS *is unforgeable, if for all PPT adversaries* $\mathcal{A}$ *there is a negligible function* $\varepsilon(\cdot)$ *such that:*

$$\Pr\left[\begin{array}{l}(\mathsf{sk},\mathsf{pk}) \leftarrow \mathsf{KeyGen}(1^\kappa),\\ (\mathsf{sk}_\mathsf{s},\mathsf{pk}_\mathsf{s}) \leftarrow \mathsf{KeyGen}_\mathsf{s}(1^\kappa),\\ \mathcal{O} \leftarrow \{\mathsf{Sig}(\cdot,\cdot,\cdot,(\mathsf{sk},\mathsf{pk}),\cdot),\\ \mathsf{San}(\cdot,\cdot,\cdot,\cdot,\mathsf{sk}_\mathsf{s}),\\ \mathsf{Prf}(\cdot,\cdot,(\mathsf{sk},\mathsf{pk}),\cdot)\},\\ (\mathsf{m}^\star,\sigma^\star) \leftarrow \mathcal{A}^{\mathcal{O}}(\mathsf{pk},\mathsf{pk}_\mathsf{s})\end{array} : \begin{array}{r}\mathsf{Verify}(\mathsf{m}^\star,\sigma^\star,\mathsf{pk},\mathsf{pk}_\mathsf{s}) = 1 \wedge\\ (\mathsf{m}^\star,\mathsf{ADM}^\star,\cdot,\mathsf{pk}_\mathsf{s}) \notin \mathcal{Q}^{\mathsf{Sig}} \wedge\\ ((\mathsf{m}^\star,\cdot),\mathsf{ADM}^\star,\mathsf{pk}) \notin \mathcal{Q}^{\mathsf{San}}\end{array}\right] \leq \varepsilon(\kappa),$$

where Sig*,* San *and* Prf *represent the oracles for the* Sign*,* Sanit *and* Proof *algorithms, respectively. The environment keeps track of the queries to* Sig *using* $\mathcal{Q}^{\mathsf{Sig}}$*. Furthermore, it maintains a list* $\mathcal{Q}^{\mathsf{San}}$ *containing the answers of* San *extended with* ADM *and* pk *from the respective oracle query. Note that* $\mathsf{ADM}^\star$ *can be recovered from* $\sigma^\star$*.*

Definition 5.19 (Immutability)**.** *An* ESS *is immutable, if for all PPT adversaries* $\mathcal{A}$ *there is a negligible function* $\varepsilon(\cdot)$ *such that:*

$$\Pr\left[\begin{array}{l}(\mathsf{sk},\mathsf{pk}) \leftarrow \mathsf{KeyGen}(1^\kappa),\\ \mathcal{O} \leftarrow \{\mathsf{Sig}(\cdot,\cdot,\cdot,(\mathsf{sk},\mathsf{pk}),\cdot),\\ \mathsf{Prf}(\cdot,\cdot,(\mathsf{sk},\mathsf{pk}),\cdot)\},\\ (\mathsf{pk}_\mathsf{s}^\star,\mathsf{m}^\star,\sigma^\star) \leftarrow \mathcal{A}^{\mathcal{O}}(\mathsf{pk})\end{array} : \begin{array}{r}\mathsf{Verify}(\mathsf{m}^\star,\sigma^\star,\mathsf{pk},\mathsf{pk}_\mathsf{s}^\star) = 1 \ \wedge\ (\\ (\cdot,\cdot,\cdot,\mathsf{pk}_\mathsf{s}^\star) \notin \mathcal{Q}^{\mathsf{Sig}} \ \vee\ \nexists\, \mathsf{m}^{\star\,(\mathsf{ADM}^\star,\mathsf{V}^\star)}\preceq \mathsf{m} :\\ (\mathsf{m},\mathsf{ADM}^\star,\mathsf{V}^\star,\mathsf{pk}_\mathsf{s}^\star) \in \mathcal{Q}^{\mathsf{Sig}})\end{array}\right] \leq \varepsilon(\kappa),$$

where the oracles and the environment variables are as in Definition 5.18.

Definition 5.20 (Privacy)**.** *An* ESS *is private, if for all PPT adversaries* $\mathcal{A}$ *there is a negligible function* $\varepsilon(\cdot)$ *such that:*

$$\Pr\left[\begin{array}{l}(\mathsf{sk},\mathsf{pk}) \leftarrow \mathsf{KeyGen}(1^\kappa),\ (\mathsf{sk}_\mathsf{s},\mathsf{pk}_\mathsf{s}) \leftarrow \mathsf{KeyGen}_\mathsf{s}(1^\kappa),\\ b \xleftarrow{R} \{0,1\},\ \mathcal{O} \leftarrow \{\mathsf{Sig}(\cdot,\cdot,\cdot,(\mathsf{sk},\mathsf{pk}),\cdot),\\ \mathsf{San}(\cdot,\cdot,\cdot,\cdot,\mathsf{sk}_\mathsf{s}),\ \mathsf{Prf}(\cdot,\cdot,(\mathsf{sk},\mathsf{pk}),\cdot),\\ \mathsf{LoR}(\cdot,\cdot,\cdot,(\mathsf{sk},\mathsf{pk}),(\mathsf{sk}_\mathsf{s},\mathsf{pk}_\mathsf{s}),b)\},\\ b^\star \leftarrow \mathcal{A}^{\mathcal{O}}(\mathsf{pk},\mathsf{pk}_\mathsf{s})\end{array} : b = b^\star\right] \leq 1/2 + \varepsilon(\kappa),$$

where Sig*,* San *and* Prf *are as in Definition 5.18.* LoR *is defined as follows:*

$\mathsf{LoR}((\mathsf{m}_0,\mathsf{MOD}_0),(\mathsf{m}_1,\mathsf{MOD}_1),\mathsf{ADM},(\mathsf{sk},\mathsf{pk}),(\mathsf{sk}_\mathsf{s},\mathsf{pk}_\mathsf{s}),b)$:
1: *Randomly choose* V *(compatible with* MOD_0 *and* MOD_1*).*
2: *If* $\mathsf{MOD}_0 \not\preceq (\mathsf{ADM},\mathsf{V}) \ \vee\ \mathsf{MOD}_1 \not\preceq (\mathsf{ADM},\mathsf{V})$*, return* $\perp$*.*
3: *If* $(\mathsf{m}_0,\mathsf{MOD}_0,\mathsf{ADM},\mathsf{V}) \not\equiv (\mathsf{m}_1,\mathsf{MOD}_1,\mathsf{ADM},\mathsf{V})$*, return* $\perp$*.*
4: *Compute* $(\sigma_b,\mathsf{SAN}_b) \leftarrow \mathsf{Sign}(\mathsf{m}_b,\ \mathsf{ADM},\mathsf{V},(\mathsf{sk},\mathsf{pk}),\mathsf{pk}_\mathsf{s})$*.*
5: *Return* $(\mathring{\mathsf{m}}_b,\mathring{\sigma}_b) \leftarrow \mathsf{Sanit}((\mathsf{m}_b,\sigma_b),\ \mathsf{MOD}_b,\mathsf{SAN}_b,\mathsf{pk},\mathsf{sk}_\mathsf{s})$*.*

Observe that since V is internally chosen (and, thus, independent of the bit b) in LoRSan, privacy holds independent of the adversaries capability to reconstruct the set limitations. Clearly, this *contradicts* a definition of privacy in a sense that sanitized signatures do not reveal the original message.

Definition 5.21 (Transparency)**.** *An* ESS *is transparent, if for all PPT adversaries* $\mathcal{A}$ *there is a negligible function* $\varepsilon(\cdot)$ *such that:*

$$\Pr\left[\begin{array}{l}(\mathsf{sk},\mathsf{pk})\leftarrow\mathsf{KeyGen}(1^\kappa),\ (\mathsf{sk}_\mathsf{s},\mathsf{pk}_\mathsf{s})\leftarrow\mathsf{KeyGen}_\mathsf{s}(1^\kappa),\\ b\xleftarrow{R}\{0,1\},\ \mathcal{O}\leftarrow\{\mathsf{Sig}(\cdot,\cdot,\cdot,(\mathsf{sk},\mathsf{pk}),\cdot),\\ \mathsf{San}(\cdot,\cdot,\cdot,\cdot,\mathsf{sk}_\mathsf{s}),\ \mathsf{Prf}(\cdot,\cdot,(\mathsf{sk},\mathsf{pk}),\cdot), \qquad : b=b^\star\\ \mathsf{SoS}(\cdot,\cdot,\cdot,\cdot,(\mathsf{sk},\mathsf{pk}),(\mathsf{sk}_\mathsf{s},\mathsf{pk}_\mathsf{s}),b)\},\\ b^\star\leftarrow\mathcal{A}^{\mathcal{O}}(\mathsf{pk},\mathsf{pk}_\mathsf{s})\end{array}\right]\leq 1/2+\varepsilon(\kappa),$$

where Sig, San *and* Prf *are as in Definition 5.18. In addition,* Prf *does not respond to queries for messages queried to* SoS*.* SoS *is defined as follows:*

$\mathsf{SoS}(\mathsf{m},\mathsf{ADM},\mathsf{V},\mathsf{MOD},(\mathsf{sk},\mathsf{pk}),(\mathsf{sk}_\mathsf{s},\mathsf{pk}_\mathsf{s}),b)$:
1: *If* $\mathsf{MOD}\not\preceq(\mathsf{ADM},\mathsf{V})$*, return* $\perp$.
2: *Compute* $(\sigma,\mathsf{SAN})\leftarrow\mathsf{Sign}(\mathsf{m},\mathsf{ADM},\mathsf{V},(\mathsf{sk},\mathsf{pk}),\mathsf{pk}_\mathsf{s})$.
3: *Compute* $(\hat{\mathsf{m}},\sigma_0)\leftarrow\mathsf{Sanit}((\mathsf{m},\sigma),\mathsf{MOD},\mathsf{SAN},\mathsf{pk},\mathsf{sk}_\mathsf{s})$.
4: *Compute* $(\sigma_1,\mathsf{SAN})\leftarrow\mathsf{Sign}(\hat{\mathsf{m}},\mathsf{ADM},\mathsf{V},(\mathsf{sk},\mathsf{pk}),\mathsf{pk}_\mathsf{s})$.
5: *Return* $(\mathring{\mathsf{m}},\sigma_b)$.

Proof-restricted transparency [BFLS10]: Prf does not answer queries for messages returned by SoS. In the proof for the implication of privacy by transparency [BFF+09], SoS is used to simulate the LoR queries. Thus, note that the implication only holds if the privacy-adversary is restricted to Prf queries for messages which do not originate from LoR. To additionally rule out even stronger adversaries against privacy, i.e., such that privacy also holds after seeing proofs for the messages in question, one would need to prove privacy directly (we will later do so for our generic extension to obtain a more general result).

Definition 5.22 (Sanitizer-Accountability)**.** *An* ESS *is sanitizer-accountable, if for all PPT adversaries* $\mathcal{A}$ *there is a negligible function* $\varepsilon(\cdot)$ *such that:*

$$\Pr\left[\begin{array}{l}(\mathsf{sk},\mathsf{pk})\leftarrow\mathsf{KeyGen}(1^\kappa),\\ \mathcal{O}\leftarrow\{\mathsf{Sig}(\cdot,\cdot,\cdot,(\mathsf{sk},\mathsf{pk}),\cdot),\\ \mathsf{Prf}(\cdot,\cdot,(\mathsf{sk},\mathsf{pk}),\cdot)\},\\ (\mathsf{pk}_\mathsf{s}^\star,\mathsf{m}^\star,\sigma^\star)\leftarrow\mathcal{A}^{\mathcal{O}}(\mathsf{pk}),\\ \pi\leftarrow\mathsf{Proof}((\mathsf{m}^\star,\sigma^\star),\mathtt{SIG},\\ (\mathsf{sk},\mathsf{pk}),\mathsf{pk}_\mathsf{s}^\star)\end{array}\ :\ \begin{array}{r}\mathsf{Verify}(\mathsf{m}^\star,\sigma^\star,\mathsf{pk},\mathsf{pk}_\mathsf{s}^\star)=1\ \wedge\\ (\mathsf{m}^\star,\mathsf{ADM}^\star,\cdot,\mathsf{pk}_\mathsf{s}^\star)\notin\mathcal{Q}^{\mathsf{Sig}}\ \wedge\\ \mathsf{Judge}((\mathsf{m}^\star,\sigma^\star),\mathsf{pk},\mathsf{pk}_\mathsf{s}^\star,\pi)=\mathtt{sig}\end{array}\right]\leq\varepsilon(\kappa),$$

where the oracles are as in Definition 5.18. The environment maintains a list SIG*, containing all message-signature tuples obtained from* Sig*. Note that* $\mathsf{ADM}^\star$ *can be recovered from* $\sigma^\star$*.*

Definition 5.23 (Signer-Accountability)**.** *An* ESS *is signer-accountable, if for all PPT adversaries* $\mathcal{A}$ *there is a negligible function* $\varepsilon(\cdot)$ *such that:*

$$\Pr\left[\begin{array}{l}(\mathsf{sk}_\mathsf{s},\mathsf{pk}_\mathsf{s})\leftarrow\mathsf{KeyGen}_\mathsf{s}(1^\kappa),\\ \mathcal{O}\leftarrow\{\mathsf{San}(\cdot,\cdot,\cdot,\cdot,\mathsf{sk}_\mathsf{s})\},\\ (\mathsf{pk}^\star,\mathsf{m}^\star,\sigma^\star,\pi^\star)\leftarrow\mathcal{A}^{\mathcal{O}}(\mathsf{pk}_\mathsf{s})\end{array}\ :\ \begin{array}{r}\mathsf{Verify}(\mathsf{m}^\star,\sigma^\star,\mathsf{pk}^\star,\mathsf{pk}_\mathsf{s})=1\ \wedge\\ ((\mathsf{m}^\star,\cdot),\mathsf{ADM}^\star,\mathsf{pk}^\star)\notin\mathcal{Q}^{\mathsf{San}}\ \wedge\\ \mathsf{Judge}((\mathsf{m}^\star,\sigma^\star),\mathsf{pk}^\star,\mathsf{pk}_\mathsf{s},\pi^\star)=\mathtt{san}\end{array}\right]\leq\varepsilon(\kappa),$$

where San *as well as* $\mathcal{Q}^{\mathsf{San}}$ *are as in Definition 5.18.*

5.3.2 Rethinking Privacy for ESS

In the following, we consider alternatives to the standard privacy property, i.e., (strong) unlinkability, and finally come up with a notion denoted as strong privacy which captures privacy for ESS in the original sense of sanitizable signatures.

Revisiting Unlinkability. The notion of unlinkability for sanitizable signatures has been introduced in [BFLS10] as a stronger notion of privacy (which implies the usual privacy property). Below we adapt the unlinkability property to the model for ESS.

Definition 5.24 (Unlinkability)**.** *An* ESS *is unlinkable, if for all PPT adversaries* $\mathcal{A}$ *there is a negligible function* $\varepsilon(\cdot)$ *such that:*

$$\Pr\left[\begin{array}{l}(\mathsf{sk},\mathsf{pk}) \leftarrow \mathsf{KeyGen}(1^\kappa), (\mathsf{sk}_\mathsf{s},\mathsf{pk}_\mathsf{s}) \leftarrow \mathsf{KeyGen}_\mathsf{s}(1^\kappa),\\ b \xleftarrow{R} \{0,1\},\ \mathcal{O} \leftarrow \{\mathsf{Sig}(\cdot,\cdot,\cdot,(\mathsf{sk},\mathsf{pk}),\cdot),\\ \mathsf{San}(\cdot,\cdot,\cdot,\cdot,\mathsf{sk}_\mathsf{s}),\ \mathsf{Prf}(\cdot,\cdot,(\mathsf{sk},\mathsf{pk}),\cdot), \quad : \ b = b^\star\\ \mathsf{LoR}(\cdot,\cdot,\cdot,(\mathsf{sk},\mathsf{pk}),(\mathsf{sk}_\mathsf{s},\mathsf{pk}_\mathsf{s}),b)\},\\ b^\star \leftarrow \mathcal{A}^{\mathcal{O}}(\mathsf{pk},\mathsf{pk}_\mathsf{s})\end{array}\right] \leq 1/2 + \varepsilon(\kappa),$$

where Sig, San *and* Prf *are as in Definition 5.18 and* $\mathcal{O}^{\mathsf{LoRSanit}}$ *operates as follows:*

$\mathsf{LoR}((\mathsf{m}_0,\mathsf{MOD}_0,\mathsf{SAN}_0,\sigma_0),(\mathsf{m}_1,\mathsf{MOD}_1,\mathsf{SAN}_1,\sigma_1),\mathsf{ADM},(\mathsf{sk},\mathsf{pk}),(\mathsf{sk}_\mathsf{s},\mathsf{pk}_\mathsf{s}),b)$:
1: *If* $\mathsf{MOD}_0 \not\preceq (\mathsf{ADM},\mathsf{V}_0) \ \vee \ \mathsf{MOD}_1 \not\preceq (\mathsf{ADM},\mathsf{V}_1)$, *return* $\perp$.
2: *If* $(\mathsf{m}_0,\mathsf{MOD}_0,\mathsf{ADM},\mathsf{V}_0) \not\equiv (\mathsf{m}_1,\mathsf{MOD}_1,\mathsf{ADM},\mathsf{V}_1)$, *return* $\perp$.
3: *If for any* $i \in \{0,1\}$, $\mathsf{Verify}((\mathsf{m}_i,\sigma_i),\mathsf{pk},\mathsf{pk}_\mathsf{s}) = 0$, *return* $\perp$
4: *Return* $(\hat{\mathsf{m}}_b,\hat{\sigma}_b) \leftarrow \mathsf{Sanit}((\mathsf{m}_b,\sigma_b),\mathsf{MOD}_b,\mathsf{SAN}_b,\mathsf{pk},\mathsf{sk}_\mathsf{s})$.

Note that V_0 *and* V_1 *can be reconstructed from* SAN_0 *and* SAN_1*, respectively. Furthermore, note that for answers from the oracle* LoR*, the oracle* Sanit *is restricted to queries for modifications which are covered by both set limitations* V_0 *and* V_1*, which were initially submitted to* LoR*.*

In [BPS13], an even stronger notion, i.e., strong unlinkability, has been proposed. It requires that unlinkability must even hold for signers. The corresponding definition (adapted to the model for ESS) is provided below.

Definition 5.25 (Strong Unlinkability)**.** *An* ESS *is strongly unlinkable, if for all PPT adversaries* $\mathcal{A}$ *there is a negligible function* $\varepsilon(\cdot)$ *such that:*

$$\Pr\left[\begin{array}{l}(\mathsf{sk}_\mathsf{s},\mathsf{pk}_\mathsf{s}) \leftarrow \mathsf{KeyGen}_\mathsf{s}(1^\kappa),\ b \xleftarrow{R} \{0,1\},\\ \mathcal{O} \leftarrow \{\mathsf{San}(\cdot,\cdot,\cdot,\cdot,\mathsf{sk}_\mathsf{s}),\ \mathsf{LoR}(\cdot,\cdot,\cdot,\cdot,(\mathsf{sk}_\mathsf{s},\mathsf{pk}_\mathsf{s}),b)\}, \ : \ b = b^\star\\ b^\star \leftarrow \mathcal{A}^{\mathcal{O}}(\mathsf{pk}_\mathsf{s})\end{array}\right] \leq 1/2 + \varepsilon(\kappa),$$

where the oracles are as in Definition 5.24, except that the signer is no longer fixed by the environment and $\mathcal{A}$ *can query the* San *and* LoR *oracle with arbitrary (dishonest) signer public keys.*

While (strong) unlinkability covers privacy for the `LimitSet` extension in the original sense of privacy[17], it seems very hard to construct efficient (strongly) unlinkable schemes that support the `LimitSet` extension. Unfortunately, it is not possible to simply extend existing (strongly) unlinkable constructions [BFLS10, BPS13, FKM+16] by the `LimitSet` extension. To illustrate why, we revisit the design principle of such schemes. Here, upon Sign, the signer issues two signatures. The first signature, σ_{FIX}, only covers the fixed message blocks and the public key of the sanitizer, whereas the second signature, σ_{FULL}, covers the whole message together with the public key of the signer (and the public key of the sanitizer [BPS13]). Upon Sanit, the sanitizer simply issues a new signature σ_{FULL}, whereas the signature σ_{FIX} remains unchanged. Finally, upon Verify, one verifies whether σ_{FIX} is valid under pk and σ_{FULL} is either valid under pk or pk_{s} for a given message m and ADM. Thereby, the signature scheme used for σ_{FIX} is a deterministic signature scheme, while the scheme used for σ_{FULL} can either also be a deterministic signature scheme [BPS13], a group/ring signature scheme [BFLS10], or a signature scheme with rerandomizable keys [FKM+16].

When extending these schemes to also support the `LimitSet` extension, it is clear that the set limitations need to be fixed by the signer and must not be modifiable by the sanitizer. One simple way to realize the `LimitSet` extension would be to additionally include some unambiguous encoding of the limitations V in σ_{FIX} and check whether the message is consistent with the defined limitations upon Verify. Obviously, this extension does not influence unforgeability and immutability and the scheme is still (publicly) accountable. Furthermore also privacy holds, since the set limitations which are included in the challenge tuple in the privacy game are randomly chosen inside LoR. However, unlinkability can not hold for the following reason: When querying the oracle LoR in the unlinkability game, the adversary can choose set limitations V_0 and V_1 such that $\mathsf{MOD}_0 \preceq (\mathsf{ADM}, \mathsf{V}_0)$, $\mathsf{MOD}_1 \preceq (\mathsf{ADM}, \mathsf{V}_1)$ and $(\mathsf{m}_0, \mathsf{MOD}_0, \mathsf{ADM}, \mathsf{V}_0) \equiv (\mathsf{m}_1, \mathsf{MOD}_1, \mathsf{ADM}, \mathsf{V}_1)$, but $\mathsf{V}_0 \neq \mathsf{V}_1$. For the corresponding signatures $\sigma_0 = (\sigma_{\mathsf{FIX}_0}, \sigma_{\mathsf{FULL}_0})$, $\sigma_1 = (\sigma_{\mathsf{FIX}_1}, \sigma_{\mathsf{FULL}_1})$ submitted to the oracle, this means that $\sigma_{\mathsf{FIX}_0} \neq \sigma_{\mathsf{FIX}_1}$ which yields a trivial distinguisher for the unlinkability game.

As an alternative, one may think of separately signing each message contained in the limited sets (using a deterministic signature scheme), where only the signatures corresponding to the chosen messages are revealed. However, to prevent forgeries where message blocks are re-used in other signatures (i.e., mix-and-match like attacks [BFLS09]), it would be required to also include some message-specific identifier in each signature. Again, it is easy to see that this would provide a trivial distinguisher for the (strong) unlinkability game.

Clearly, the requirement that the limited sets are fixed by the signer and cannot be modified later is not only specific to the aforementioned constructions, but is inherent to all constructions of such schemes. To circumvent the aforementioned issues, one could make use of more sophisticated primitives, which, however, come at the cost of significant computational overhead and complexity

[17] Note that the ability to reconstruct the set limitations for $\mathring{\sigma}_b$ obtained via LoR would imply a trivial distinguisher for the unlinkability game.

of the scheme. This is confirmed by the only known unlinkable construction supporting `LimitSet` [CL13]. It is computationally very expensive due to a high number of bilinear map applications and the use of non-interactive zero-knowledge proofs of knowledge in the computationally expensive target group of the bilinear map. Moreover, it is proven secure only in a model which does not consider all security requirements of sanitizable signatures (as it is tailored to their black-box construction of anonymous credentials) and thus does not represent an ESS.

A Strengthened Notion for Privacy. Surprisingly, our requirement that the set limitations remain concealed can be met by a simple extension of the conventional privacy property. We call the extended property *strong privacy*.[18] As we will see, this modification allows to obtain efficient implementations from secure existing ones in a black-box fashion. We modify the privacy game such that the set limitations in LoR can be submitted per message, i.e., LoR takes $(\mathsf{m}_0, \mathsf{MOD}_0, \mathsf{V}_0), (\mathsf{m}_1, \mathsf{MOD}_1, \mathsf{V}_1), \mathsf{ADM}$. This means that V_0 and V_1 can be different and only need an overlap such that after applying MOD_0 and MOD_1 the messages $\mathring{\mathsf{m}}_0$ and $\mathring{\mathsf{m}}_1$ are identical. More formally, the game is defined as follows:

Definition 5.26 (Strong Privacy)**.** *An* ESS *is strongly private, if for all PPT adversaries $\mathcal{A}$ there is a negligible function $\varepsilon(\cdot)$ such that:*

$$\Pr\left[\begin{array}{l} (\mathsf{sk},\mathsf{pk}) \leftarrow \mathsf{KeyGen}(1^\kappa), (\mathsf{sk}_\mathsf{s},\mathsf{pk}_\mathsf{s}) \leftarrow \mathsf{KeyGen}_\mathsf{s}(1^\kappa), \\ b \xleftarrow{R} \{0,1\}, \mathcal{O} \leftarrow \{\mathsf{Sig}(\cdot,\cdot,\cdot,(\mathsf{sk},\mathsf{pk}),\cdot), \\ \mathsf{San}(\cdot,\cdot,\cdot,\cdot,\mathsf{sk}_\mathsf{s}),\ \mathsf{Prf}(\cdot,\cdot,(\mathsf{sk},\mathsf{pk}),\cdot), \qquad : \ b = b^\star \\ \mathsf{LoR}(\cdot,\cdot,\cdot,(\mathsf{sk},\mathsf{pk}),(\mathsf{sk}_\mathsf{s},\mathsf{pk}_\mathsf{s}),b)\}, \\ b^\star \leftarrow \mathcal{A}^{\mathcal{O}}(\mathsf{pk},\mathsf{pk}_\mathsf{s}) \end{array}\right] \leq \frac{1}{2} + \varepsilon(\kappa),$$

where the oracles Sign, Sanit *and* Proof *are defined as in Definition 5.18. The oracle* LoR *is defined as follows:*

$\mathsf{LoR}((\mathsf{m}_0,\mathsf{MOD}_0,\mathsf{V}_0),(\mathsf{m}_1,\mathsf{MOD}_1,\mathsf{V}_1),\mathsf{ADM},(\mathsf{sk},\mathsf{pk}),(\mathsf{sk}_\mathsf{s},\mathsf{pk}_\mathsf{s}),b)$:
1: *If* $\mathsf{MOD}_0 \not\preceq (\mathsf{ADM},\mathsf{V}_0) \ \vee \ \mathsf{MOD}_1 \not\preceq (\mathsf{ADM},\mathsf{V}_1)$, *return* $\perp$.
2: *If* $(\mathsf{m}_0,\mathsf{MOD}_0,\mathsf{ADM},\mathsf{V}_0) \not\equiv (\mathsf{m}_1,\mathsf{MOD}_1,\mathsf{ADM},\mathsf{V}_1)$, *return* $\perp$.
3: *Compute* $(\sigma_b,\mathsf{SAN}_b) \leftarrow \mathsf{Sign}(\mathsf{m}_b,\mathsf{ADM},\mathsf{V}_b,(\mathsf{sk},\mathsf{pk}),\mathsf{pk}_\mathsf{s})$.
4: *Return* $(\mathring{\mathsf{m}}_b,\mathring{\sigma}_b) \leftarrow \mathsf{Sanit}((\mathsf{m}_b,\sigma_b),\mathsf{MOD}_b,\mathsf{SAN}_b,\mathsf{pk},\mathsf{sk}_\mathsf{s})$.

Note that for answers from the oracle LoR*, the oracle* San *is restricted to queries for modifications which are covered by both set limitations V_0 and V_1, which were initially submitted to* LoR*.*

Theorem 5.6. *Privacy is strictly weaker than strong privacy, while (strong) unlinkability is strictly stronger than strong privacy.*

As mentioned in [CJ10], the extension of the model regarding `LimitSet` does not influence the relations of the properties shown in [BFF+09]. That is, *unforgeability* is implied by *accountability*, *(proof-restricted) privacy* is implied by

[18] In [dMPPS14b], a security notion called strong privacy has been introduced for *plain* sanitizable signatures. Our notion of strong privacy is unrelated to their notion and does not conflict with their notion as ours is only meaningful in context of ESS.

(proof-restricted) transparency and *immutability* is still independent of the other properties. What remains for the proof of Theorem 5.6 is to unveil the relations of *strong privacy* to the other privacy related notions. We now prove a number of lemmas to finally obtain the desired result.

Lemma 5.22. *Not every transparent* ESS *is strongly private.*

We prove Lemma 5.22 by counterexample.

Proof. Let us consider an instantiation of Scheme 5.6 with a *correct*, *unforgeable*, *immutable*, *private*, *(proof-restricted) transparent* and *accountable* sanitizable signature scheme. Further, assume that the accumulator scheme is distinguishable. The ability to distinguishing two accumulators implies an adversary against strong privacy. □

This proof also yields the following corollary.

