

Christian Maierhofer

User Centered and Privacy Preserving Identity Management

An Austrian Use Case

Graz University of Technology

IAIK

Head: Univ.-Prof. Dipl.-Ing. Dr.techn. Reinhard Posch

Supervisor: Dipl.-Ing. Dr.techn. Klaus Stranacher

Graz, October 2014

This document is set in Palatino, compiled with pdfL^AT_EX2e and Biber.

The L^AT_EX template from Karl Voit is based on KOMA script and can be found online: <https://github.com/novoid/LaTeX-KOMA-template>

Statutory Declaration

I declare that I have authored this thesis independently, that I have not used other than the declared sources/resources, and that I have explicitly marked all material which has been quoted either literally or by content from the used sources.

Graz, _____
Date

Signature

Eidesstattliche Erklärung¹

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.

Graz, am _____
Datum

Unterschrift

¹Beschluss der Curricula-Kommission für Bachelor-, Master- und Diplomstudien vom 10.11.2008; Genehmigung des Senates am 1.12.2008

Zusammenfassung

Identifizierung und Authentifizierung, unter Verwendung von qualifizierten Identitätsdaten, bilden heutzutage die Grundlage für Abläufe im Bereich des eGovernments. Die momentane Umsetzung in Österreich basiert auf einem elektronischen Identitätskonzept namens Bürgerkarte. Bei diesem Ansatz werden sämtliche Identitätsdaten auf der Bürgerkarte an den Serviceprovider weitergegeben, unabhängig, ob sie für den Zugriff auf die angeforderten Ressourcen benötigt werden. Das Ziel dieser Masterarbeit ist die Implementierung eines benutzerzentrierten und die Privatsphäre schützenden Identitätsmanagementmodells. Das implementierte Modell ermöglicht einerseits die Auswahl der zu übertragenden Identitätsattribute und andererseits werden diese Daten vom Bürger bis zum Serviceprovider nur noch verschlüsselt übertragen. Dieses Konzept ermöglicht den Betrieb des Identitätsproviders in der Cloud und bringt so die Vorteile der Skalierbarkeit und Verfügbarkeit mit sich.

Abstract

Qualified identification and authentication are one of the main pillars of eGovernment processes nowadays. The current implementation in Austria is based on an eID concept called the Austrian Citizen Card. In the current approach the identity data, that is stored on the Citizen Card, is sent to the the service provider for identification even if not all of the data is processed or needed by the service provider to grant access to a requested resource. This work focuses on a proof of concept implementation of a user-centered and privacy preserving identity management model. The implemented model provides selective identity disclosure, thus enabling a citizen to forward qualified identity data without the need to reveal all of her privacy-sensitive data to a service provider. The model, this thesis is based on, also enables the deployment of the identity provider in a semi-trusted environment, like the public cloud, while still preserving privacy. Deploying applications within the cloud results in a high level of availability and scalability which are important factors within the Austrian eGovernment infrastructure.

Contents

Abstract	v
1 Introduction	1
1.1 Motivation	1
1.2 Outline	1
2 Electronic Signatures	3
2.1 Introduction	3
2.2 Conventional Digital Signatures	4
2.2.1 Signature Creation and Verification	4
2.2.2 RSA Algorithm	6
2.2.3 Certificate Validation	7
2.3 Modifiable Signatures	9
2.3.1 Redactable Signatures	11
2.3.2 Sanitizable Signatures	13
2.4 Conclusion	18
3 Identity Management in Austria	19
3.1 Introduction	19
3.2 Identity Management	20
3.3 Electronic Identity - Austrian Citizen Card	22
3.3.1 Legal Framework	24
3.3.2 Citizen Card Features	25
3.3.3 Identity Link and sPIN	26
3.4 Austrian Identity Provider	27
3.5 Conclusion	32
4 User-Centered and Privacy-Preserving Model	35
4.1 Introduction	35

Contents

4.2	Model	35
4.3	Blank Digital Signatures	36
4.3.1	Introduction	36
4.3.2	Principle	37
4.3.3	The Scheme	39
4.3.4	Properties	40
4.4	Proxy Re-encryption	41
4.4.1	Introduction	41
4.4.2	Principle	42
4.4.3	The Scheme	42
4.4.4	Properties	44
4.5	Conclusion	44
5	Implementation	47
5.1	Introduction	47
5.2	Demo Application SRA/CSP	48
5.2.1	Registration Process	48
5.3	Identification Process	51
5.4	Modifications in MOCCA	52
5.4.1	Implementation Details	52
5.4.2	Virtual Citizen Card	55
5.5	Modifications in MOA-ID	57
5.5.1	Implementation Details	57
5.6	Modifications for Online Applications	59
5.7	Identity Link Details	60
5.7.1	Signature Format	60
5.7.2	Citizen Certificate	60
5.8	Security Layer	62
5.9	Proxy Re-encryption Library	63
5.10	Conclusion	64
6	Evaluation	67
6.1	Introduction	67
6.2	Cloud Deployment of the IdP	68
6.2.1	Private Environment Architecture	68
6.2.2	Cloud Computing	70
6.2.3	Jelastic	72

Contents

6.2.4	Security Threats	74
6.3	Hardware Platform Evaluation	82
6.3.1	TrustZone	83
6.3.2	Conclusion Hardware Platform	86
7	Summary and Conclusions	89
7.1	Summary	89
7.2	Conclusions	90
8	Acronyms	93
9	Apendix	95
9.1	BDS-Message Signature Format	95
9.2	Identity Link of the current eID Solution	97
	Bibliography	101

List of Figures

2.1	Signature Generation Process	5
2.2	Signature Verification Process	6
2.3	Certificate Chain Validation	10
2.4	Redactable Signatures	14
2.5	Sanitizable Signatures	17
3.1	Identity Management Components	20
3.2	Identity Management Models	23
3.3	ssPIN Calculation based on sPIN and Sector Identifier	28
3.4	Sequence Diagram Identification Procedure.	31
4.1	User-Centered and Privacy-Preserving Model	36
4.2	Principle of the Blank Digital Signature Scheme	38
4.3	Principle of a Proxy Re-encryption Scheme	43
5.1	Sequence Diagram BDS identification	53
5.2	MOCCA Block Diagram	56
5.3	Login Block Diagram	58
5.4	XAdES Signature Block	61
6.1	Apache Tomcat and Webserver Architecture	69
6.2	The Jelastic Environment Configuration	73
6.3	The Jelastic Environment Configuration	74
6.4	Ensure Binding between IDL and AuthBlock	76
6.5	CAmazon Web Services CloudHSM Architecture	78
6.6	Hybrid IdM Model using a HSM	80
6.7	The Principle of the ARM TrustZone	84
6.8	Concept CC-App on TrustZone	85

1 Introduction

1.1 Motivation

Qualified identification and authentication play a fundamental role within eGovernment processes nowadays. In Austria an electronic identity (eID) concept, called the Austrian Citizen Card, is used. The current approach foresees a deployment of the so-called identity provider within a private environment normally provided by the service provider. This identity provider is responsible for carrying out qualified identification and authentication based on a citizen's identity data. Within the last decade it became popular to move data and applications into the cloud. Even if the cloud offers many advantages, like scalability and availability, current cloud solutions provide no provable or certified security of the data that is stored in the cloud at all. To mitigate this issue this work focuses on a centralized public cloud deployment of the identity provider that may be used by citizens for identification at online applications. The user centered model implemented within this thesis provides to important features. On the one hand it provides selective identity disclosure by enabling a citizen to select which identity attributes may be forwarded to service provider. On the other hand it preserves privacy by enabling processing of encrypted identity data all the way from the citizen to the service provider, while still providing qualified identification and authentication.

1.2 Outline

Including this introduction chapter this thesis consists of 7 chapters. Chapter 2 will provide information regarding electronic signatures. At first con-

1 Introduction

ventional digital signature creation and verification, the RSA algorithm and details regarding the public key certificate verification are discussed. The second part of chapter 2 presents another type of signatures, called modifiable signature schemes and will explain redactable and sanitizable signatures in more detail. Modifiable signature scheme provide a mechanism to modify signed data in a specified way, without breaking the originally applied signature.

Chapter 3 provides an overview of different identity management models and detailed information regarding the current Austrian eID concept. This chapter also includes basics regarding the legal framework for identification and authentication in Austria as well as implementation details about the Citizen Card and the identity provider.

After the current Austrian eID solution has been discussed, chapter 4 provides the identity management model this thesis is based on. Beneath a short model description, this chapter also includes details regarding two cryptographic schemes used within this thesis. The first one are blank digital signatures which are used as redactable signatures within this thesis to enable selective identity disclosure. The second scheme is a proxy re-encryption scheme used for processing encrypted identity data from the citizen up to the service provider.

Chapter 5 presents the implementation details. It provides details about the modifications and extensions that had to be carried out within the different Austrian eID software components. The target of the implementation was to implement the model presented within chapter 4.

Chapter 6 will address different cloud deployment issues that have to be considered when deploying applications within the public cloud and it provides some solutions to this issues. This chapter also evaluates a hardware platform that may be used as an underlying platform for the Citizen Card concept in the future.

Chapter 7 finally will draw a conclusion over the whole thesis and will provide ideas for future work.

2 Electronic Signatures

2.1 Introduction

With the transition from paper-based to electronic data in the last two to three decades new challenges regarding IT-security arose. While handwritten signatures are sufficient within a legal point of view for paper-based documents an equivalent for digital data had to be found. Basically a signature, whether handwritten or electronic, must provide the following three security properties:

- **Authenticity** Signatures ensure that no one is able to impersonate the originator of a signed document.
- **Integrity** Signatures ensure that a modification of the document, after it has been signed, has to be detectable within the verification process.
- **Non-repudiation** The originator of the document is not able to deny that she has signed the document.

As [Fillingham, 1997] pointed out in his work the properties **authenticity** and **non-repudiation** are provided by handwritten signatures, e.g. a seller may compare the signature on the back of a buyer's credit card with the buyer's signature on the bill. But the **integrity** of a document may not be guaranteed by the signature itself. Even if indelible ink or tamper-evident paper is used the level of data integrity of a hand-signed document stays very low. A common way to provide a basic level of integrity is to issue a copy of the signed document to the involved entities for comparison against the original. The electronic equivalent to a handwritten signature, that provides the three properties mentioned above, is called digital signature and will be discussed in the following section.

2.2 Conventional Digital Signatures

Digital signatures are based on public key cryptography also known as asymmetric cryptography. In contrast to symmetric key cryptography, where the same key is used for encryption and decryption, every entity that wants to share data with other entities in a secure manner, has to create a key pair consisting of a private and a public key. The private key has to be kept private by the signatory and the public key can be given out to everybody else. The public key may be wrapped in a special data structure called certificate and signed by a trusted authority usually called certification authority (CA). This certificate is then published for encryption and signature verification processes using the so-called public key infrastructure (PKI). The certificate itself contains, beneath the public key, additional information like the certificate validity period, the signatory identifier and the signature value calculated over the certificate based on the CA's private signing key. This information is used for the certificate validation that will be discussed in section 2.2.3. The following section will provide basic informations regarding the operations provided by public key cryptography.

2.2.1 Signature Creation and Verification

There are two main use cases for public key cryptography, data encryption and decryption on the one hand and signature creation and verification on the other hand. Since within this master thesis only the signature operations are used this section will describe the signature generation process and the signature verification process in more detail. Figure 2.1 shows a block diagram containing the main building blocks for the signature generation.

The input data block represents the data the signature has to be calculated for. In theory it would be possible to use this data as input for the signature creation function, but in terms of performance it is more efficient to first calculate a fixed length value representing the input data and sign this value. This is done by using a hash function like SHA-3 [Bertoni et al., 2011] or similar. A hash function is a one-way function, which means that it is not invertible. Another important property of a good hash function is that a single flipping bit on the input should lead to a large variation of the

2.2 Conventional Digital Signatures

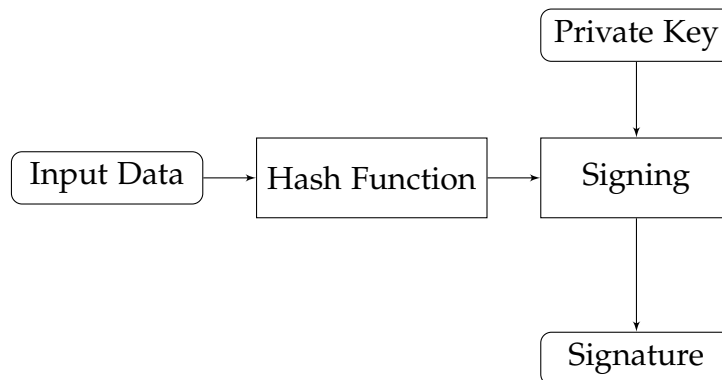


Figure 2.1: Signature Generation Process

output value, so it is much easier to (visually) compare two hash values belonging to two different documents than to compare the two documents themselves especially if they only differ in e.g. one letter. The hash value always has the same length, no matter how long the input data was, and in most cases it is much shorter than the document itself which enables faster signature calculation. In the next step the hash value is signed using the private key of the signatory in combination with an appropriate algorithm like RSA [Rivest, Shamir, and Adleman, 1978]. The RSA algorithm will be described in more detail in one of the next sections. The output value of this algorithm represents the the digital signature that is bound to the input data and the signatory holding the private key.