Corollary 5.5. *Not every private* ESS *is strongly private.*

To show that strong privacy is a strictly stronger notion than privacy, we additionally need to show that the following lemma holds.

Lemma 5.23. *Every strongly private* ESS *is also private.*

To prove this, we show that we can construct an efficient adversary $\mathcal{A}^{\mathsf{SP}}$ against strong privacy using an efficient adversary $\mathcal{A}^{\mathsf{P}}$ against privacy.

Proof. $\mathcal{A}^{\mathsf{SP}}$ simply forwards the calls to the oracles $\mathsf{Sig}, \mathsf{San}, \mathsf{Prf}$, whereas the oracle LoR is simulated as follows: Upon every query $(\mathsf{m}_0, \mathsf{MOD}_0)$, $(\mathsf{m}_1, \mathsf{MOD}_1)$, ADM of $\mathcal{A}^{\mathsf{P}}$, $\mathcal{A}^{\mathsf{SP}}$ internally chooses random set limitations V such that $\mathsf{MOD}_0 \preceq (\mathsf{ADM}, \mathsf{V})$, $\mathsf{MOD}_1 \preceq (\mathsf{ADM}, \mathsf{V})$. Then $\mathcal{A}^{\mathsf{SP}}$ forwards the query $(\mathsf{m}_0, \mathsf{MOD}_0, \mathsf{V})$, $(\mathsf{m}_1, \mathsf{MOD}_1, \mathsf{V})$, ADM to its own LoR oracle and returns the result to $\mathcal{A}^{\mathsf{P}}$. Eventually, $\mathcal{A}^{\mathsf{P}}$ outputs a bit b which is forwarded by $\mathcal{A}^{\mathsf{SP}}$. It is easy to see that the winning probability of $\mathcal{A}^{\mathsf{SP}}$ is identical to that of $\mathcal{A}^{\mathsf{P}}$. □

Below, we show that unlinkability is strictly stronger than strong privacy.

Lemma 5.24. *Not every strongly private* ESS *is (strongly) unlinkable.*

We prove Lemma 5.24 by counterexample.

Proof. Let us consider an instantiation of Scheme 5.6 with a *correct*, *unforgeable*, *immutable*, *private*, *(proof-restricted) transparent* and *accountable* sanitizable signature scheme which does *not* fulfill unlinkability. By Theorem 5.8, we can extend it to be strongly private by using an indistinguishable accumulator. □

Lemma 5.25. *Every unlinkable* ESS *is also strongly private.*

To prove Lemma 5.25, we show that we can construct an efficient adversary $\mathcal{A}^{\mathsf{U}}$ against unlinkability using an efficient adversary $\mathcal{A}^{\mathsf{SP}}$ against strong privacy.

Proof. Likewise to the proof of Lemma 5.23, $\mathcal{A}^{\mathsf{U}}$ simply forwards the calls to the oracles $\mathsf{Sig}, \mathsf{San}, \mathsf{Prf}$, whereas the oracle LoR is simulated as follows: Upon every query $(\mathsf{m}_0, \mathsf{MOD}_0, \mathsf{v}_0)$, $(\mathsf{m}_1, \mathsf{MOD}_1, \mathsf{v}_1)$, ADM of $\mathcal{A}^{\mathsf{SP}}$, $\mathcal{A}^{\mathsf{U}}$ obtains $(\sigma_0, \mathsf{SAN}_0) \leftarrow \mathsf{Sig}(\mathsf{m}_0, \mathsf{ADM}, \mathsf{v}_0), (\sigma_1, \mathsf{SAN}_1) \leftarrow \mathcal{O}^{\mathsf{Sign}}(\mathsf{m}_1, \mathsf{ADM}, \mathsf{v}_1)$ using its own Sig oracle. Then $\mathcal{A}^{\mathsf{U}}$ forwards the query $(\mathsf{m}_0, \mathsf{MOD}_0, \mathsf{SAN}_0, \sigma_0)$, $(\mathsf{m}_1, \mathsf{MOD}_1, \mathsf{SAN}_1, \sigma_1)$, ADM to its own $\mathcal{O}^{\mathsf{LoR}}$ oracle and returns the result to $\mathcal{A}^{\mathsf{SP}}$. Eventually, $\mathcal{A}^{\mathsf{SP}}$ outputs a bit b which is forwarded by $\mathcal{A}^{\mathsf{U}}$. It is easy to see that the winning probability of $\mathcal{A}^{\mathsf{U}}$ is identical to that of $\mathcal{A}^{\mathsf{SP}}$. □

Taking all the above results together, Theorem 5.6 follows.

5.3.3 Black-Box Extension of Sanitizable Signatures

Provably secure existing constructions of ESS build up on concrete existing sanitizable signature schemes. As it turns out, we can even obtain a more general result, i.e., we obtain an ESS that only makes black-box use of sanitizable signatures in the model of [BFF+09, GQZ10] and secure accumulators. The so obtained black-box construction of an ESS then fulfills all the security notions of the underlying sanitizable signature scheme.

Before we continue, we recall the general paradigm for instantiating `LimitSet` (cf. [CJ10, KL06]).

Paradigm 5.1. *For each* `LimitSet` *block, use a secure accumulator to accumulate the set of admissible replacements. The respective message blocks are then replaced with the corresponding accumulator value, i.e., the accumulators are included in the same way as fixed message blocks. Conversely, the actually chosen message blocks for each* `LimitSet` *block are included in the same way as variable message blocks (since they change on every sanitization). Finally, the signature is augmented by the witnesses corresponding to the actual message blocks, while the remaining witnesses are only known to the signer and the sanitizer.*

We introduce our generic construction (that follows Paradigm 5.1) in Scheme 5.6, where we use $(\mathbf{KeyGen}, \mathbf{KeyGen_s}, \mathbf{Sign}, \mathbf{Sanit}, \mathbf{Verify}, \mathbf{Proof}, \mathbf{Judge})$ to denote the algorithms of the underlying sanitizable signature scheme. We define two operators ϕ and ψ to manipulate sets $S = \{(k_1, v_1), \ldots, (k_n, v_n)\}$ of key-value pairs. Thereby, we assume the keys $k_1, \ldots, k_n$ to be unique. The operator $\phi(\cdot, \cdot)$ takes a key k and a set S, obtains the tuple (k_i, v_i) with $k = k_i$ from S, and returns v_i. If no such tuple exists, $\perp$ is returned. Similarly, the operator $\psi(\cdot, \cdot, \cdot)$, takes a key k, a value v_i' and a set S and obtains the tuple (k_i, v_i) with $k = k_i$ from S. It returns $(S \setminus \{(k_i, v_i)\}) \cup \{(k_i, v_i')\}$ and $\perp$ if no such tuple exists.

We will prove the security of Scheme 5.6 using similar arguments as in [CJ10], but relying on the abstract model of [BFF+09, GQZ10], instead of specific properties of the used sanitizable signature scheme.

Observations. Now, we discuss some observations related to the instantiation of the `LimitSet` extension using accumulators. As discussed in the previous

$\underline{\mathsf{KeyGen}(1^\kappa)}$: Fix a sanitizable signature scheme $\{\mathbf{KeyGen}, \mathbf{KeyGen_s}, \mathbf{Sign}, \mathbf{Sanit}, \mathbf{Verify}, \mathbf{Proof}, \mathbf{Judge}\}$, run $(\mathbf{sk}, \mathbf{pk}) \leftarrow \mathbf{KeyGen}(1^\kappa)$, fix an accumulator scheme $\mathsf{A} = \{\mathsf{Gen}, \mathsf{Eval}, \mathsf{WitCreate}, \mathsf{Verify}\}$, run $(\mathsf{sk}_\Lambda, \mathsf{pk}_\Lambda) \leftarrow \mathsf{A.Gen}(1^\kappa, \infty)$, and return $(\mathsf{sk}, \mathsf{pk}) \leftarrow ((\mathbf{sk}, \mathsf{sk}_\Lambda), (\mathbf{pk}, \mathsf{pk}_\Lambda))$.

$\underline{\mathsf{KeyGen_s}(1^\kappa)}$: Run $(\mathbf{sk_s}, \mathbf{pk_s}) \leftarrow \mathbf{KeyGen_s}(1^\kappa)$ and return $(\mathsf{sk_s}, \mathsf{pk_s}) \leftarrow (\mathbf{sk_s}, \mathbf{pk_s})$.

$\underline{\mathsf{Sign}(\mathsf{m}, \mathsf{ADM}, \mathsf{v}, (\mathsf{sk}, \mathsf{pk}), \mathsf{pk_s})}$: Parse m as $(\mathsf{m}_i)_{i\in[n]}$, ADM as $(\mathsf{B}_i)_{i\in[n]}$, v as $\{(i, \mathsf{M}_i)\}_{i\in[m]}$ with $m \le n$, set $\nu \leftarrow \emptyset$, $\mathsf{WIT} \leftarrow \emptyset$ and run

for $i = 1 \dots n$ **if** $\mathsf{B}_i = \mathtt{lim}$ **do**:

$\mathsf{M}_i \leftarrow \phi(i, \mathsf{v})$, $(\Lambda_i, \mathsf{aux}_i) \leftarrow \mathsf{A.Eval}((\mathsf{sk}_\Lambda, \mathsf{pk}_\Lambda), \mathsf{M}_i)$, $\mathsf{WIT}_i \leftarrow \emptyset$,

$\forall\, v_j \in \mathsf{M}_i : \mathsf{wit}_{i_j} \leftarrow \mathsf{A.WitCreate}((\mathsf{sk}_\Lambda, \mathsf{pk}_\Lambda), \Lambda_i, \mathsf{aux}_i, v_j)$, $\mathsf{WIT}_i \leftarrow \mathsf{WIT}_i \cup \{(v_j, \mathsf{wit}_{i_j})\}$,

$\nu_i \leftarrow (i, (\phi(\mathsf{m}_i, \mathsf{WIT}_i), \Lambda_i)), \nu \leftarrow \nu \cup \{\nu_i\}, \mathsf{WIT} \leftarrow \mathsf{WIT} \cup \{(i, \mathsf{WIT}_i)\}$,

$\mathsf{B}_i \leftarrow \mathtt{var}$, $\mathsf{m} \leftarrow \mathsf{m}||(\Lambda_i, i)$, $\mathsf{ADM} \leftarrow \mathsf{ADM}||(\mathtt{fix})$.

endfor.

Run $\delta \leftarrow \mathbf{Sign}(\mathsf{m}, \mathsf{ADM}, (\mathbf{sk}, \mathbf{pk}), \mathbf{pk_s})$, and return (σ, SAN), where $\sigma \leftarrow (\delta, \nu)$, $\mathsf{SAN} \leftarrow (\emptyset, \mathsf{WIT})$.

$\underline{\mathsf{Sanit}((\mathsf{m}, \sigma), \mathsf{MOD}, \mathsf{SAN}, \mathsf{pk}, \mathsf{sk_s})}$: Parse (m, σ) as $((\mathsf{m}_i)_{i\in[n]}, (\delta, \nu))$, MOD as $\{(i, \mathring{\mathsf{m}}_i)\}_{i\in I}$ with $I \subseteq [n]$, SAN as $(\emptyset, \mathsf{WIT})$, obtain $\mathsf{ADM} = (\mathsf{B}_i)_{i\in[n]}$ from σ, and run

for $i \in I$ **if** $\mathsf{B}_i = \mathtt{var} \ \wedge\ \phi(i, \mathsf{WIT}) \neq \perp$ **do**:

$\mathsf{WIT}_i \leftarrow \phi(i, \mathsf{WIT})$, $\mathsf{wit} \leftarrow \phi(\mathring{\mathsf{m}}_i, \mathsf{WIT}_i)$, $(\cdot, \Lambda_i) \leftarrow \phi(i, \nu)$, $\mathring{\nu} \leftarrow \psi(i, (\mathsf{wit}, \Lambda_i), \nu)$.

endfor.

Run $\mathring{\delta} \leftarrow \mathbf{Sanit}(\mathit{Ext}(\mathsf{m}, \sigma), \mathsf{MOD}, \mathbf{pk}, \mathbf{sk_s})$ and return $\mathring{\sigma} = (\mathring{\delta}, \mathring{\nu})$.

$\underline{\mathsf{Verify}((\mathsf{m}, \sigma), \mathsf{pk}, \mathsf{pk_s})}$: Parse (m, σ) as $((\mathsf{m}_i)_{i=1}^n, (\delta, \nu))$ and run

for $i = 1 \dots n$ **if** $\mathsf{B}_i = \mathtt{var} \ \wedge\ \phi(i, \nu) = \perp$ **do**:

$(\mathsf{wit}_{i_j}, \Lambda_i) \leftarrow \phi(i, \nu)$, **if**$[\mathsf{A.Verify}(\mathsf{pk}_\Lambda, \Lambda_i, \mathsf{wit}_{i_j}, \mathsf{m}_i) = 0]$ { **return** 0 }.

endfor.

Finally return 1 if $\mathbf{Verify}(\mathit{Ext}(\mathsf{m}, \sigma), \mathbf{pk}, \mathbf{pk_s}) = 1$ holds, and 0 otherwise.

$\underline{\mathsf{Proof}((\mathsf{m}, \sigma), \{(\mathsf{m}_j, \sigma_j)\}_{j=0}^q, (\mathsf{sk}, \mathsf{pk}), \mathsf{pk_s})}$: Return $\mathbf{Proof}(\mathit{Ext}(\mathsf{m}, \sigma), \{\mathit{Ext}(\mathsf{m}_j, \sigma_j)\}_{j=0}^q, (\mathbf{sk}, \mathbf{pk}), \mathbf{pk_s})$.

$\underline{\mathsf{Judge}((\mathsf{m}, \sigma), \mathsf{pk}, \mathsf{pk_s}, \pi)}$: Return $\mathbf{Judge}(\mathit{Ext}(\mathsf{m}, \sigma), \mathbf{pk}, \mathbf{pk_s}, \pi)$.

$\underline{\mathit{Ext}(\mathsf{m}, \sigma)}$: On input $(\mathsf{m}, \sigma) = ((\mathsf{m}_i)_{i=1}^n, \sigma)$,

for $i = 1 \dots n$ **do**:

$(\mathsf{wit}_{i_j}, \Lambda_i) \leftarrow \phi(i, \nu)$, **if** $[(\mathsf{wit}_{i_j}, \Lambda_i) \neq \perp]$ { set $\mathsf{m} \leftarrow \mathsf{m}||(\Lambda_i, i)$ }.

endfor.

Return (m, σ).

Scheme 5.6: Black-box construction of ESS from any sanitizable signature scheme.

section, it seems to be hard to design generic extensions that also preserve unlinkability [BFLS10, BPS13]. Furthermore, the abstract model does not consider the signer as an adversary, which gives some freedom regarding the implemen-

tations of certain algorithms and the choice of the accumulator scheme. As mentioned in Section 3.1, the abstract model of accumulators assumes a trusted setup. It is, however, beneficial that the signer runs the $\mathsf{A.Gen}$ algorithm to be able to perform more efficient updates using the trapdoor.

Theorem 5.7. *When instantiating Scheme 5.6 with a sanitizable signature scheme* SSS *that provides security properties* Ψ *in the model of [BFF+09, GQZ10], a secure accumulator scheme, and an unambiguous encoding of the message blocks, one obtains an* ESS *that provides security properties* Ψ*.*

First, we emphasize that—while our model includes the extensions regarding ADM from [GQZ10]—the proof does not rely on these extensions. This means that our black-box extension is also applicable to schemes in the model of [BFF+09]. Below we prove Theorem 5.7 by proving Lemma 5.26-5.32.

Lemma 5.26. *Instantiating Scheme 5.6 with a correct* SSS *and a correct accumulator* A *yields a correct* ESS*.*

The lemma above straightforwardly follows from inspection and the correctness of the underlying primitives; the proof is omitted.

Lemma 5.27. *Instantiating Scheme 5.6 with an unforgeable* SSS *yields an unforgeable* ESS*.*

Proof. Assume an efficient adversary $\mathcal{A}$ against unforgeability. We show how this adversary can be turned into an efficient adversary $\mathcal{B}$ against unforgeability of the underlying sanitizable signature scheme SSS. To do so, we describe a reduction $\mathcal{R}$ such that $(\mathcal{A}, \mathcal{R})$ form $\mathcal{B}$. Firstly, $\mathcal{R}$ obtains $\mathbf{pk}$ and $\mathbf{pk}_\mathsf{s}$ from the challenger, runs $(\mathsf{sk}_\Lambda, \mathsf{pk}_\Lambda) \leftarrow \mathsf{A.Gen}(1^\kappa)$ and starts $\mathcal{A}$ on $((\mathbf{pk}, \mathsf{pk}_\Lambda), \mathbf{pk}_\mathsf{s})$. $\mathcal{R}$ implements the oracles by computing the accumulator related parts itself and for the rest it uses the oracles of the unforgeability challenger of the SSS. Now, by definition, $\mathcal{A}$ outputs $(\mathsf{m}^\star, \sigma^\star) = (\mathsf{m}^\star, (\delta^\star, \nu^\star))$ such that $\mathsf{Verify}(\mathsf{m}^\star, \sigma^\star, \mathsf{pk}, \mathsf{pk}_\mathsf{s}) = 1 \ \wedge \ (\mathsf{m}^\star, \mathsf{ADM}^\star, \cdot, \mathsf{pk}_\mathsf{s}) \notin \mathcal{Q}^{\mathsf{Sig}} \ \wedge \ ((\mathsf{m}^\star, \cdot), \mathsf{ADM}^\star, \mathsf{pk}) \notin \mathcal{Q}^{\mathsf{San}}$. This means that $\mathcal{R}$ can output $(\mathit{Ext}(\mathsf{m}^\star, \sigma^\star), \delta^\star)$ as a forgery for SSS.[19] Since we assume an unambiguous encoding, the probability to break unforgeability is upper bounded by $\varepsilon_{\mathsf{uf}}(\kappa)$, which concludes the proof. □

Lemma 5.28. *Instantiating Scheme 5.6 with an immutable* SSS *and an indistinguishable accumulator* A *yields an immutable* ESS*.*

Proof. Assume an efficient adversary $\mathcal{A}$ against immutability. We show how this adversary can be turned (1) into an efficient adversary $\mathcal{B}_1$ against immutability of the underlying sanitizable signature SSS scheme, or (2) into an efficient adversary $\mathcal{B}_2$ against the collision freeness of the underlying accumulator. To do so, we describe two reductions $\mathcal{R}_1, \mathcal{R}_2$ such that $\mathcal{B}_i = (\mathcal{A}, \mathcal{R}_i), i \in \{1, 2\}$.

[19] Observe that the chosen message block per `LimitSet` block is also included as variable element, while the accumulators Λ_i together with the positions i of the `LimitSet` blocks in the message are treated as additional fixed elements (cf. *Ext* in Scheme 5.6). Thus, *every* forgery is a forgery for the underlying SSS scheme.

$\underline{\mathcal{R}_1}$: $\mathcal{R}_1$ obtains $\mathbf{pk}$ from the challenger, runs $(\mathsf{sk}_\Lambda, \mathsf{pk}_\Lambda) \leftarrow \mathsf{A.Gen}(1^\kappa)$ and starts $\mathcal{A}$ on $(\mathbf{pk}, \mathsf{pk}_\Lambda)$. $\mathcal{R}_1$ implements the oracles by computing the accumulator related parts itself and for the rest it uses the oracles of the immutability challenger of the SSS. Eventually, $\mathcal{A}$ outputs $(\mathsf{pk}_\mathsf{s}^\star, \mathsf{m}^\star, \sigma^\star) = (\mathbf{pk}_\mathsf{s}^\star, \mathsf{m}^\star, (\delta^\star, \nu^\star))$ such that $\mathsf{Verify}(\mathsf{m}^\star, \sigma^\star, \mathsf{pk}, \mathsf{pk}_\mathsf{s}^\star) = 1 \ \wedge \ ((\cdot,\cdot,\cdot,\mathsf{pk}_\mathsf{s}^\star) \notin \mathcal{Q}^{\mathsf{Sig}} \ \vee \ \nexists \ \mathsf{m}^{\star\,(\mathsf{ADM}^\star, \mathsf{V}^\star)}\!\preceq \mathsf{m} : (\mathsf{m}, \mathsf{ADM}^\star, \mathsf{V}^\star, \mathsf{pk}_\mathsf{s}^\star) \in \mathcal{Q}^{\mathsf{Sig}})$. If $\nexists \ \mathsf{m}^{\star\,(\mathsf{ADM}^\star, \mathsf{V}^\star)}\!\preceq \mathsf{m} : (\mathsf{m}, \mathsf{ADM}^\star, \mathsf{V}^\star, \mathsf{pk}_\mathsf{s}^\star) \in \mathcal{Q}^{\mathsf{Sig}})$ because of a modification of a `LimitSet` element that is not covered by $\mathsf{V}^\star$, then $\mathcal{R}_1$ aborts (since we are in the other case). Otherwise, $\mathcal{R}_1$ can output $(\mathbf{pk}_\mathsf{s}^\star, Ext(\mathsf{m}^\star, \sigma^\star), \delta^\star)$ and wins the immutability game of SSS.

$\underline{\mathcal{R}_2}$: $\mathcal{R}_2$ obtains pk_Λ from the challenger, runs $(\mathbf{sk}, \mathbf{pk}) \leftarrow \mathbf{KeyGen}(1^\kappa)$ and starts $\mathcal{A}$ on $(\mathbf{pk}, \mathsf{pk}_\Lambda)$. $\mathcal{R}_2$ can simulate the oracles by computing the SSS related parts itself and calling the respective oracles for the accumulator related computations. $\mathcal{R}_2$ keeps track of the accumulators, contained sets and (optionally) the used randomizers obtained from the oracles via a list L_Λ. Eventually, $\mathcal{A}$ outputs $(\mathsf{pk}_\mathsf{s}^\star, \mathsf{m}^\star, \sigma^\star) = (\mathbf{pk}_\mathsf{s}^\star, \mathsf{m}^\star, (\delta^\star, \nu^\star))$ such that $\mathsf{Verify}(\mathsf{m}^\star, \sigma^\star, \mathsf{pk}, \mathsf{pk}_\mathsf{s}^\star) = 1 \ \wedge \ ((\cdot,\cdot,\cdot,\mathsf{pk}_\mathsf{s}^\star) \notin \mathcal{Q}^{\mathsf{Sig}} \ \vee \ \nexists \ \mathsf{m}^{\star\,(\mathsf{ADM}^\star, \mathsf{V}^\star)}\!\preceq \mathsf{m} : (\mathsf{m}, \mathsf{ADM}^\star, \mathsf{V}^\star, \mathsf{pk}_\mathsf{s}^\star) \in \mathcal{Q}^{\mathsf{Sig}})$. If $(\cdot,\cdot,\cdot,\mathsf{pk}_\mathsf{s}^\star) \notin \mathcal{Q}^{\mathsf{Sig}}$, $\mathcal{R}_2$ aborts. If $\nexists \ \mathsf{m}^{\star\,(\mathsf{ADM}^\star, \mathsf{V}^\star)}\!\preceq \mathsf{m} : (\mathsf{m}, \mathsf{ADM}^\star, \mathsf{V}^\star, \mathsf{pk}_\mathsf{s}^\star) \in \mathcal{Q}^{\mathsf{Sig}})$ because of a modification of a `LimitSet` element that is not covered by $\mathsf{V}^\star$, we know that there is at least one tuple $(i, (\mathsf{wit}_{i_j}, \Lambda_i)) \in \nu$ such that $\mathsf{A.Verify}(\mathsf{pk}_\Lambda, \Lambda_i, \mathsf{wit}_{i_j}, \mathsf{m}_i) = 1$ but $\mathsf{m}_i \notin \mathsf{M}_i$, where M_i is the set contained in Λ_i. Now, $\mathcal{R}_2$ can look up the set M_i and the corresponding randomizer r_i in L_Λ and return $(\mathsf{wit}_{i_j}, \mathsf{m}_i, \mathsf{M}_i, r_i)$ as a collision for the accumulator. Otherwise, $\mathcal{R}_2$ aborts (since we are in the other case).

Overall Bound. The simulations in both reductions are indistinguishable from a real game. In front of an adversary we randomly guess the adversary's strategy, inducing a loss of $1/2$. Because of the unambiguous encoding, our reductions always succeed if the forger succeeds. This means that the probability to break immutability is upper-bounded by $2 \cdot \max\{\varepsilon_{\mathsf{im}}(\kappa), \varepsilon_{\mathsf{cf}}(\kappa)\}$. □

Lemma 5.29. *Instantiating Scheme 5.6 with a private* SSS *yields a private* ESS.

Proof. Assume an efficient adversary $\mathcal{A}$ against privacy. We show how this adversary can be turned into an efficient adversary $\mathcal{B}$ against privacy of the underlying sanitizable signature scheme SSS. To do so, we describe a reduction $\mathcal{R}$ such that $(\mathcal{A}, \mathcal{R})$ form $\mathcal{B}$. Firstly, $\mathcal{R}$ obtains $\mathbf{pk}$ and $\mathbf{pk}_\mathsf{s}$ from the challenger, runs $(\mathsf{sk}_\Lambda, \mathsf{pk}_\Lambda) \leftarrow \mathsf{A.Gen}(1^\kappa)$ and starts $\mathcal{A}$ on $((\mathbf{pk}, \mathsf{pk}_\Lambda), \mathsf{pk}_\mathsf{s})$. $\mathcal{R}$ implements the oracles by computing the accumulator related parts itself and for the rest it uses the oracles of the privacy challenger of the SSS. Eventually $\mathcal{A}$ outputs a bit $b^\star$, which can be used by $\mathcal{R}$ to win the privacy game of the SSS, where the winning probability is that of $\mathcal{A}$. Now, we argue why this is the case: In the privacy game, the set limitations v are internally chosen in LoR such that they are compatible with both submitted challenge messages. This, means that the `LimitSet` related signature components are independent of the actual bit b, which concludes the proof. □

Lemma 5.30. *Instantiating Scheme 5.6 with a transparent* SSS *yields a transparent* ESS*.*

Proof. Assume an efficient adversary $\mathcal{A}$ against transparency. We show how this adversary can be turned into an efficient adversary $\mathcal{B}$ against transparency of the underlying sanitizable signature scheme SSS. To do so, we describe a reduction $\mathcal{R}$ such that $(\mathcal{A}, \mathcal{R})$ form $\mathcal{B}$. Firstly, $\mathcal{R}$ obtains $\mathbf{pk}$ and $\mathbf{pk}_\mathsf{s}$ from the challenger, runs $(\mathsf{sk}_\Lambda, \mathsf{pk}_\Lambda) \leftarrow \mathsf{A.Gen}(1^\kappa)$ and starts $\mathcal{A}$ on $((\mathbf{pk}, \mathsf{pk}_\Lambda), \mathsf{pk}_\mathsf{s})$. $\mathcal{R}$ implements the oracles by computing the accumulator related parts itself and for the rest uses the oracles of the transparency challenger of the SSS. Eventually, $\mathcal{A}$ outputs a bit $b^\star$ and $\mathcal{R}$ forwards this bit to the transparency challenger of the SSS and wins the game—the winning probability is that of $\mathcal{A}$. To see this, observe that in SoS the same set limitations v are used in both, the signed and the sanitized message. □

Lemma 5.31. *Instantiating Scheme 5.6 with a sanitizer-accountable* SSS *yields a sanitizer-accountable* ESS*.*

Proof. Assume an efficient adversary $\mathcal{A}$ against sanitizer-accountability. We show how this adversary can be turned into an efficient adversary $\mathcal{B}$ against sanitizer-accountability of the underlying sanitizable signature scheme SSS. To do so, we describe a reduction $\mathcal{R}$ such that $(\mathcal{A}, \mathcal{R})$ form $\mathcal{B}$. Firstly, $\mathcal{R}$ obtains $\mathbf{pk}$ from the challenger, runs $(\mathsf{sk}_\Lambda, \mathsf{pk}_\Lambda) \leftarrow \mathsf{A.Gen}(1^\kappa)$ and starts $\mathcal{A}$ on $(\mathbf{pk}, \mathsf{pk}_\Lambda)$. $\mathcal{R}$ implements the oracles by computing the accumulator related parts itself and for the rest uses the oracles of the sanitizer-accountability challenger of the SSS. Now, by definition, $\mathcal{A}$ outputs $(\mathsf{pk}_\mathsf{s}^\star, \mathsf{m}^\star, \sigma^\star) = (\mathbf{pk}_\mathsf{s}^\star, \mathsf{m}^\star, (\delta^\star, \nu^\star))$ such that $\mathsf{Verify}(\mathsf{m}^\star, \sigma^\star, \mathsf{pk}, \mathsf{pk}_\mathsf{s}^\star) = 1 \ \wedge \ (\mathsf{m}^\star, \mathsf{ADM}^\star, \cdot, \mathsf{pk}_\mathsf{s}^\star) \notin \mathcal{Q}^{\mathsf{Sig}} \ \wedge \ \mathsf{Judge}((\mathsf{m}^\star, \sigma^\star), \mathsf{pk}, \mathsf{pk}_\mathsf{s}^\star, \mathsf{Proof}((\mathsf{m}^\star, \sigma^\star), \mathtt{SIG}, (\mathsf{sk}, \mathsf{pk}), \mathsf{pk}_\mathsf{s}^\star))) = \mathtt{sig}$. Consequently, $\mathcal{R}$ can output $(\mathbf{pk}_\mathsf{s}^\star, Ext(\mathsf{m}^\star, \sigma^\star), \delta^\star)$ and wins the sanitizer-accountability game of the SSS (the argumentation why is analogous to the one in the unforgeability proof). □

Lemma 5.32. *Instantiating Scheme 5.6 with a signer-accountable* SSS *yields a signer-accountable* ESS*.*

Proof. Assume an efficient adversary $\mathcal{A}$ against signer-accountability. We show how this adversary can be turned into an efficient adversary $\mathcal{B}$ against signer-accountability of the underlying sanitizable signature scheme SSS. To do so, we describe a reduction $\mathcal{R}$ such that $(\mathcal{A}, \mathcal{R})$ form $\mathcal{B}$. Firstly, $\mathcal{R}$ obtains $\mathbf{pk}_\mathsf{s}$ from the challenger and starts $\mathcal{A}$ on $\mathbf{pk}_\mathsf{s}$. $\mathcal{R}$ implements the oracles by computing the accumulator related parts itself and for the rest uses the oracles of the signer-accountability challenger of the SSS. Eventually, $\mathcal{A}$ outputs $(\mathsf{pk}^\star, \mathsf{m}^\star, \sigma^\star, \pi^\star) = ((\mathbf{pk}^\star, \mathsf{pk}_\Lambda^\star), \mathsf{m}^\star, (\delta^\star, \nu^\star), \pi^\star)$ such that $\mathsf{Verify}(\mathsf{m}^\star, \sigma^\star, \mathsf{pk}^\star, \mathsf{pk}_\mathsf{s}) = 1 \ \wedge \ ((\mathsf{m}^\star, \cdot), \mathsf{ADM}^\star, \mathsf{pk}^\star) \notin \mathcal{Q}^{\mathsf{San}} \ \wedge \ \mathsf{Judge}((\mathsf{m}^\star, \sigma^\star), \mathsf{pk}^\star, \mathsf{pk}_\mathsf{s}, \pi^\star) = \mathtt{san}$. Consequently, $\mathcal{A}$ outputs $(\mathbf{pk}^\star, Ext(\mathsf{m}^\star, \sigma^\star), \delta^\star, \pi^\star)$ and wins the signer-accountability of the SSS (the argumentation why is analogous to the one in the unforgeability proof). □

5.3.4 Achieving Strong Privacy

Now we show how strongly private ESS can be constructed from private sanitizable signature schemes in a black-box fashion. Basically, this can be achieved by applying the conversion in Scheme 5.6 and instantiating `LimitSet` using an accumulator that provides the indistinguishability property.