The complement to the signing function is the signature verification function that is presented in figure 2.2. This verification process may be applied by every entity receiving

- the original document,
- the signature and
- the public key (certificate)

At first the hash value is revealed from the signature value by using the signatory's public key (certificate) revealing the hash value 1. Then the original document is hashed, using the same hash function as in the signature creation procedure, resulting in the hash value 2. In the last step this two values are compared to each other. If they match the signature is valid, if not

2 Electronic Signatures

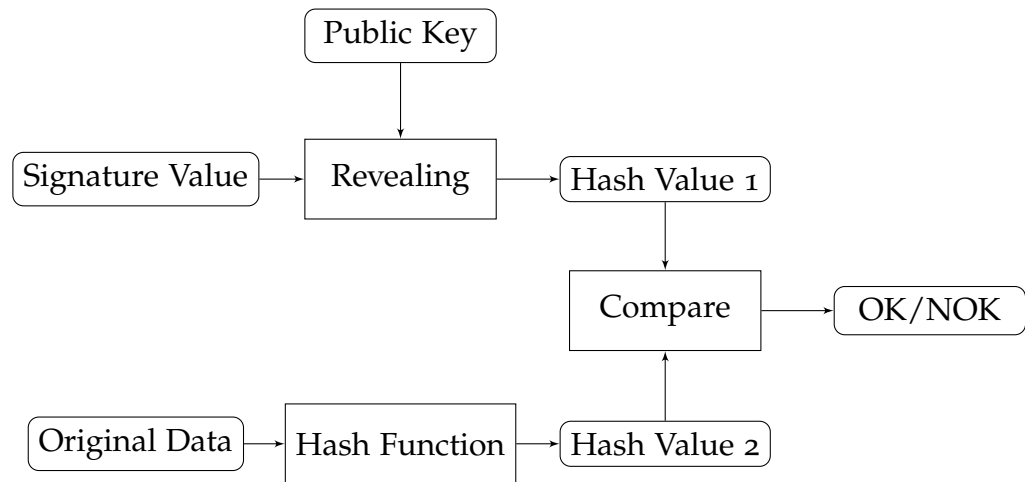


Figure 2.2: Signature Verification Process

the signature is not valid. Either the document was modified or the public key does not belong to the entity the signature was created by. Because the public key certificate is directly involved within the signature verification process, it is, beneath the cryptographic check described before, also necessary to perform a certificate validation which will be described within section 2.2.2. The following section now provides the basic principle of the RSA algorithm that may be used for signature creation and verification.

2.2.2 RSA Algorithm

This algorithm was invented by [Rivest, Shamir, and Adleman, 1978] and even if it has been invented more than 30 years ago it is still used within cryptographic systems. The RSA algorithm basically defines the steps key generation, signature creation and verification that will be described in more detail here.

- **Key Generation**

- two large prime numbers are generated, namely p and q
- the product $n = p \cdot q$ is calculated
- the Euler ϕ function is calculated via $\phi(n) = (p - 1) \cdot (q - 1)$

2.2 Conventional Digital Signatures

- a number e is chosen so that e and $\phi(n)$ are coprime.
- a number d that fulfills $e \cdot d \equiv 1 \text{ modulo } (\phi(n))$

The tuple (e, n) represent the public key and (d, n) represent the private key.

- **Signature Generation**

- a message m is created
- a hash value $h = \text{HASH}(m)$ is calculated
- the signature $s = h^d \text{ modulo } (n)$ is calculated
- the message m and the signature s are forwarded to the verifying entity

- **Signature Verification**

- the verifying entity receives the message m and the signature s
- the hash h is revealed using the senders public key by calculating $h = s^e \text{ modulo } (n) = (h^d)^e \text{ modulo } (n) = h^{d \cdot e} \text{ modulo } (n) = h$
- the hash value is calculated over the message m by $h_{calc} = \text{HASH}(m)$
- if the revealed hash h and the calculated value h_{calc} are equal the signature is valid

The security of this cryptographic system is based on the problem of factoring a product of two large prime numbers if only the product is known. If an attacker is able to factor the product n , thus revealing p and q , she is able to calculate $\phi(n)$ thus the secret key d . More details regarding the security of RSA can be found in [Rivest, Shamir, and Adleman, 1978]. In this section the basic mathematics used within the RSA algorithm have been presented. The next chapter contains another important part of the signature creation and verification process, namely the certificate validation.

2.2.3 Certificate Validation

Without ensuring the validity of the certificate holding the signatory's public key it would be useless to validate a signature belonging to a dedicated document. Basically anybody could have created a key pair, wrap the public key into a certificate and self-sign it. To circumvent this problem the process of certificate validation is applied. A certificate represents information about

2 Electronic Signatures

the person holding the secret key belonging to the public key within the certificate as well as signature meta information in a standardized format like X.509 defined by [Cooper et al., 2008]. Some of the important fields defined within this standard are

- **Issuer** - represents the entity that signed the certificate.
- **Subject** - represents the entity the certificate is issued to.
- **Serial number** - represents a unique identifier for every certificate issued by a certification authority (CA).
- **Validity** - defines the validity period of the certificate.
- **Public Key** - represents the public key belonging to the unique, private key in possession of the entity represented by the subject.
- **Signature** - represents the signature value calculated over the certificate with the issuer's private signing key.

The validation of a certificate consists of a few steps. The first step is to build a certificate chain up to a trusted root certificate as shown in Figure 2.3. A typical chain may consist of the user certificate at the lowest level, 0 to n intermediate certificates at the middle level and a self-signed, trusted root certificate at the top level. After the chain is built the following validation steps are performed.

- **Revocation Check** - A certificate may be revoked within the validity period. Reasons for a revocation may be a compromised private signing key or the invalidation of the data within a certificate, e.g. changed last name because of marriage. There are two main mechanisms for performing revocation checks:
 - Certificate revocation lists (CRL) - a list containing revoked certificates, identified by the serial number, revocation point of time and additional information defined in the RFC-5280 by [Cooper et al., 2008]. The list itself is also digitally signed by the publishing CA to verify its integrity.
 - Online certificate status protocol (OCSP) - the state of a certificate can be checked by issuing a request to a server, the so-called OCSP-responder defined in the RFC-2560 by [Myers et al., 1999]. The OCSP-responder returns a response containing the status of the requested certificate. This response is digitally signed by the OCSP-responder and thus it may be verified by the client.

2.3 Modifiable Signatures

In contrast to CRLs, that are only updated within regular intervals, an OCSP response will always contain up-to-date certificate revocation informations.

- **Time Validity Check** - Every certificate is only valid within a specified time frame. The two entries in the certificate defining the validity period are *notBefore* and *notAfter*. Based on the used model the the validity period is verified in different ways:
 - **Chain Model** - using this model every certificate within the chain has to be valid at the point of time it was used for creating it's signature. The model does not consider if a certificate within the chain has been invalidated since the signature creation.
 - **Shell Model (PKIX)** - the PKIX model is more restrictive. The involved signatures also have to be valid at the signature verification point of time. So a certificate that is valid within the the chain model does not have to be valid within the PKIX model.

The checks described above are applied to all the certificates within the chain. Only if all certificates within the chain contain valid signatures, are not revoked and the current date is within the validity period the result of the certificate verification process is positive.

2.3 Modifiable Signatures

As described in the former sections a main feature of standard digital signatures is to provide data integrity. This means that every modification of the signed data leads to a broken signature. Although in some fields of applications it may be useful if certain, predefined parts within a set of data could be changed or redacted without breaking the signature, e.g. in terms of privacy it could be necessary to redact some parts within a document containing privacy-sensitive data before it is forwarded to another entity. This problem is known as the Digital Document Sanitizing Problem, published by [Miyazaki et al., 2003]. The first modifiable (editable) signature schemes have been proposed more than ten years ago. The following sections will provide an overview regarding two approaches, redactable and sanitizable signatures. **Redactable** signatures are the most basic types of modifiable

2 Electronic Signatures

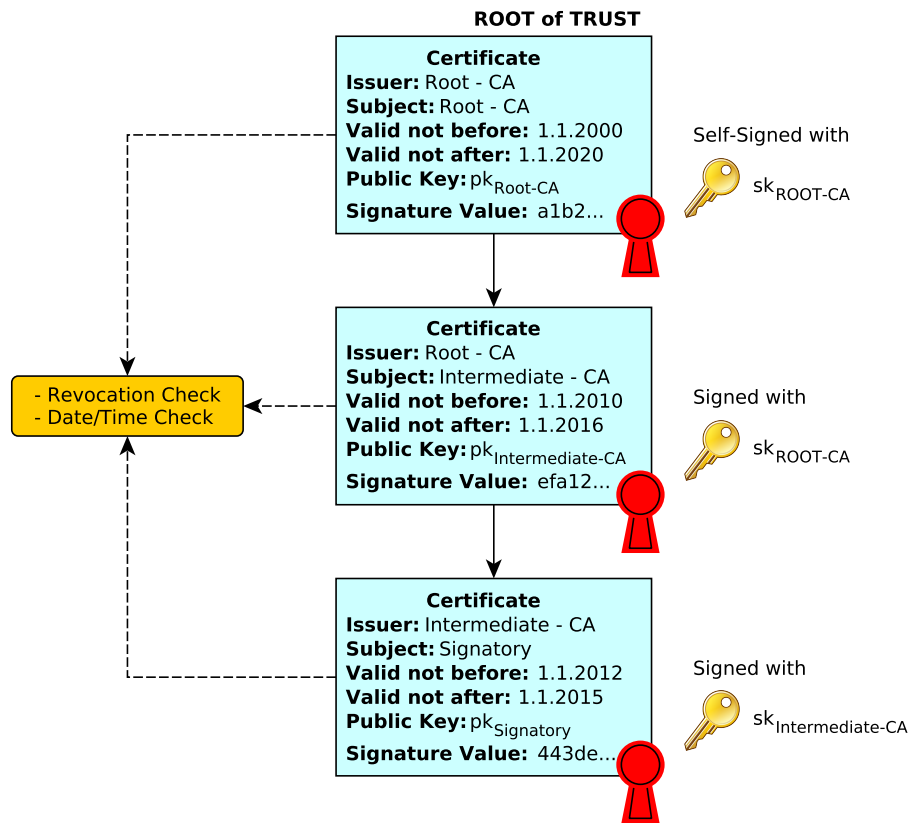


Figure 2.3: Certificate Chain Validation

signatures where **every entity** receiving the signed data is able to delete or exchange parts of it by a single character without invalidating the signature. The other type of modifiable signatures are **sanitizable** signatures where only **pre-defined entities** are able to modify **pre-defined parts** of data.

2.3.1 Redactable Signatures

This type of signatures were invented by [Johnson et al., 2002] and [Steinfeld, Bull, and Zheng, 2002] at the same time. The idea behind their model is to enable a person, called redactor or censor, to delete or replace sub-strings within a message by a special character without breaking the signature, so a third person will still be able to validate the signature. Figure 2.4 illustrates the basic working principle of this type of signatures. At first the whole message is split into blocks, e.g. every word could represent a block.

$$m = m_1, m_2, m_3 \dots m_n \quad (2.1)$$

In the second step a hash value is calculated for every block.

$$h_i = \text{HASH}(m_i) \text{ for } i = 1..n \quad (2.2)$$

Now a combined hash value is generated from the previous calculated ones by concatenating and hashing them again.

$$h_{combined} = \text{HASH}(\text{CONCAT}(h_1, h_2, \dots, h_n)) \quad (2.3)$$

This combined hash value is now digitally signed by the signatory using her secret key sk as shown in section 2.2 resulting in the Signature sk .

$$S_{orig} = \text{SIGN}_{sk}(h_{combined}) \quad (2.4)$$

Now the signature S_{orig} and the original message m are forwarded to the censor. The censor now has the possibility to redact some of the blocks within the message m resulting in a modified message $m_{modified}$. If the censor redacts a block m_x , by replacing it with a special character, like '*', she also has to store the original hash value h_x belonging to the original block value, the position x and the original signature S_{orig} within the adapted signature

2 Electronic Signatures

$S_{adapted}$. Now every entity in posses of $S_{adapted}$ and the modified message $m_{modified}$ is able to verify the validity of the document. The verifier gains no information about the original content of the block m_x except the hash value h_x . The verifier now executes the following steps. She calculates the hash values for every block, except for the blocks that have been redacted by the censor. This information can be extracted from the signature $S_{adapted}$.

$$h_i = HASH(m_i) \text{ for } i = 1..n \quad (2.5)$$

The hash value for block x is read from the signature $S_{adapted}$. Then the combined hash value is calculated as before by

$$h_{combinednew} = HASH(CONCAT(h_1, h_2, \dots, h_x, \dots, h_n)) \quad (2.6)$$

and compared to the original combined hash value $h_{combined}$, revealed from the signature S_{orig} using the signatory's public key pk

$$h_{combined} = REVEAL_{pk}(S_{orig}) \quad (2.7)$$

If $h_{combined}$ equals $h_{combinednew}$ the message signature is valid.

Although this scheme looks quite promising problems may arise if the message m contains some kind of predictable message blocks. An example could be questions that may only be answered by a simple 'yes' or 'no'. If an attacker is able to determine that only two possibilities to an answer are available she can determine the selected answer by simply calculating the two hash values for 'yes' and 'no' and compare it to hash value of the redacted block that is contained in the signature $S_{adapted}$. To circumvent this problem the usage of so-called commitments instead of hash functions was proposed by [Steinfeld, Bull, and Zheng, 2002]. As described by [Slamanig and Rass, 2011] a commitment provides to features:

- **Hiding** - the message may not be revealed from the commitment by anybody without the secret information.
- **Binding** - the message may not be changed by anybody, especially not by the creator, without the secret information.

Using the concept of commitments the process of creating redactable signatures changes as follows. The signatory generates a random value r_i for every message block m_i , calculates the commitments c_i by

$$c_i = HASH(CONCAT(m_i, r_i)) \text{ for } i = 1..n \quad (2.8)$$

and calculates the combined hash value by

$$h_{combined} = HASH(CONCAT(c_1, c_2, \dots, c_n)). \quad (2.9)$$

The signature S_{orig} is calculated as before over the hash value $h_{combined}$ using the signing key sk . In contrast to the simple approach the signature has to be extended to include the list of random values (randomizer) r_i resulting in the modified signature tuple S_{orig}^r where

$$S_{orig}^r = (S_{orig}, (r_1, r_2, \dots, r_n)). \quad (2.10)$$

If a censor now wants to redact a message block m_x she replaces the block by a '*' resulting in

$$m_{modified} = (m_1, m_2, \dots, m_x, \dots, m_n) \quad (2.11)$$

and she also replaces r_x by the value $h_x^r = HASH(CONCAT(m_x, r_x))$ in the signature resulting in

$$S_{adapted}^r = (S_{orig}, (r_1, r_2, \dots, h_x^r, \dots, r_n)). \quad (2.12)$$

The verification will not be explained here because it is pretty similar to the hash-only-based process described before. The commitment-based scheme solves the problem of guessing a message block according to a hash value because it is practically infeasible to reveal m_x from h_x^r without knowing r_x . A disadvantage arising with the scheme is lacking performance in terms of storage. The signature here has to contain, beneath the original signature value, a tuple containing the randomizer values. So the storage needed increases linear with the message blocks ($O(n)$). A better approach, discussed by [Slamanig and Rass, 2011], is to use a hash tree for storing the randomizer values which may reduce the storage to $O(\log(n))$.

2.3.2 Sanitizable Signatures

Sanitizable signatures were invented by [Ateniese et al., 2005]. In contrast to redactable signatures, where the censor is able to delete any parts of the message, in this model the parts that may be modified by the censor

2 Electronic Signatures

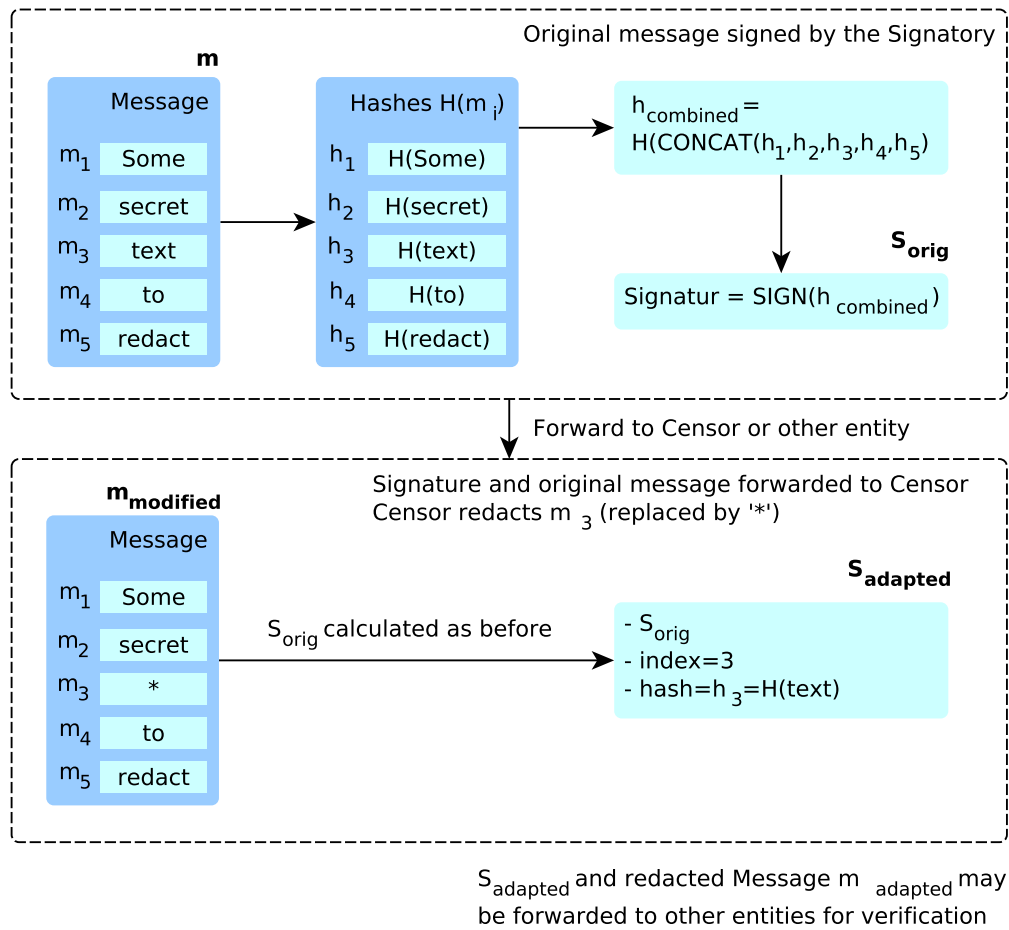


Figure 2.4: Redactable Signatures

2.3 Modifiable Signatures

are predefined by the signatory. Modification, in contrast to redactable signatures, here means not only the redaction of message blocks it means replacing blocks with any or predefined content. Additionally the signatory has to define the censor that is allowed to change the message without breaking the signature. The hash-based commitment used within redactable signatures is replaced by so-called chameleon hash functions invented by [Krawczyk and Rabin, 1998]. Chameleon hash functions involve a secret and public key belonging to a dedicated censor as well as randomizer. Without the knowledge of the secret key, chameleon hash functions are resistant against calculating collisions or pre-images. If the private key and the randomizer are known it is possible to calculate hash collisions in an efficient way. A commitment based on chameleon hash functions is calculated by

$$c_i = CHASH_{pk_{censor}}(m_i, r_i) \quad (2.13)$$

where c_i is the commitment, m_i the message block, r_i the randomizer, pk_{censor} the censor's public key and $CHASH_{pk_{censor}}$ the chameleon hash function under the censor's public key. The owner of the private key sk_{censor} is able to efficiently calculate hash collisions, meaning he is able to find a pair (m_i^*, r_i^*) with $(m_i^* \neq m_i)$ that fulfills the equation

$$c_i = CHASH_{pk_{censor}}(m_i, r_i) = CHASH_{pk_{censor}}(m_i^*, r_i^*) \quad (2.14)$$

in an efficient manner. Every entity in possess of c_i , (m_i, r_i) respectively (m_i^*, r_i^*) and the censors public key pk_{censor} is able to verify if c_i is a valid commitment to m_i respectively m_i^* .

Based on chameleon hash functions a sanitizable signature creation, shown in figure 2.5, requires the following steps: Key pairs have to be generated and public keys have to be published. This model utilizes the signatory's secret sk_{signer} and public key pk_{signer} as well as the censor's keys sk_{censor} and pk_{censor} . Again the message m is split into n blocks

$$m = m_1, m_2, m_3 \dots m_n \quad (2.15)$$

Using the concept of chameleon hash functions, also known as a commitment with trap door, the signatory first has to generate a random value r_i for every message block m_i and calculates the chameleon hash ch_i for the **sanitizable** message blocks

$$ch_i = CHASH_{pk_{censor}}(CONCAT(m_i, r_i)) \quad (2.16)$$

2 Electronic Signatures

and for the **non-sanitizable** message blocks

$$h_j = H(\text{CONCAT}(m_j, r_j)) \quad (2.17)$$

where $\text{CHASH}_{pk_{\text{censor}}}$ represents the chameleon hash function under the censor's public key and H represents a standard hash function like it was used within the redactable signature scheme. The combined hash value is a concatenation of all the hash values

$$h_{\text{combined}} = \text{CONCAT}(ch_i, h_j). \quad (2.18)$$

The signature S_{orig} is then calculated over the hash value h_{combined} using the signing key sk_{signer} . In this scheme the randomizer values r_i have to be added to the original signature, resulting in the modified signature tuple S_{orig}^r where

$$S_{\text{orig}}^r = (S_{\text{orig}}, (r_1, r_2, \dots, r_n)). \quad (2.19)$$

Now the message m and the signature S_{orig}^r are forwarded to the designated censor, in possession of the private key sk_{censor} . To keep the original hash values for the sanitizable blocks when the censor changes the content of these blocks she has to calculate chameleon hash collisions using her private key. So for a given c_j and a modified message block m_j^* she wants to set, she has to calculate

$$c_j = \text{CHASH}_{pk_{\text{censor}}}(m_j, r_j) = \text{CHASH}_{sk_{\text{censor}}}(m_j^*, r_j^*). \quad (2.20)$$

The randomizer value r_j^* also has to be included within the signature S_{orig}^r resulting in a new signature tuple

$$S_{\text{censored}}^r = (S_{\text{orig}}, (r_1, r_2, \dots, r_j^*, \dots, r_n)). \quad (2.21)$$

This tuple together with the censored message m is then forwarded to another entity that is able to verify the signature by calculating the standard hash value $H(\text{CONCAT}(m_j, r_j))$ over the fixed message blocks and the chameleon hash value $\text{CHASH}_{pk_{\text{censor}}}$ over the changed blocks. These hash values are concatenated resulting in h_{revealed} and the original hash value is revealed from S_{orig} using the signatory's public key pk_{signer} resulting in h_{combined} . The signature is valid, which means that it has not been modified or it has been censored by an authorized entity, if the hash values h_{revealed} and h_{combined} match.

2.3 Modifiable Signatures

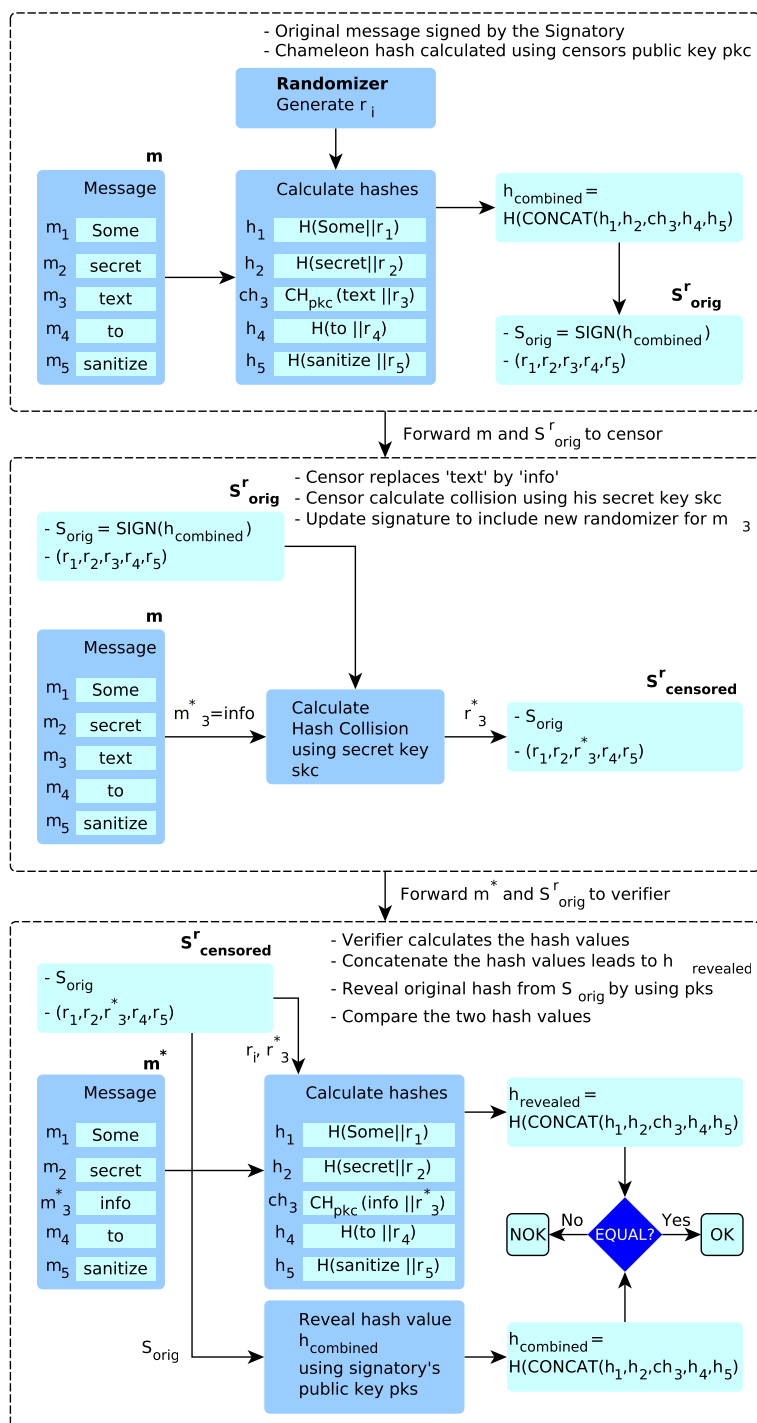


Figure 2.5: Sanitizable Signatures

2.4 Conclusion

This chapter provided a basic introduction into state of the art digital signatures as well as an insight into the more recent primitives of modifiable respectively editable signature schemes. While digital signatures are a powerful tool within today's information and communication technologies (ICT) and used extensively for e.g. electronic identification and authentication, file protection, secure email transfer, secure web access, etc., modifiable signatures are mainly used within the academic area and some specialized sections like eHealth, as for instance presented by [Slamanig and Rass, 2010]. This thesis will provide a basic concept how modifiable signatures may be used within identity management procedures to ensure privacy and authenticity. The implementation will be based on the current Austrian identity management system that will be described in more detail within the next chapter.

3 Identity Management in Austria

3.1 Introduction

In the last decade the number of online services in the private and public sector increased massively. Especially governmental processes have made a transition from face-to-face communication at public authorities to online procedures. The wealth of these procedures are combined under the term e-Government. As [Stranacher et al., 2013] pointed out in their paper, Austria is very dedicated in providing electronic governmental procedures to every citizen. As an example help.gv.at [help.gv.at, 2014] provides about 350 online forms and had over 1.2 million visitors a month in the year 2014. When participating in such online procedures a main focus has to be set on secure identification and authentication. The process of identification and authentication, the so called identity management, in Austria is based on a concept called the Austrian Citizen Card that represents the electronic identity within Austria. The cryptographic primitives used within the eID concept are based on standards like digital signature schemes and cryptographic hash functions. The qualified electronic signatures, based on a qualified certificate, are equivalent to handwritten signatures by law. This chapter will provide information regarding the identity management in Austria from a technical and a legal point of view as well as information regarding different identity management models that have been developed over the last years.

3.2 Identity Management

Identity management systems consist of some core components shown in figure 3.1. A **user** wants to access an **online application (OA)** provided by a **service provider (SP)**. The SP forwards an identification request for the requesting user to the **identity provider (IdP)** also known as identity assertion provider. From now on the IdP is responsible for managing the identification and authentication of the user that requested the resource at the SP. Depending on the configuration it may request a simple user-name password combination or even provide some kind of two-factor authentication based on a security token possessed by the user. After the identification and authentication process between the user and the IdP has been completed, the user's identity information is forwarded to the SP that may grant or deny access to the requested resource. Based on this basic procedure different identity management models have emerged over the last years. These different models have been presented and compared by [Zwattendorfer, Zefferer, and Stranacher, 2014] and a few of them will be presented in the following.