Theorem 5.8. *Let* ESS *obtained using Scheme 5.6 be private and* (AGen, AEval, AWitCreate, AVerify) *be an indistinguishable accumulator, then* ESS *is strongly private.*

Proof. We prove the theorem above by using a sequence of games. Thereby, we denote the event that the adversary wins Game i by S_i and let k be the overall number of accumulators created within LoR.

Game 0: The original strong privacy game.

Game 1: As in the original game, but we modify the oracle LoR to firstly compute $\mathsf{v} \leftarrow \mathsf{v}_0 \cap \mathsf{v}_1$ and to set $\mathsf{v}_0 \leftarrow \mathsf{v}$, $\mathsf{v}_1 \leftarrow \mathsf{v}$.

Transition Game 0 → Game 1: A distinguisher between Game 0 and Game 1 is a distinguisher for the indistinguishability game of the accumulator, i.e., $|\Pr[S_0] - \Pr[S_1]| \leq k \cdot \varepsilon_{\mathsf{ind}}(\kappa)$.[20]

In Game 1, the signatures are computed with respect to $\mathsf{v}_0 \cap \mathsf{v}_1$ in LoR. This means that the `LimitSet` related values are independent of the bit b (similar as when randomly choosing v). Thus, from the adversary's viewpoint, Game 1 is equivalent to the conventional privacy game, meaning that $\mathsf{Pr}[S_1] \leq \frac{1}{2} + \varepsilon_{\mathsf{priv}}(\kappa)$. In further consequence, we obtain $\mathsf{Pr}[S_0] \leq \frac{1}{2} + \varepsilon_{\mathsf{priv}}(\kappa) + k \cdot \varepsilon_{\mathsf{ind}}(\kappa)$ which concludes the proof. □

We also note that it might be an option to use cfw-indistinguishable accumulators instead of indistinguishable accumulators if the chosen random value x_r can not be efficiently guessed. This would resemble the suggestion of [KL06], who informally mentioned that additionally accumulating a random value might prevent the adversary from guessing the set limitations.

5.4 Homomorphic Proxy Re-Authenticators

Proxy re-cryptography [BBS98] is a powerful concept which allows proxies to transform cryptographic objects under one key to cryptographic objects under another key using a transformation key (a so called re-key). In particular, proxy re-encryption has shown to be of great practical interest in cloud scenarios such as data storage [CD16, BBL16], data sharing [XXW+16], publish-subscribe [BGP+16] as well as cloud-based identity management [NAL12, NA14,

[20] For compactness, we exchange all accumulators in a single game change and note that it is straight forward to unroll the exchange of the accumulators to k simple game changes.

ZSSH14, SSZ14]. In contrast, other proxy re-primitives, and in particular proxy re-signatures (or MACs), seem to unleash their full potential not before considering them in combination with homomorphic properties on the message space. Interestingly, however, this direction has received no attention so far. To this end, we introduce the notion of homomorphic proxy re-authenticators (HPRAs), which allows distinct senders to authenticate data under their own keys, and an evaluator (aggregator) can transform these single signatures or message authentication codes (MACs) to a MAC under a receiver's key without knowing it. Most importantly, the aggregator can evaluate arithmetic circuits (functions) on the inputs so that the resulting MAC corresponds to the evaluation of the respective function. Furthermore, we investigate whether we can hide the input messages from the aggregator. On the way to solve this, we formally define the notion of homomorphic proxy re-encryption (HPRE). We see data aggregation as the central application of our framework, but want to stress that it is not limited to this application.

Motivation. Data aggregation is an important task in the Internet of Things (IoT) and cloud computing. We observe a gap in existing work as the important issue of end-to-end authenticity and verifiability of computations on the data (aggregation results) is mostly ignored. We address this issue and propose a versatile non-interactive solution which is tailored to a multi-user setting. The additional authenticity features of our solution add robustness against errors occurring during transmission or aggregation even in the face of a non-trusted aggregator.

Multi-User Data Aggregation. Assume a setting where n senders, e.g., sensor nodes, regularly report data to some entity denoted the aggregator. The aggregator collects the data and then reports computations (evaluations of functions) on these data to a receiver. For example, consider environmental monitoring of hydroelectric plants being located in a mountainous region, where small sensors are used for monitoring purposes. Due to the lack of infrastructure (e.g., very limited cell coverage) sensors are not directly connected to the Internet and collected data is first sent to a gateway running at the premise of some telecommunication provider. This gateway aggregates the data and forwards it to some cloud service operated by the receiver.

Obviously, when the involved parties communicate via public networks, security related issues arise. Apart from achieving security against outsiders, there are also security and privacy related issues with respect to the involved parties.

In general, we identify three main goals. (1) End-to-end authenticity, i.e., protecting data items from unauthorized manipulation and preserving the source authenticity. (2) Concealing the original data from the aggregator and the receiver, and, even further, concealing the result of the computation from the aggregator. Clearly, in (2) we also want to conceal data from any outsider. (3) Establishing independent secret keys for the involved parties so that they do not share a single secret. Latter facilitates a dynamic setting.

Below, we present such an aggregation scenario, discuss why straightforward solutions fall short, and sketch our solution. Then, we discuss the problems popping up when we require stronger privacy guarantees and show how our primitives help to overcome these issues.

Authenticity & Input Privacy. In our first scenario, the n senders each hold their own signing key and within every period sender i reports a signed data item d_i to the aggregator. The aggregator must be able to evaluate functions $f \in \mathcal{F}$ (where $\mathcal{F}$ is some suitable class of functions, e.g., linear functions) on $d_1, \ldots, d_n$ so that a receiver will be convinced of the authenticity of the data and the correctness of the computation without fully trusting the aggregator (recall the end-to-end authenticity requirement). Moreover, although the inputs to the aggregator are not private, we still want them to be hidden relative to the function f, i.e., so that a receiver only learns what is revealed by f and $\hat{d} = f(d_1, \ldots, d_n)$, as a receiver might not need to learn the single input values.

A central goal is that the single data sources have individual keys. Thus, we can not directly employ homomorphic signatures (or MACs). Also the recent concept of multikey-homomorphic signatures [FMNP16, i5, LTWC16] does not help: even though they allow homomorphic operations on the key space, they do not consider transformations to some specific target key.[21] With HPRAs we can realize this, as the aggregator (who holds re-keys from the senders to some receiver) can transform all the single signatures or MACs to a MAC under the receiver's key (without having access to it). Moreover, due to the homomorphic property, a MAC which corresponds to the evaluation of a function f on the inputs can be computed. The receiver can then verify the correctness of the computation, i.e., that $\hat{d} = f(d_1, \ldots, d_n)$, and the authenticity of the used inputs (without explicitly learning them) using its independent MAC key.

Adding Output Privacy. In our second scenario, we additionally want data privacy guarantees with respect to the aggregator. This can be crucial if the aggregator is running in some untrusted environment, e.g., the cloud. We achieve this by constructing an output private HPRA. In doing so, one has to answer the question as how to confidentially provide the result of the computation to the receiver and how to guarantee the authenticity (verifiability) of the computation. We tackle this issue by introducing a HPRE where the homomorphism is compatible to the one of the HPRA. The sources then additionally encrypt the data under their own keys and the aggregator re-encrypts the individual ciphertexts to a ciphertext under a receiver's key and evaluates the same function f as on the MACs on the ciphertexts. This enables the receiver to decrypt the result $\hat{d}$ using its *own* decryption key and to verify the MAC on $\hat{d}$ together with a description of the function f. In addition, we use a trick to prevent public verifiability of the signatures from the single data sources, as public verifiabil-

[21] While the homomorphic properties might allow one to define a function mapping to a target key, it is unclear whether handing over the description of such a function to a proxy would maintain the security requirements posed by our application.

ity potentially leaks the signed data items which trivially would destroy output privacy.

Contribution. Our contributions can be summarized as follows.

- We introduce the notion of homomorphic proxy re-authenticators (HPRA). Our framework tackles multi-user data aggregation in a dynamic setting. For the first time, we thereby consider independent keys of the single parties, the verifiability of the evaluation of general functions on the authenticated inputs by the sources, as well as privacy with respect to the aggregator.
- As a means to achieve the strong privacy requirements imposed by our security model, we formally define the notion of homomorphic proxy re-encryption (HPRE), which may be of independent interest.
- We present two modular constructions of HPRA schemes for the family of linear function classes, which differ regarding the strength of the provided privacy guarantees. On our way, we establish various novel building blocks. Firstly, we present a linearly homomorphic MAC which is suitable to be used in our construction. Secondly, to achieve the stronger privacy guarantees, we construct a HPRE scheme for the family of linear function classes. All our proofs are modular in the sense that we separately prove the security of our building blocks; our overall proofs then build upon the results obtained for the building blocks. Thus, our building blocks may as well easily be used in other constructions.

Related Work. In this paragraph we review related work. As our focus is on non-interactive approaches, we omit interactive approaches where clients download all the data, decrypt them locally, compute a function, and send the results back along with a zero-knowledge proof of correctness (as, e.g., in [DL11]).

Proxy Re-Cryptography. Proxy re-encryption (PRE) [BBS98] allows a semi-trusted proxy to transform a message encrypted under the key of some party into a ciphertext to the same message under a key of another party, where the proxy performing the re-encryption learns nothing about the message. This primitive has been introduced in [BBS98], further studied in [ID03] and the first strongly secure constructions have been proposed by Ateniese et al. in [AFGH06]. Boneh et al. construct PRE in the symmetric setting [BLMR13]. Follow-up work focuses on even stronger (IND-CCA2 secure) schemes (cf. [CH07, LV11, NAL15, NAL16]). Since we, however, require certain homomorphic properties, we focus on IND-CPA secure schemes (as IND-CCA2 security does not allow any kind of malleability). In previous work by Ayday et al. [ARHR13], a variant of the linearly homomorphic Paillier encryption scheme and proxy encryption in the sense of [ID03] were combined. Here, the holder of a key splits the key and gives one part to the proxy and one to the sender, with the drawback that the secret key is exposed when both collude. We are looking for proxy re-encryption that is homomorphic, works in a multi-user setting but is collusion-safe and non-interactive, i.e., re-encryption keys can be computed by the sender using

only the public key of the receiver without any interaction and a collusion of sender and proxy does not reveal the receiver's key. Also note that, as our focus is on practically efficient constructions, we do not build upon fully homomorphic encryption [Gen09], which allows to build HPRE using the rather expensive bootstrapping technique. In concurrent work Ma et al. [MLO16] follow this approach and propose a construction of a PRE scheme with homomorphic properties which additionally achieves key privacy. They build upon [GSW13] using the bootstrapping techniques in [AP14] and apply some modifications for key privacy. While their construction can be seen as a HPRE in our sense, they do not formally define a corresponding security model and we are not aware of a suitable formalization for our purposes.

Proxy re-signatures, i.e., the signature analogue to proxy re-encryption, have been introduced in [BBS98] and formally studied in [ID03]. Later, [AH05] introduced stronger security definitions, constructions and briefly discussed some applications. However, the schemes in [AH05] and follow up schemes [LV08] do not provide a homomorphic property and it is unclear how they could be extended. The concept of *homomorphic proxy re-authenticators*, which we propose, or a related concept, has to the best of our knowledge not been studied before.

Homomorphic Authenticators. General (non-interactive) verifiable computing techniques (cf. [WB15] for a recent overview) are very expressive, but usually prohibitive regarding proof computation (proof size and verification can, however, be very small and cheap respectively). In addition, the function and/or the data needs to be fixed at setup time and inputs are not authenticated. Using homomorphic authenticators allows evaluations of functions on authenticated inputs under a single key (cf. [Cat14] for a recent overview). They are dynamic with respect to the authenticated data and the evaluated function, and also efficient for interesting classes of functions. Evaluating results is typically not more efficient than computing the function (unless using an amortized setting [BFR13, CFW14]). Yet, they provide benefits when saving bandwidth is an issue and/or the inputs need to be hidden from evaluators (cf. [LDPW14, CMP14]). Computing on data authenticated under different keys using so called multi-key homomorphic authenticators [FMNP16, i5, LTWC16], has only very recently been considered. Even though they are somewhat related, they are no replacement for what we are proposing.

Aggregator-Oblivious Encryption (AOE). AOE [RN10, SCR+11] considers data provided by multiple producers, which is aggregated by a semi-honest aggregator. The aggregator does not learn the single inputs but only the final result. Follow-up work [JL13, LEM14, BJL16] improved this approach in various directions. Furthermore, [CSS12] introduced a method to achieve fault tolerance, being applicable to all previous schemes. There are also other lines of work on data aggregation, e.g., [LC13, CCMT09], [LCP14, GMP14]. Very recently, [LEÖM15] combined AOE with homomorphic tags to additionally provide verifiability of the aggregated results. Here, every user has a tag key and the aggregator ad-

ditionally aggregates the tags. Verification can be done under a pre-distributed combined fixed tag key. Their approach is limited to a single function (the sum) and requires a shared secret key-setting, which can be problematic.

In all previous approaches it is impossible to hide the outputs (i.e., the aggregation results) from the aggregator. In contrast to only hiding the inputs, we additionally want to hide the outputs. In addition, we do not want to assume a trusted distribution of the keys, but every sender should authenticate and encrypt under his own key and the aggregator can then perform re-operations (without any secret key) to the receiver.

5.4.1 Formal Security Model

We introduce homomorphic proxy re-authenticators (HPRAs) and rigorously formalize a suitable security model capturing the requirements discussed above. Recall that our goal is formalize a scheme where multiple signers create signatures under their own keys. Those signatures can then be transformed to a MAC, which is valid under the receiver's key, and which authenticates the evaluation of some function on the authenticated data items provided by the potentially different signers. Our goal is to obtain a flexible framework with various possible instantiations. Accordingly, our definitions are rather generic. We stress that both the source and receiver re-key generation, besides the secret key of the executing party, only require public inputs, i.e., are non-interactive.

Definition 5.27. *A homomorphic proxy re-authenticator* (HPRA) *for a family of function classes* $\{\mathcal{F}_{\mathsf{PP}}\}$ *is a tuple of PPT algorithms* $(\mathsf{Gen}, \mathsf{SGen}, \mathsf{VGen}, \mathsf{Sign}, \mathsf{Verify}, \mathsf{SRGen}, \mathsf{VRGen}, \mathsf{Agg}, \mathsf{AVerify})$*, where* Verify *is optional. They are defined as follows:*

$\mathsf{Gen}(1^\kappa, \ell)$: *Takes security parameter* κ *and vector length* ℓ *and outputs parameters* PP*, determining the message space* $\mathcal{M}^\ell$*, the function class* $\mathcal{F}_{\mathsf{PP}}$ *containing functions* $f : (\mathcal{M}^\ell)^n \to \mathcal{M}^\ell$ *with* $\ell, n \leq \mathsf{poly}(\kappa)$*, as well as a tag space being exponentially large in* κ*.*

$\mathsf{SGen}(\mathsf{PP})$: *Takes parameters* PP *as input, and outputs a signer key* $(\mathsf{id}, \mathsf{sk}, \mathsf{pk})$*.*

$\mathsf{VGen}(\mathsf{PP})$: *Takes parameters* PP*, and outputs a secret MAC key* mk *and public auxiliary information* aux*.*

$\mathsf{Sign}(\mathsf{sk}, \vec{m}, \tau)$: *Takes a signer secret key* sk*, a message vector* $\vec{m}$*, and a tag* τ *as input, and outputs a signature* σ*.*

$\mathsf{Verify}(\mathsf{pk}, \vec{m}, \tau, \sigma)$: *Takes a signer public key* pk*, a message vector* $\vec{m}$*, a tag* τ*, and a signature* σ *as input, and outputs a bit* b*.*

$\mathsf{SRGen}(\mathsf{sk}_i, \mathsf{aux})$: *Takes a signer secret key* sk_i*, some auxiliary information* aux*, and outputs a re-encryption key* rk_i*.*

$\mathsf{VRGen}(\mathsf{pk}_i, \mathsf{mk}, \mathsf{rk}_i)$: *Takes a signer public key* pk_i *and a MAC key* mk*, as well as a re-encryption key* rk_i *as input, and outputs an aggregation key* ak_i*.*

$\mathsf{Agg}((\mathsf{ak}_i)_{i\in[n]}, (\sigma_i)_{i\in[n]}, \tau, f)$: *Takes n aggregation keys $(\mathsf{ak}_i)_{i\in[n]}$, n signatures $(\sigma_i)_{i\in[n]}$, a tag τ, and a function $f \in \mathcal{F}_{\mathsf{PP}}$ as input, and outputs an aggregate authenticated message vector Φ.*

$\mathsf{AVerify}(\mathsf{mk}, \Phi, \mathsf{ID}, f)$: *Takes a MAC key mk, an aggregate authenticated message vector Φ, n identifiers $\mathsf{ID} = (\mathsf{id}_i)_{i\in[n]}$, and a function $f \in \mathcal{F}_{\mathsf{PP}}$. It outputs a message vector and a tag $(\vec{m}, \tau)$ on success and $(\bot, \bot)$ otherwise.*

Security Properties. Below we define the oracles, where the public parameters and the keys generated in the security games are implicitly available to the oracles. While most oracle definitions are fairly easy to comprehend and therefore not explicitly explained, we note that the RoS oracle is used to model the requirement that signatures do not leak the signed data in a real-or-random style. The environment maintains the initially empty sets $\mathtt{HU}$ and $\mathtt{CU}$ of honest and corrupted users ($\mathtt{CU}$ is only set in the output privacy game). Further, it maintains the initially empty sets $\mathtt{SK}$, $\mathtt{RK}$ and $\mathtt{AK}$ of signer, re-encryption and aggregation keys, and an initially empty set $\mathtt{SIG}$ of message-identity pairs.

$\mathsf{SG}(i)$: If $\mathtt{SK}[i] \neq \bot$ return $\bot$. Otherwise run $(\mathsf{id}_i, \mathsf{sk}_i, \mathsf{pk}_i) \leftarrow \mathsf{SGen}(\mathsf{PP})$, set $\mathtt{SK}[i] \leftarrow (\mathsf{id}_i, \mathsf{sk}_i, \mathsf{pk}_i)$, and, if $i \notin \mathtt{CU}$ set $\mathtt{HU} \leftarrow \mathtt{HU} \cup \{i\}$. Return $(\mathsf{id}_i, \mathsf{pk}_i)$.

$\mathsf{SKey}(i)$: If $i \notin \mathtt{HU}$ return $\bot$. Otherwise return $\mathtt{SK}[i]$.

$\mathsf{Sig}((j_i)_{i\in[n]}, (\vec{m}_i)_{i\in[n]})$: If $\mathtt{SK}[j_i] = \bot$ for any $i \in [n]$, or there exists $u, v \in [n], u \neq v$ so that $j_u = j_v$, return $\bot$. Otherwise sample a random tag τ and compute $(\sigma_{j_i} \leftarrow \mathsf{Sign}(\mathtt{SK}[j_i][2], \vec{m}_i, \tau))_{i\in[n]}$, set $\mathtt{SIG}[\tau] \leftarrow \mathtt{SIG}[\tau] \cup \{(\vec{m}_i, \mathtt{SK}[j_i][1])\}$ for $i \in [n]$, and return $(\sigma_{j_i})_{i\in[n]}$ and τ.

$\mathsf{RoS}((j_i)_{i\in[n]}, (\vec{m}_i)_{i\in[n]}, b)$: If $\mathtt{SK}[j_i] = \bot$ or $j_i \in \mathtt{CU}$ for any $i \in [n]$ return $\bot$. Otherwise sample τ uniformly at random and if $b = 0$ compute $(\sigma_{j_i} \leftarrow \mathsf{Sign}(\mathtt{SK}[j_i][2], \vec{m}_i, \tau))_{i\in[n]}$. Else choose $(\vec{r}_i)_{i\in[n]} \xleftarrow{R} (\mathcal{M}^\ell)^n$ where $\mathcal{M}$ is the message space and compute $(\sigma_{j_i} \leftarrow \mathsf{Sign}(\mathtt{SK}[j_i][2], \vec{r}_i, \tau))_{i\in[n]}$. Finally, return $(\sigma_{j_i})_{i\in[n]}$.

$\mathsf{SR}(i)$: If $\mathtt{SK}[i] = \bot \vee \mathtt{RK}[i] \neq \bot$ return $\bot$. Else, set $\mathtt{RK}[i] \leftarrow \mathsf{SRGen}(\mathtt{SK}[i][2], \mathsf{aux})$ and return $\mathtt{RK}[i]$.

$\mathsf{VR}(i)$: If $\mathtt{SK}[i] = \bot \vee \mathtt{RK}[i] = \bot \vee \mathtt{AK}[i] \neq \bot$ return $\bot$. Else, set $\mathtt{AK}[i] \leftarrow \mathsf{VRGen}(\mathtt{SK}[i][3], \mathsf{mk}, \mathtt{RK}[i])$.

$\mathsf{VRKey}(i)$: Return $\mathtt{AK}[i]$.

$\mathsf{A}((\sigma_{j_i})_{i\in[n]}, (j_i)_{i\in[n]}, \tau, f)$: Check validity of all σ_{j_i}, whether $f \in \mathcal{F}_{\mathsf{PP}}$, whether $\mathtt{SIG}[\tau] = \bot$, and return $\bot$ if any check fails. Further, check whether there exists $u, v \in [n], u \neq v$ so that $j_u = j_v$ and return $\bot$ if so. Obtain $(\mathsf{ak}_{j_i})_{i\in[n]}$ from $\mathtt{AK}$ and return $\bot$ if $\mathtt{AK}[j_i] = \bot$ for any $i \in [n]$. Set $\mathtt{SIG}[\tau] \leftarrow \bigcup_{i\in[n]}\{(\vec{m}_{j_i}, \mathtt{SK}[j_i][1])\}$ and return $\Phi \leftarrow \mathsf{Agg}((\mathsf{ak}_{j_i})_{i\in[n]}, (\sigma_{j_i})_{i\in[n]}, \tau, f)$.

We require a HPRA to be correct, signer unforgeable, aggregator unforgeable, and input private. We formally introduce the security notions below. Intuitively, correctness requires that everything works as intended if everyone behaves honestly.

Definition 5.28 (Correctness)**.** *A* HPRA *for a family of function classes* $\{\mathcal{F}_{\mathsf{PP}}\}$ *is correct, if for all* κ*, for all* $\ell \leq \mathsf{poly}(\kappa)$*, for all* $\mathsf{PP} \leftarrow \mathsf{Gen}(1^\kappa, \ell)$ *determining* $\mathcal{F}_{\mathsf{PP}}$*, for all* $n \leq \mathsf{poly}(\kappa)$*, for all* $((\mathsf{id}_i, \mathsf{sk}_i, \mathsf{pk}_i) \leftarrow \mathsf{SGen}(\mathsf{PP}))_{i\in[n]}$*, for all* $(\mathsf{mk}, \mathsf{aux}) \leftarrow \mathsf{VGen}(\mathsf{PP})$*, for all* $(\vec{m}_i)_{i\in[n]}$*, for all* τ*, for all* $(\sigma_i \leftarrow \mathsf{Sign}(\mathsf{sk}_i, \vec{m}_i, \tau))_{i\in[n]}$*, for all* $(\mathsf{ak}_i \leftarrow \mathsf{VRGen}(\mathsf{pk}_i, \mathsf{mk}, \mathsf{SRGen}(\mathsf{sk}_i, \mathsf{aux})))_{i\in[n]}$*, for all* $f \in \mathcal{F}_{\mathsf{PP}}$*, for all* $\Phi \leftarrow \mathsf{Agg}((\mathsf{ak}_i)_{i\in[n]}, (\sigma_i)_{i\in[n]}, \tau, f)$ *it holds that* $(\mathsf{Verify}(\mathsf{pk}_i, \vec{m}_i, \tau, \sigma_i) = 1)_{i\in[n]}$ *and that* $\mathsf{AVerify}(\mathsf{mk}, \Phi, \mathsf{ID}, f) = 1$*, where we sometimes omit to make the domains of the values over which we quantify explicit for brevity.*

Signer unforgeability requires that, as long as the aggregator remains honest, no coalition of dishonest signers can produce a valid aggregate authenticated message vector Φ with respect to function $f \in \mathcal{F}_{\mathsf{PP}}$ so that Φ is outside of the range of f evaluated on arbitrary combinations of actually signed vectors. Aggregator unforgeability is the natural counterpart of signer unforgeability, where the aggregator is dishonest while the signers are honest.[22]

Definition 5.29 (T-Unforgeability)**.** *Let* $\mathsf{T} \in \{\mathit{Signer}, \mathit{Aggregator}\}$*. A* HPRA *for a family of function classes* $\{\mathcal{F}_{\mathsf{PP}}\}$ *is* T*-unforgeable, if for all PPT adversaries* $\mathcal{A}$ *there is a negligible function* $\varepsilon(\cdot)$ *such that*

$$\Pr\left[\begin{array}{l} \mathsf{PP} \leftarrow \mathsf{Gen}(1^\kappa, \ell), \\ (\mathsf{mk}, \mathsf{aux}) \leftarrow \mathsf{VGen}(\mathsf{PP}), \\ (\Phi^\star, \mathsf{ID}^\star, f^\star) \leftarrow \mathcal{A}^{\mathcal{O}_\mathsf{T}}(\mathsf{PP}, \mathsf{aux}), \\ (\vec{m}, \tau) \leftarrow \mathsf{AVerify}(\mathsf{mk}, \Phi^\star, \mathsf{ID}^\star, f^\star) \end{array} : \begin{array}{r} \vec{m} \neq \bot \ \wedge \ f^\star \in \mathcal{F}_{\mathsf{PP}} \ \wedge \\ 0 < n, \ell \leq \mathsf{poly}(\kappa) \ \wedge \\ \left(\nexists\ (\vec{m}_j)_{j\in[n]} : (\forall\ j \in [n] : \right. \\ \left. (\vec{m}_j, \mathsf{id}_j^\star) \in \mathtt{SIG}[\tau]\right) \wedge \\ f^\star((\vec{m}_j)_{j\in[n]}) = \vec{m}\big) \end{array}\right] \leq \varepsilon(\kappa),$$

where $\mathcal{O}_\mathsf{T} := \{\mathsf{SG}(\cdot), \mathsf{SKey}(\cdot), \mathsf{SR}(\cdot), \mathsf{VR}(\cdot), \mathsf{A}(\cdot,\cdot,\cdot)\}$ *for* $\mathsf{T} = \mathit{Signer}$ *and* $\mathcal{O}_\mathsf{T} := \{\mathsf{SG}(\cdot), \mathsf{Sig}(\cdot,\cdot), \mathsf{SR}(\cdot), \mathsf{VR}(\cdot), \mathsf{VRKey}(\cdot)\}$ *for* $\mathsf{T} = \mathit{Aggregator}$*.*

Input privacy captures the requirement that an aggregate authenticated message vector does not leak more about the inputs to f as the evaluation result and the description of f would leak on their own.