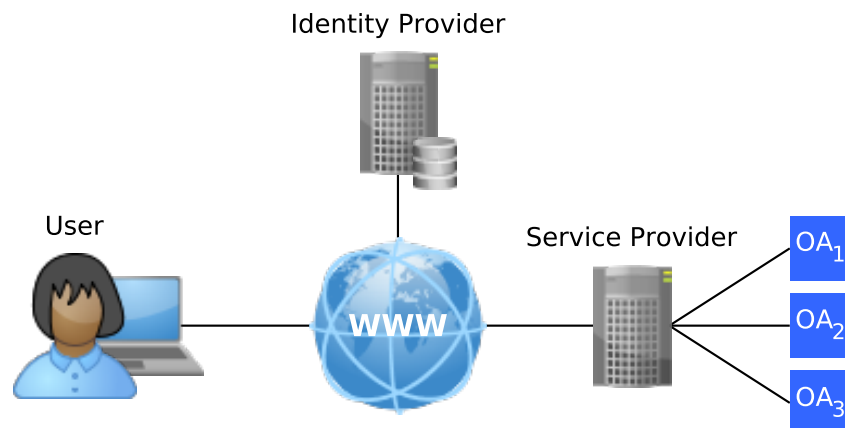


Figure 3.1: Identity Management Components

- The simplest model is the so-called **isolated model**. Here the SP and the IdP are merged into one domain as shown in figure 3.2. This

3.2 Identity Management

basically means that the identification and authentication process is controlled by the SP and the identity data is stored directly at the SP. Also identity management operations, like adding or removing new users, may only be carried out by this specific service provider. Another disadvantage is the fact that the identity data provided by this IdP can not be used for the identification at other SPs so a user has to register at multiple IdPs if she wants to access another SP's applications.

- The **central model** provides a architecture where multiple SPs use a shared central IdP, see figure 3.2. The IdP is responsible for user registration, identification and authentication. Identity data is stored within a central repository in control of the centralized IdP. If a user wants to access an OA run by a SP, the SP forwards the user to the IdP that manages the identification and authentication procedure. If the procedure succeeded a token containing the required identity information is assembled and forwarded to the SP that may grant or deny access based on this information.
- In contrast to the former models the **user-centric model**, shown in figure 3.2, foresees that the identity data is under sole control of the user, e.g. stored on a secure token like a smart card. So if the user wants to access the OA she is forwarded to the IdP. The IdP requests the identity data from the user and may perform some type of two-factor authentication based on a challenge-response protocol. Then the IdP may forward a token containing the identity information to the SP that may grant or deny access. The Austrian IdM solution is also based on the user-centric approach.

The following models are pretty similar to the former presented ones but they are deployed in the public cloud. The advantage of the cloud deployment is the scalability. In theory the cloud offers nearly unlimited hardware resources that may be assigned dynamically to every application based on the actual load.

- The **identity as a cloud model** represents the cloud-pendant to the isolated model with the only difference that the environment the SP and IdP run in is the public cloud. The big disadvantage of this model is that the organization moving the identity service to the public cloud

3 Identity Management in Austria

also loses the control over their identity data which leads to a big security issue in terms of data protection.

- The **identity to the cloud model** represents the cloud-variant to the central identity model. Here the IdP and the SP are separated entities but in contrast to the central model the SP and its OAs are deployed in the public cloud while the IdP still runs in a private environment, thus the identity data stays under full control of the organization.
- The last model presented here is the **identity from the cloud model**. In this model both, the IdP and the SP, are moved to the public cloud but the entities are deployed at different cloud providers. Although this model may utilize the full scalability features provided by the cloud, it may lead to problems regarding data protection. In the public cloud it is very hard to determine where your plain identity data is actually stored or what is really done with it. A precondition for using this model is to fully trust the IdP respectively the cloud provider.

As described above many different types of IdM models exist. The model used within the current eID solution is based on the user-centric model where every citizen is in possession of a smart card containing the identity data together with signature key material. In the Austrian use-case the IdP is deployed within a private environment. In most cases every SP runs its own IdP instance. This master thesis is also based on this user-centric approach but in addition it foresees the deployment of the IdP at a public cloud provider. Because of data protection some additional security mechanisms, that will be described later, have to be implemented within the Austrian eID solution. The next chapters will describe the Austrian eID concept in more detail.

3.3 Electronic Identity - Austrian Citizen Card

The main building block of the Austrian identity management concept is the so called Austrian Citizen Card that represents the electronic identity (eID) in Austria. The Austrian Citizen Card is a functional concept that is not bound to dedicated hardware. It is possible to activate the functionality on smart cards, like the health insurance card or cash cards; another option

3.3 Electronic Identity - Austrian Citizen Card

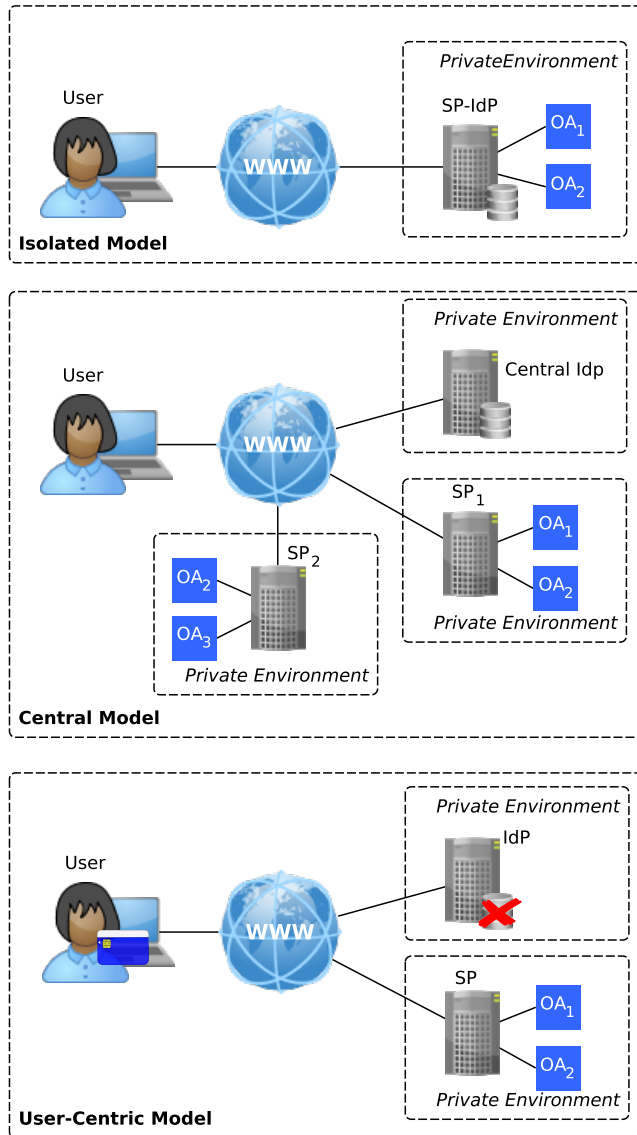


Figure 3.2: Identity Management Models

3 Identity Management in Austria

is the so called mobile phone signature¹, which basically is the Citizen Card implementation for mobile phones. The Citizen Card Environment (CCE) is responsible for the communication between the IdP and the Citizen Card and thus it provides the abstraction of the Citizen Card implementation used by the citizen. Beneath the technical background the following chapters will also provide some information regarding the legal framework the Austrian eID is based on.

3.3.1 Legal Framework

Digital signatures based on public key cryptography are important components within the Austrian eID concept and therefore the legal basis has been defined within the Signature Directive (SigD) by [The Council of the European Union, 2000]. The directive is implemented in Austria within the Austrian Signature Act (SigG) by [Republik Oesterreich, 2010]. The directive provides information regarding the use of electronic signatures and their legal background. The directive defines three types of electronic signatures, that are

- Simple electronic signature
 - electronic data that is
 - used for authentication
- Advanced electronic signature
 - simple electronic signature
 - linked to the signatory
 - capable of identifying the signatory
 - created with equipment under the citizen's sole control
 - linked to the signed data
 - change of linked data is recognizable
- Qualified electronic signature²
 - advanced electronic signature
 - based on a qualified certificate (QC)

¹<https://www.handy-signatur.at/>

²Term not defined in SigD

3.3 Electronic Identity - Austrian Citizen Card

- created using a secure signature creation device (SSCD)

While simple and advanced electronic signatures also have to be accepted as exhibit at the court only the **qualified electronic signature** is the legal equivalent to a handwritten signature except a few limitations (Article 5, SigG). A qualified electronic signature is based on a qualified certificate issued by a certification service provider. A qualified certificate has to include the following attributes:

- the note that it is a qualified certificate
- a unique name identifying the entity that issued the certificate
- signatory's name
- signature verification data (i.e. public key)
- certificate's validity period
- unique serial number
- the signature

The entity responsible for providing, respectively certifying, qualified certificates is represented by the certification service provider (CSP). A CSP within Austria, issuing qualified certificates, has to act according to the Austrian Signature Law [Republik Oesterreich, 2010] (SigG). Before a CSP issues a certificate it has to verify the citizen's identity based on an official photo identification or another, in terms of reliability equal, proof of identity (SigG section 8). After the identity has been proven, the certificate is signed by the CSP and written onto the smart card. It contains the public key that is bound to the private key stored within a secure location on the smart card. Based on these legal preliminaries the Citizen Card has been invented as a main building block of the Austrian eID concept and will be presented within the next sections.

3.3.2 Citizen Card Features

Identification and authentication are the fundamentals when participating in online services, so enough information has to be stored on the card to uniquely identify a person. In Austria a unique identifier, the so called CRR number, is assigned to every citizen and is stored within a central register of residences (CRR). Due to legal requirements concerning privacy it is

3 Identity Management in Austria

not allowed to use this CPR number directly, so the number is encrypted³ under the private key of the sourcePIN Register Authority (SRA) and is called sourcePIN (sPIN). This sPIN, together with the person's first name and last name is wrapped into an XML data structure called the identity link (IDL) [Hollosi and Karlinger, 2005]. Next a signing key pair, consisting of a public and private key is created and the public key is also added to the IDL. The private key, used for creating qualified electronic signatures, is stored in a secure location on the card and it can not be extracted from the card. The whole IDL is digitally signed by the SRA and also stored on the citizen card. Basically the IDL, in combination with a digital signature, is used for qualified identification and authentication. The identification is carried out by sending the IDL to an IdP. The IdP verifies the IDL's content and the SRA's signature over the IDL and returns data block⁴ to the citizen that she has to sign. This step is referred as authentication process and is carried out by applying a qualified electronic signature using her secret key stored on the smart card. The creation of the signature is secured by entering a PIN, so the signature is bound to something the citizen has (the Citizen Card) and something the citizen knows (the PIN) also known as two-factor authentication. This qualified electronic signatures are equivalent to hand-written signatures by law [The Council of the European Union, 2000]. In addition to identification, authentication and electronic signatures the Citizen Card also provides data storage functionality. E.g. the IDL and the qualified certificate are stored within the card storage. This storage locations are organized in so called info boxes.

3.3.3 Identity Link and sPIN

In Austria every person is uniquely identified by the IDL, respectively the sPIN. A sample IDL can be found in the appendix 9.2. An IDL is based on the Security Assertion Markup Language (SAML). All the identity relevant data is placed within a *saml:Assertion* element. The *saml:Subject* element contains all the citizen's personal data, which are first name, last name, date of birth and sPIN within the *pr:Identification* tag. The public keys are

³Triple-DES

⁴E.g. citizen's name, OA's technical parameters, current time... within an AUTH-block

3.4 Austrian Identity Provider

contained within *saml:Attribute* tags. The appended IDL contains two keys, the first one is an ECDSA key and the second one is a RSA key. The ECDSA key is the public key responsible for verifying qualified electronic signatures that are equivalent to handwritten signatures. The RSA public key may be used for data encryption and its private key for general signature creation by the citizen. The *dsig:Signature* block contains, beneath the signature value itself, information regarding the signed data, e.g. transformations are defined and references to the signed data are set. The signature value is calculated by the SRA and thus the SRA's public key certificate is also included within the IDL for its verification.

As already mentioned the sPIN is the anchor of a unique identification within Austria. Without the use of the sPIN data twins, sharing the same name and date of birth, would arise which could lead to problems within public and private online procedures. The problem using the sPIN directly for identification within online procedures is that a person would be completely traceable across multiple services when forwarding this sPIN to the service providers (SP). To avoid this problem, a sector identifier is assigned to every service, e.g. the sector 'Steuern' gets the acronym 'ST' and 'Bauen und Wohnen' gets the acronym 'BW'. If a citizen now wants to access a service, a sector-specific sPIN (ssPIN) is calculated based on the sector and the original sPIN. This calculation utilizes the SHA-1 hash function and thus it is not invertible, see figure 3.3. If, for example, $ssPIN_{BW}$ and $ssPIN_{ST}$ are calculated from sPIN it is not possible to calculate sPIN from $ssPIN_{BW}$ or $ssPIN_{ST}$. In addition there is no way to map $ssPIN_{BW}$ to $ssPIN_{ST}$ and vice versa. Based on this calculation it is possible to uniquely identify each citizen within a specific service but it is not possible to trace a person over multiple services. The usage of ssPINs within the identification and authentication procedures in Austria will be clarified in the following section.

3.4 Austrian Identity Provider

This section will provide informations regarding the identification and authentication process in Austria. First the different roles involved and then the general process flow will be explained in detail.

3 Identity Management in Austria

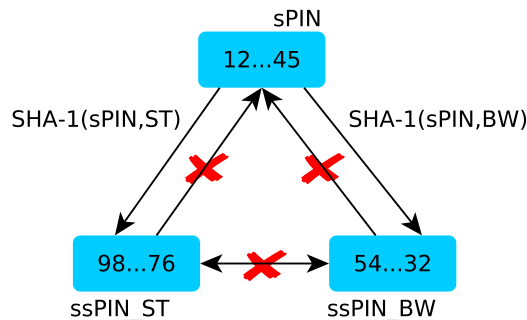


Figure 3.3: ssPIN Calculation based on sPIN and Sector Identifier

- The **Service Provider (SP)** is an entity providing online applications (OA), like banking applications or e-governmental applications. The applications are accessed using a standard web browser. In contrast to the standard user-name/password combination qualified identity data based on the Austrian eID should be used within this process.
- The **Identity Provider's (IdP)** task is to request the IDL from the user's Citizen Card, verify the citizen's and the SRA's signature and create the identity data sent to the SP. In Austria the Module for Online Applications - Identification (MOA-ID)⁵ is used as IdP.
- The **Citizen** is the entity that wants to login at an OA using her identity credentials stored on her Citizen Card. The OA is accessed via a standard web browser.
- The **Citizen Card Environment (CCE)** is the gateway between the application initiating a signature process and the Citizen Card. On the one hand it provides an interface to the applications, e.g. via http(s), that implements the Security Layer specification [Hollosi and Karlinger, 2014] and on the other hand it provides the communication to a smart card via the Security Token Abstraction Layer (STAL) protocol. Different CCE implementations are available. In this thesis the open source CCE Modular Open Citizen Card Architecture (MOCCA)⁶ is used and extended by the required functionality. It is a Java imple-

⁵<https://joinup.ec.europa.eu/software/moa-idspss/description>

⁶<https://joinup.ec.europa.eu/software/mocca/description>

3.4 Austrian Identity Provider

mentation where the security layer commands are sent to MOCCA via HTTP binding using port 3495 (http) and port 3496 (https). MOCCA is available as an online version running within the context of a Java applet and as a local version running within a local deployed jetty⁷ web server.

Figure 3.4 shows a sequence diagram representing the identification and authentication procedure. The following sequence diagram is based on the informations provided by [Stranacher et al., 2013].

- **Step 1** The citizen starts her browser and opens a web site where she wants to access an OA, providing protected resources, that is run by a SP.
- **Step 2** The OA performs an access check. If the citizen is already authenticated the process is done and the citizen may access the requested resource.
- **Step 3,4** If the citizen is not authenticated, she is redirected to MOA-ID, representing the IdP, via her browser.
- **Step 5,6** An HTML form, including an XML structure to request the IDL, is generated by MOA-ID. This InfoBoxReadRequest is again redirected to the CCE via the citizen's browser.
- **Step 7,8** The CCE carries out the communication to the smart card reader that the Citizen Card is attached to. The InfoBoxReadRequest is forwarded to the Citizen Card, the IDL is read from the card and sent back to MOA-ID.
- **Step 9** In this step MOA-ID verifies the IDL. The IDL was signed by the SRA when it was created and thus it is possible to verify the signature using the SRA's public key. If the signature is valid the citizen is identified.
- **Step 10** Based on the IDL a signature request, containing the AUTH-Block to be signed, is generated. It contains textual information about the citizen and about the OA the citizen wants to access. The signature request is forwarded to the CCE.
- **Step 11,12** The CCE forwards the data to be signed to the citizen card where it is signed using the citizens private key. The signed data structure is then returned to MOA-ID.

⁷<http://sourceforge.net/projects/jetty/>

3 Identity Management in Austria

- **Step 13** By verifying the signature using the citizen's public key the citizen is now authenticated against the IdP. MOA-ID creates a so-called SAML assertion representing the signed data and additional information like technical parameters and signing-time and stores it into its database. In addition a SAML artifact, a pointer to the SAML assertion, is created. MOA-ID is also responsible for calculating the ssPIN from the sPIN within the IDL. The Security Assertion Markup Language (SAML), [OASIS Security Services, 2002], represents an XML framework used for exchanging authentication data between entities.
- **Step 14,15,16** Including the SAML artifact, created in the former step, the citizen is now redirected from MOA-ID to the OA via the CCE and the web browser.
- **Step 17,18,19** The OA is able to fetch the SAML assertion by sending the SAML artifact to MOA-ID. Depending on the informations within the SAML assertion the citizen is allowed to access the requested resources or not. In the last step the citizen is forwarded to the resource or to an error page if access is not granted.

3.4 Austrian Identity Provider

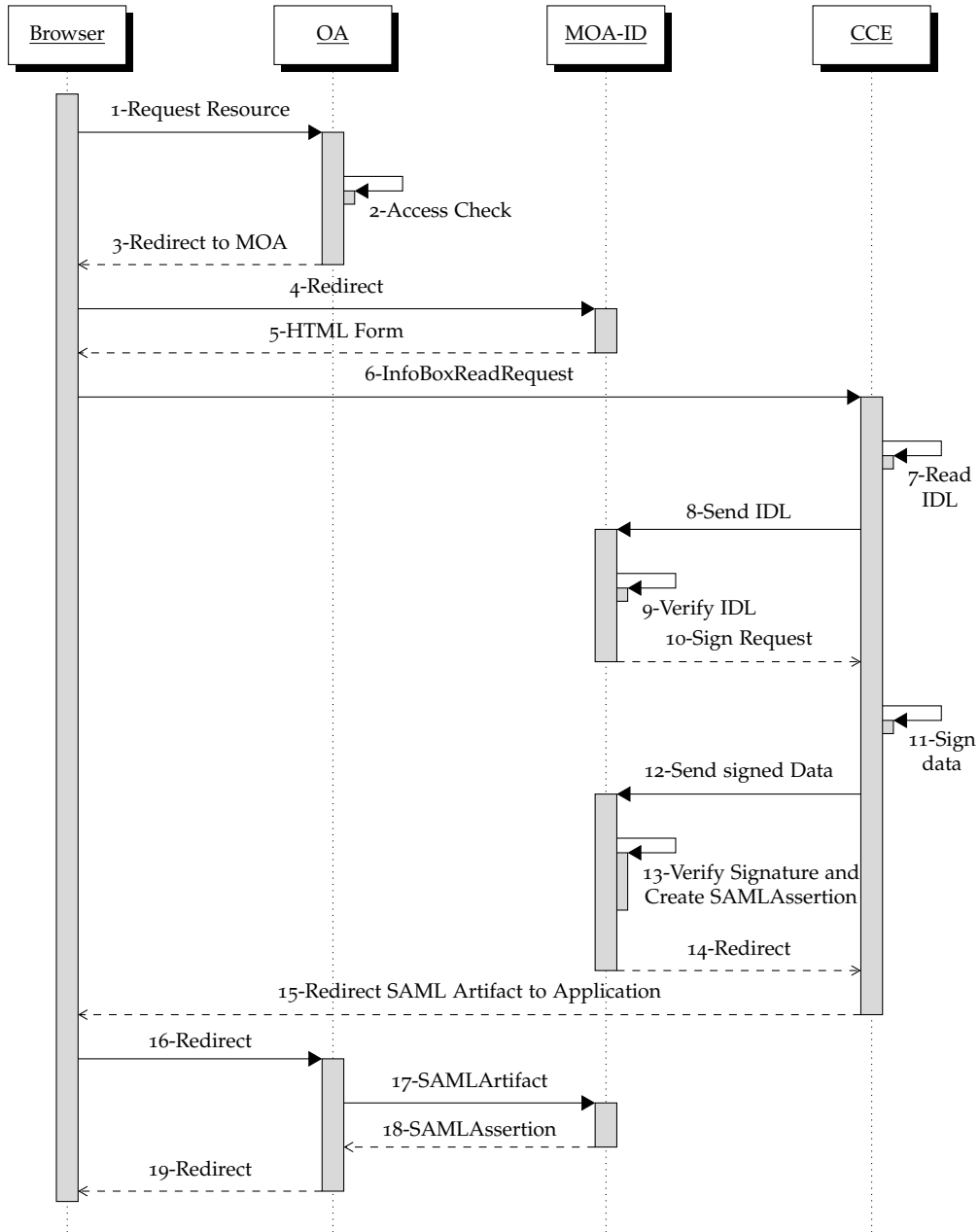


Figure 3.4: Sequence Diagram Identification Procedure.

3.5 Conclusion

From the identification and authentication point of view the current implementation of the Austrian process is very well designed and provides a huge amount of security. The whole procedure was designed under the assumption, that the IdP (MOA-ID) is deployed within a private and trusted environment. In most cases nowadays MOA-ID is deployed within the infrastructure used by the service provider offering the online applications. The privacy of the citizens data, in this case, is basically based on a trust relationship between the citizen and the SP/IdP. This means that the citizen trusts the IdP that it does not misuse the provided identity data. The approach presented within this master thesis foresees a deployment of the IdP within the public cloud. In the early days of cloud computing the main concerns moving services to the public cloud were about availability and reliability. In the last few years also concerns regarding privacy and data protection became a big issue. Thinking of the IdP use case where the IDL has to be sent to the IdP via a secured https connection, the data is secure on the transport channel between the citizen and the IdP. In the usual setup the IdP needs to get the IDL attributes as plain text. As long as the IdP can be fully trusted, like it is the case in a local IdP deployment by the SP, this should be fine, but in the case of a public cloud deployment this relationship has to be modified. Basically a cloud provider could do everything with the data it receives. The approach presented within this thesis utilizes a cryptographic scheme where the IdP, respectively the cloud provider, never actually receives plain identity data. Although the IdP is able to ensure that the encrypted identity data represents a citizen in possess of qualified signature creation data. The SP is then able to decrypt the identity data and grant or deny access to the requested resource.

Another important point covered within this thesis is the so-called selective disclosure. Imagine a simple online application that a person is only allowed to use when she is over 16 years old. When using the Citizen Card to login to such an application, all the data contained within the IDL (first and last name, data of birth, ssPIN) is sent to the SP to be processed. Even if you would fully trust this SP it simply is not necessary to forward all the identity data. So the second target of this thesis is to provide a technique to choose which subset of the identity data should be forwarded to the SP. Although

3.5 Conclusion

the SP is able to define which subset of identity data it needs, the decision what data actually will be forwarded to the SP will be under the citizen's sole control.

To achieve the requirements within this master thesis two pretty innovative cryptographic concepts have to be implemented. The underlying model and the concepts will be described in detail within the next chapter.

4 User-Centered and Privacy-Preserving Model

4.1 Introduction

The current Austrian eID solution, presented within chapter 3, basically relies on a trust-relationship between the citizen and the IdP respectively the SP. Based on the fact that every SP deploys its own IdP it is not problematic in terms of security to forward a citizen's identity data in plain (via a secure channel). The model, see figure 4.1, implemented within this thesis foresees a centralized deployment of the IdP within the public cloud and has been proposed by [Slamanig, Stranacher, and Zwattendorfer, 2014]. According to their model different SPs, running in different environments, share an IdP deployed within the public cloud. According to the IdM models presented in section 3.2 the model implemented within this thesis basically represents a variant of the central model, where the identity data stays under the sole control of the citizen on a secure token. But in contrast to the central model, where the IdP is deployed within a private environment, the IdP in the implemented model is deployed within the public cloud. The following section will provide an overview regarding the implemented model while the later sections will discuss the cryptographic primitives used within this thesis.

4.2 Model

The principle of the model this thesis is based on is shown in figure 4.1. The arrows represent the identity-data flow within the model. The green

4 User-Centered and Privacy-Preserving Model

arrow, from the user to the SP via the IdP, represents encrypted identity data and the red trace represents where the plain identity data is available. The implemented model provides encrypted identity data from the citizen's token, over the IdP up to the SP where only the requested attributes get decrypted. In addition this model enables the citizen to select which identity attributes should be forwarded to the SP and which should be blanked, known as identity data disclosure. To implement this model two innovative cryptographic concepts had to be used within this thesis. For selective disclosure so-called blank digital signatures and for the encrypted identity data a proxy re-encryption scheme have been used. Both concepts will be presented in more detail within the following sections.

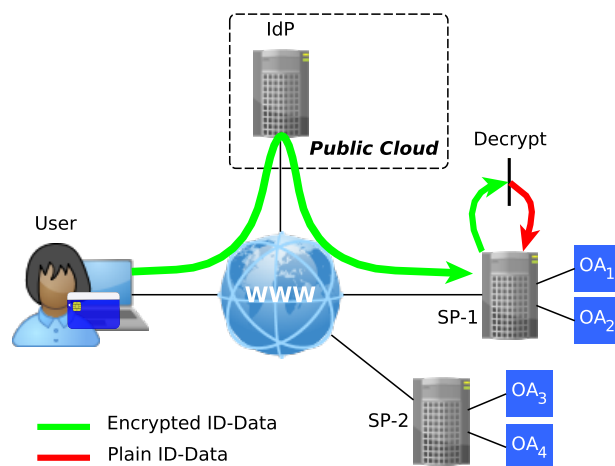


Figure 4.1: User-Centered and Privacy-Preserving Model

4.3 Blank Digital Signatures

4.3.1 Introduction

Digital signatures, see section 2.1, provide a common way to validate a document's authenticity and integrity after it has been signed. Every entity,

4.3 Blank Digital Signatures

in possess of the signatory's public verification key, is able to verify the signature. Although in some cases it could be useful if a **dedicated** entity would be able to modify **pre-defined** parts of a message after the signature has been created without invalidating the signature. This concept is known as identity data disclosure and is based on a type of cryptographic schemes that are called modifiable signature schemes. Details regarding two of this schemes, namely redactable and sanitizable signatures, have already been provided in section 2.3. In this thesis a novel scheme, called blank digital signatures (BDS), invented by [Hanser and Slamanig, 2013], will be used.

4.3.2 Principle

As a motivation figure 4.2 represents an example of a use case for such a cryptographic scheme. A businesswoman has to order a certain amount of a product every week. This order has to be signed digitally by the woman herself. Because the businesswoman is very busy she wants to move the ordering process to her secretary. The intent behind using BDS is the following: the businesswoman could create a so-called BDS-template as shown in listing 4.1 (step 1). This XML scheme is based on the implementation by [Derler, 2013] done within his master thesis. The BDS scheme basically defines three different types of template entries, which are

- **fix** - the value must not be changed by a proxy.
- **exch** - the value must be set to exactly one value chosen from the list of available values.
- **blank** - the value of this element may be set to any value by the proxy.