Definition 5.30 (Input Privacy)**.** *A* HPRA *for a family of function classes* $\{\mathcal{F}_{\mathsf{PP}}\}$ *is input private if for all* $\kappa \in \mathbb{N}$*, for all* $\ell \leq \mathsf{poly}(\kappa)$*, for all* $\mathsf{PP} \leftarrow \mathsf{Gen}(1^\kappa, \ell)$ *determining* $\mathcal{F}_{\mathsf{PP}}$*, for all* $f \in \mathcal{F}_{\mathsf{PP}}$ *implicitly defining* n*, for all tags* τ*, and for all* $(\vec{m}_{11}, \ldots, \vec{m}_{n1})$ *and* $(\vec{m}_{12}, \ldots, \vec{m}_{n2})$ *where* $f(\vec{m}_{11}, \ldots, \vec{m}_{n1}) = f(\vec{m}_{12}, \ldots, \vec{m}_{n2})$*, for all* $(\mathsf{mk}, \mathsf{aux}) \leftarrow \mathsf{VGen}(\mathsf{PP})$*, for all* $((\mathsf{sk}_i, \mathsf{pk}_i) \leftarrow \mathsf{SGen}(\mathsf{PP}))_{i\in[n]}, (\mathsf{ak}_i \leftarrow \mathsf{SRGen}($

[22] It is impossible to consider both, signers and aggregators, to be dishonest at the same time, as such a coalition could essentially authenticate everything. This is in contrast to the setting of proxy re-encryption, where it makes sense to model security in the face of receivers colluding with the proxy.

$\mathsf{sk}_i, \mathsf{aux}, \mathsf{VRGen}(\mathsf{pk}_i, \mathsf{mk}))_{i\in[n]}$, *the following distributions are identical:*

$$\{\mathsf{Agg}((\mathsf{ak}_i)_{i\in[n]}, (\mathsf{Sign}(\mathsf{sk}_i, \vec{m}_{i1}, \tau))_{i\in[n]}, \tau, f)\},$$
$$\{\mathsf{Agg}((\mathsf{ak}_i)_{i\in[n]}, (\mathsf{Sign}(\mathsf{sk}_i, \vec{m}_{i2}, \tau))_{i\in[n]}, \tau, f)\}.$$

Additionally, a HPRA may provide output privacy. It models that the aggregator neither learns the inputs nor the result of the evaluation of f.

Definition 5.31 (Output Privacy)**.** *A* HPRA *for a family of function classes* $\{\mathcal{F}_{\mathsf{PP}}\}$ *is output private, if for all PPT adversaries* $\mathcal{A}$ *there is a negligible function* $\varepsilon(\cdot)$ *such that:*

$$\Pr\left[\begin{array}{l|l} \mathsf{PP} \leftarrow \mathsf{Gen}(1^\kappa, \ell), (\mathtt{CU}, \mathtt{ST}) \leftarrow \mathcal{A}(\mathsf{PP}), b \xleftarrow{R} \{0,1\}, & \\ (\mathsf{mk}, \mathsf{aux}) \leftarrow \mathsf{VGen}(\mathsf{PP}), \mathcal{O} \leftarrow \{\mathsf{SG}(\cdot), \mathsf{SKey}(\cdot), & : \quad b = b^\star \\ \mathsf{RoS}(\cdot,\cdot,b), \mathsf{SR}(\cdot), \mathsf{VR}(\cdot), \mathsf{VRKey}(\cdot)\}, & \\ b^\star \leftarrow \mathcal{A}^{\mathcal{O}}(\mathsf{aux}, \mathtt{ST}) & \end{array}\right] \leq 1/2 + \varepsilon(\kappa).$$

5.4.2 An Input Private Scheme for Linear Functions

Now we present our first HPRA for the family of linear function classes $\{\mathcal{F}^{\mathsf{lin}}_{\mathsf{PP}}\}$. The main challenge we face is to construct a signature scheme with an associated linearly homomorphic MAC scheme, where the translation of the signatures under one key to a MAC under some other key works out. Since we believe that our linearly homomorphic MAC may as well be useful in other settings we present it as a standalone building block and then proceed with our full construction, where the MAC is used as a submodule. Both build upon the ideas used in the signature scheme presented in [BFKW09].

A Suitable Linearly Homomorphic MAC. We present our linearly homomorphic MAC in Scheme 5.7. We can not recycle the security arguments from [BFKW09] as we require the ability to submit arbitrary tags τ to the Sig oracle (cf. Definition 2.19). Thus we directly prove unforgeability.

$\underline{\mathsf{Gen}(\kappa, \ell)}$: Run $\mathsf{BG} \leftarrow \mathsf{BGGen}(1^\kappa)$, fix $H : \{0,1\}^* \to \mathbb{G}$, choose $(g_i)_{i\in[\ell]} \xleftarrow{R} (\mathbb{G}^*)^\ell$, and return $\mathsf{PP} \leftarrow (\mathsf{BG}, H, (g_i)_{i\in[\ell]}, \ell)$.

$\underline{\mathsf{KeyGen}(\mathsf{PP})}$: Choose $\alpha \xleftarrow{R} \mathbb{Z}_p$ and return $\mathsf{sk} \leftarrow (\mathsf{PP}, \alpha)$.

$\underline{\mathsf{Sign}(\mathsf{sk}, \vec{v}, \mathsf{id}, \tau)}$: Parse sk as (PP, α) and return $\mu \leftarrow e(H(\tau||\mathsf{id}) \cdot \prod_{j\in[\ell]} g_j^{v_j}, g^\alpha)$.

$\underline{\mathsf{Comb}(\mathsf{PP}, f, (\mu_i)_{i\in[n]})}$: Parse f as $(\omega_i)_{i\in[n]}$ and return $\mu \leftarrow \prod_{i\in[n]} \mu_i^{\omega_i}$.

$\underline{\mathsf{Verify}(\mathsf{sk}, \vec{v}, \mu, \tau, (\mathsf{id}_i)_{i\in[n]}, f)}$: Parse sk as (PP, α), f as $(\omega_i)_{i\in[n]}$, and output 1 if the following holds, and 0 otherwise: $\mu = e(\prod_{i\in[n]} H(\tau||\mathsf{id}_i)^{\omega_i} \prod_{j\in[\ell]} g_j^{v_j}, g^\alpha)$

Scheme 5.7: Linearly homomorphic MAC based on [BFKW09].

Lemma 5.33. *If the bilinear BDDH assumption holds, then Scheme 5.7 is an unforgeable linearly homomorphic MAC in the ROM.*

We weaken BDDH to BDDH', where the adversary is given $(\mathsf{BG}, g^a, \mathbf{g}^b, \mathbf{g}^c)$ with $a, b \xleftarrow{R} \mathbb{Z}_q$ and needs to decide whether $c = ab$, or c is random in $\mathbb{Z}_q$. Clearly, BDDH' is weaker than BDDH: given a BDDH instance $(\mathsf{BG}, g^a, g^b, g^c, \mathbf{g}^d)$ one can use the BDDH' distinguisher on $(\mathsf{BG}, g^a, e(g^b, g^c), \mathbf{g}^d)$.

Proof. We prove Lemma 5.33 under BDDH' (and, therefore, BDDH) and let $q_{\mathsf{R}} \leq \mathsf{poly}(\kappa)$ be the number of random oracle queries (also including the calls we use internally in the reduction). We obtain a BDDH' instance $(\mathsf{BG}, g^a, \mathbf{g}^b, \mathbf{g}^c)$, choose $((x_i, y_i) \xleftarrow{R} {\mathbb{Z}_q}^2)_{i\in[q_{\mathsf{R}}]}$, set $((g^{a_i}, \mathbf{g}^b, \mathbf{g}^{c_i}) \leftarrow ((g^a)^{x_i} g^{y_i}), \mathbf{g}^b, (\mathbf{g}^c)^{x_i} (\mathbf{g}^b)^{y_i})_{i\in[q_{\mathsf{R}}]}$, choose $(u_i)_{i\in[\ell]} \xleftarrow{R} \mathbb{Z}_q^\ell$, and set $(g_i \leftarrow g^{u_i})_{i\in[\ell]}$. Finally, we set $\mathsf{PP} \leftarrow (\mathsf{BG}, H, (g_i)_{i\in[\ell]})$, start $\mathcal{A}$ on PP and simulate the oracles as follows. We use a counter j initialized to 1 and an initially empty list $\mathtt{H}$.

$H(x)$: If $x \notin \mathtt{H}$ set $\mathtt{H}[x] \leftarrow (g^{a_j}, \mathbf{g}^{c_j})$ and $j \leftarrow j + 1$. Return $\mathtt{H}[x][1]$.

$\mathsf{Sig}((\vec{v}_i)_{i\in[n]}, (\mathsf{id}_i)_{i\in[n]}, \tau)$: As the original oracle, except that the values $(\mu_i)_{i\in[n]}$ are computed as follows. For $i \in [n]$ call $H(\tau||\mathsf{id}_i)$, set $\mu_i \leftarrow \mathtt{H}[\tau||\mathsf{id}_i][2] \cdot \prod_{j\in[\ell]} (\mathbf{g}^b)^{v_{i_j} \cdot u_j}$ and return $(\mu_i)_{i\in[n]}$.

Now, if the BDDH' instance is valid the simulation is perfect. If the BDDH' instance is invalid the responses of the Sig oracle are uniformly random and independent values from $\mathbb{G}_T$ and therefore do not reveal anything about the MAC key. In this case the adversary can only guess a forgery with probability $1/q$. Both cases are computationally indistinguishable. □

Our Input Private Construction. In Scheme 5.8 we present our HPRA construction for the family of linear function classes $\{\mathcal{F}^{\mathsf{lin}}_{\mathsf{PP}}\}$. It allows to authenticate vectors of length ℓ, so that the same function can be evaluated per vector component. In our application scenario we have $\ell = 1$. We allow to parametrize our construction with an algorithm $\mathsf{Eval}(\cdot, \cdot)$, which defines how to compute $f \in \mathcal{F}^{\mathsf{lin}}_{\mathsf{PP}}$ on the message vector. When directly instantiating Scheme 5.8, Eval is defined as $\mathsf{Eval}(f, (\vec{m}_i)_{i\in[n]}) := f((\vec{m}_i)_{i\in[n]})$. Also note that the signature σ output by Sign also carries the message. This is just an artifact required for the generic extension presented in Scheme 5.10 and can be omitted when instantiating the plain Scheme 5.8.

To prove the security of our construction, we require a novel assumption which we introduce below. We call this assumption—which follows from the Uber-assumption [Boy08] with $R = S = \langle 1, 1/U, U, V, W\rangle, T = \langle 1\rangle, f = UVW$—extended bilinear computational Diffie-Hellman assumption (eBCDH).

Definition 5.32 (Extended Bilinear Computational Diffie-Hellman Assumption (eBCDH))**.** *The eBCDH assumption holds relative to* BGGen*, if for all PPT adversaries* $\mathcal{A}$ *there is a negligible function* $\varepsilon(\cdot)$ *such that*

$$\Pr\left[\begin{array}{l|l} \mathsf{BG} \leftarrow \mathsf{BGGen}(1^\kappa, 1), \ \mathsf{u}, \mathsf{v}, \mathsf{w} \xleftarrow{R} \mathbb{Z}_\mathsf{q} & \mathbf{h} = e(g, g)^{uvw} \\ \mathbf{h} \leftarrow \mathcal{A}(\mathsf{BG}, g^{1/u}, g^u, g^v, g^w) & \end{array}\right] \leq \varepsilon(\kappa).$$

$\underline{\mathsf{Gen}(1^\kappa, \ell)}$: Run $\mathsf{BG} \leftarrow \mathsf{BGGen}(1^\kappa)$, fix $H : \mathbb{Z}_q \to \mathbb{G}$, choose $(g_i)_{i\in[\ell]} \xleftarrow{R} \mathbb{G}^\ell$, and return $\mathsf{PP} \leftarrow (\mathsf{BG}, H, (g_i)_{i\in[\ell]}, \ell)$.

$\underline{\mathsf{SGen}(\mathsf{PP})}$: Choose $\beta \xleftarrow{R} \mathbb{Z}_q$, set $\mathsf{id} \leftarrow g^\beta$, $\mathsf{pk} \leftarrow (\mathsf{PP}, g^\beta, g^{1/\beta})$, $\mathsf{sk} \leftarrow (\mathsf{pk}, \beta)$, and return $(\mathsf{id}, \mathsf{sk}, \mathsf{pk})$.

$\underline{\mathsf{VGen}(\mathsf{PP})}$: Choose $\alpha \xleftarrow{R} \mathbb{Z}_q$, set $\mathsf{aux} \leftarrow \emptyset$, $\mathsf{mk} \leftarrow (\mathsf{PP}, \alpha)$ and return $(\mathsf{mk}, \mathsf{aux})$.

$\underline{\mathsf{Sign}(\mathsf{sk}, \vec{m}, \tau)}$: Parse sk as $(((\mathsf{BG}, H, (g_i)_{i\in[\ell]}, \ell), g^\beta, \cdot), \beta)$, compute and return $\sigma \leftarrow (\sigma', \vec{m})$, where

$$\sigma' \leftarrow \Big(H(\tau||g^\beta) \cdot \prod_{i=1}^{\ell} g_i^{m_i}\Big)^\beta.$$

$\underline{\mathsf{Verify}(\mathsf{pk}, \vec{m}, \tau, \sigma)}$: Parse pk as $((\mathsf{BG}, H, (g_i)_{i\in[\ell]}, \ell), g^\beta, \cdot)$, and σ as $(\sigma', \vec{m}')$, and return 1 if the following holds and 0 otherwise:

$$e(H(\tau||g^\beta) \cdot \prod_{i=1}^{\ell} g_i^{m_i}, g^\beta) = e(\sigma, g) \quad \wedge \quad \vec{m} = \vec{m}'.$$

$\underline{\mathsf{SRGen}(\mathsf{sk}_i, \mathsf{aux})}$: Return $\mathsf{rk}_i \leftarrow \emptyset$.

$\underline{\mathsf{VRGen}(\mathsf{pk}_i, \mathsf{mk}, \mathsf{rk}_i)}$: Parse pk_i as $(\cdot, \cdot, g^{1/\beta_i})$, mk as $(\cdot, \alpha)$, and return $\mathsf{ak}_i \leftarrow (g^{1/\beta_i})^\alpha$.

$\underline{\mathsf{Agg}((\mathsf{ak}_i)_{i\in[n]}, (\sigma_i)_{i\in[n]}, \tau, f)}$: Parse f as $(\omega_i)_{i\in[n]}$, and for $i \in [n]$ parse σ_i as $(\sigma_i', \vec{m}_i)$ and return $\Phi \leftarrow (\mathsf{Eval}(f, (\vec{m}_i)_{i\in[n]}), \mu, \tau)$, where

$$\mu \leftarrow \prod_{i\in[n]} e(\sigma_i'^{\omega_i}, \mathsf{ak}_i).$$

$\underline{\mathsf{AVerify}(\mathsf{mk}, \Phi, \mathsf{ID}, f)}$: Parse mk as (PP, α), Φ as $(\vec{m}, \mu, \tau)$, ID as $(g^{\beta_i})_{i\in[n]}$ and f as $(\omega_i)_{i\in[n]}$ and return $(\vec{m}, \tau)$ if the following holds, and $(\perp, \perp)$ otherwise:

$$\mu' = \Big(\prod_{i=1}^{n} e(g^{\omega_i}, H(\tau||g^{\beta_i})) \cdot e(\prod_{i=1}^{\ell} g_i^{m_i}, g)\Big)^\alpha$$

Scheme 5.8: HPRA scheme for $\mathcal{F}_{\mathsf{lin}}$ parametrized by Eval.

Theorem 5.9. *If the linearly homomorphic MAC from Scheme 5.7 is unforgeable and the eBCDH assumption holds, then Scheme 5.8 represents a signer unforgeable, aggregator unforgeable and input private* HPRA *for family of linear function classes* $\{\mathcal{F}_{\mathsf{PP}}^{\mathsf{lin}}\}$ *in the ROM.*

We prove Theorem 5.9 by proving Lemmas 5.34-5.36.

Lemma 5.34. *If Scheme 5.7 is unforgeable, then Scheme 5.8 is signer unforgeable.*

Proof. We show that an efficient adversary $\mathcal{A}$ against signer unforgeability can efficiently be turned into an efficient adversary $\mathcal{B}$ against unforgeability of Scheme 5.7 by presenting a reduction $\mathcal{R}$, which interacts with the unforgeability

challenger of Scheme 5.7 and simulates the environment for $\mathcal{A}$, i.e., so that $\mathcal{B} = (\mathcal{A}, \mathcal{R})$.

$\mathcal{R}$ obtains PP from $\mathcal{C}$, starts $\mathcal{A}$ on PP and $\mathsf{aux} = \emptyset$ and simulates the environment for $\mathcal{A}$ as follows.

$\mathsf{VR}(i)$: If $\mathtt{SK}[i] = \bot \ \vee \ \mathtt{RK}[i] = \bot$ return $\bot$, otherwise set $\mathtt{AK}[i] \leftarrow \top$.

$\mathsf{A}((\sigma_{j_i})_{i\in[n]}, (j_i)_{i\in[n]}, \tau, f)$: Check whether any $(\mathsf{Verify}(\mathsf{pk}_{j_i}, \vec{m}_{j_i}, \tau, \sigma_{j_i}) = 0)_{i\in[n]}$, whether there is any duplicate index j in $(j_i)_{i\in[n]}$, whether $\mathtt{SIG}[\tau] \neq \bot$, or whether $\mathtt{AK}[j_i] = \bot$ for any $i \in [n]$, and return $\bot$ if so. Otherwise, set $\mathtt{SIG}[\tau] \leftarrow \bigcup_{i\in[n]}\{(\vec{m}_{j_i}, \mathsf{id}_{j_i})\}$ compute $(\mu_i)_{i\in[n]} \leftarrow \mathcal{C}.\mathsf{Sig}((\vec{m}_{j_i})_{i\in[n]}, (\mathsf{id}_{j_i})_{i\in[n]}, \tau)$ and return $\Lambda \leftarrow (\mathsf{Eval}(f, (m_{j_i})_{i\in[n]}), \mu, \tau)$, where $\mu \leftarrow \prod_{i\in[n]} \mu_i^{\omega_i}$.

The oracles SG, SKey, and SR are simulated honestly. If $\mathcal{A}$ eventually outputs a forgery $(\Lambda^\star, \mathsf{ID}^\star, f^\star)$ the reduction $\mathcal{R}$ can parse $\Lambda^\star$ as $(\vec{m}, \mu, \tau)$, forward $(\vec{m}, \mu, \tau, \mathsf{ID}^\star, f^\star)$ to $\mathcal{C}$ and wins the unforgeability game with the same probability as $\mathcal{A}$ breaks HPRA unforgeability. □

Lemma 5.35. *If the eBCDH assumption holds, then Scheme 5.8 is aggregator unforgeable in the ROM.*

Our proof is along the lines of [BFKW09], but under a slightly different, novel assumption, i.e., *eBCDH*.

Proof. We construct an efficient algorithm $\mathcal{R}$ which turns an efficient algorithm $\mathcal{A}$ breaking aggregator unforgeability into an efficient *eBCDH* solver. $\mathcal{R}$ internally maintains the lists $\mathtt{Rnd}$ and $\mathtt{H}$, which are initially empty. $\mathcal{R}$ obtains an extended bilinear CDH instance $eBCDH_\kappa \leftarrow (\mathsf{BG}, g^{1/\beta}, g^{\beta}, g^{\alpha/\beta}, g^{\gamma})$ relative to the security parameter κ and runs the following modified Gen algorithm Gen' to obtain PP.

$\mathsf{Gen}'(eBCDH_\kappa, \ell)$: For $i \in [\ell]$, choose $\mathtt{Rnd}[i] \leftarrow (s_i, t_i) \xleftarrow{R} \mathbb{Z}_q^2$ and set $g_i \leftarrow (g^\gamma)^{s_i} g^{t_i}$. Fix $H : \mathbb{Z}_q \to \mathbb{G}$ and return $\mathsf{PP} \leftarrow (\mathsf{BG}, H, (g_i)_{i\in[\ell]}, \ell)$

Then, $\mathcal{R}$ starts $\mathcal{A}$ on PP and $\mathsf{aux} = \emptyset$, where the oracles are simulated as follows and we assume that all oracles have implicit access to the state of $\mathcal{R}$ (in particular to $\mathtt{Rnd}$ and $\mathtt{H}$):

$H(x)$: If $x \in \mathtt{H}$ return $\mathtt{H}[x][1]$. Otherwise choose $(\rho, \nu) \xleftarrow{R} \mathbb{Z}_q^2$, set $\mathtt{H}[x] \leftarrow ((g^\gamma)^\rho g^\nu, \rho, \nu)$ and return $\mathtt{H}[x][1]$.

$\mathsf{SG}(i)$: Choose $\xi \xleftarrow{R} \mathbb{Z}_q^*$, and set $\mathtt{SK}[i] \leftarrow (\xi, (g^\beta)^\xi, (g^{1/\beta})^{1/\xi})$. Then, return $((g^\beta)^\xi, (g^{1/\beta})^{1/\xi})$.

$\mathsf{Sig}((j_i)_{i\in[n]}, (\vec{m}_{j_i})_{i\in[n]})$: As the original oracle, except: choose $\tau \xleftarrow{R} \mathbb{Z}_q$ and for all $i \in [n]$ choose $\nu_i \xleftarrow{R} \mathbb{Z}_q$. If $\tau || \mathtt{SK}[j_i][2] \in \mathtt{H}$ for any $i \in [n]$ abort. Otherwise, for all $i \in [n]$, set $\rho_i \leftarrow -\sum_{k\in[\ell]} \mathtt{Rnd}[k][1] \cdot m_{j_i}[k]$, and $\mathtt{H}[\tau || \mathtt{SK}[j_i][2]] \leftarrow ((g^\gamma)^{\rho_i} g^{\nu_i}, \rho_i, \nu_i)$, compute $\delta \leftarrow \nu_i + \sum_{k\in[\ell]} m_{j_i}[k] \cdot \mathtt{Rnd}[k][2]$ and return $(\sigma_{j_i} \leftarrow \mathtt{SK}[j_i][2]^\delta)_{i\in[n]}$ and τ.

$\mathsf{VR}(i)$: If $\mathtt{SK}[i] = \bot \ \vee \ \mathtt{RK}[i] = \bot \ \vee \ \mathtt{AK}[i] \neq \bot$ return $\bot$. Otherwise, set $\mathtt{AK}[i] \leftarrow (g^{\alpha/\beta})^{1/\mathtt{SK}[i][1]}$.

The oracles SR and VRKey are simulated honestly. If $\mathcal{A}$ eventually outputs a valid forgery $(\Lambda^\star, \mathsf{ID}^\star, f^\star) = ((\vec{m}^\star, \mu^\star, \tau^\star), \mathsf{ID}^\star, f^\star)$ with $f^\star = (\omega_i^\star)_{i\in[n]}$, we know that it is of the form

$$\begin{aligned}\mu^\star =& \big(e(\prod_{i\in[\ell]} g_i^{m_i^\star}, g) \cdot \prod_{i\in[n]} e(g^{\omega_i^\star}, H(\tau^\star||\mathsf{id}_i^\star))\big)^\alpha = \\ & (\mathbf{g}^{\alpha\gamma})^{\sum_{i\in[\ell]} m_i^\star \cdot \mathtt{Rnd}[i][1]} (\mathbf{g}^{\alpha})^{\sum_{i\in[\ell]} m_i^\star \cdot \mathtt{Rnd}[i][2]} \cdot \\ & (\mathbf{g}^{\alpha\gamma})^{\sum_{i\in[n]} \omega_i^\star \cdot \mathtt{H}[\tau^\star||\mathsf{pk}_i^\star][2]} (\mathbf{g}^{\alpha})^{\sum_{i\in[n]} \omega_i^\star \cdot \mathtt{H}[\tau^\star||\mathsf{pk}_i^\star][3]}.\end{aligned}$$

Then we let

$$\begin{aligned}\psi &= \sum_{i\in[\ell]} m_i^\star \cdot \mathtt{Rnd}[i][1] + \sum_{i\in[n]} \omega_i^\star \cdot \mathtt{H}[\tau^\star||\mathsf{pk}_i^\star][2], \\ \upsilon &= \sum_{i\in[\ell]} m_i^\star \cdot \mathtt{Rnd}[i][2] + \sum_{i\in[n]} \omega_i^\star \cdot \mathtt{H}[\tau^\star||\mathsf{pk}_i^\star][3],\end{aligned}$$

and output $\mathbf{g}^{\alpha\gamma} \leftarrow (\mu^\star \cdot e(g^\beta, g^{\alpha/\beta})^{-\upsilon})^{1/\psi}$ as $eBCDH$ solution. The simulation is negligibly close to the original game: all values are identically distributed; the signature is uniquely determined by the responses of the random oracle and the key, and, thus, also identically distributed as a real signature. Collisions in the answers of the random oracle only occur with negligible probability. Consequently, also an abort happens with negligible probability. What remains is to analyze the probability that the forgery output by $\mathcal{A}$ is of the form that we actually extract $\mathbf{g}^{\alpha\gamma}$, i.e., we have that $\psi \neq 0$ which we define as event E. By the same argumentation as [BFKW09] we obtain that $\Pr[\neg E] = 1/q$ which concludes the proof (cf. [BFKW09, Proof of Theorem 6] for a more detailed probability analysis). □

Lemma 5.36. *Scheme 5.8 is input private.*

Proof. Both input privacy ensembles are identical. □

5.4.3 Adding Output Privacy

An additional goal is that the aggregator neither learns the input nor the output (output privacy). On our way to achieve this, we formally define the notion of homomorphic proxy-re encryption (HPRE) and develop an instantiation for the family of linear function classes $\{\mathcal{F}_{\mathsf{PP}}^{\mathsf{lin}}\}$. Based on this, we extend Scheme 5.8 to additionally provide output privacy.

Homomorphic Proxy Re-Encryption. A homomorphic proxy re-encryption scheme (HPRE) is a PRE which additionally allows the homomorphic evaluation of functions on the ciphertexts. This functionality firstly allows to aggregate messages encrypted under the same public key, and, secondly, to transform the ciphertext holding the evaluation of a function to a ciphertext for another entity, when given the respective proxy re-encryption key. We stress that if the initial ciphertexts are with respect to different public keys, then one can use the respective re-encryption keys to transform them to a common public key before evaluating the function. More formally:

Definition 5.33. *A* HPRE *for a family of function classes* $\{\mathcal{F}_{\mathsf{PP}}\}$ *is a* PRE *with an additional evaluation algorithm* Eval.

$\mathsf{Eval}(\mathsf{PP}, f, \vec{c})$: *This algorithm takes public parameters* PP*, a function* $f \in \mathcal{F}_{\mathsf{PP}}$*, and a vector of ciphertexts* $\vec{c} = (c_i)_{i\in[n]}$ *to messages* $(m_i)_{i\in[n]}$ *all under public key* pk*, and outputs a ciphertext* c *to message* $f((m_i)_{i\in[n]})$ *under* pk*.*

Additionally, we require the following compactness notion (analogous to [CF15]).

Definition 5.34 (Compactness)**.** *A* HPRE *for a family of function classes* $\{\mathcal{F}_{\mathsf{PP}}\}$ *is called compact if for all* $\mathsf{PP} \leftarrow \mathsf{PGen}(1^\kappa)$ *and for all* $f \in \mathcal{F}_{\mathsf{PP}}$ *the running time of the algorithms* $\vec{\mathsf{Dec}}$ *is bounded by a fixed polynomial in the security parameter* κ*.*

Besides the straightforward adoption of correctness, IND-CPA$^-$ remains identical (Eval is a public algorithm). However, we require an IND-CPA$^-$ variant, where the adversary may adaptively choose the targeted user. To the best of our knowledge, such a notion does not exist for PRE. We introduce such a notion (termed mt-IND-CPA$^-$) and show that it is implied by the conventional IND-CPA notions.