Using BDS a dedicated entity responsible for instantiating the template has to be defined by the businesswoman (step 2). Now the created template is signed by the businesswoman (step 3) and forwarded to her secretary (step 4). Whenever the secretary now has to place an order, she instantiates the template (step 5), thus creating a message as shown in listing 4.2. This message, representing the order, is then signed by the secretary (step 6) and forwarded to the company selling the product (step 7). The seller is then able to verify the signature using the businesswoman's and the secretary's public keys.

4 User-Centered and Privacy-Preserving Model

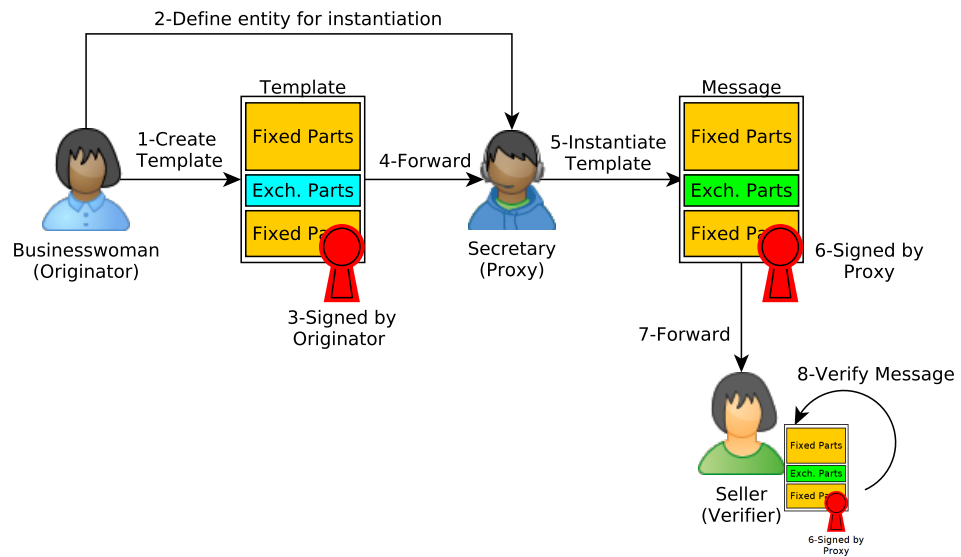


Figure 4.2: Principle of the Blank Digital Signature Scheme

```

1 <?xml version="1.0" encoding="UTF-8"
2   standalone="yes"?>
3 <template id="1234">
4   <templateentry>
5     <message type="fix" length="0">
6       <text>Hi, I would like to order</text>
7     </message>
8   </templateentry>
9   <templateentry>
10    <message type="exch" length="0">
11      <text>1</text>
12    </message>
13    <message type="exch" length="0">
14      <text>5</text>
15    </message>
16    <message type="exch" length="0">
17      <text>10</text>
18    </message>
19  </templateentry>
20  <templateentry>
21    <message type="fix" length="0">
22      <text>pieces of your product</text>
23    </message>
24  </templateentry>
25 </template>

```

Listing 4.1: BDS Template example

4.3 Blank Digital Signatures

```
1 <?xml version="1.0" encoding="UTF-8"  
2   standalone="yes"?>  
3 <message>  
4   <messageentry type="fix" length="0">  
5     <text>Hi, I would like to order</text>  
6   </messageentry>  
7   <messageentry type="exch" length="0">  
8     <text>5</text>  
9   </messageentry>  
10  <messageentry type="fix" length="0">  
11    <text>pieces of your product</text>  
12  </messageentry>  
13 </message>
```

Listing 4.2: BDS Message example

This simple example should have illustrated the basic principle of blank digital signatures. The following section will provide detailed information regarding the used signature scheme.

4.3.3 The Scheme

Before going into detail, concerning the algorithms used within this scheme, a short description of the involved entities (roles) will be given.

- For creating the key pairs a **trusted third party (TTP)** is needed. It also creates the public parameters needed within the scheme.
- The **originator** is the entity specifying a BDS template. Such a template may include fixed, exchangeable and blank values. This template is signed using the originator's secret key.
- The **proxy** receives the assigned permission from the originator to create template instances based on the signed template thus creating BDS messages. This message is signed using the proxy's secret key.
- The **verifier** receives the message and is able to verify the signature using the originator's and proxy's public keys. The BDS verification basically checks if the received message is a valid instantiation of the template without seeing the exchangeable values that have not been selected.

After the roles are defined the algorithms used within the BDS scheme have to be defined according to [Hanser and Slamanig, 2013]:

4 User-Centered and Privacy-Preserving Model

- $pp = \text{KeyGen}(\kappa, t)$ represents the algorithm used for generating the public parameters pp . κ is the security parameter and t specifies the template's maximum size. The public parameter generation is carried out by the TTP.
- $(\sigma_\tau, sk_p^\tau) = \text{Sign}(\tau, pp, sk_O, pk_P)$ is executed by the originator. τ represents the BDS template, sk_O the originator's signing key and pk_P the proxy's verification key. σ_τ represents the BDS template's signature and sk_p^τ is the template dependent private key that is forwarded to the proxy.
- $V_\tau = \text{Verify}_\tau(\tau, \sigma_\tau, pp, pk_O, sk_p^\tau, pk_P)$ is executed by the proxy to verify the template's signature. pk_O is the public verification key belonging to the originator. V_τ is a bit set to true if the template signature is valid, else false.
- $\sigma_M = \text{Inst}(\tau, M, \sigma_T, pp, sk_p^\tau, sk_P)$ is executed by the proxy. M represents the BDS message which is an instantiation of the Template τ and sk_P is the proxy's signing key. σ_m represents the message' signature value.
- $V_M = \text{Verify}_M(M, \sigma_M, pp, pk_P, pk_O)$ is the verification algorithm to verify the BDS message' signature. This algorithm can be executed by everybody who has access to the message M , the public parameters pp and the public keys pk_O and pk_P . V_M is a bit set to true if the message signature is valid (an instance of the Template T), else false.

The keys sk_P, pk_P, sk_O and pk_O are standard digital signature keys (e.g. ECDSA), thus standard digital signatures are used to provide the authenticity and integrity of the template and message signatures. BDS are based on polynomial commitments and elliptic curve pairings, but details regarding this concepts will not be given here, because that would go beyond the scope of this thesis.

4.3.4 Properties

Compared to redactable and sanitizeable signature schemes BDS support the hiding of unchosen values. Using BDS a verifier is not able to gain any information regarding the template's exchangeable values. This is an important feature that plays a fundamental role within this thesis. The BDS

scheme may be used as a redactable scheme by defining each redactable entry as an exchangeable value that may be set to the actual value or a special sign, like a '*', that marks the value as redacted.

4.4 Proxy Re-encryption

4.4.1 Introduction

Proxy Re-encryption (PRE) is based on a very basic concept. A document, encrypted for a user A, should be re-encrypted by a third entity that another user B is able to decrypt it using her secret key while the third entity has no access to the unencrypted data. A naive approach could be based on a trusted third entity that is in possess of secret keys belonging to registered users. A message encrypted under user A's public key could then be forwarded to the trusted entity where it is decrypted using A's secret key and encrypted again using B's public. Then the new cipher text is forwarded to B who is able do decrypt it using her private key. Even if this would work it fails in terms of data security. First the private keys have to be stored on a server not under the sole control by the user and second the data to be encrypted is available at the server in plain. PRE in contrast provides much more security in terms of data privacy. First the keys have not to be stored anywhere else and second the proxy never processes plain data while it performs the re-encryption. A typical use case for PRE could be the following scenario: Alice wants to share data, encrypted for her, with multiple other users, let us call them Bob and Eve. Alice could now simply decrypt the data encrypted for her using her private key, encrypt it twice with Bob's and Eve's key and place it on a server, where they both could download it. Using PRE in this case could lead to the following solution. Alice encrypts the data once. Additionally she calculates re-encryption keys for Bob and Eve using their public keys and uploads the encrypted data and the PRE-keys to the server. If Bob now wants to download the data, the server re-encrypts it using the re-encryption key calculated by Alice for Bob. The same applies for Eve. So instead of saving the encrypted data multiple times on the server it is only stored once plus the re-encryption keys for

4 User-Centered and Privacy-Preserving Model

every user who should be able to download the file. After the file has been downloaded the server may delete the re-encryption key.

4.4.2 Principle

Figure 4.3 represents a typical setup of a PRE scheme. It includes two users Alice and Bob who are in posses of a key pair for encryption and decryption. The proxy is responsible for performing the re-encryption from user Alice to Bob. In step one a document is encrypted under Alice' public key pk_A , called c_A . Alice would now like to enable forwarding the encrypted document to Bob, so that he is able to decrypt it using his private key sk_B , without the need for Alice to be on-line to decrypt the cipher text c_A using her secret key sk_A , encrypt it for Bob using his public key pk_B and then forwarding the cipher text c_B to Bob. Therefore Alice has to create a so-called re-encryption key $rk_{A \rightarrow B}$ based on her private key and Bob's public key. In step 2 the encrypted document and the re-encryption key are forwarded to the proxy. The proxy's task in step 3 is to re-encrypt the cipher text c_A to the cipher text c_B without revealing the plain text using the re-encryption key $rk_{A \rightarrow B}$. In step 4 the cipher text c_B is forwarded to Bob. In step 5 Bob is able to decrypt c_B using his secret key sk_B thus revealing the original document. Now that the concept behind PRE has been illustrated the scheme's cryptographic algorithms will be discussed in more detail within the next section.

4.4.3 The Scheme

In the following the PRE scheme presented by [Ateniese et al., 2006] will be discussed in more detail. The cryptographic primitives are defined by the scheme.

- The **Key Generation** operation outputs Alice' private key $sk_A = a$ and public key $pk_A = g^a$ where g represents a system parameter based on bilinear maps.
- Alice calculates the **Re-Encryption Key** using her private key $sk_A = a$ and Bobs public key $pk_B = g^b$ by calculating $rk_{A \rightarrow B} = g^{\frac{b}{a}}$.

4.4 Proxy Re-encryption

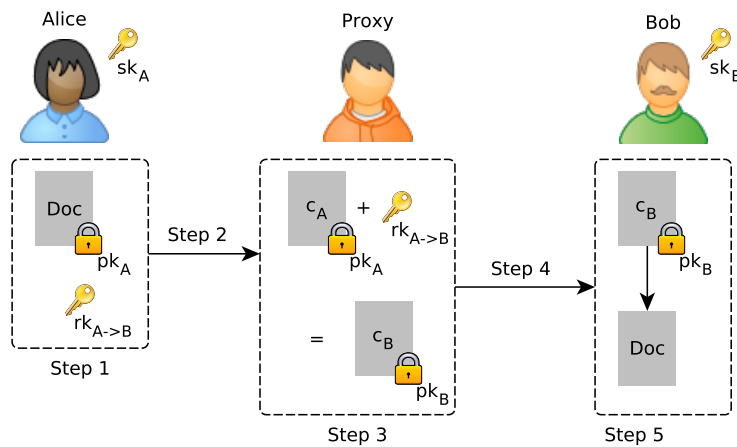


Figure 4.3: Principle of a Proxy Re-encryption Scheme

- A **First Level Encryption** means encrypting a plain text m using pk_A in such a way, that only Alice (the holder of sk_A) is able to decrypt it. The cipher text is represented by a tuple $c = (Z^{ak}, mZ^k)$ where Z is again a system parameter and k a random element.
- A **Second Level Encryption** means encrypting a plain text m using pk_A in such a way, that Alice and Bob are able to decrypt the cipher text $c = (g^{ak}, mZ^k)$.
- The **Re-encryption** can be executed on every cipher text c that was created using a second level encryption. So from $c_A = (g^{ak}, mZ^k)$ using the re-encryption key $rk_{A \rightarrow B} = g^{\frac{b}{a}}$ the first level cipher text $c_B = (Z^{bk}, mZ^k)$ is calculated and forwarded to Bob.
- For a **First Level Cipher Text Decryption** of a cipher text $c = (\alpha, \beta)$ the original message is decrypted by $m = \frac{\beta}{\alpha^{1/a}}$ under Alice' private key $sk_A = a$.
- For a **Second Level Cipher Text Decryption** of a cipher text $c = (\alpha, \beta)$ the original message is decrypted by $m = \frac{\beta}{e(\alpha, g)^{1/b}}$ under Bob's private key $sk_B = b$ where e is a bilinear map $e : G1 \times G1 \rightarrow G2$.

This scheme has been implemented in the NICS crypto library by [Nuñez, 2013]. It provides a basic, functional implementation of the presented scheme and it utilizes the Java pairing-based cryptography library [De Caro and

4 User-Centered and Privacy-Preserving Model

Iovino, 2011]. It had to be slightly extended to be usable within the current eID implementation, details will be provided within chapter 5.

After the scheme operations have been defined, the next section will provide some of the scheme's properties.

4.4.4 Properties

This section will shortly describe the properties provided by the used scheme.

- it is **unidirectional**. It is possible to re-encrypt cipher texts from Alice to Bob using the re-encryption key $rk_{A \rightarrow B}$ but not vice versa.
- it is **non-interactive**. Re-encryption keys are generated only by Alice using Bob's public key without the interaction of a trusted third party. Also the re-encryption key generation does not involve Bob's private key.
- it is **key optimal**. The required storage for the proxy and Bob remains constant, independent of the amount of delegations. This is not the case for some other schemes.
- it is **collusion-safe**. Even if the proxy and Bob would cooperate under evil intent, they would not be able to determine Alice's secret key, nor they would be able to reveal the plain text according to a first-level cipher text created by Alice.
- it is **non-transitive**. The proxy alone is not able to generate new re-encryption keys from already received ones. This feature is very important, because if the proxy colludes with an evil party EVE the proxy could re-encrypt the data for EVE, she would decrypt it and send it back to the proxy.

4.5 Conclusion

The first part of this chapter introduced the concept of the IdM model implemented within this thesis. It foresees, in contrast to the current Austrian eID implementation where every SP deploys its own IdP, the deployment

4.5 Conclusion

of an IdP within the public cloud. Because the processed identity data has to be protected in terms of privacy the target of this work is to process only encrypted identity data on the whole path from the citizen via the IdP up to the SP. Another target of the implemented approach is to increase a citizen's privacy by providing selective identity data disclosure. The rest of this chapter provides information regarding the two underlying cryptographic schemes used within this model. The first one is the blank digital signature scheme which is used as a type of redactable signature scheme for selective attribute disclosure. The second one is a proxy re-encryption scheme used for re-encrypting cipher texts for user A to cipher texts for user B carried out by a third entity without accessing the plain texts. Chapter 5 will provide details on how the current eID solution is extended by this two schemes to provide the required functionality.

5 Implementation

5.1 Introduction

For implementing the user-centered and privacy-preserving eID concept, presented within chapter 4, the components used within the current Austrian eID implementation had to be modified. At first an application representing the entities responsible for generating and certifying key material and identity data, namely the SRA and CSP, had to be created. This application is also responsible for carrying out the registration process and creating the IDL. For the implementation of the identification and authentication procedure the CCE (MOCCA) and the IdP (MOA-ID) have also been modified to be capable of handling PRE and BDS. MOCCA uses BDS for creating BDS messages, representing the IDL, by instantiating the BDS template based on the sector of the OA and the required identity attributes. MOA-ID basically only requires the BDS message verification algorithm to verify that the BDS message represents a valid instantiation of the BDS template signed issued by the SRA and stored only on the Citizen Card. The PRE implementation is required by MOCCA, respectively the Citizen Card, for calculating the re-encryption key used by MOA-ID for re-encrypting the identity attributes that are forwarded to the OA. The Citizen Card within this thesis is implemented as a virtual smart card within the context of MOCCA.

While section 5.2 to section 5.6 provide implementation details according to the different eID components section 5.7 gives information regarding the IDL structure, respectively it's signature that is based on the XAdES standard. The same section also contains information regarding the legal basis for using pseudonyms within a citizen's certificate. While section 5.8 includes some basic information about the Security Layer, section 5.9 provides details

5 Implementation

regarding the extensions that had to be carried out within the used PRE library.

5.2 Demo Application SRA/CSP

In the current implementation of the Austrian eID concept two main authorities are needed for generating the IDL, attesting the DSS key pairs and the certificates. On the one hand there is the SRA that is responsible for the sPIN calculation and for signing the IDL. On the other hand is the CSP that is responsible for the certification of the key material, thus creating public key certificates. In the first part of the implementation the procedure of generating the IDL and additional key material is implemented within a demo application. Basically the application is a web application that offers a simple web form where a citizen's data can be entered and afterwards the required key material, certificates and the IDL is generated. The keys and certificates are stored within Java key store objects that are distributed manually to the involved entities. For a real use case a public key infrastructure (PKI) would be used for the distribution but for a proof of concept implementation this approach is sufficient. The following section provides details regarding the registration process.

5.2.1 Registration Process

The implementation is based on the registration process presented by [Slamanig, Stranacher, and Zwattendorfer, 2014] and has been discussed within section 4.2. The roles that are involved within this process are the CSP, the SRA, the Citizen (Citizen Card), MOA-ID (IdP) and the SP. In the following the necessary steps are described.

- **CSP's tasks**

- At first the DSS key pairs are generated and the public keys for the involved entities are certified by the CSP. The created keys are the signing key sk_{SRA}^{DSS} and the public key pk_{SRA}^{DSS} for the SRA,

5.2 Demo Application SRA/CSP

sk_{MOA}^{DSS} and pk_{MOA}^{DSS} for the IdP and $sk_{Citizen}^{DSS}$ and $pk_{Citizen}^{DSS}$ for the citizen.

- In the next step the public parameters pp^{PRE} for the PRE are generated together with the public $pk_{Citizen}^{PRE}$ and private $pk_{Citizen}^{PRE}$ PRE keys and are forwarded to the Citizen and the SP ($pk_{SP}^{PRE}, sk_{SP}^{PRE}$). The public parameters and the public PRE keys are certified by the CSP. The citizen's key pair will be stored on the Citizen Card in a later step.
- Now the CSP creates and certifies the public parameters pp^{BDS} for the BDS.

• SRA's tasks

- The SRA creates a modified IDL compared to the IDL used within the current eID solution. So beneath the first name, last name, date of birth and sPIN it also includes pre-calculated ssPINs for all available sectors as defined by [Republik Oesterreich, 2014]. So the IDL is represented by a list of n tuples of the form $[(A_1, a_1), (A_2, a_2), \dots, (A_n, a_n)]$, where A_x represents the key and a_x the value.
- Afterwards every single attribute value is encrypted using the citizen's public PRE key $pk_{Citizen}^{PRE}$ resulting in an encrypted IDL*. The encryption is a second level encryption according to the PRE scheme presented in section 4.4. This means, that the cipher text may be decrypted by the citizen herself or re-encrypted by the proxy using a re-encryption key calculated by the citizen under her secret key and the SP's public key.
- This IDL* is the base for the BDS template generation where every attribute is marked as an exchangeable value to be exchanged by a '*'. This means that also the '*' gets encrypted using $pk_{Citizen}^{PRE}$. A single template entry may look like shown in listing 5.1. This example shows that only the attribute identifier, in this case $pr : GivenName$, is included as plain text. Using this data structure the citizen will be able to redact an entry by selecting the encrypted '*' value. This template is then signed by the SRA using the BDS-signing algorithm. This calculation results in the template signature σ_τ and the template dependent private key $sk_{Citizen}^\tau$ which will be stored on the Citizen Card. So only the citizen that

5 Implementation

is in possession of the key is able to create valid BDS messages based on the template.

- In the last step the data defined in table 5.1 is stored on the Citizen Card.

Name	Description
σ_{τ}	Template signature
τ	BDS template representing the IDL
$sk_{Citizen}^{\tau}$	Template dependent private key for creating BDS messages
pp^{BDS}	Public parameters specified for BDS in certificate
$sk_{Citizen}^{DSS}$	Citizen's DSS signing key
$sk_{Citizen}^{PRE}$	Citizen's private PRE key
$pk_{Citizen}^{PRE}$	Citizen's public PRE key in certificate
pp^{PRE}	Public parameters specified for PRE in certificate

Table 5.1: Content stored on Citizen Card

```

1  ...
2  <templateentry>
3    <message type="fix" length="0">
4      <text><pr:GivenName</text>
5    </message>
6  </templateentry>
7  <templateentry>
8    <message type="exch" length="0">
9      <text>ckdienv7e.....80930408</text>
10   </message>
11   <message type="exch" length="0">
12     <text>jfj3odkje.....jwjdoejf</text>
13   </message>
14 </templateentry>
15 <templateentry>
16   <message type="fix" length="0">
17     <text></pr:GivenName</text>
18   </message>
19 </templateentry>
20 ...

```

Listing 5.1: BDS Template Entry

5.3 Identification Process

Before details regarding the modifications and implementations within the Austrian eID components will be given, the identification process implemented within this master thesis, as shown in figure 5.1, will be described in more detail.

1. The citizen request a resource at the SP.
2. The SP creates a request, including his public PRE key pk_{SP}^{PRE} and the sector the SP operates in, and redirects the citizen to MOA-ID.
3. In MOA-ID every OA is registered and the attributes required for the authorization by the SP are pre-configured. So MOA-ID creates a request that includes the required attributes, the sector and pk_{SP}^{PRE} and sends it to the citizen.
4. The citizen now reads the BDS template from the card.
5. In the redaction step the sPIN and all ssPIN that do not match the specified sector are redacted. This means that the exchangeable BDS values are set to the encrypted '*' value. Additionally the CCE provides the information to the user which attributes the SP wants to read and the user is then able to agree to forward the required attributes or she may even change the attribute set to be forwarded via a user interface.
6. After the template content is selected the BDS template is instantiated using the citizens private DSS key $sk_{Citizen}^{DSS}$ and the template dependent private key $sk_{Citizen}^{\tau}$. This calculation is done on the virtual card and outputs the message signature σ_M belonging to the BDS message M .
7. In the next step the re-encryption key $rk_{Citizen \rightarrow SP}$ is calculated. Therefore the SP's public PRE key pk_{SP}^{PRE} is sent to the virtual card and processed together with the citizen's private key $sk_{Citizen}^{PRE}$.
8. Now the BDS message M , the signature σ_M and the re-encryption key $rk_{Citizen \rightarrow SP}$ are returned to MOA-ID.
9. MOA-ID is now able to verify the message signature by using the BDS message verification algorithm. If the verification was successful, the message represents a valid BDS template instantiation.
10. MOA-ID now initializes a re-encryption of all the attributes within the message using the re-encryption key $rk_{Citizen \rightarrow SP}$. Actually MOA-ID only has to re-encrypt the ssPIN belonging to the SP's sector and the other attributes requested by SP.

5 Implementation

11. MOA-ID creates an AuthBlock containing the re-encrypted identity attributes, then it creates a SAMLAssertion A based on this block and sign the assertion using it's signing key sk_{MOA}^{DSS} .
12. MOA-ID also creates a SAMLArtifact referencing the SAMLAssertion and sends it to the SP.
13. The SP is now able to pick up the SAMLAssertion stored at the IdP by issuing a request containing the SAMLArtifact.
14. The SP is able to verify the signature. If it is valid, it decrypts the attributes using it's PRE key sk_{SP}^{PRE} .
15. Depending on the received attributes the SP is able to grant or deny access to the requested resource.

5.4 Modifications in MOCCA

As already shown in sequence diagram 5.1 the IdP checks, according to the requested OA, which attributes are required for the authentication of the citizen and forwards this attributes together with the sector and the SP's public re-encryption key to the CCE (MOCCA). MOCCA now presents a user interface to the user where the requested attributes may be checked and even changed by her. In the next step MOCCA reads the IDL (BDS template) from the card and creates a BDS message by blanking the the attributes that should not be forwarded to the SP. This message is then sent to the card for creating the BDS signature as well as creating the re-encryption key for the IdP based on the SP's public PRE key. From a user's point of view the only difference to the current process of identification and authentication consists of the notification and selection of the identity attributes to be forwarded to the SP.

5.4.1 Implementation Details

This section briefly describes the changes within the affected classes. Block diagram 5.2 shows the data flow from the receiving servlet down to the (virtual) Citizen Card. The SL command is sent via HTTP POST to the *BKURequestHandler* class. It contains the IDL read request, the required

5.4 Modifications in MOCCA

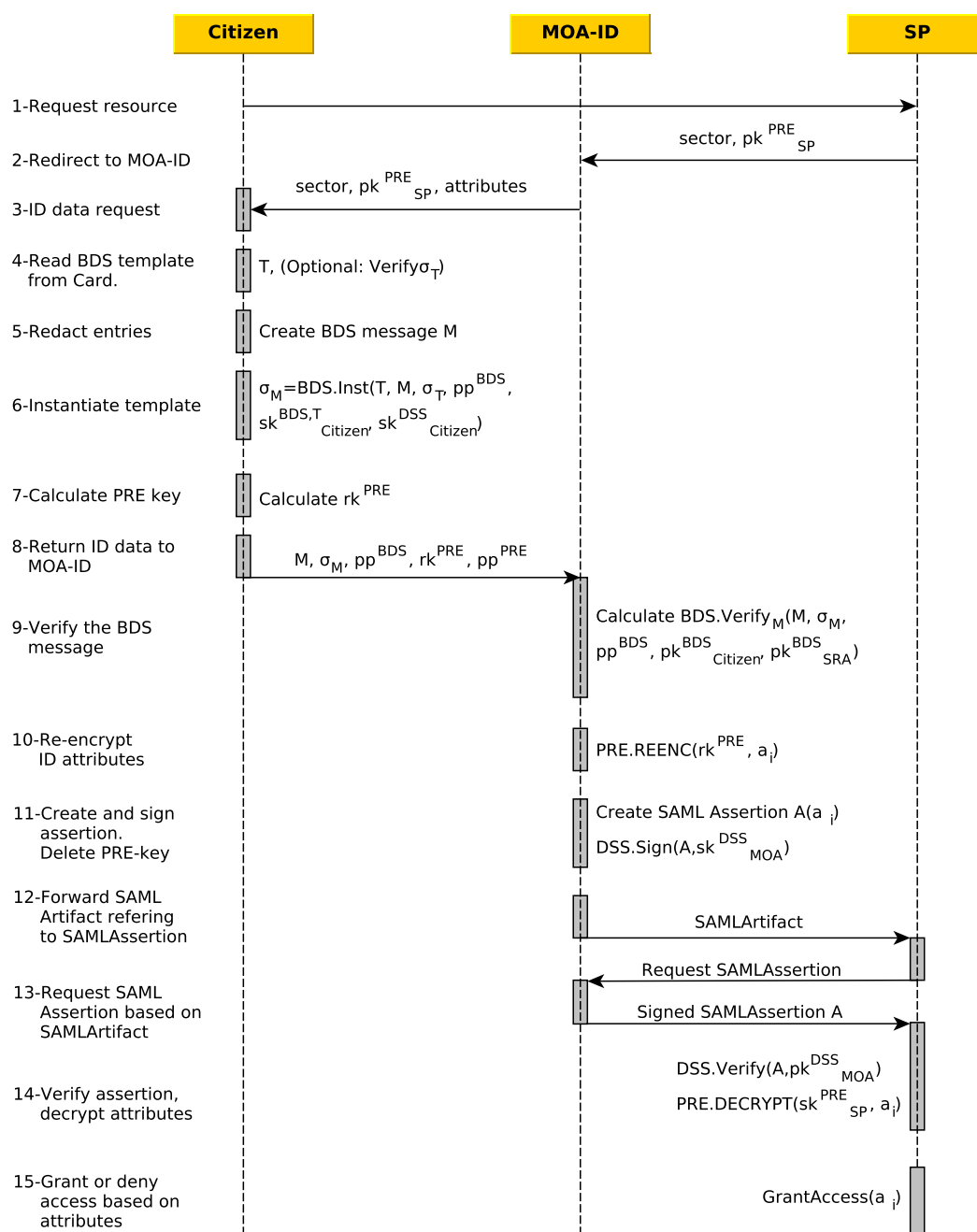


Figure 5.1: Sequence Diagram representing the Identification Process using BDS and PRE

5 Implementation

attributes, the OA sector and the SP's public PRE key. Here the *HTTPBindingProcessor* is instantiated and the received data is forwarded to it. The *HTTPBindingProcessor* has to handle the following tasks:

- extract form parameters, e.g. *dataUrl*, *target*, needed attributes, etc.
- check request against configuration, e.g. match *dataUrl* against a white list, check signature layout version, check XAdES version, etc.
- state machine controlled request processing:
 - process request - invoke according command
 - handle *dataURL* - check return codes and forward or redirect response according to received code
 - transform result - transform the output if required according to a specified style sheet
- error handling - create error message if an error occurred
- consume request - check request parameters for completeness, e.g. correct content type, transfer encoding, source URL, etc.