Definition 5.35 (mt-IND-CPA$^-$)**.** *A* (H)PRE *is* mt-IND-CPA$^-$ *secure, if for all PPT adversaries* $\mathcal{A}$ *there is a negligible function* $\varepsilon(\cdot)$ *such that*

$$\Pr\left[\begin{array}{l} \mathsf{PP} \leftarrow \mathsf{PGen}(1^\kappa), b \xleftarrow{R} \{0,1\}, \\ (\mathsf{sk}_h, \mathsf{pk}_h) \leftarrow \mathsf{KeyGen}(\mathsf{PP}), \mathcal{O} \leftarrow \{\mathsf{G}(\cdot), \mathsf{RG}(\cdot)\}, \\ (m_0, m_1, i^\star, \mathsf{st}) \leftarrow \mathcal{A}^{\mathcal{O}}(\mathsf{PP}, \mathsf{pk}_h), \\ c \leftarrow \mathsf{Enc}^2(m_b, \mathsf{pk}_{i^\star}), b^\star \leftarrow \mathcal{A}(\mathsf{st}, c) \end{array} : \; b = b^\star \right] \leq 1/2 + \varepsilon(\kappa),$$

where the environment holds an initially empty list HU*.* G *and* RG *are defined as:*

$\mathsf{G}(i)$: *If* $\mathtt{HU}[i] \neq \perp$ *return* $\perp$*. Otherwise, run* $(\mathsf{sk}_i, \mathsf{pk}_i) \leftarrow \mathsf{KeyGen}(\mathsf{PP})$*, set* $\mathtt{HU}[i] \leftarrow (\mathsf{sk}_i, \mathsf{pk}_i)$*, and return* pk_i*.*

$\mathsf{RG}(i)$: *If* $\mathtt{HU}[i] = \perp$ *return* $\perp$*. Otherwise, set* $\mathsf{rk}_{i\to h} \leftarrow \mathsf{ReGen}(\mathtt{HU}[i][1], \mathsf{pk}_h)$ *and return* $\mathsf{rk}_{i\to j}$*.*

Lemma 5.37. *Every* IND-CPA$^-$ *(and thus every* IND-CPA*) secure* PRE *also satisfies* mt-IND-CPA$^-$ *security.*

Proof. We prove the lemma by bounding the success probability in the mt-IND-CPA$^-$ game relative to the IND-CPA$^-$ bound. Therefore, we let q_{G} be the number of queries to the G oracle.

Game 0: The mt-IND-CPA$^-$ game.

Game 1: As Game 0, but we guess the index $i^\star$ beforehand. If our guess is wrong we abort.

Transition$^{0\to1}$: We have that $\Pr[S_0] = q_{\mathsf{G}} \cdot \Pr[S_1]$.

Game 2: As Game 1, but we engage with an IND-CPA′ challenger, obtain $(\mathsf{pp}, \mathsf{pk}_t, \mathsf{pk}_h, \mathsf{rk}_{t\to h})$, set $\mathtt{HU}[i^\star] \leftarrow (\bot, \mathsf{pk}_t, \mathsf{rk}_{t\to h})$ and start $\mathcal{A}$ on $(\mathsf{pp}, \mathsf{pk}_h)$. Furthermore, we simulate the oracles when queried for user $i^\star$ as follows:

$\mathsf{G}(i^\star)$: Return $\mathtt{HU}[i^\star][2]$ (i.e., pk_t).

$\mathsf{RG}(i^\star)$: Return $\mathtt{HU}[i^\star][3]$ (i.e., $\mathsf{rk}_{t\to h}$).

The oracle calls for the remaining indexes are simulated honestly.

Transition$^{1\to2}$: This change is conceptual.

Whenever an adversary in Game 2 outputs its guess $b^\star$, we can forward $b^\star$ to the IND-CPA$^-$ challenger and win whenever the adversary wins Game 2. As we have that $\Pr[S_1] = \Pr[S_2] \leq \varepsilon_{\mathsf{cpa}^-}(\kappa)$, and, thus, $\Pr[S_0] \leq q_{\mathsf{G}} \cdot \varepsilon_{\mathsf{cpa}^-}(\kappa)$, this concludes the proof. □

HPRE **Construction for the Family of Linear Function Classes.** We state our construction in Scheme 5.9. Essentially, we build on the PRE scheme in [AFGH06, third attempt] and turn it into a HPRE for the family of linear function classes $\{\mathcal{F}^{\mathsf{lin}}_{\mathsf{PP}}\}$, henceforth referred to as $\mathsf{HPRE}_{\mathsf{lin}}$. For the desired homomorphism we use a standard trick in the context of ElGamal-like encryption schemes: we encode messages $m \in \mathbb{Z}_q$ into the exponent and encrypt $\mathbf{g}^m$. Decryption then yields $m' = \mathbf{g}^m$ and one additionally needs to compute $m = \log_{\mathbf{g}} m'$ to obtain m. Thus, for the schemes to remain efficient, the size of the message space needs to be polynomial in the security parameter. While this might sound quite restrictive, we stress that in practical settings one deals with numerical values where messages in the order of millions to billions are by far sufficient. Thus, this type of decryption is not a limitation and entirely practical.

As Eval is a public algorithm it does not influence IND-CPA security. Thus, our argumentation is identical to [AFGH06] and we can use the following theorem.

Theorem 5.10 (cf. [AFGH06])**.** *If the eDBDH assumption holds in* $(\mathbb{G}, \mathbb{G}_T)$ *then Scheme 5.9 is an* IND-CPA *secure* $\mathsf{HPRE}_{\mathsf{lin}}$.

We also note that compactness of Scheme 5.9 (Definition 5.34) is easy to verify.

$\underline{\mathsf{PGen}(1^\kappa)}$: Run $\mathsf{BG} \leftarrow \mathsf{BGGen}(1^\kappa)$, and return $\mathsf{pp} \leftarrow \mathsf{BG}$.

$\underline{\mathsf{KeyGen}(\mathsf{pp})}$: Choose $(a_1, a_2) \xleftarrow{R} \mathbb{Z}_q^2$, and return $(\mathsf{rsk}_A, \mathsf{rpk}_A) \leftarrow ((a_1, a_2), (\mathbf{g}^{a_1}, g^{a_2}))$.

$\underline{\mathsf{ReGen}(\mathsf{rsk}_A, \mathsf{rpk}_B)}$: Parse rsk_A as $(a_{1A}, \cdot)$ and rpk_B as $(\cdot, g^{a_{2B}})$ and return $\mathsf{rk}_{A\to B} \leftarrow (g^{a_{2B}})^{a_{1A}}$.

$\underline{\mathsf{Enc}^1(\mathsf{rpk}, m)}$: Parse rpk as $(\mathbf{g}^{a_1}, \cdot)$, choose $k \xleftarrow{R} \mathbb{Z}_q$, and return $c \leftarrow (\mathbf{g}^k, \mathbf{g}^m \cdot (\mathbf{g}^{a_1})^k, 1)$

$\underline{\mathsf{Enc}^2(\mathsf{rpk}, m)}$: Parse rpk as $(\mathbf{g}^{a_1}, \cdot)$, choose $k \xleftarrow{R} \mathbb{Z}_q$, and return $c \leftarrow (g^k, \mathbf{g}^m \cdot (\mathbf{g}^{a_1})^k, 2)$

$\underline{\mathsf{ReEnc}(\mathsf{rk}_{A\to B}, c_A)}$: Parse c_A as $(c_1, c_2, 2)$ and return $c \leftarrow (e(c_1, \mathsf{rk}_{A\to B}), c_2, R)$

$\underline{\mathsf{Dec}^1(\mathsf{rsk}, c)}$: Parse c as (c_1, c_2, c_3) and rsk as (a_1, a_2), and return $\mathbf{g}^m \leftarrow c_2 \cdot c_1^{-a_1}$ if $c_3 = 1$ and $\mathbf{g}^m \leftarrow c_2 \cdot c_1^{-1/a_2}$ if $c_3 = R$.

$\underline{\mathsf{Dec}^2(\mathsf{rsk}, c)}$: Parse c as $(c_1, c_2, 2)$ and rsk as (a_1, a_2), and return $\mathbf{g}^m \leftarrow c_2 \cdot e(g, c_1^{-a_1})$.

$\underline{\mathsf{Eval}(\mathsf{pp}, f, \vec{c})}$: Parse f as $(\omega_1, \ldots, \omega_n)$ and $\vec{c}$ as $(c_i)_{i\in[n]}$, and return $c \leftarrow \prod_{i\in[n]} c_i^{\omega_i}$, where multiplication and exponentiation is component-wise.

Scheme 5.9: $\mathsf{HPRE}_{\mathsf{lin}}$ based on [AFGH06, third attempt].

$\mathsf{HPRE}_{\mathsf{lin}}$ for Vectors. We extend $\mathsf{HPRE}_{\mathsf{lin}}$ to vectors over $\mathbb{Z}_q$, while preserving the support for re-encryption and the homomorphic properties. It turns out that we can employ a communication efficient solution. That is, borrowing the idea of randomness re-use from [BBKS07] and applying it to $\mathsf{HPRE}_{\mathsf{lin}}$, we can reduce the size of the ciphertexts as long as no re-encryption is performed. Upon setup, we have to fix a maximum length ℓ of the message vectors, which needs to be additionally provided to the PGen algorithm. The secret and the public keys are then of the form $\mathsf{rsk} \leftarrow (\mathsf{rsk}_i)_{i\in[\ell]} = ((a_{1i}, a_{2i}))_{i\in[\ell]}$, $\mathsf{rpk} \leftarrow (\mathsf{rpk}_i)_{i\in[\ell]} = ((\mathbf{g}^{a_{1i}}, g^{a_{2i}}))_{i\in[\ell]}$, where $(a_{1i}, a_{2i})_{i\in[\ell]} \xleftarrow{R} (\mathbb{Z}_q^2)^\ell$. First and second level encryption are defined as

$$\mathsf{Enc}^1_\ell(\mathsf{rpk}, \vec{m}) \coloneqq (\mathbf{g}^k, (\mathbf{g}^{m_i} \cdot \mathsf{rpk}_i[1]^k)_{i\in[\ell]}, 1), \text{ and}$$
$$\mathsf{Enc}^2_\ell(\mathsf{rpk}, \vec{m}) \coloneqq (g^k, (\mathbf{g}^{m_i} \cdot \mathsf{rpk}_i[1]^k)_{i\in[\ell]}, 2), \text{ respectively.}$$

Decryption $\mathsf{Dec}^j_\ell(\cdot, \cdot)$ of a ciphertext $(c[1], (c[i+1])_{i\in[\ell]}, j)$ is defined as

$$\mathsf{Dec}^1_\ell(\mathsf{rsk}, \vec{c}) \coloneqq (c[i+1] \cdot c[1]^{-\mathsf{rsk}_i[1]})_{i\in[\ell]}, \text{ and}$$
$$\mathsf{Dec}^2_\ell(\mathsf{rsk}, \vec{c}) \coloneqq (c[i+1] \cdot e(c[1], g^{-\mathsf{rsk}_i[1]}))_{i\in[\ell]}.$$

Re-encryption key generation is defined as

$$\mathsf{ReGen}_\ell(\mathsf{rsk}_A, \mathsf{rpk}_B) \coloneqq (((\mathsf{rpk}_B)_i[2])^{(\mathsf{rsk}_A)_i[1]})_{i\in[\ell]}.$$

From a second level ciphertext $\vec{c}_A$ for A and a re-encryption key $\mathsf{rk}_{A\to B}$, a ciphertext $\vec{c}_B$ for B is computed as

$$\vec{c}_B \leftarrow \mathsf{ReEnc}(\mathsf{rk}_{A\to B}, \vec{c}_A) \coloneqq ((e(c_A[1], \mathsf{rk}_{A\to B}[i]), c_A[i+1]))_{i\in[\ell]}.$$

Note that re-encrypted ciphertexts have a different form. Thus we do not need to add the level as suffix. Decryption $\mathsf{Dec}^1_\ell(\cdot,\cdot)$ for re-encrypted ciphertexts is

$$\mathsf{Dec}^1_\ell(\mathsf{rsk},(c_i)_{i\in[\ell]}) \coloneqq (c_i[2]\cdot c_i[1]^{-1/\mathsf{rsk}_i[2]})_{i\in[\ell]}.$$

Theorem 5.11. *If the eDBDH assumption holds, then the extension of* $\mathsf{HPRE}_{\mathsf{lin}}$ *as described above, yields an* IND-CPA *secure* $\mathsf{HPRE}_{\mathsf{lin}}$ *for vectors.*

Proof (Sketch). IND-CPA security of the original scheme implies Theorem 5.11 under a polynomial loss: using ℓ hybrids, where in hybrid i ($1 \le i \le \ell$) the i-th ciphertext component is exchanged by random under the original strategy in [AFGH06].

Combining the theorem above with Lemma 5.37 yields:

Corollary 5.6. *The extension of* $\mathsf{HPRE}_{\mathsf{lin}}$ *as described above yields an* mt-IND-CPA$^-$ *secure* $\mathsf{HPRE}_{\mathsf{lin}}$ *for vectors.*

Putting the Pieces Together: Output Privacy. Our idea is to combine Scheme 5.8 with the $\mathsf{HPRE}_{\mathsf{lin}}$ presented above. In doing so, we face some obstacles. First, a naïve combination of those primitives does not suit our needs: one can still verify guesses for signed messages using solely the signatures, since signatures are publicly verifiable. Second, switching to a MAC for the data sources is also no option, as this would require an interactive re-key generation. This is excluded by our model as we explicitly want to avoid it. Thus, we pursue a different direction and turn the signatures used in Scheme 5.8 into a MAC-like primitive by blinding a signature with a random element g^r. An aggregated MAC holding an evaluation of f is then blinded by $g^{f(\dots,r,\dots)}$, i.e., the receiver needs to evaluate the function f on all the blinding values from the single sources. Now the question arises as how to transmit the blinding values to the receiver. Using our $\mathsf{HPRE}_{\mathsf{lin}}$ for vectors yields an arguably elegant solution: by treating the randomness as an additional vector component, we can use the re-encryption features of the $\mathsf{HPRE}_{\mathsf{lin}}$. More importantly, by executing the $\mathcal{EV}$ algorithm the aggregator simultaneously evaluates the function f on the data and on the randomness so that the receiver can directly obtain the blinding value $f(\dots,r,\dots)$ upon decryption.

Note on the Instantiation. Augmenting Scheme 5.8 to obtain Scheme 5.10 using $\mathsf{HPRE}_{\mathsf{lin}}$ requires an alternative decryption strategy for the vector component containing r, as r is uniformly random in $\mathbb{Z}_q$ and can thus not be efficiently recovered. Fortunately, obtaining $r \in \mathbb{Z}_q$ is not required, as g^r (resp. $\mathbf{g}^r$) is sufficient to unblind the signature (resp. MAC). Those values are efficiently recoverable.

Theorem 5.12. *If Scheme 5.8 is signer and aggregator unforgeable, and* $\mathsf{HPRE}_{\mathsf{lin}}$ *for vectors is* mt-IND-CPA$^-$ *secure, then Scheme 5.10 is a signer and aggregator unforgeable, input and output private* HPRA *for the family of linear function classes* $\{\mathcal{F}^{\mathsf{lin}}_{\mathsf{PP}}\}$.

$\underline{\mathsf{Gen}(1^\kappa, \ell)}$: Fix a homomorphic $\mathsf{PRE} = (\mathsf{Gen}, \mathsf{KeyGen}, \vec{\mathsf{Enc}}, \vec{\mathsf{Dec}}, \mathsf{ReGen}, \mathsf{ReEnc}, \mathsf{Eval})$ for class $\mathcal{F}_{\mathsf{lin}}$ and the $\mathsf{HPRA}(\mathsf{Eval}) = (\mathbf{Gen}, \mathbf{SGen}, \mathbf{VGen}, \mathbf{Sign}, \mathbf{Verify}, \mathbf{SRGen}, \mathbf{VRGen}, \mathbf{Agg}, \mathbf{AVerify})$ from Scheme 5.8 such that $\mathcal{M}_{\mathsf{PRA}} \subseteq \mathcal{M}_{\mathsf{PRE}}$, run $\mathsf{pp}_s \leftarrow \mathbf{Gen}(1^\kappa, \ell)$, $\mathsf{pp}_e \leftarrow \mathsf{PGen}(1^\kappa, \ell + 1)$, and return $\mathsf{pp} \leftarrow (\mathsf{pp}_s, \mathsf{pp}_e)$.

$\underline{\mathsf{SGen}(\mathsf{pp})}$: Run $(\mathbf{id}, \mathbf{sk}, \mathbf{pk}) \leftarrow \mathbf{SGen}(\mathsf{pp}_s)$, $(\mathsf{rsk}, \mathsf{rpk}) \leftarrow \mathsf{KeyGen}(\mathsf{pp}_e)$, and return $(\mathsf{id}, \mathsf{sk}, \mathsf{pk}) \leftarrow (\mathbf{id}, (\mathbf{sk}, \mathsf{rsk}, \mathsf{rpk}), \mathbf{pk})$.

$\underline{\mathsf{VGen}(\mathsf{pp})}$: Run $(\mathbf{mk}, \mathbf{aux}) \leftarrow \mathbf{VGen}(\mathsf{pp}_s)$, $(\mathsf{rsk}, \mathsf{rpk}) \leftarrow \mathsf{KeyGen}(\mathsf{pp}_e)$, and return $(\mathsf{mk}, \mathsf{aux}) \leftarrow ((\mathbf{mk}, \mathsf{rsk}), (\mathbf{aux}, \mathsf{rpk}))$.

$\underline{\mathsf{Sign}(\mathsf{sk}, \vec{m}, \tau)}$: Parse sk as $(\mathbf{sk}, \cdot, \mathsf{rpk})$, choose $r \xleftarrow{R} \mathbb{Z}_q$, and return $\sigma \leftarrow (\sigma' \cdot g^r, \vec{c})$, where

$$(\sigma', \cdot) \leftarrow \mathbf{Sign}(\mathbf{sk}, \vec{m}, \tau) \text{ and } \vec{c} \leftarrow \mathsf{Enc}^2_{\ell+1}(\mathsf{rpk}, \vec{m}||r).$$

$\underline{\mathsf{SRGen}(\mathsf{sk}_i, \mathsf{aux})}$: Parse sk_i as $(\mathbf{sk}_i, \mathsf{rsk}_i, \mathsf{rpk}_i)$ and aux as $(\mathbf{aux}, \mathsf{rpk})$. Obtain $\mathbf{rk}_i \leftarrow \mathbf{SRGen}(\mathbf{sk}_i, \mathbf{aux})$ and $\mathsf{prk}_i \leftarrow \mathsf{ReGen}(\mathsf{rsk}_i, \mathsf{rpk})$, and return $\mathsf{rk}_i \leftarrow (\mathbf{rk}_i, \mathsf{prk}_i)$.

$\underline{\mathsf{VRGen}(\mathsf{pk}_i, \mathsf{mk}, \mathsf{rk}_i)}$: Parse pk_i as $\mathbf{pk}_i$ and mk as $(\mathbf{mk}, \cdot)$, obtain $\mathbf{ak}_i \leftarrow \mathbf{VRGen}(\mathbf{pk}_i, \mathbf{mk})$ and return $\mathsf{ak}_i \leftarrow (\mathbf{ak}_i, \mathsf{rk}_i)$.

$\underline{\mathsf{Agg}((\mathsf{ak}_i)_{i\in[n]}, (\sigma_i)_{i\in[n]}, \tau, f)}$: For $i \in [n]$ parse ak_i as $(\mathbf{ak}_i, (\mathbf{rk}_i, \mathsf{prk}_i))$, σ_i as $(\sigma'_i, \vec{c}_i)$. Output $\Lambda \leftarrow (\vec{c}', \mu, \tau)$, where

$$(\vec{c}'_i \leftarrow \mathsf{ReEnc}(\mathsf{prk}_i, \vec{c}_i))_{i\in[n]},\ (\vec{c}', \mu, \tau) \leftarrow \mathbf{Agg}((\mathbf{ak}_i)_{i\in[n]}, (\sigma'_i, \vec{c}'_i)_{i\in[n]}, f).$$

$\underline{\mathsf{AVerify}(\mathsf{mk}, \Lambda, \mathsf{ID}, f)}$: Parse mk as $(\mathbf{mk}, \mathsf{rsk})$ and Λ as $(\vec{c}, \mu, \tau)$, obtain $\vec{m}'||r \leftarrow \mathsf{Dec}^1_{\ell+1}(\mathsf{rsk}, \vec{c})$ and return $(\vec{m}, \tau)$ if the following holds, and $(\perp, \perp)$ otherwise:

$$\mathbf{AVerify}(\mathbf{mk}, (\vec{m}, \mu \cdot (\mathbf{g}^r)^{-1}, \tau), \mathsf{ID}, f) = 1$$

Scheme 5.10: Output private HPRA scheme for the family of linear function classes $\{\mathcal{F}^{\mathsf{lin}}_{\mathsf{PP}}\}$ with $\mathsf{Eval}(\cdot,\cdot) \coloneqq \mathsf{HPRE.Eval}(\mathsf{pp_e}, \cdot, \cdot)$

We prove Theorem 5.12 by proving Lemma 5.38-5.40.

Lemma 5.38. *If Scheme 5.8 is signer and aggregator unforgeable, then Scheme 5.10 is so as well.*

Proof. Additional encryption does not influence unforgeability. □

Lemma 5.39. *Scheme 5.10 is input private.*

Proof. Both input privacy ensembles are identical. □

Lemma 5.40. *If* $\mathsf{HPRE}_{\mathsf{lin}}$ *for vectors is* $\mathsf{mt\text{-}IND\text{-}CPA}^-$ *secure, then Scheme 5.10 is output private.*

Proof. We prove output privacy using a sequence of games, and let q_{RoS} be the cumulative number of signing calls within the RoS queries.

Game 0: The original privacy game with bit $b = 0$.

Game $1_i (1 \leq i < q_{\mathsf{RoS}})$**:** As the previous game, but we modify i-th Sign run within RoS as follows:

$\mathsf{Sign}(\mathsf{sk}, \vec{m}, \tau)$: Parse sk as $(\mathbf{sk}, \mathsf{rsk}, \mathsf{rpk})$, choose $r \xleftarrow{R} \mathbb{Z}_q$, $\boxed{\text{choose } \vec{\rho} \xleftarrow{R} \mathbb{Z}_q^{\ell+1}}$, and return $\sigma \leftarrow (\sigma' \cdot g^r, \vec{c})$, where $(\sigma', \cdot) \leftarrow \mathbf{Sign}(\mathbf{sk}, \vec{m}, \tau)$ and $\vec{c} \leftarrow \mathsf{Enc}^2_{\ell+1}(\mathsf{rpk}, \boxed{\vec{\rho}})$.

Transition$^{1_i \to 1_{i+1}}$**:** We show that the success probability of a distinguisher $\mathsf{D}^0 \to \mathsf{D}^{1_1}$ (resp. $\mathsf{D}^{1_i \to 1_{i+1}}$) is bounded by $\varepsilon_{\mathsf{mt\text{-}cpa}^-}(\kappa)$. In doing so, we present a hybrid game, which, based on the bits chosen by the challengers, interpolates between Game 0 and Game 1_1 (resp. Game 1_i and Game 1_{i+1}). Thereby, we use $\mathcal{C}_\kappa$ to denote an $\mathsf{mt\text{-}IND\text{-}CPA}^-$ challenger for $\mathsf{HPRE}_{\mathsf{lin}}$ for vectors. Firstly, we run the modified Gen algorithm Gen':

$\mathsf{Gen}'(1^\kappa, \ell)$: Run $\mathsf{PP}_s \leftarrow \mathbf{Gen}(1^\kappa, \ell)$, obtain $\boxed{(\mathsf{PP}_e, \mathsf{pk}_h) \leftarrow \mathcal{C}_\kappa}$, store pk_h and return $\mathsf{PP} \leftarrow (\mathsf{PP}_s, \mathsf{PP}_e)$.

Secondly, we run the modified VGen algorithm VGen':

$\mathsf{VGen}'(\mathsf{PP})$: Run $(\mathbf{mk}, \mathbf{aux}) \leftarrow \mathbf{VGen}(\mathsf{PP}_s)$, $\boxed{\mathsf{rpk} \leftarrow \mathsf{pk}_h}$, set $\boxed{\mathsf{rsk} \leftarrow \bot}$. Return $(\mathsf{mk}, \mathsf{aux}) \leftarrow ((\mathbf{mk}, \mathsf{rsk}), (\mathbf{aux}, \mathsf{rpk}))$.

Thirdly, we modify the oracles SG as well as SR.

$\mathsf{SG}(i)$: If $\mathtt{SK}[i] \neq \bot$ return $\bot$. Otherwise, run $(\mathsf{id}_i, \mathsf{sk}_i, \mathsf{pk}_i) \leftarrow \mathsf{SGen}'(\mathsf{PP}, i, \mathtt{CU})$, set $\mathtt{SK}[i] \leftarrow (\mathsf{id}_i, \mathsf{sk}_i, \mathsf{pk}_i)$ and return $(\mathsf{id}_i, \mathsf{pk}_i)$, where SGen' is defined as follows:

- $\mathsf{SGen}'(\mathsf{PP}, i, \mathtt{CU})$: Run $(\mathbf{id}, \mathbf{sk}, \mathbf{pk}) \leftarrow \mathbf{SGen}(\mathsf{PP}_s)$. If $i \in \mathtt{CU}$, run $(\mathsf{rsk}, \mathsf{rpk}) \leftarrow \mathsf{KeyGen}(\mathsf{PP}_e)$. Otherwise, obtain $\mathsf{rpk} \leftarrow \mathcal{C}_\kappa.\mathsf{G}(i)$, set $\mathsf{rsk} \leftarrow \bot$. In any case set $(\mathsf{id}, \mathsf{sk}, \mathsf{pk}) \leftarrow (\mathbf{id}, (\mathbf{sk}, \mathsf{rsk}, \mathsf{rpk}), \mathbf{pk})$.

$\mathsf{SR}(i)$: If $\mathtt{SK}[i] = \bot \vee \mathtt{RK}[i] \neq \bot$ return $\bot$. Otherwise, if $i \in \mathtt{CU}$ set $\mathtt{RK}[i] \leftarrow \mathsf{SRGen}(\mathtt{SK}[i][2], \mathsf{aux})$ and return $\mathtt{RK}[i]$. If $i \notin \mathtt{CU}$ simulate the oracle as for $i \in \mathtt{CU}$, but with the following modified SRGen algorithm SRGen':

- $\mathsf{SRGen}'(\mathsf{sk}_i, \mathsf{aux})$: Parse sk_i as $(\mathbf{sk}_i, \bot, \mathsf{rpk}_i)$ and aux as $(\mathbf{aux}, \mathsf{rpk})$. Return $\mathsf{rk}_i \leftarrow (\mathbf{rk}_i, \mathsf{prk}_i)$, where $\mathbf{rk}_i \leftarrow \mathbf{SRGen}(\mathbf{sk}_i, \mathbf{aux})$, and $\mathsf{prk}_i \leftarrow \mathcal{C}_\kappa.\mathsf{RG}(i)$.

Finally, we modify the i-th Sign run within RoS as follows:

$\mathsf{Sign}(\mathsf{sk}_i, \vec{m}, \tau)$: Parse sk_i as $(\mathbf{sk}_i, \bot, \mathsf{rpk}_i)$, choose $r \xleftarrow{R} \mathbb{Z}_q$, choose $\vec{\rho} \xleftarrow{R} \mathbb{Z}_q^{\ell+1}$, and return $\sigma \leftarrow (\sigma' \cdot g^r, \vec{c})$, where $(\sigma', \cdot) \leftarrow \mathbf{Sign}(\mathbf{sk}_i, \vec{m}, \tau)$, and $\vec{c} \leftarrow \mathcal{C}_\kappa(\vec{m}||r, \vec{\rho}, i)$.

Here, $\vec{c} \leftarrow \mathcal{C}_\kappa(\vec{m}||r, \vec{\rho}, i)$ denotes that a challenge ciphertext with respect to messages $\vec{m}||r$ and $\vec{\rho}$, and signer i is obtained from the challenger. The oracles SKey, VR, and VRKey, are simulated honestly. Now, the bit b chosen by $\mathcal{C}_\kappa$ switches between the distributions in Game i and Game $i + 1$.

In Game $1_{q_{\mathsf{RoS}}}$ all ciphertexts are encryptions of random values. The g^r's unconditionally hide the signed messages in the signatures (signatures can be viewed as Pedersen commitments under randomness r), i.e., signatures are distributed as signatures on random vectors, and we are in the game where $b = 1$; the distinguishing probability between the original Game 0 and Game $1_{q_{\mathsf{RoS}}}$ is negligible. □

6 Conclusion

In this thesis we cryptographically addressed several important questions in the context of privacy preserving processing of authenticated data. These questions gained more and more relevance in recent years due to many security and privacy related challenges arising from the following two developments. First, outsourcing computations to, and storing data at cloud computing providers, who are typically not fully trusted, became an increasingly followed trend. Second, we can observe a tremendous growth in the number of computing devices which allow us to ubiquitously access the distributed infrastructures and services emerging from this trend.

From a technical viewpoint, we improved upon the state of the art in various directions. We developed novel schemes and protocols as well as generalizations and extensions of existing schemes and protocols which are suitable to reduce the required trust assumptions in the parties being involved in processing the data, while simultaneously also taking privacy issues into account. Throughout the thesis we followed a modular approach which is also resembled by the structure of this thesis.

We started by establishing various basic primitives being well suited as building blocks in our constructions. However, we want to stress that they address more fundamental issues which may also be of interest for other areas in cryptography. In the context of cryptographic accumulators we introduced a comprehensive unified security model. We used this model to analyze the security of existing accumulator constructions, and also set accumulators being secure in this model into relation with other primitives. Most importantly, our security model includes the first meaningful formalization of indistinguishability, which is a central requirement in many applications. In this work we also presented novel constructions of accumulator schemes and proved them secure in our model. In

the area of key-homomorphic signatures we established a framework which allows us to gain further insights into the properties of plain signature schemes being required to use them in advanced signature primitives (including universal designated verifier signatures and ring signatures) and zero-knowledge argument systems with strong security properties. Besides being of theoretical interest it turns our that our constructions are surprisingly efficient and even yield to instantiations without random oracles which are favorable regarding efficiency when compared to existing work.

We then proposed a new group signature scheme which provides security in the strong BSZ model [BSZ05]. We were, thereby, able to make efficiency gains by using a paradigm to construct group signatures, which was previously unknown to be compatible with such a strong security model. Bottom line, we outperform all existing group signature schemes being proven secure in a comparably strong model and, in addition, even outperform the popular BBS short group signature scheme [BBS04] regarding signature size. As many applications of group signatures require particularly efficient signature generation, our results are an important step towards better acceptance of group signatures in practice. Compared to other schemes we use slightly stronger/riskier, yet very plausible assumptions.

Somewhat orthogonal to our results on group signatures, we presented several contributions in the area of malleable signature schemes. First, we proposed a unified framework for redactable signatures being independent of the concrete data structure to be signed. On top of that we introduced three generic constructions for different message structures. Our constructions are based on indistinguishable accumulators and other standard primitives, yielding various practically efficient instantiations. Second, we built upon this framework to extend redactable signatures with two additional important privacy features being motivated by the requirements of cloud-based document sharing scenarios: signer anonymity and designated verifiers. We formally modeled this primitive and presented two constructions. The first one is a generic construction which is more of theoretical interest, whereas the second one is a particularly efficient construction. The efficiency gains we made were obtained by combining our results on key-homomorphic signatures with our results on redactable signatures. Third, we focused on extensions of sanitizable signatures with a feature to limit the allowed modifications by the sanitizer to signer-defined sets of messages per block (`LimitSet`). We found that existing formalizations do not guarantee the privacy one would expect, as they do not require the sets of possible modifications to remain concealed. To this end we strengthened the security model with respect to this observation and showed how to generically extend plain sanitizable signatures to also support the `LimitSet` extension using (indistinguishable) accumulators. Finally, we turned to a variant of malleable signatures allowing to evaluate functions on authenticated data vectors. In our work, we combined the malleability properties with features known from proxy re-cryptography: an aggregator can transform data vectors authenticated under different keys of various data sources to data vectors authenticated under a receiver's key, while

at the same time being able to evaluate functions on the authenticated vectors. We introduced a model which formally captures our requirements and developed novel building blocks to eventually come up with a modular construction of our newly introduced primitive covering the class of linear functions.

Summing up, the work presented in this thesis can be seen as a framework which addresses important privacy issues arising when third parties, who are not fully trusted, process authenticated data. In our work we aimed to find a suitable tradeoff between computational efficiency and modularity. That is, while most of our constructions are generic, and thus flexible enough to be instantiated using different primitives and hardness assumptions, there are efficient instantiations of all our constructions which are well suited for deployment in practical applications.

6.1 Open Questions and Future Work

Finally, we shed light on open questions and interesting directions for future work. In general, we observe that malleability of signature schemes, or malleability of cryptographic objects in general, seems to be a useful tool to address various practical as well as theoretical issues one faces when designing cryptographic schemes and protocols with advanced features and/or strong security properties. So the broader future direction we aspire is to further investigate cryptographic objects with malleable properties.

Besides that, we want to briefly sketch open questions which are directly tangled with the work presented in this thesis. Some questions which remained open in our work were already addressed in follow-up work. For completeness, we also include these follow-up results (which were already discussed in more detail in the respective technical chapters) in our discussion below.

Cryptographic accumulators received quite some attention in the recent time and meanwhile there are multiple works which mainly address stronger variants of indistinguishability and more expressive set operations [RY16, GOP+16, ZKP17, BCD+17]. What is still open is to investigate further relationships of accumulators to other primitives with similar functionalities, in the fashion as we have investigated their relation to commitments and zero-knowledge sets. In the area of key-homomorphic signatures, we have already demonstrated their usefulness in a broad spectrum of applications. It would, nevertheless, be interesting to find further applications where key homomorphisms are useful to obtain compellingly efficient and simple schemes and protocols—especially schemes and protocols which come with security proofs without random oracles. In addition, it seems that malleability properties on the key space are also a valuable tool when it comes to achieving tightness in security proofs. It would thus be interesting to develop new techniques for tight security proofs being based on key homomorphisms.

Our work on group signatures centrally focuses on schemes with particularly efficient signature generation and verification, providing strong security guarantees. One open point for future work would be to weaken the required

underlying assumptions while still preserving the efficiency. Another question to investigate would be to which extent our construction paradigm is useful to achieve instantiations which do not require random oracles. Finally, while we informally sketched an extension to the stronger variant of opening soundness using our results on plaintext (in-)equality proofs in [c4], a formal treatment of this issue is still open.

In our work on extended privacy for redactable signatures, it remained open whether it is possible to achieve a stronger notion of simulatability where the adversary can adaptively query the simulation oracle (RoD). While an extension seems straight forward for our generic construction, we currently do not see how to adapt our construction which is based on key-homomorphisms. In the context of sanitizable signatures we have quite recently seen another efficient instantiation of plain unlinkable sanitizable signatures in the standard model [LZCS16]. We note that—likewise to the paradigm to obtain unlinkability in [FKM+16]—also the paradigm which is used in [LZCS16] is not helpful to obtain practically efficient unlinkable ESS. Accordingly it is still open to come up with a practically efficient construction of unlinkable ESS. In this context we observe similarities between unlinkable ESS and anonymous credential systems.[1] So a fruitful direction might be to study relations between them. Then, unlinkable ESS can also benefit from advances in the context of attribute based anonymous credential systems and vice versa. We, however, note that a notion like unlinkability will often be too strong for a primitive like ESS, since most real world documents will contain elements which allow to link the sanitized documents despite the unlinkability of the signature. Finally, in the context of homomorphic proxy re-encryption an instantiation for function classes beyond linear ones, as well as a construction in the standard model remained open.

[1] In fact, in [CL13] it was shown that anonymous credential schemes can be constructed from a variant of sanitizable signature schemes. However, they use a model which is tailored to showing this implication and does not consider all security properties of sanitizable signatures. Showing an implication in the other direction is open at all.

Bibliography

[AB09] Shweta Agrawal and Dan Boneh. Homomorphic macs: Mac-based integrity for network coding. In *Applied Cryptography and Network Security, 7th International Conference, ACNS 2009, Paris-Rocquencourt, France, June 2-5, 2009. Proceedings*, pages 292–305, 2009.

[ABC+12] Jae Hyun Ahn, Dan Boneh, Jan Camenisch, Susan Hohenberger, Abhi Shelat, and Brent Waters. Computing on authenticated data. In *Theory of Cryptography - 9th Theory of Cryptography Conference, TCC 2012, Taormina, Sicily, Italy, March 19-21, 2012. Proceedings*, pages 1–20, 2012.

[ABC+15] Jae Hyun Ahn, Dan Boneh, Jan Camenisch, Susan Hohenberger, Abhi Shelat, and Brent Waters. Computing on authenticated data. *J. Cryptology*, 28(2):351–395, 2015.

[ACdMT05] Giuseppe Ateniese, Daniel H. Chou, Breno de Medeiros, and Gene Tsudik. Sanitizable Signatures. In *Computer Security - ESORICS 2005, 10th European Symposium on Research in Computer Security, Milan, Italy, September 12-14, 2005, Proceedings*, pages 159–177, 2005.

[ACHdM05] Giuseppe Ateniese, Jan Camenisch, Susan Hohenberger, and Breno de Medeiros. Practical group signatures without random oracles. *IACR Cryptology ePrint Archive*, 2005:385, 2005.

[ACJ+12] Balamurugan Anandan, Chris Clifton, Wei Jiang, Mummoorthy Murugesan, Pedro Pastrana-Camacho, and Luo Si. t-plausibility: Generalizing words to desensitize text. *Trans. Data Privacy*, 5(3):505–534, 2012.

[ACJT00] Giuseppe Ateniese, Jan Camenisch, Marc Joye, and Gene Tsudik. A practical and provably secure coalition-resistant group signature scheme. In *Advances in Cryptology - CRYPTO 2000, 20th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 2000, Proceedings*, pages 255–270, 2000.

[ADK+13] Masayuki Abe, Bernardo David, Markulf Kohlweiss, Ryo Nishimaki, and Miyako Ohkubo. Tagged one-time signatures: Tight

security and optimal tag size. In *Public-Key Cryptography - PKC 2013 - 16th International Conference on Practice and Theory in Public-Key Cryptography, Nara, Japan, February 26 - March 1, 2013. Proceedings*, pages 312–331, 2013.

[AFGH06] Giuseppe Ateniese, Kevin Fu, Matthew Green, and Susan Hohenberger. Improved proxy re-encryption schemes with applications to secure distributed storage. *ACM Trans. Inf. Syst. Secur.*, 9(1):1–30, 2006.

[AH05] Giuseppe Ateniese and Susan Hohenberger. Proxy re-signatures: new definitions, algorithms, and applications. In *Proceedings of the 12th ACM Conference on Computer and Communications Security, CCS 2005, Alexandria, VA, USA, November 7-11, 2005*, pages 310–319, 2005.

[AHI11] Benny Applebaum, Danny Harnik, and Yuval Ishai. Semantic security under related-key attacks and applications. In *Innovations in Computer Science - ICS 2010, Tsinghua University, Beijing, China, January 7-9, 2011. Proceedings*, pages 45–60, 2011.

[ALP12] Nuttapong Attrapadung, Benoît Libert, and Thomas Peters. Computing on Authenticated Data: New Privacy Definitions and Constructions. In *Advances in Cryptology - ASIACRYPT 2012 - 18th International Conference on the Theory and Application of Cryptology and Information Security, Beijing, China, December 2-6, 2012. Proceedings*, pages 367–385, 2012.

[ALP13] Nuttapong Attrapadung, Benoît Libert, and Thomas Peters. Efficient completely context-hiding quotable and linearly homomorphic signatures. In *Public-Key Cryptography - PKC 2013 - 16th International Conference on Practice and Theory in Public-Key Cryptography, Nara, Japan, February 26 - March 1, 2013. Proceedings*, pages 386–404, 2013.

[AN11] Tolga Acar and Lan Nguyen. Revocation for delegatable anonymous credentials. In *Public Key Cryptography - PKC 2011 - 14th International Conference on Practice and Theory in Public Key Cryptography, Taormina, Italy, March 6-9, 2011. Proceedings*, pages 423–440, 2011.

[AP14] Jacob Alperin-Sheriff and Chris Peikert. Faster bootstrapping with polynomial error. In *Advances in Cryptology - CRYPTO 2014 - 34th Annual Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2014, Proceedings, Part I*, pages 297–314, 2014.

[ARHR13] Erman Ayday, Jean Louis Raisaro, Jean-Pierre Hubaux, and Jacques Rougemont. Protecting and evaluating genomic privacy

in medical tests and personalized medicine. In *Proceedings of the 12th annual ACM Workshop on Privacy in the Electronic Society, WPES 2013, Berlin, Germany, November 4, 2013*, pages 95–106, 2013.

[ATSM09] Man Ho Au, Patrick P. Tsang, Willy Susilo, and Yi Mu. Dynamic universal accumulators for DDH groups and their application to attribute-based anonymous credential systems. In *Topics in Cryptology - CT-RSA 2009, The Cryptographers' Track at the RSA Conference 2009, San Francisco, CA, USA, April 20-24, 2009. Proceedings*, pages 295–308, 2009.

[BB04a] Dan Boneh and Xavier Boyen. Secure identity based encryption without random oracles. In *Advances in Cryptology - CRYPTO 2004, 24th Annual International CryptologyConference, Santa Barbara, California, USA, August 15-19, 2004, Proceedings*, pages 443–459, 2004.

[BB04b] Dan Boneh and Xavier Boyen. Short signatures without random oracles. In *Advances in Cryptology - EUROCRYPT 2004, International Conference on the Theory and Applications of Cryptographic Techniques, Interlaken, Switzerland, May 2-6, 2004, Proceedings*, pages 56–73, 2004.

[BB12] Jordan Brown and Douglas M. Blough. Verifiable and redactable medical documents. In *AMIA 2012, American Medical Informatics Association Annual Symposium, Chicago, Illinois, USA, November 3-7, 2012*, 2012.

[BBD+10] Christina Brzuska, Heike Busch, Özgür Dagdelen, Marc Fischlin, Martin Franz, Stefan Katzenbeisser, Mark Manulis, Cristina Onete, Andreas Peter, Bertram Poettering, and Dominique Schröder. Redactable Signatures for Tree-Structured Data: Definitions and Constructions. In *Applied Cryptography and Network Security, 8th International Conference, ACNS 2010, Beijing, China, June 22-25, 2010. Proceedings*, pages 87–104, 2010.

[BBKS07] Mihir Bellare, Alexandra Boldyreva, K. Kurosawa, and Jessica Staddon. Multirecipient encryption schemes: How to save on bandwidth and computation without sacrificing security. *IEEE Trans. Information Theory*, 53(11):3927–3943, 2007.

[BBL16] Olivier Blazy, Xavier Bultel, and Pascal Lafourcade. Two secure anonymous proxy-based data storages. In *Proceedings of the 13th International Joint Conference on e-Business and Telecommunications (ICETE 2016) - Volume 4: SECRYPT, Lisbon, Portugal, July 26-28, 2016.*, pages 251–258, 2016.

[BBL17] Fabrice Benhamouda, Florian Bourse, and Helger Lipmaa. Cca-secure inner-product functional encryption from projective hash functions. In *Public-Key Cryptography - PKC 2017 - 20th IACR International Conference on Practice and Theory in Public-Key Cryptography, Amsterdam, The Netherlands, March 28-31, 2017, Proceedings*, pages 36–66, 2017.

[BBM09] David Bauer, Douglas M. Blough, and Apurva Mohan. Redactable signatures on data with dependencies and their application to personal health records. In *Proceedings of the 2009 ACM Workshop on Privacy in the Electronic Society, WPES 2009, Chicago, Illinois, USA, November 9, 2009*, pages 91–100, 2009.

[BBS98] Matt Blaze, Gerrit Bleumer, and Martin Strauss. Divertible protocols and atomic proxy cryptography. In *Advances in Cryptology - EUROCRYPT '98, International Conference on the Theory and Application of Cryptographic Techniques, Espoo, Finland, May 31 - June 4, 1998, Proceeding*, pages 127–144, 1998.

[BBS04] Dan Boneh, Xavier Boyen, and Hovav Shacham. Short group signatures. In *Advances in Cryptology - CRYPTO 2004, 24th Annual International CryptologyConference, Santa Barbara, California, USA, August 15-19, 2004, Proceedings*, pages 41–55, 2004.

[BC14] Dan Boneh and Henry Corrigan-Gibbs. Bivariate polynomials modulo composites and their applications. In *Advances in Cryptology - ASIACRYPT 2014 - 20th International Conference on the Theory and Application of Cryptology and Information Security, Kaoshiung, Taiwan, R.O.C., December 7-11, 2014. Proceedings, Part I*, pages 42–62, 2014.

[BCC+15] Jonathan Bootle, Andrea Cerulli, Pyrros Chaidos, Essam Ghadafi, Jens Groth, and Christophe Petit. Short accountable ring signatures based on DDH. In *Computer Security - ESORICS 2015 - 20th European Symposium on Research in Computer Security, Vienna, Austria, September 21-25, 2015, Proceedings, Part I*, pages 243–265, 2015.

[BCC+16] Jonathan Bootle, Andrea Cerulli, Pyrros Chaidos, Essam Ghadafi, and Jens Groth. Foundations of fully dynamic group signatures. In *Applied Cryptography and Network Security - 14th International Conference, ACNS 2016, Guildford, UK, June 19-22, 2016. Proceedings*, pages 117–136, 2016.

[BCD+17] Foteini Baldimtsi, Jan Camenisch, Maria Dubovitskaya, Anna Lysyanskaya, Leonid Reyzin, Kai Samelin, and Sophia Yakoubov. Accumulators with applications to anonymity-preserving revocation. In *IEEE European Symposium on Security and Privacy, EuroS&P 2017, Paris, France, April 26-28, 2017*, 2017.

[BCM11] Mihir Bellare, David Cash, and Rachel Miller. Cryptography secure against related-key attacks and tampering. In *Advances in Cryptology - ASIACRYPT 2011 - 17th International Conference on the Theory and Application of Cryptology and Information Security, Seoul, South Korea, December 4-8, 2011. Proceedings*, pages 486–503, 2011.

[BCN+10] Patrik Bichsel, Jan Camenisch, Gregory Neven, Nigel P. Smart, and Bogdan Warinschi. Get shorty via group signatures without encryption. In *Security and Cryptography for Networks, 7th International Conference, SCN 2010, Amalfi, Italy, September 13-15, 2010. Proceedings*, pages 381–398, 2010.

[BD17] Razvan Barbulescu and Sylvain Duquesne. Updating key size estimations for pairings. *IACR Cryptology ePrint Archive*, 2017:334, 2017.

[BdM93] Josh Cohen Benaloh and Michael de Mare. One-way accumulators: A decentralized alternative to digital sinatures (extended abstract). In *Advances in Cryptology - EUROCRYPT '93, Workshop on the Theory and Application of of Cryptographic Techniques, Lofthus, Norway, May 23-27, 1993, Proceedings*, pages 274–285, 1993.

[Ber15] Daniel J. Bernstein. Multi-user schnorr security, revisited. *IACR Cryptology ePrint Archive*, 2015:996, 2015.

[BF11] Dan Boneh and David Mandell Freeman. Homomorphic Signatures for Polynomial Functions. In *Advances in Cryptology - EUROCRYPT 2011 - 30th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tallinn, Estonia, May 15-19, 2011. Proceedings*, pages 149–168, 2011.

[BF14] Mihir Bellare and Georg Fuchsbauer. Policy-Based Signatures. In *Public-Key Cryptography - PKC 2014 - 17th International Conference on Practice and Theory in Public-Key Cryptography, Buenos Aires, Argentina, March 26-28, 2014. Proceedings*, pages 520–537, 2014.

[BFF+09] Christina Brzuska, Marc Fischlin, Tobias Freudenreich, Anja Lehmann, Marcus Page, Jakob Schelbert, Dominique Schröder, and Florian Volk. Security of Sanitizable Signatures Revisited. In *Public Key Cryptography - PKC 2009, 12th International Conference on Practice and Theory in Public Key Cryptography, Irvine, CA, USA, March 18-20, 2009. Proceedings*, pages 317–336, 2009.

[BFG13] David Bernhard, Georg Fuchsbauer, and Essam Ghadafi. Efficient signatures of knowledge and DAA in the standard model. In

Applied Cryptography and Network Security - 11th International Conference, ACNS 2013, Banff, AB, Canada, June 25-28, 2013. Proceedings, pages 518–533, 2013.

[BFKW09] Dan Boneh, David Mandell Freeman, Jonathan Katz, and Brent Waters. Signing a linear subspace: Signature schemes for network coding. In *Public Key Cryptography - PKC 2009, 12th International Conference on Practice and Theory in Public Key Cryptography, Irvine, CA, USA, March 18-20, 2009. Proceedings*, pages 68–87, 2009.

[BFLS09] Christina Brzuska, Marc Fischlin, Anja Lehmann, and Dominique Schröder. Santizable signatures: How to partially delegate control for authenticated data. In *BIOSIG 2009 - Proceedings of the Special Interest Group on Biometrics and Electronic Signatures, 17.-18. September 2009 in Darmstadt, Germany*, pages 117–128, 2009.

[BFLS10] Christina Brzuska, Marc Fischlin, Anja Lehmann, and Dominique Schröder. Unlinkability of Sanitizable Signatures. In *Public Key Cryptography - PKC 2010, 13th International Conference on Practice and Theory in Public Key Cryptography, Paris, France, May 26-28, 2010. Proceedings*, pages 444–461, 2010.

[BFP+15] Abhishek Banerjee, Georg Fuchsbauer, Chris Peikert, Krzysztof Pietrzak, and Sophie Stevens. Key-homomorphic constrained pseudorandom functions. In *Theory of Cryptography - 12th Theory of Cryptography Conference, TCC 2015, Warsaw, Poland, March 23-25, 2015, Proceedings, Part II*, pages 31–60, 2015.

[BFR13] Michael Backes, Dario Fiore, and Raphael M. Reischuk. Verifiable delegation of computation on outsourced data. In *2013 ACM SIGSAC Conference on Computer and Communications Security, CCS'13, Berlin, Germany, November 4-8, 2013*, pages 863–874, 2013.

[BFS14] Xavier Boyen, Xiong Fan, and Elaine Shi. Adaptively secure fully homomorphic signatures based on lattices. *IACR Cryptology ePrint Archive*, 2014:916, 2014.

[BFW15] David Bernhard, Marc Fischlin, and Bogdan Warinschi. Adaptive proofs of knowledge in the random oracle model. In *Public-Key Cryptography - PKC 2015 - 18th IACR International Conference on Practice and Theory in Public-Key Cryptography, Gaithersburg, MD, USA, March 30 - April 1, 2015, Proceedings*, pages 629–649, 2015.

[BGG+14] Dan Boneh, Craig Gentry, Sergey Gorbunov, Shai Halevi, Valeria Nikolaenko, Gil Segev, Vinod Vaikuntanathan, and Dhinakaran

Vinayagamurthy. Fully key-homomorphic encryption, arithmetic circuit ABE and compact garbled circuits. In *Advances in Cryptology - EUROCRYPT 2014 - 33rd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Copenhagen, Denmark, May 11-15, 2014. Proceedings*, pages 533–556, 2014.

[BGI14] Elette Boyle, Shafi Goldwasser, and Ioana Ivan. Functional Signatures and Pseudorandom Functions. In *Public-Key Cryptography - PKC 2014 - 17th International Conference on Practice and Theory in Public-Key Cryptography, Buenos Aires, Argentina, March 26-28, 2014. Proceedings*, pages 501–519, 2014.

[BGLS03] Dan Boneh, Craig Gentry, Ben Lynn, and Hovav Shacham. Aggregate and verifiably encrypted signatures from bilinear maps. In *Advances in Cryptology - EUROCRYPT 2003, International Conference on the Theory and Applications of Cryptographic Techniques, Warsaw, Poland, May 4-8, 2003, Proceedings*, pages 416–432, 2003.

[BGP+16] Cristian Borceaa, Arnab "Bobby" Deb Guptaa, Yuriy Polyakova, Kurt Rohloffa, and Gerard Ryana. Picador: End-to-end encrypted publish-subscribe information distribution with proxy re-encryption. *Future Generation Comp. Syst.*, 62:119–127, 2016.

[BJ08] Ali Bagherzandi and Stanislaw Jarecki. Multisignatures using proofs of secret key possession, as secure as the diffie-hellman problem. In *Security and Cryptography for Networks, 6th International Conference, SCN 2008, Amalfi, Italy, September 10-12, 2008. Proceedings*, pages 218–235, 2008.

[BJL16] Fabrice Benhamouda, Marc Joye, and Benoît Libert. A new framework for privacy-preserving aggregation of time-series data. *ACM Trans. Inf. Syst. Secur.*, 18(3):10:1–10:21, 2016.

[BJLS16] Christoph Bader, Tibor Jager, Yong Li, and Sven Schäge. On the impossibility of tight cryptographic reductions. In *Advances in Cryptology - EUROCRYPT 2016 - 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Vienna, Austria, May 8-12, 2016, Proceedings, Part II*, pages 273–304, 2016.

[BK10] Zvika Brakerski and Yael Tauman Kalai. A framework for efficient signatures, ring signatures and identity based encryption in the standard model. *IACR Cryptology ePrint Archive*, 2010:86, 2010.

[BKM09] Adam Bender, Jonathan Katz, and Ruggero Morselli. Ring signatures: Stronger definitions, and constructions without random oracles. *J. Cryptology*, 22(1):114–138, 2009.

[BLL00] Ahto Buldas, Peeter Laud, and Helger Lipmaa. Accountable certificate management using undeniable attestations. In *CCS 2000, Proceedings of the 7th ACM Conference on Computer and Communications Security, Athens, Greece, November 1-4, 2000.*, pages 9–17, 2000.

[BLL02] Ahto Buldas, Peeter Laud, and Helger Lipmaa. Eliminating counterevidence with applications to accountable certificate management. *Journal of Computer Security*, 10(3):273–296, 2002.

[BLMR13] Dan Boneh, Kevin Lewi, Hart William Montgomery, and Ananth Raghunathan. Key homomorphic prfs and their applications. In *Advances in Cryptology - CRYPTO 2013 - 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part I*, pages 410–428, 2013.

[BLS02] Paulo S. L. M. Barreto, Ben Lynn, and Michael Scott. Constructing elliptic curves with prescribed embedding degrees. In *Security in Communication Networks, Third International Conference, SCN 2002, Amalfi, Italy, September 11-13, 2002. Revised Papers*, pages 257–267, 2002.

[BLS04] Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the weil pairing. *J. Cryptology*, 17(4):297–319, 2004.

[BMW03] Mihir Bellare, Daniele Micciancio, and Bogdan Warinschi. Foundations of group signatures: Formal definitions, simplified requirements, and a construction based on general assumptions. In *Advances in Cryptology - EUROCRYPT 2003, International Conference on the Theory and Applications of Cryptographic Techniques, Warsaw, Poland, May 4-8, 2003, Proceedings*, pages 614–629, 2003.

[BN05] Paulo S. L. M. Barreto and Michael Naehrig. Pairing-friendly elliptic curves of prime order. In *Selected Areas in Cryptography, 12th International Workshop, SAC 2005, Kingston, ON, Canada, August 11-12, 2005, Revised Selected Papers*, pages 319–331, 2005.

[Bol03] Alexandra Boldyreva. Threshold signatures, multisignatures and blind signatures based on the gap-diffie-hellman-group signature scheme. In *Public Key Cryptography - PKC 2003, 6th International Workshop on Theory and Practice in Public Key Cryptography, Miami, FL, USA, January 6-8, 2003, Proceedings*, pages 31–46, 2003.

[Bou00] Fabrice Boudot. Efficient proofs that a committed number lies in an interval. In *Advances in Cryptology - EUROCRYPT 2000,*

International Conference on the Theory and Application of Cryptographic Techniques, Bruges, Belgium, May 14-18, 2000, Proceeding, pages 431–444, 2000.

[Boy08] Xavier Boyen. The uber-assumption family. In *Pairing-Based Cryptography - Pairing 2008, Second International Conference, Egham, UK, September 1-3, 2008. Proceedings*, pages 39–56, 2008.

[BP97] Niko Barić and Birgit Pfitzmann. Collision-free accumulators and fail-stop signature schemes without trees. In *Advances in Cryptology - EUROCRYPT '97, International Conference on the Theory and Application of Cryptographic Techniques, Konstanz, Germany, May 11-15, 1997, Proceeding*, pages 480–494, 1997.

[BP14] Abhishek Banerjee and Chris Peikert. New and improved key-homomorphic pseudorandom functions. In *Advances in Cryptology - CRYPTO 2014 - 34th Annual Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2014, Proceedings, Part I*, pages 353–370, 2014.

[BPS12] Christina Brzuska, Henrich Christopher Pöhls, and Kai Samelin. Non-interactive public accountability for sanitizable signatures. In *Public Key Infrastructures, Services and Applications - 9th European Workshop, EuroPKI 2012, Pisa, Italy, September 13-14, 2012, Revised Selected Papers*, pages 178–193, 2012.

[BPS13] Christina Brzuska, Henrich Christopher Pöhls, and Kai Samelin. Efficient and Perfectly Unlinkable Sanitizable Signatures without Group Signatures. In *Public Key Infrastructures, Services and Applications - 10th European Workshop, EuroPKI 2013, Egham, UK, September 12-13, 2013, Revised Selected Papers*, pages 12–30, 2013.

[BPT12] Mihir Bellare, Kenneth G. Paterson, and Susan Thomson. RKA security beyond the linear barrier: Ibe, encryption and signatures. In *Advances in Cryptology - ASIACRYPT 2012 - 18th International Conference on the Theory and Application of Cryptology and Information Security, Beijing, China, December 2-6, 2012. Proceedings*, pages 331–348, 2012.

[BPW12] David Bernhard, Olivier Pereira, and Bogdan Warinschi. How not to prove yourself: Pitfalls of the fiat-shamir heuristic and applications to helios. In *Advances in Cryptology - ASIACRYPT 2012 - 18th International Conference on the Theory and Application of Cryptology and Information Security, Beijing, China, December 2-6, 2012. Proceedings*, pages 626–643, 2012.

[BR93] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *CCS '93, Proceedings of the 1st ACM Conference on Computer and Communications Security, Fairfax, Virginia, USA, November 3-5, 1993.*, pages 62–73, 1993.

[Bra00] Stefan Brands. *Rethinking Public-Key Infrastructures and Digital Certificates: Building in Privacy.* MIT Press, 2000.

[BSZ05] Mihir Bellare, Haixia Shi, and Chong Zhang. Foundations of Group Signatures: The Case of Dynamic Groups. In *Topics in Cryptology - CT-RSA 2005, The Cryptographers' Track at the RSA Conference 2005, San Francisco, CA, USA, February 14-18, 2005, Proceedings*, pages 136–153, 2005.

[BW06] Xavier Boyen and Brent Waters. Compact group signatures without random oracles. In *Advances in Cryptology - EUROCRYPT 2006, 25th Annual International Conference on the Theory and Applications of Cryptographic Techniques, St. Petersburg, Russia, May 28 - June 1, 2006, Proceedings*, pages 427–444, 2006.

[BW07] Xavier Boyen and Brent Waters. Full-domain subgroup hiding and constant-size group signatures. In *Public Key Cryptography - PKC 2007, 10th International Conference on Practice and Theory in Public-Key Cryptography, Beijing, China, April 16-20, 2007, Proceedings*, pages 1–15, 2007.

[CA89] David Chaum and Hans Van Antwerpen. Undeniable signatures. In *Advances in Cryptology - CRYPTO '89, 9th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 1989, Proceedings*, pages 212–216, 1989.

[Can01] Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *42nd Annual Symposium on Foundations of Computer Science, FOCS 2001, 14-17 October 2001, Las Vegas, Nevada, USA*, pages 136–145, 2001.

[Cat14] Dario Catalano. Homomorphic signatures and message authentication codes. In *Security and Cryptography for Networks - 9th International Conference, SCN 2014, Amalfi, Italy, September 3-5, 2014. Proceedings*, pages 514–519, 2014.

[CCJT13] Sébastien Canard, Iwen Coisel, Amandine Jambert, and Jacques Traoré. New results for the practical use of range proofs. In *Public Key Infrastructures, Services and Applications - 10th European Workshop, EuroPKI 2013, Egham, UK, September 12-13, 2013, Revised Selected Papers*, pages 47–64, 2013.

[CCMT09] Claude Castelluccia, Aldar C.-F. Chan, Einar Mykletun, and Gene Tsudik. Efficient and provably secure aggregation of encrypted data in wireless sensor networks. *TOSN*, 5(3):20:1–20:36, 2009.

[CCS08] Jan Camenisch, Rafik Chaabouni, and Abhi Shelat. Efficient protocols for set membership and range proofs. In *Advances in Cryptology - ASIACRYPT 2008, 14th International Conference on the Theory and Application of Cryptology and Information Security, Melbourne, Australia, December 7-11, 2008. Proceedings*, pages 234–252, 2008.

[CD16] Sébastien Canard and Julien Devigne. Highly privacy-protecting data sharing in a tree structure. *Future Generation Comp. Syst.*, 62:119–127, 2016.

[CDDT12] Sébastien Canard, Nicolas Desmoulins, Julien Devigne, and Jacques Traoré. On the implementation of a pairing-based cryptographic protocol in a constrained device. In *Pairing-Based Cryptography - Pairing 2012 - 5th International Conference, Cologne, Germany, May 16-18, 2012, Revised Selected Papers*, pages 210–217, 2012.

[CDHK15] Jan Camenisch, Maria Dubovitskaya, Kristiyan Haralambiev, and Markulf Kohlweiss. Composable & Modular Anonymous Credentials: Definitions and Practical Constructions. *IACR Cryptology ePrint Archive*, 2015:580, 2015.

[CDS94] Ronald Cramer, Ivan Damgård, and Berry Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In *Advances in Cryptology - CRYPTO '94, 14th Annual International Cryptology Conference, Santa Barbara, California, USA, August 21-25, 1994, Proceedings*, pages 174–187, 1994.

[CF13] Dario Catalano and Dario Fiore. Vector commitments and their applications. In *Public-Key Cryptography - PKC 2013 - 16th International Conference on Practice and Theory in Public-Key Cryptography, Nara, Japan, February 26 - March 1, 2013. Proceedings*, pages 55–72, 2013.

[CF15] Dario Catalano and Dario Fiore. Using linearly-homomorphic encryption to evaluate degree-2 functions on encrypted data. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, Denver, CO, USA, October 12-6, 2015*, pages 1518–1529, 2015.

[CFW14] Dario Catalano, Dario Fiore, and Bogdan Warinschi. Homomorphic signatures with efficient verification for polynomial functions.

In *Advances in Cryptology - CRYPTO 2014 - 34th Annual Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2014, Proceedings, Part I*, pages 371–389, 2014.

[CG04] Jan Camenisch and Jens Groth. Group signatures: Better efficiency and new theoretical aspects. In *Security in Communication Networks, 4th International Conference, SCN 2004, Amalfi, Italy, September 8-10, 2004, Revised Selected Papers*, pages 120–133, 2004.

[CGH04] Ran Canetti, Oded Goldreich, and Shai Halevi. The random oracle methodology, revisited. *J. ACM*, 51(4):557–594, 2004.

[CGS07] Nishanth Chandran, Jens Groth, and Amit Sahai. Ring signatures of sub-linear size without random oracles. In *Automata, Languages and Programming, 34th International Colloquium, ICALP 2007, Wroclaw, Poland, July 9-13, 2007, Proceedings*, pages 423–434, 2007.

[CH02] Jan Camenisch and Els Van Herreweghen. Design and Implementation of the *Idemix* Anonymous Credential System. In *ACM CCS*. ACM, 2002.

[CH07] Ran Canetti and Susan Hohenberger. Chosen-ciphertext secure proxy re-encryption. In *Proceedings of the 2007 ACM Conference on Computer and Communications Security, CCS 2007, Alexandria, Virginia, USA, October 28-31, 2007*, pages 185–194, 2007.

[CH10] Philippe Camacho and Alejandro Hevia. On the impossibility of batch update for cryptographic accumulators. In *Progress in Cryptology - LATINCRYPT 2010, First International Conference on Cryptology and Information Security in Latin America, Puebla, Mexico, August 8-11, 2010, Proceedings*, pages 178–188, 2010.

[Cha85] David Chaum. Security Without Identification: Transaction Systems to Make Big Brother Obsolete. *Commun. ACM*, 28(10):1030–1044, 1985.

[Cha94] David Chaum. Designated confirmer signatures. In *Advances in Cryptology - EUROCRYPT '94, Workshop on the Theory and Application of Cryptographic Techniques, Perugia, Italy, May 9-12, 1994, Proceedings*, pages 86–91, 1994.

[CHKM10] Sanjit Chatterjee, Darrel Hankerson, Edward Knapp, and Alfred Menezes. Comparing two pairing-based aggregate signature schemes. *Des. Codes Cryptography*, 55(2-3):141–167, 2010.

[CHKO08] Philippe Camacho, Alejandro Hevia, Marcos A. Kiwi, and Roberto Opazo. Strong accumulators from collision-resistant hashing. In

Information Security, 11th International Conference, ISC 2008, Taipei, Taiwan, September 15-18, 2008. Proceedings, pages 471–486, 2008.

[CHL+13] Melissa Chase, Alexander Healy, Anna Lysyanskaya, Tal Malkin, and Leonid Reyzin. Mercurial commitments with applications to zero-knowledge sets. *J. Cryptology*, 26(2):251–279, 2013.

[CHP12] Jan Camenisch, Susan Hohenberger, and Michael Østergaard Pedersen. Batch verification of short signatures. *J. Cryptology*, 25(4):723–747, 2012.

[CJ10] Sébastien Canard and Amandine Jambert. On Extended Sanitizable Signature Schemes. In *Topics in Cryptology - CT-RSA 2010, The Cryptographers' Track at the RSA Conference 2010, San Francisco, CA, USA, March 1-5, 2010. Proceedings*, pages 179–194, 2010.

[CJL12] Sébastien Canard, Amandine Jambert, and Roch Lescuyer. Sanitizable signatures with several signers and sanitizers. In *Progress in Cryptology - AFRICACRYPT 2012 - 5th International Conference on Cryptology in Africa, Ifrance, Morocco, July 10-12, 2012. Proceedings*, pages 35–52, 2012.

[CKS09] Jan Camenisch, Markulf Kohlweiss, and Claudio Soriente. An accumulator based on bilinear maps and efficient revocation for anonymous credentials. In *Public Key Cryptography - PKC 2009, 12th International Conference on Practice and Theory in Public Key Cryptography, Irvine, CA, USA, March 18-20, 2009. Proceedings*, pages 481–500, 2009.

[CL02a] Jan Camenisch and Anna Lysyanskaya. A Signature Scheme with Efficient Protocols. In *Security in Communication Networks, Third International Conference, SCN 2002, Amalfi, Italy, September 11-13, 2002. Revised Papers*, pages 268–289, 2002.

[CL02b] Jan Camenisch and Anna Lysyanskaya. Dynamic accumulators and application to efficient revocation of anonymous credentials. In *Advances in Cryptology - CRYPTO 2002, 22nd Annual International Cryptology Conference, Santa Barbara, California, USA, August 18-22, 2002, Proceedings*, pages 61–76, 2002.

[CL04] Jan Camenisch and Anna Lysyanskaya. Signature Schemes and Anonymous Credentials from Bilinear Maps. In *Advances in Cryptology - CRYPTO 2004, 24th Annual International Cryptology-Conference, Santa Barbara, California, USA, August 15-19, 2004, Proceedings*, pages 56–72, 2004.

[CL06] Melissa Chase and Anna Lysyanskaya. On Signatures of Knowledge. In *Advances in Cryptology - CRYPTO 2006, 26th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 2006, Proceedings*, pages 78–96, 2006.

[CL13] Sébastien Canard and Roch Lescuyer. Protecting Privacy by Sanitizing Personal Data: A New Approach to Anonymous Credentials. In *8th ACM Symposium on Information, Computer and Communications Security, ASIA CCS '13, Hangzhou, China - May 08 - 10, 2013*, pages 381–392, 2013.

[CLM08] Sébastien Canard, Fabien Laguillaumie, and Michel Milhau. Trapdoor Sanitizable Signatures and Their Application to Content Protection. In *Applied Cryptography and Network Security, 6th International Conference, ACNS 2008, New York, NY, USA, June 3-6, 2008. Proceedings*, pages 258–276, 2008.

[CLX09] Ee-Chien Chang, Chee Liang Lim, and Jia Xu. Short Redactable Signatures Using Random Trees. In *Topics in Cryptology - CT-RSA 2009, The Cryptographers' Track at the RSA Conference 2009, San Francisco, CA, USA, April 20-24, 2009. Proceedings*, pages 133–147, 2009.

[CMP14] Dario Catalano, Antonio Marcedone, and Orazio Puglisi. Authenticating computation on groups: New homomorphic primitives and applications. In *Advances in Cryptology - ASIACRYPT 2014 - 20th International Conference on the Theory and Application of Cryptology and Information Security, Kaoshiung, Taiwan, R.O.C., December 7-11, 2014, Proceedings, Part II*, pages 193–212, 2014.

[CS97] Jan Camenisch and Markus Stadler. Efficient Group Signature Schemes for Large Groups (Extended Abstract). In *Advances in Cryptology - CRYPTO '97, 17th Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 1997, Proceedings*, pages 410–424, 1997.

[CSF+08] D. Cooper, S. Santesson, S. Farrell, S. Boeyen, R. Housley, and W. Polk. Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile. RFC 5280 (Proposed Standard), May 2008.

[CSS12] T.-H. Hubert Chan, Elaine Shi, and Dawn Song. Privacy-preserving stream aggregation with fault tolerance. In *Financial Cryptography and Data Security - 16th International Conference, FC 2012, Kralendijk, Bonaire, Februray 27-March 2, 2012, Revised Selected Papers*, pages 200–214, 2012.

[CvH91] David Chaum and Eugène van Heyst. Group Signatures. In *Advances in Cryptology - EUROCRYPT '91, Workshop on the*

Theory and Application of of Cryptographic Techniques, Brighton, UK, April 8-11, 1991, Proceedings, pages 257–265, 1991.

[CX08] Ee-Chien Chang and Jia Xu. Remote integrity check with dishonest storage server. In *Computer Security - ESORICS 2008, 13th European Symposium on Research in Computer Security, Málaga, Spain, October 6-8, 2008. Proceedings*, pages 223–237, 2008.

[Dam10] Ivan Damgård. On Σ-protocols, 2010. Manuscript.

[DHLW10] Yevgeniy Dodis, Kristiyan Haralambiev, Adriana López-Alt, and Daniel Wichs. Efficient public-key cryptography in the presence of key leakage. In *Advances in Cryptology - ASIACRYPT 2010 - 16th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 5-9, 2010. Proceedings*, pages 613–631, 2010.

[DKNS04] Yevgeniy Dodis, Aggelos Kiayias, Antonio Nicolosi, and Victor Shoup. Anonymous identification in ad hoc groups. In *Advances in Cryptology - EUROCRYPT 2004, International Conference on the Theory and Applications of Cryptographic Techniques, Interlaken, Switzerland, May 2-6, 2004, Proceedings*, pages 609–626, 2004.

[DL11] George Danezis and Benjamin Livshits. Towards ensuring client-side computational integrity. In *Proceedings of the 3rd ACM Cloud Computing Security Workshop, CCSW 2011, Chicago, IL, USA, October 21, 2011*, pages 125–130, 2011.

[dMLPP12] Hermann de Meer, Manuel Liedel, Henrich C. Pöhls, and Joachim Posegga. Indistinguishability of One-Way Accumulators. Technical Report MIP-1210, Faculty of Computer Science and Mathematics (FIM), University of Passau, 2012.

[dMPPS14a] Hermann de Meer, Henrich C. Pöhls, Joachim Posegga, and Kai Samelin. Redactable Signature Schemes for Trees with Signer-Controlled Non-Leaf-Redactions. In Mohammad S. Obaidat and Joaquim Filipe, editors, *E-Business and Telecommunications*, volume 455 of *CCIS*, pages 155–171. Springer, 2014.

[dMPPS14b] Hermann de Meer, Henrich Christopher Pöhls, Joachim Posegga, and Kai Samelin. On the relation between redactable and sanitizable signature schemes. In *Engineering Secure Software and Systems - 6th International Symposium, ESSoS 2014, Munich, Germany, February 26-28, 2014, Proceedings*, pages 113–130, 2014.

[DMS16] Yevgeniy Dodis, Ilya Mironov, and Noah Stephens-Davidowitz. Message transmission with reverse firewalls - secure communication on corrupted machines. In *Advances in Cryptology - CRYPTO 2016 - 36th Annual International Cryptology Conference, Santa*

Barbara, CA, USA, August 14-18, 2016, Proceedings, Part I, pages 341–372, 2016.

[DP06] Cécile Delerablée and David Pointcheval. Dynamic fully anonymous short group signatures. In *Progressin Cryptology - VIETCRYPT 2006, First International Conferenceon Cryptology in Vietnam, Hanoi, Vietnam, September 25-28, 2006, Revised Selected Papers*, pages 193–210, 2006.

[DT08] Ivan Damgård and Nikos Triandopoulos. Supporting non-membership proofs with bilinear-map accumulators. *IACR Cryptology ePrint Archive*, 2008:538, 2008.

[EG14] Alex Escala and Jens Groth. Fine-tuning groth-sahai proofs. In *Public-Key Cryptography - PKC 2014 - 17th International Conference on Practice and Theory in Public-Key Cryptography, Buenos Aires, Argentina, March 26-28, 2014. Proceedings*, pages 630–649, 2014.

[FF13] Marc Fischlin and Nils Fleischhacker. Limitations of the meta-reduction technique: The case of schnorr signatures. In *Advances in Cryptology - EUROCRYPT 2013, 32nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Athens, Greece, May 26-30, 2013. Proceedings*, pages 444–460, 2013.

[FHS15a] Georg Fuchsbauer, Christian Hanser, and Daniel Slamanig. Practical Round-Optimal Blind Signatures in the Standard Model. In *Advances in Cryptology - CRYPTO 2015 - 35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part II*, pages 233–253, 2015.

[FHS15b] Georg Fuchsbauer, Christian Hanser, and Daniel Slamanig. Structure-preserving signatures on equivalence classes and constant-size anonymous credentials. *IACR Cryptology ePrint Archive*, 2014:944, 2015.

[Fis05] Marc Fischlin. Communication-efficient non-interactive proofs of knowledge with online extractors. In *Advances in Cryptology - CRYPTO 2005: 25th Annual International Cryptology Conference, Santa Barbara, California, USA, August 14-18, 2005, Proceedings*, pages 152–168, 2005.

[FKM+16] Nils Fleischhacker, Johannes Krupp, Giulio Malavolta, Jonas Schneider, Dominique Schröder, and Mark Simkin. Efficient unlinkable sanitizable signatures from signatures with re-randomizable keys. In *Public-Key Cryptography - PKC 2016 - 19th*

IACR International Conference on Practice and Theory in Public-Key Cryptography, Taipei, Taiwan, March 6-9, 2016, Proceedings, Part I, pages 301–330, 2016.

[FKMV12] Sebastian Faust, Markulf Kohlweiss, Giorgia Azzurra Marson, and Daniele Venturi. On the non-malleability of the fiat-shamir transform. In *Progress in Cryptology - INDOCRYPT 2012, 13th International Conference on Cryptology in India, Kolkata, India, December 9-12, 2012. Proceedings*, pages 60–79, 2012.

[FLZ14] Prastudy Fauzi, Helger Lipmaa, and Bingsheng Zhang. Efficient non-interactive zero knowledge arguments for set operations. In *Financial Cryptography and Data Security - 18th International Conference, FC 2014, Christ Church, Barbados, March 3-7, 2014, Revised Selected Papers*, pages 216–233, 2014.

[FMNP16] Dario Fiore, Aikaterini Mitrokotsa, Luca Nizzardo, and Elena Pagnin. Multi-key homomorphic authenticators. In *Advances in Cryptology - ASIACRYPT 2016 - 22nd International Conference on the Theory and Application of Cryptology and Information Security, Hanoi, Vietnam, December 4-8, 2016, Proceedings, Part II*, pages 499–530, 2016.

[FN02] Nelly Fazio and Antonio Nicolisi. Cryptographic Accumulators: Definitions, Constructions and Applications. Technical report, 2002.

[FS86] Amos Fiat and Adi Shamir. How to Prove Yourself: Practical Solutions to Identification and Signature Problems. In *Advances in Cryptology - CRYPTO '86, Santa Barbara, California, USA, 1986, Proceedings*, pages 186–194, 1986.

[Gam85] Taher El Gamal. A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms. *IEEE Transactions on Information Theory*, 31(4):469–472, 1985.

[GCZ16] Steven Goldfeder, Melissa Chase, and Greg Zaverucha. Efficient post-quantum zero-knowledge and signatures. *IACR Cryptology ePrint Archive*, 2016:1110, 2016.

[Gen09] Craig Gentry. Fully homomorphic encryption using ideal lattices. In *Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009, Bethesda, MD, USA, May 31 - June 2, 2009*, pages 169–178, 2009.

[GHKW16] Romain Gay, Dennis Hofheinz, Eike Kiltz, and Hoeteck Wee. Tightly cca-secure encryption without pairings. In *Advances in*

Cryptology - EUROCRYPT 2016 - 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Vienna, Austria, May 8-12, 2016, Proceedings, Part I, pages 1–27, 2016.

[GJKW07] Eu-Jin Goh, Stanislaw Jarecki, Jonathan Katz, and Nan Wang. Efficient signature schemes with tight reductions to the diffie-hellman problems. *J. Cryptology*, 20(4):493–514, 2007.

[GJM02] Philippe Golle, Stanislaw Jarecki, and Ilya Mironov. Cryptographic primitives enforcing communication and storage complexity. In *Financial Cryptography, 6th International Conference, FC 2002, Southampton, Bermuda, March 11-14, 2002, Revised Papers*, pages 120–135, 2002.

[GK15] Jens Groth and Markulf Kohlweiss. One-out-of-many proofs: Or how to leak a secret and spend a coin. In *Advances in Cryptology - EUROCRYPT 2015 - 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part II*, pages 253–280, 2015.

[GLW12] Shafi Goldwasser, Allison B. Lewko, and David A. Wilson. Bounded-collusion IBE from key homomorphism. In *Theory of Cryptography - 9th Theory of Cryptography Conference, TCC 2012, Taormina, Sicily, Italy, March 19-21, 2012. Proceedings*, pages 564–581, 2012.

[GMP14] Felix Günther, Mark Manulis, and Andreas Peter. Privacy-enhanced participatory sensing with collusion resistance and data aggregation. In *Cryptology and Network Security - 13th International Conference, CANS 2014, Heraklion, Crete, Greece, October 22-24, 2014. Proceedings*, pages 321–336, 2014.

[GMS02] Steven D. Galbraith, John Malone-Lee, and Nigel P. Smart. Public key signatures in the multi-user setting. *Inf. Process. Lett.*, 83(5):263–266, 2002.

[GMY03] Juan A. Garay, Philip D. MacKenzie, and Ke Yang. Strengthening zero-knowledge protocols using signatures. In *Advances in Cryptology - EUROCRYPT 2003, International Conference on the Theory and Applications of Cryptographic Techniques, Warsaw, Poland, May 4-8, 2003, Proceedings*, pages 177–194, 2003.

[GMY06] Juan A. Garay, Philip D. MacKenzie, and Ke Yang. Strengthening zero-knowledge protocols using signatures. *J. Cryptology*, 19(2):169–209, 2006.

[Gol01] Oded Goldreich. *The Foundations of Cryptography - Volume 1, Basic Techniques*. Cambridge University Press, 2001.

[Gol08] Oded Goldreich. *Computational Complexity - A Conceptual Perspective*. Cambridge University Press, 2008.

[GOP+16] Esha Ghosh, Olga Ohrimenko, Dimitrios Papadopoulos, Roberto Tamassia, and Nikos Triandopoulos. Zero-knowledge accumulators and set algebra. In *Advances in Cryptology - ASIACRYPT 2016 - 22nd International Conference on the Theory and Application of Cryptology and Information Security, Hanoi, Vietnam, December 4-8, 2016, Proceedings, Part II*, pages 67–100, 2016.

[GOT15] Esha Ghosh, Olga Ohrimenko, and Roberto Tamassia. Zero-knowledge authenticated order queries and order statistics on a list. In *Applied Cryptography and Network Security - 13th International Conference, ACNS 2015, New York, NY, USA, June 2-5, 2015, Revised Selected Papers*, pages 149–171, 2015.

[GQZ10] Junqing Gong, Haifeng Qian, and Yuan Zhou. Fully-Secure and Practical Sanitizable Signatures. In *Information Security and Cryptology - 6th International Conference, Inscrypt 2010, Shanghai, China, October 20-24, 2010, Revised Selected Papers*, pages 300–317, 2010.

[Gro06] Jens Groth. Simulation-sound NIZK proofs for a practical language and constant size group signatures. In *Advances in Cryptology - ASIACRYPT 2006, 12th International Conference on the Theory and Application of Cryptology and Information Security, Shanghai, China, December 3-7, 2006, Proceedings*, pages 444–459, 2006.

[Gro07] Jens Groth. Fully anonymous group signatures without random oracles. In *Advances in Cryptology - ASIACRYPT 2007, 13th International Conference on the Theory and Application of Cryptology and Information Security, Kuching, Malaysia, December 2-6, 2007, Proceedings*, pages 164–180, 2007.

[GS02] Craig Gentry and Alice Silverberg. Hierarchical id-based cryptography. In *Advances in Cryptology - ASIACRYPT 2002, 8th International Conference on the Theory and Application of Cryptology and Information Security, Queenstown, New Zealand, December 1-5, 2002, Proceedings*, pages 548–566, 2002.

[GS07] Jens Groth and Amit Sahai. Efficient non-interactive proof systems for bilinear groups. *IACR Cryptology ePrint Archive*, 2007:155, 2007.

[GS08] Jens Groth and Amit Sahai. Efficient Non-interactive Proof Systems for Bilinear Groups. In *Advances in Cryptology - EUROCRYPT 2008, 27th Annual International Conference on the*

Theory and Applications of Cryptographic Techniques, Istanbul, Turkey, April 13-17, 2008. Proceedings, pages 415–432, 2008.

[GSW13] Craig Gentry, Amit Sahai, and Brent Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In *Advances in Cryptology - CRYPTO 2013 - 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part I*, pages 75–92, 2013.

[GTH02] Michael T. Goodrich, Roberto Tamassia, and Jasminka Hasic. An efficient dynamic and distributed cryptographic accumulator. In *Information Security, 5th International Conference, ISC 2002 Sao Paulo, Brazil, September 30 - October 2, 2002, Proceedings*, pages 372–388, 2002.

[GVW15] Sergey Gorbunov, Vinod Vaikuntanathan, and Daniel Wichs. Leveled Fully Homomorphic Signatures from Standard Lattices. In *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, STOC 2015, Portland, OR, USA, June 14-17, 2015*, pages 469–477, 2015.

[HCCN15] Jung Yeon Hwang, Liqun Chen, Hyun Sook Cho, and DaeHun Nyang. Short Dynamic Group Signature Scheme Supporting Controllable Linkability. *IEEE Transactions on Information Forensics and Security*, 10(6):1109–1124, 2015.

[HHH+08] Stuart Haber, Yasuo Hatano, Yoshinori Honda, William G. Horne, Kunihiko Miyazaki, Tomas Sander, Satoru Tezoku, and Danfeng Yao. Efficient signature schemes supporting redaction, pseudonymization, and data deidentification. In *Proceedings of the 2008 ACM Symposium on Information, Computer and Communications Security, ASIACCS 2008, Tokyo, Japan, March 18-20, 2008*, pages 353–362, 2008.

[HLC+13] Jung Yeon Hwang, Sokjoon Lee, Byung-Ho Chung, Hyun Sook Cho, and DaeHun Nyang. Group Signatures with Controllable Linkability for Dynamic Membership. *Inf. Sci.*, 222:761–778, 2013.

[HLhC+11] Jung Yeon Hwang, Sokjoon Lee, Byung ho Chung, Hyun Sook Cho, and DaeHun Nyang. Short Group Signatures with Controllable Linkability. In *Lightweight Security Privacy: Devices, Protocols and Applications (LightSec), 2011 Workshop on*, pages 44–52, March 2011.

[HS13a] Christian Hanser and Daniel Slamanig. Blank Digital Signatures. In *8th ACM Symposium on Information, Computer and Communications Security, ASIA CCS '13, Hangzhou, China - May 08 - 10, 2013*, pages 95–106, 2013.

[HS13b] Christian Hanser and Daniel Slamanig. Warrant-Hiding Delegation-by-Certificate Proxy Signature Schemes. In *Progress in Cryptology - INDOCRYPT 2013 - 14th International Conference on Cryptology in India, Mumbai, India, December 7-10, 2013. Proceedings*, pages 60–77, 2013.

[HS14] Christian Hanser and Daniel Slamanig. Structure-Preserving Signatures on Equivalence Classes and Their Application to Anonymous Credentials. In *Advances in Cryptology - ASIACRYPT 2014 - 20th International Conference on the Theory and Application of Cryptology and Information Security, Kaoshiung, Taiwan, R.O.C., December 7-11, 2014. Proceedings, Part I*, pages 491–511, 2014.

[ID03] Anca-Andreea Ivan and Yevgeniy Dodis. Proxy cryptography revisited. In *Proceedings of the Network and Distributed System Security Symposium, NDSS 2003, San Diego, California, USA*, 2003.

[IN83] K. Itakura and K. Nakamura. A public-key cryptosystem suitable for digital multisignatures. *NEC Research & Development*, 71, 1983.

[JL13] Marc Joye and Benoît Libert. A scalable scheme for privacy-preserving aggregation of time-series data. In *Financial Cryptography and Data Security - 17th International Conference, FC 2013, Okinawa, Japan, April 1-5, 2013, Revised Selected Papers*, pages 111–125, 2013.

[JMSW02] Robert Johnson, David Molnar, Dawn Xiaodong Song, and David Wagner. Homomorphic Signature Schemes. In *Topics in Cryptology - CT-RSA 2002, The Cryptographer's Track at the RSA Conference, 2002, San Jose, CA, USA, February 18-22, 2002, Proceedings*, pages 244–262, 2002.

[JSI96] Markus Jakobsson, Kazue Sako, and Russell Impagliazzo. Designated verifier proofs and their applications. In *Advances in Cryptology - EUROCRYPT '96, International Conference on the Theory and Application of Cryptographic Techniques, Saragossa, Spain, May 12-16, 1996, Proceeding*, pages 143–154, 1996.

[Kat10] Jonathan Katz. *Digital Signatures*. Springer, 2010.

[KB13] Ashish Kundu and Elisa Bertino. Privacy-Preserving Authentication of Trees and Graphs. *Int. J. Inf. Sec.*, 12(6):467–494, 2013.

[KB16] Taechan Kim and Razvan Barbulescu. Extended tower number field sieve: A new complexity for the medium prime case. In *Advances in Cryptology - CRYPTO 2016 - 36th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2016, Proceedings, Part I*, pages 543–571, 2016.

[KL06] Marek Klonowski and Anna Lauks. Extended Sanitizable Signatures. In *Information Security and Cryptology - ICISC 2006, 9th International Conference, Busan, Korea, November 30 - December 1, 2006, Proceedings*, pages 343–355, 2006.

[KL07] Jonathan Katz and Yehuda Lindell. *Introduction to Modern Cryptography*. Chapman and Hall/CRC Press, 2007.

[KM07] Neal Koblitz and Alfred Menezes. Another look at generic groups. *Adv. in Math. of Comm.*, 1(1):13–28, 2007.

[KM15] Neal Koblitz and Alfred J. Menezes. The random oracle model: a twenty-year retrospective. *Des. Codes Cryptography*, 77(2-3):587–610, 2015.

[KMP16] Eike Kiltz, Daniel Masny, and Jiaxin Pan. Optimal security proofs for signatures from identification schemes. In *Advances in Cryptology - CRYPTO 2016 - 36th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2016, Proceedings, Part II*, pages 33–61, 2016.

[KSS08] Ezekiel J. Kachisa, Edward F. Schaefer, and Michael Scott. Constructing brezing-weng pairing-friendly elliptic curves using elements in the cyclotomic field. In *Pairing-Based Cryptography - Pairing 2008, Second International Conference, Egham, UK, September 1-3, 2008. Proceedings*, pages 126–135, 2008.

[KW03] Jonathan Katz and Nan Wang. Efficiency improvements for signature schemes with tight security reductions. In *Proceedings of the 10th ACM Conference on Computer and Communications Security, CCS 2003, Washington, DC, USA, October 27-30, 2003*, pages 155–164, 2003.

[KY05] Aggelos Kiayias and Moti Yung. Group signatures with efficient concurrent join. In *Advances in Cryptology - EUROCRYPT 2005, 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Aarhus, Denmark, May 22-26, 2005, Proceedings*, pages 198–214, 2005.

[KZG10] Aniket Kate, Gregory M. Zaverucha, and Ian Goldberg. Constant-size commitments to polynomials and their applications. In *Advances in Cryptology - ASIACRYPT 2010 - 16th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 5-9, 2010. Proceedings*, pages 177–194, 2010.

[Lac16] Marie-Sarah Lacharité. Security of bls and bgls signatures in a multi-user setting. Arcticcrypt 2016 Talk, `http://arcticcrypt.b.uib.no/files/2016/07/Slides-Lacharite.pdf`, 2016.

[LAN02] Helger Lipmaa, N. Asokan, and Valtteri Niemi. Secure vickrey auctions without threshold trust. In *Financial Cryptography, 6th International Conference, FC 2002, Southampton, Bermuda, March 11-14, 2002, Revised Papers*, pages 87–101, 2002.

[LC13] Qinghua Li and Guohong Cao. Efficient privacy-preserving stream aggregation in mobile sensing with low aggregation error. In *Privacy Enhancing Technologies - 13th International Symposium, PETS 2013, Bloomington, IN, USA, July 10-12, 2013. Proceedings*, pages 60–81, 2013.

[LCP14] Qinghua Li, Guohong Cao, and Thomas F. La Porta. Efficient and privacy-aware data aggregation in mobile sensing. *IEEE Trans. Dependable Sec. Comput.*, 11(2):115–129, 2014.

[LDPW14] Junzuo Lai, Robert H. Deng, HweeHwa Pang, and Jian Weng. Verifiable computation on outsourced encrypted data. In *Computer Security - ESORICS 2014 - 19th European Symposium on Research in Computer Security, Wroclaw, Poland, September 7-11, 2014. Proceedings, Part I*, pages 273–291, 2014.

[LEM14] Iraklis Leontiadis, Kaoutar Elkhiyaoui, and Refik Molva. Private and dynamic time-series data aggregation with trust relaxation. In *Cryptology and Network Security - 13th International Conference, CANS 2014, Heraklion, Crete, Greece, October 22-24, 2014. Proceedings*, pages 305–320, 2014.

[LEÖM15] Iraklis Leontiadis, Kaoutar Elkhiyaoui, Melek Önen, and Refik Molva. PUDA - privacy and unforgeability for data aggregation. In *Cryptology and Network Security - 14th International Conference, CANS 2015, Marrakesh, Morocco, December 10-12, 2015, Proceedings*, pages 3–18, 2015.

[Lip03] Helger Lipmaa. On diophantine complexity and statistical zero-knowledge arguments. In *Advances in Cryptology - ASIACRYPT 2003, 9th International Conference on the Theory and Application of Cryptology and Information Security, Taipei, Taiwan, November 30 - December 4, 2003, Proceedings*, pages 398–415, 2003.

[Lip12] Helger Lipmaa. Secure accumulators from euclidean rings without trusted setup. In *Applied Cryptography and Network Security - 10th International Conference, ACNS 2012, Singapore, June 26-29, 2012. Proceedings*, pages 224–240, 2012.

[LLM+16] Benoît Libert, San Ling, Fabrice Mouhartem, Khoa Nguyen, and Huaxiong Wang. Signature schemes with efficient protocols and dynamic group signatures from lattice assumptions. In *Advances in Cryptology - ASIACRYPT 2016 - 22nd International Conference on the Theory and Application of Cryptology and Information*

Security, Hanoi, Vietnam, December 4-8, 2016, Proceedings, Part II, pages 373–403, 2016.

[LLX07] Jiangtao Li, Ninghui Li, and Rui Xue. Universal Accumulators with Efficient Nonmembership Proofs. In *Applied Cryptography and Network Security, 5th International Conference, ACNS 2007, Zhuhai, China, June 5-8, 2007, Proceedings*, pages 253–269, 2007.

[LMPY16] Benoît Libert, Fabrice Mouhartem, Thomas Peters, and Moti Yung. Practical "signatures with efficient protocols" from simple assumptions. In *Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security, AsiaCCS 2016, Xi'an, China, May 30 - June 3, 2016*, pages 511–522, 2016.

[LOS+06] Steve Lu, Rafail Ostrovsky, Amit Sahai, Hovav Shacham, and Brent Waters. Sequential aggregate signatures and multisignatures without random oracles. In *Advances in Cryptology - EUROCRYPT 2006, 25th Annual International Conference on the Theory and Applications of Cryptographic Techniques, St. Petersburg, Russia, May 28 - June 1, 2006, Proceedings*, pages 465–485, 2006.

[LPY15] Benoît Libert, Thomas Peters, and Moti Yung. Short group signatures via structure-preserving signatures: Standard model security from simple assumptions. In *Advances in Cryptology - CRYPTO 2015 - 35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part II*, pages 296–316, 2015.

[LRD+15] Thomas Lorünser, Charles Bastos Rodriguez, Denise Demirel, Simone Fischer-Hübner, Thomas Groß, Thomas Länger, Mathieu des Noes, Henrich C. Pöhls, Boris Rozenberg, and Daniel Slamanig. Towards a New Paradigm for Privacy and Security in Cloud Services. In *New LEIT projects on Security-by-Design - 4th Cyber Security and Privacy EU Forum, CSP Forum 2015, Brussels, Belgium, April 28 - 29*. Springer, 2015.

[LTWC16] Russell W. F. Lai, Raymond K. H. Tai, Harry W. H. Wong, and Sherman S. M. Chow. A zoo of homomorphic signatures: Multi-key and key-homomorphism. *IACR Cryptology ePrint Archive*, 2016:834, 2016.

[LV08] Benoît Libert and Damien Vergnaud. Multi-use unidirectional proxy re-signatures. In *Proceedings of the 2008 ACM Conference on Computer and Communications Security, CCS 2008, Alexandria, Virginia, USA, October 27-31, 2008*, pages 511–520, 2008.

[LV11] Benoît Libert and Damien Vergnaud. Unidirectional chosen-ciphertext secure proxy re-encryption. *IEEE Trans. Information Theory*, 57(3):1786–1802, 2011.

[LWB05] Helger Lipmaa, Guilin Wang, and Feng Bao. Designated verifier signature schemes: Attacks, new security notions and a new construction. In *Automata, Languages and Programming, 32nd International Colloquium, ICALP 2005, Lisbon, Portugal, July 11-15, 2005, Proceedings*, pages 459–471, 2005.

[LZCS16] Russell W. F. Lai, Tao Zhang, Sherman S. M. Chow, and Dominique Schröder. Efficient sanitizable signatures without random oracles. In *Computer Security - ESORICS 2016 - 21st European Symposium on Research in Computer Security, Heraklion, Greece, September 26-30, 2016, Proceedings, Part I*, pages 363–380, 2016.

[Mau11] Ueli Maurer. Constructive cryptography - A new paradigm for security definitions and proofs. In *Theory of Security and Applications - Joint Workshop, TOSCA 2011, Saarbrücken, Germany, March 31 - April 1, 2011, Revised Selected Papers*, pages 33–56, 2011.

[MGGR13] Ian Miers, Christina Garman, Matthew Green, and Aviel D. Rubin. Zerocoin: Anonymous distributed e-cash from bitcoin. In *2013 IEEE Symposium on Security and Privacy, SP 2013, Berkeley, CA, USA, May 19-22, 2013*, pages 397–411, 2013.

[MHI06] Kunihiko Miyazaki, Goichiro Hanaoka, and Hideki Imai. Digitally signed document sanitizing scheme based on bilinear maps. In *Proceedings of the 2006 ACM Symposium on Information, Computer and Communications Security, ASIACCS 2006, Taipei, Taiwan, March 21-24, 2006*, pages 343–354, 2006.

[MIM+05] Kunihiko Miyazaki, Mitsuru Iwamura, Tsutomu Matsumoto, Ryôichi Sasaki, Hiroshi Yoshiura, Satoru Tezuka, and Hideki Imai. Digitally Signed Document Sanitizing Scheme with Disclosure Condition Control. *IEICE Transactions*, 88-A(1):239–246, 2005.

[MLO16] Chunguang Ma, Juyan Li, and Weiping Ouyang. A homomorphic proxy re-encryption from lattices. In *Provable Security - 10th International Conference, ProvSec 2016, Nanjing, China, November 10-11, 2016, Proceedings*, pages 353–372, 2016.

[MPV09] Jean Monnerat, Sylvain Pasini, and Serge Vaudenay. Efficient deniable authentication for signatures. In *ACNS*, 2009.

[MR11] Ueli Maurer and Renato Renner. Abstract cryptography. In *Innovations in Computer Science - ICS 2010, Tsinghua University, Beijing, China, January 7-9, 2011. Proceedings*, pages 1–21, 2011.

[MRK03] Silvio Micali, Michael O. Rabin, and Joe Kilian. Zero-knowledge sets. In *44th Symposium on Foundations of Computer Science (FOCS 2003), 11-14 October 2003, Cambridge, MA, USA, Proceedings*, pages 80–91, 2003.

[MS04] Alfred Menezes and Nigel P. Smart. Security of signature schemes in a multi-user setting. *Des. Codes Cryptography*, 33(3):261–274, 2004.

[MSM+15] Hiraku Morita, Jacob C. N. Schuldt, Takahiro Matsuda, Goichiro Hanaoka, and Tetsu Iwata. On the security of the schnorr signature scheme and DSA against related-key attacks. In *Information Security and Cryptology - ICISC 2015 - 18th International Conference, Seoul, South Korea, November 25-27, 2015, Revised Selected Papers*, pages 20–35, 2015.

[MSS16] Alfred Menezes, Palash Sarkar, and Shashank Singh. Challenges with assessing the impact of NFS advances on the security of pairing-based cryptography. *IACR Cryptology ePrint Archive*, 2016:1102, 2016.

[MV13] Atefeh Mashatan and Serge Vaudenay. A Fully Dynamic Universal Accumulator. *Proceedings of the Romanian Academy*, 14:269–285, 2013.

[NA14] David Nuñez and Isaac Agudo. Blindidm: A privacy-preserving approach for identity management as a service. *Int. J. Inf. Sec.*, 13(2):199–215, 2014.

[NAL12] David Nuñez, Isaac Agudo, and Javier Lopez. Integrating openid with proxy re-encryption to enhance privacy in cloud-based identity services. In *4th IEEE International Conference on Cloud Computing Technology and Science Proceedings, CloudCom 2012, Taipei, Taiwan, December 3-6, 2012*, pages 241–248, 2012.

[NAL15] David Nuñez, Isaac Agudo, and Javier Lopez. A parametric family of attack models for proxy re-encryption. In *IEEE 28th Computer Security Foundations Symposium, CSF 2015, Verona, Italy, 13-17 July, 2015*, pages 290–301, 2015.

[NAL16] David Nuñez, Isaac Agudo, and Javier Lopez. On the application of generic cca-secure transformations to proxy re-encryption. *Security and Communication Networks*, 9(12):1769–1785, 2016.

[Ngu05] Lan Nguyen. Accumulators from bilinear pairings and applications. In *Topics in Cryptology - CT-RSA 2005, The Cryptographers' Track at the RSA Conference 2005, San Francisco, CA, USA, February 14-18, 2005, Proceedings*, pages 275–292, 2005.

[NS04] Lan Nguyen and Reihaneh Safavi-Naini. Efficient and provably secure trapdoor-free group signature schemes from bilinear pairings. In *Advances in Cryptology - ASIACRYPT 2004, 10th International Conference on the Theory and Application of Cryptology and Information Security, Jeju Island, Korea, December 5-9, 2004, Proceedings*, pages 372–386, 2004.

[Nyb96a] Kaisa Nyberg. Commutativity in Cryptography. In *1st International Trier Conference in Functional Analysis*. Walter Gruyter & Co, 1996.

[Nyb96b] Kaisa Nyberg. Fast accumulated hashing. In *Fast Software Encryption, Third International Workshop, Cambridge, UK, February 21-23, 1996, Proceedings*, pages 83–87, 1996.

[OP01] Tatsuaki Okamoto and David Pointcheval. The gap-problems: A new class of problems for the security of cryptographic schemes. In *Public Key Cryptography, 4th International Workshop on Practice and Theory in Public Key Cryptography, PKC 2001, Cheju Island, Korea, February 13-15, 2001, Proceedings*, pages 104–118, 2001.

[Ora] Oracle. Java™ Cryptography Architecture (JCA) Reference Guide. `http://docs.oracle.com/javase/7/docs/technotes/guides/security/crypto/CryptoSpec.html`.

[PB10] Kun Peng and Feng Bao. Vulnerability of a non-membership proof scheme. In *SECRYPT 2010 - Proceedings of the International Conference on Security and Cryptography, Athens, Greece, July 26-28, 2010, SECRYPT is part of ICETE - The International Joint Conference on e-Business and Telecommunications*, pages 419–422, 2010.

[PK14] Henrich Christopher Pöhls and Markus Karwe. Redactable signatures to control the maximum noise for differential privacy in the smart grid. In *Smart Grid Security - Second International Workshop, SmartGridSec 2014, Munich, Germany, February 26, 2014, Revised Selected Papers*, pages 79–93, 2014.

[PS14] Henrich Christopher Pöhls and Kai Samelin. On Updatable Redactable Signatures. In *Applied Cryptography and Network Security - 12th International Conference, ACNS 2014, Lausanne, Switzerland, June 10-13, 2014. Proceedings*, pages 457–475, 2014.

[PS15] Henrich Christopher Pöhls and Kai Samelin. Accountable Redactable Signatures. In *10th International Conference on Availability, Reliability and Security, ARES 2015, Toulouse, France, August 24-27, 2015*, pages 60–69, 2015.

[PS16] David Pointcheval and Olivier Sanders. Short randomizable signatures. In *Topics in Cryptology - CT-RSA 2016 - The Cryptographers' Track at the RSA Conference 2016, San Francisco, CA, USA, February 29 - March 4, 2016, Proceedings*, pages 111–126, 2016.

[PSdMP12] Henrich Christopher Pöhls, Kai Samelin, Hermann de Meer, and Joachim Posegga. Flexible redactable signature schemes for trees - extended security model and construction. In *SECRYPT 2012 - Proceedings of the International Conference on Security and Cryptography, Rome, Italy, 24-27 July, 2012*, pages 113–125, 2012.

[PSP11] Henrich Christopher Pöhls, Kai Samelin, and Joachim Posegga. Sanitizable signatures in XML signature - performance, mixing properties, and revisiting the property of transparency. In *Applied Cryptography and Network Security - 9th International Conference, ACNS 2011, Nerja, Spain, June 7-10, 2011. Proceedings*, pages 166–182, 2011.

[PSPdM12] H. C. Pöhls, K. Samelin, J. Posegga, and H. de Meer. Length-hiding redactable signatures from one-way accumulators in $O(n)$. Technical report, 2012.

[PZ13] Christian Paquin and Greg Zaverucha. U-Prove Cryptographic Specification V1.1, Revision 3. Technical report, Microsoft Corporation, 2013.

[RN10] Vibhor Rastogi and Suman Nath. Differentially private aggregation of distributed time-series with transformation and encryption. In *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2010, Indianapolis, Indiana, USA, June 6-10, 2010*, pages 735–746, 2010.

[Rot11] Ron Rothblum. Homomorphic encryption: From private-key to public-key. In *Theory of Cryptography - 8th Theory of Cryptography Conference, TCC 2011, Providence, RI, USA, March 28-30, 2011. Proceedings*, pages 219–234, 2011.

[RST01] Ronald L. Rivest, Adi Shamir, and Yael Tauman. How to leak a secret. In *Advances in Cryptology - ASIACRYPT 2001, 7th International Conference on the Theory and Application of Cryptology and Information Security, Gold Coast, Australia, December 9-13, 2001, Proceedings*, pages 552–565, 2001.

[RY07] Thomas Ristenpart and Scott Yilek. The power of proofs-of-possession: Securing multiparty signatures against rogue-key attacks. In *Advances in Cryptology - EUROCRYPT 2007, 26th Annual International Conference on the Theory and Applications*

of Cryptographic Techniques, Barcelona, Spain, May 20-24, 2007, Proceedings, pages 228–245, 2007.

[RY16] Leonid Reyzin and Sophia Yakoubov. Efficient asynchronous accumulators for distributed PKI. In *Security and Cryptography for Networks - 10th International Conference, SCN 2016, Amalfi, Italy, August 31 - September 2, 2016, Proceedings*, pages 292–309, 2016.

[San99] Tomas Sander. Efficient accumulators without trapdoor extended abstract. In *Information and Communication Security, Second International Conference, ICICS'99, Sydney, Australia, November 9-11, 1999, Proceedings*, pages 252–262, 1999.

[SBWP03] Ron Steinfeld, Laurence Bull, Huaxiong Wang, and Josef Pieprzyk. Universal designated-verifier signatures. In *Advances in Cryptology - ASIACRYPT 2003, 9th International Conference on the Theory and Application of Cryptology and Information Security, Taipei, Taiwan, November 30 - December 4, 2003, Proceedings*, pages 523–542, 2003.

[SBZ01] Ron Steinfeld, Laurence Bull, and Yuliang Zheng. Content Extraction Signatures. In *Information Security and Cryptology - ICISC 2001, 4th International Conference Seoul, Korea, December 6-7, 2001, Proceedings*, pages 285–304, 2001.

[Sch91] Claus-Peter Schnorr. Efficient signature generation by smart cards. *J. Cryptology*, 4(3):161–174, 1991.

[Sch17] Berry Schoenmakers. Lecture notes cryptographic protocols. Version 1.3, 2017.

[SCR+11] Elaine Shi, T.-H. Hubert Chan, Eleanor G. Rieffel, Richard Chow, and Dawn Song. Privacy-preserving aggregation of time-series data. In *Proceedings of the Network and Distributed System Security Symposium, NDSS 2011, San Diego, California, USA, 6th February - 9th February 2011*, 2011.

[Sho94] Peter W. Shor. Polynominal time algorithms for discrete logarithms and factoring on a quantum computer. In *Algorithmic Number Theory, First International Symposium, ANTS-I, Ithaca, NY, USA, May 6-9, 1994, Proceedings*, page 289, 1994.

[Sho97] Victor Shoup. Lower bounds for discrete logarithms and related problems. In *Advances in Cryptology - EUROCRYPT '97, International Conference on the Theory and Application of Cryptographic Techniques, Konstanz, Germany, May 11-15, 1997, Proceeding*, pages 256–266, 1997.

[Sho04] Victor Shoup. Sequences of Games: A Tool for Taming Complexity in Security Proofs. *IACR Cryptology ePrint Archive*, 2004:332, 2004.

[SKZ13] Klaus Stranacher, Vesna Krnjic, and Thomas Zefferer. Trust and reliability for public sector data. In *International Conference on e-Business and e-Government*, volume 73, pages 124 – 132, 2013.

[Sla12] Daniel Slamanig. Dynamic Accumulator Based Discretionary Access Control for Outsourced Storage with Unlinkable Access - (Short Paper). In *Financial Cryptography and Data Security - 16th International Conference, FC 2012, Kralendijk, Bonaire, Februray 27-March 2, 2012, Revised Selected Papers*, pages 215–222, 2012.

[SNF11] Amang Sudarsono, Toru Nakanishi, and Nobuo Funabiki. Efficient proofs of attributes in pairing-based anonymous credential system. In *Privacy Enhancing Technologies - 11th International Symposium, PETS 2011, Waterloo, ON, Canada, July 27-29, 2011. Proceedings*, pages 246–263, 2011.

[SPB+12a] Kai Samelin, Henrich Christopher Pöhls, Arne Bilzhause, Joachim Posegga, and Hermann de Meer. On Structural Signatures for Tree Data Structures. In *Applied Cryptography and Network Security - 10th International Conference, ACNS 2012, Singapore, June 26-29, 2012. Proceedings*, pages 171–187, 2012.

[SPB+12b] Kai Samelin, Henrich Christopher Pöhls, Arne Bilzhause, Joachim Posegga, and Hermann de Meer. Redactable signatures for independent removal of structure and content. In *Information Security Practice and Experience - 8th International Conference, ISPEC 2012, Hangzhou, China, April 9-12, 2012. Proceedings*, pages 17–33, 2012.

[SR10] Daniel Slamanig and Stefan Rass. Generalizations and extensions of redactable signatures with applications to electronic healthcare. In *Communications and Multimedia Security, 11th IFIP TC 6/TC 11 International Conference, CMS 2010, Linz, Austria, May 31 - June 2, 2010. Proceedings*, pages 201–213, 2010.

[SS08] Siamak Fayyaz Shahandashti and Reihaneh Safavi-Naini. Construction of universal designated-verifier signatures and identity-based signatures from standard signatures. In *Public Key Cryptography - PKC 2008, 11th International Workshop on Practice and Theory in Public-Key Cryptography, Barcelona, Spain, March 9-12, 2008. Proceedings*, pages 121–140, 2008.

[SS16] Palash Sarkar and Shashank Singh. New complexity trade-offs for the (multiple) number field sieve algorithm in non-prime fields.

In *Advances in Cryptology - EUROCRYPT 2016 - 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Vienna, Austria, May 8-12, 2016, Proceedings, Part I*, pages 429–458, 2016.

[SSE+12] Yusuke Sakai, Jacob C. N. Schuldt, Keita Emura, Goichiro Hanaoka, and Kazuo Ohta. On the Security of Dynamic Group Signatures: Preventing Signature Hijacking. In *Public Key Cryptography - PKC 2012 - 15th International Conference on Practice and Theory in Public Key Cryptography, Darmstadt, Germany, May 21-23, 2012. Proceedings*, pages 715–732, 2012.

[SSU14] Daniel Slamanig, Raphael Spreitzer, and Thomas Unterluggauer. Adding Controllable Linkability to Pairing-Based Group Signatures for Free. In *Information Security - 17th International Conference, ISC 2014, Hong Kong, China, October 12-14, 2014. Proceedings*, pages 388–400, 2014.

[SSZ14] Daniel Slamanig, Klaus Stranacher, and Bernd Zwattendorfer. User-Centric Identity as a Service-Architecture for eIDs with Selective Attribute Disclosure. In *19th ACM Symposium on Access Control Models and Technologies, SACMAT '14, London, ON, Canada - June 25 - 27, 2014*, pages 153–164, 2014.

[STY00] Tomas Sander, Amnon Ta-Shma, and Moti Yung. Blind, auditable membership proofs. In *Financial Cryptography, 4th International Conference, FC 2000 Anguilla, British West Indies, February 20-24, 2000, Proceedings*, pages 53–71, 2000.

[Swe02] Latanya Sweeney. Achieving k-anonymity privacy protection using generalization and suppression. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 10(5):571–588, 2002.

[TW14] Stefano Tessaro and David A. Wilson. Bounded-collusion identity-based encryption from semantically-secure public-key encryption: Generic constructions with short ciphertexts. In *Public-Key Cryptography - PKC 2014 - 17th International Conference on Practice and Theory in Public-Key Cryptography, Buenos Aires, Argentina, March 26-28, 2014. Proceedings*, pages 257–274, 2014.

[TX03] Gene Tsudik and Shouhuai Xu. Accumulating Composites and Improved Group Signing. In *Advances in Cryptology - ASIACRYPT 2003, 9th International Conference on the Theory and Application of Cryptology and Information Security, Taipei, Taiwan, November 30 - December 4, 2003, Proceedings*, pages 269–286, 2003.

[UW14] Thomas Unterluggauer and Erich Wenger. Efficient pairings and ECC for embedded systems. In *Cryptographic Hardware and Embedded Systems - CHES 2014 - 16th International Workshop, Busan, South Korea, September 23-26, 2014. Proceedings*, pages 298–315, 2014.

[Ver06] Damien Vergnaud. New extensions of pairing-based signatures into universal designated verifier signatures. In *Automata, Languages and Programming, 33rd International Colloquium, ICALP 2006, Venice, Italy, July 10-14, 2006, Proceedings, Part II*, pages 58–69, 2006.

[Wat05] Brent Waters. Efficient identity-based encryption without random oracles. In *Advances in Cryptology - EUROCRYPT 2005, 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Aarhus, Denmark, May 22-26, 2005, Proceedings*, pages 114–127, 2005.

[WB15] Michael Walfish and Andrew J. Blumberg. Verifying computations without reexecuting them. *Commun. ACM*, 58(2):74–84, 2015.

[WHT+12] Zhen Yu Wu, Chih-Wen Hsueh, Cheng-Yu Tsai, Feipei Lai, Hung-Chang Lee, and Yu-Fang Chung. Redactable signatures for signed CDA documents. *J. Medical Systems*, 36(3):1795–1808, 2012.

[WWP07] Peishun Wang, Huaxiong Wang, and Josef Pieprzyk. A new dynamic accumulator for batch updates. In *Information and Communications Security, 9th International Conference, ICICS 2007, Zhengzhou, China, December 12-15, 2007, Proceedings*, pages 98–112, 2007.

[XXW+16] Peng Xu, Jun Xu, Wei Wang, Hai Jin, Willy Susilo, and Deqing Zou. Generally hybrid proxy re-encryption: A secure data sharing among cryptographic clouds. In *Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security, AsiaCCS 2016, Xi'an, China, May 30 - June 3, 2016*, pages 913–918, 2016.

[XY04] Shouhuai Xu and Moti Yung. Accountable ring signatures: A smart card approach. In *Smart Card Research and Advanced Applications VI, IFIP 18th World Computer Congress, TC8/WG8.8 & TC11/WG11.2 Sixth International Conference on Smart Card Research and Advanced Applications (CARDIS), 22-27 August 2004, Toulouse, France*, pages 271–286, 2004.

[YSL10] Dae Hyun Yum, Jae Woo Seo, and Pil Joong Lee. Trapdoor Sanitizable Signatures Made Easy. In *Applied Cryptography and Network Security, 8th International Conference, ACNS 2010, Beijing, China, June 22-25, 2010. Proceedings*, pages 53–68, 2010.

[ZKP17] Yupeng Zhang, Jonathan Katz, and Charalampos Papamanthou. An expressive (zero-knowledge) set accumulator. In *IEEE European Symposium on Security and Privacy, EuroS&P 2017, Paris, France, April 26-28, 2017*, 2017.

[ZS13] Bernd Zwattendorfer and Daniel Slamanig. On privacy-preserving ways to porting the austrian eid system to the public cloud. In *Security and Privacy Protection in Information Processing Systems - 28th IFIP TC 11 International Conference, SEC 2013, Auckland, New Zealand, July 8-10, 2013. Proceedings*, pages 300–314, 2013.

[ZSSH14] Bernd Zwattendorfer, Daniel Slamanig, Klaus Stranacher, and Felix Hörandner. A federated cloud identity broker-model for enhanced privacy via proxy re-encryption. In *Communications and Multimedia Security - 15th IFIP TC 6/TC 11 International Conference, CMS 2014, Aveiro, Portugal, September 25-26, 2014. Proceedings*, pages 92–103, 2014.