In the processing state the *SLCommandInvoker* is instantiated and the command is forwarded to it. Here an attached *SecurityManager* is responsible for checking if the issuing entity is allowed to request the command based on a security policy defined within MOCCA. This policy could for example define that only 'gv.at'-domains are allowed to read the IDL from the card. After the security policy has been successfully checked, the *read* method of the *IDLInfoboxImplementation* object is called. This class is the main class for creating the requests and reading the responses from the smart card. At first a *STALHelper* object is created that receives a list of requests that should be sent to the card, which are:

- read the IDL (BDS template)
- read the certificate
- read public BDS parameters

Now a small GUI showing the requested attributes is provided to the citizen where she is able to decide if she is fine with sending those attributes to the SP. Based on this selection the BDS template is instantiated and thus a BDS message is created. This message is then prepared for signing, wrapped into a signature request and added to the *STALHelper* who forwards it to the corresponding handler. In the next step the SP's public PRE key

is wrapped into a re-encryption key request and is also added to the *STALHelper* which now contains a list with the requests. According to the request-type the corresponding handler is called. For the BDS signature creation the *BDSignRequestHandler* and for the re-encryption key calculation the *PREKeyCalculationRequestHandler* is called. Those handlers are calling the corresponding smart card methods and the BDS handler additionally instantiates a GUI for requesting the secure signature PIN. After the card returns the result it is assembled to a response object, handed over to the *BKURestHandler* and sent back to the IdP where the signature is verified and the attributes are further processed.

5.4.2 Virtual Citizen Card

Because of the lack of a real smart card, providing enough performance for implementing the needed cryptographic primitives, a virtual smart card has been implemented for demonstration purposes. Basically the data available on the smart card is stored within a folder in the MOCCA base-directory on the citizen's computer. The virtual card supports the following functionalities:

- **Reading IDL** - the IDL, in this case the signed BDS template containing encrypted attributes, is returned to MOCCA.
- **Create Signature** - a BDS message containing only the selected attributes are sent to the card. The BDS signature is calculated using the private template dependent private key $sk_{Citizen}^T$ and the citizen's private DSS key $sk_{Citizen}^{DSS}$. The signature value is sent back to MOCCA and is attached to the BDS message that is forwarded to the IdP.
- **Calculate re-encryption key** - calculates the re-encryption key, which is forwarded to the IdP. The public PRE key pk_{SP}^{PRE} is sent to the card and the re-encryption key is calculated using the citizen's secret PRE key $sk_{Citizen}^{PRE}$.

Of course such a virtual card is not an accurate replacement for a hardware token like a smart card, but it is sufficient to ensure that the implemented IdM model works as expected. Based on this virtual Citizen Card a model based on secure hardware will be presented within section 6.3.1.2.

5 Implementation

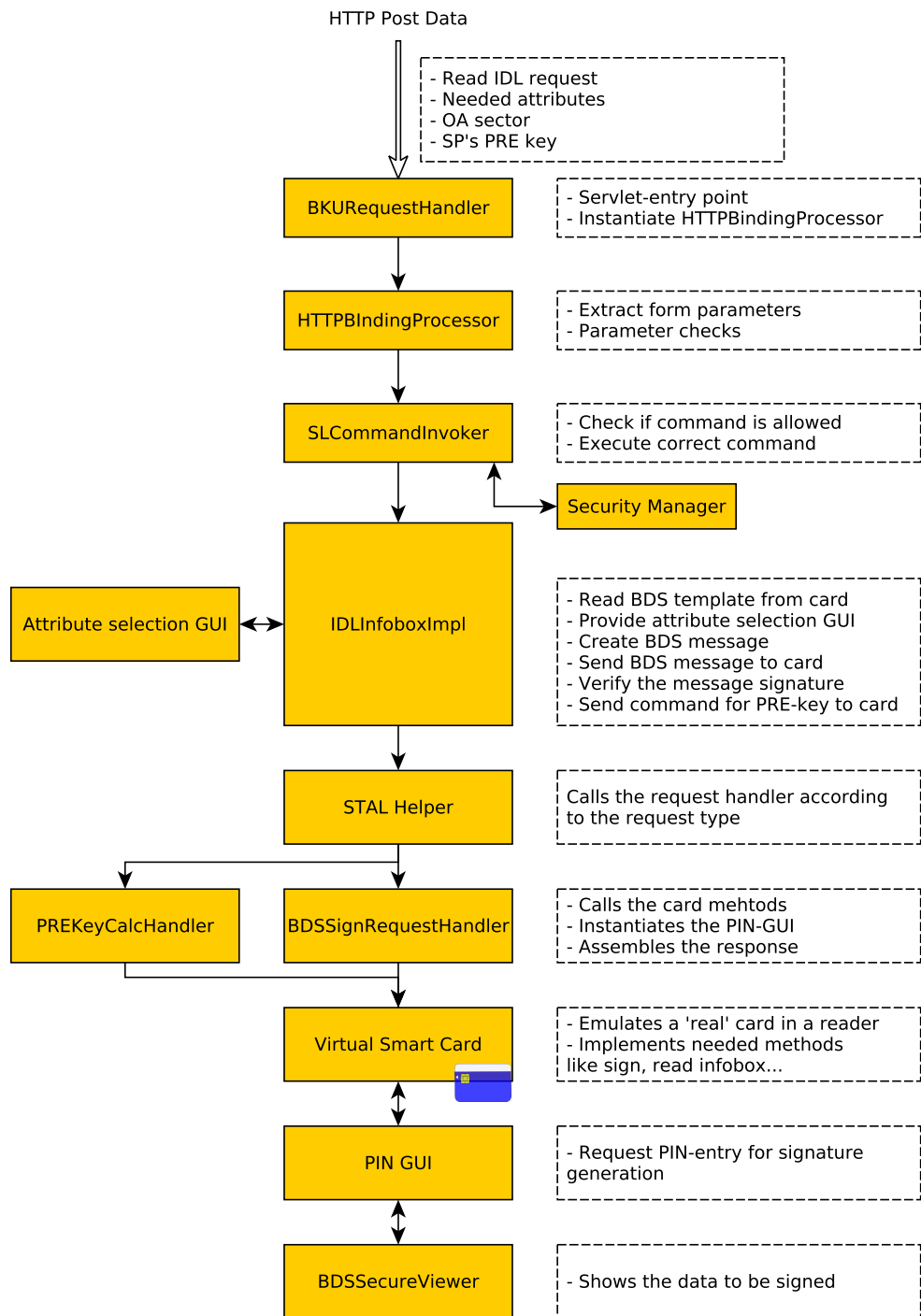


Figure 5.2: Block diagram representing the Architecture of MOCCA

5.5 Modifications in MOA-ID

The current eID implementation, as described in section 3.4, foresees a two step procedure for identification and authentication between MOA-ID and the citizen (MOCCA). In the first step the IDL is read from the card and sent back to MOA-ID where it is verified. In the second step MOA-ID creates a data block, called the AuthBlock, containing information regarding the citizen and the OA she wants to login and forwards it to MOCCA where the citizen signs it by providing her secure PIN. This signature is sent back to MOA-ID where it is verified and based on the identity data an AuthBlock is generated that may be picked up by the OA. In contrast to the current eID implementation the approach within this thesis only requires one step, as shown in figure 5.1, which results in a modified authentication sequence that had to be implemented. In this sequence MOA-ID issues a request to read the IDL from the Citizen Card including the OA sector, the SP's public PRE key and the required identity attributes. Based on the attributes the BDS template is instantiated, thus resulting in a BDS message representing the signed IDL. This BDS message is sent back to MOA-ID which is able to verify that the BDS message is a valid instance of the signed template. Based on this instance a modified AuthBlock containing the requested identity attributes may be created and picked up by the OA. Based on this sequence the following extensions had to be implemented within the current MOA-ID application:

- Forward sector and required attributes to MOCCA
- Capability for processing the modified IDL
- Implement BDS message verification algorithm
- Perform re-encryption of attributes
- Create modified AuthBlock
- Configuration for OAs has to be modified

5.5.1 Implementation Details

Figure 5.3 illustrates the basic principle of the login sequence including the applications within the blue boxes and the main classes within MOA-ID represented by the yellow boxes. After the user tries to access resources

5 Implementation

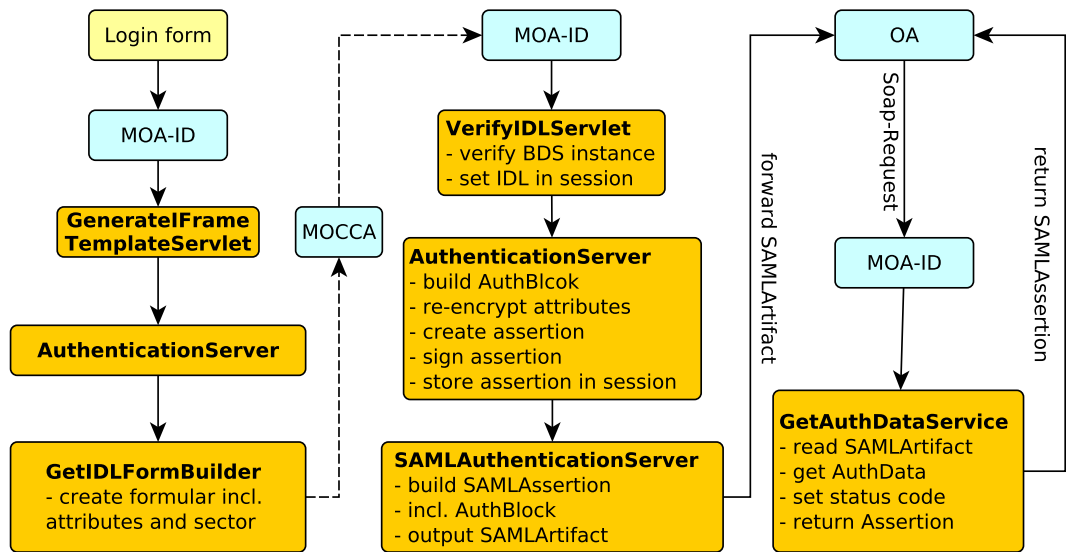


Figure 5.3: Block Diagram showing the basic Login Sequence

provided by an OA she receives a login form provided by MOA-ID. There the user has to select the CCE type that will be used for the authentication process and the *GenerateIFrameTemplateServlet* is called. Within this template the parameters, like CCE-type, are extracted from the request and the OA parameters are read from the configuration database. Within the MOA-ID configuration it is possible to configure which attributes are required for a dedicated OA. In the next step an instance of the *AuthenticationServer* class is created and the authentication is started. The *GetIdentityLinkFormBuilder* object, for generating the request for reading the IDL from the Citizen Card, is instantiated and the web form, including the requested identity attributes as well as the OA sector, is created and returned to the user's browser. The browser handles the communication with MOCCA by forwarding the Security Layer request from MOA-ID to MOCCA where the IDL (BDS message) is created and sent back to MOA-ID. MOCCA also calculates the re-encryption key, based on the citizen's private and the SP's public PRE key, and also returns it to MOA-ID.

Within MOA-ID the *VerifyIDLServlet* is called where the BDS message is verified and attached to the current MOA-session. In the next step an

5.6 Modifications for Online Applications

instance of the *AuthenticationServer* class is created where the IDL attributes are re-encrypted using the re-encryption key calculated by MOCCA. This re-encrypted attributes are then wrapped into an *AuthBlock*, and an assertion based on this *AuthBlock* is created and signed using MOA's private signing key. This signed *AuthBlock* is then stored within the MOA-session. In the next step a *SAMLArtifact*, referencing the signed *SAMLAssertion*, is created and forwarded to the OA.

The OA receiving the *SAMLArtifact* is now able to request the corresponding *SAMLAssertion*, respectively the encrypted identity attributes by issuing a request. The common way is to send a SOAP request containing the *SAMLArtifact* to MOA-ID where the *GetAuthDataService* object handles the SOAP request. It checks if a *SAMLAssertion* corresponding to the received *SAMLArtifact* is available and returns it to the OA. The OA is able to verify the assertion signature, decrypt the attributes and grant or deny access to the requested resources.

5.6 Modifications for Online Applications

Only minor modifications have to be implemented by SP's for preparing their OAs for the IdM model implemented within this thesis. The implemented sequence for secure identification and authentication consists of the following steps:

1. Forward user to MOA-ID that carries out the authentication.
2. Request SAML assertion, containing the *AuthBlock*, at MOA-ID by providing the *SAMLArtifact*.
3. Verification of the *AuthBlock*'s signature (DSS).
4. Decrypt the identity attributes using private PRE key.
5. Grant or deny access based on attributes.

Basically the sequence does not have to be modified in a fundamental way because only step 4, the PRE decryption, has to be added to the current eID authentication procedure. So only a simple decryption module, taking as input the identity attributes and the SP's private PRE key and outputs the plain identity data, has to be added by the SP.

5.7 Identity Link Details

As already described the IDL within this thesis is represented by a BDS message containing the citizen's identity attributes as well as pre-calculated ssPINs in an encrypted manner. This BDS message is signed by the Citizen and the signature element is based on a XAdES-type signature defined by [Cruellas et al., 2003] and will be described within the following section 5.7.1.

5.7.1 Signature Format

To ensure the authenticity of the IDL the signature block is bound to the BDS message. This signature block, see appendix 9.1, contains all information for verifying that the provided IDL is a valid instantiation of the template (signed by the SRA) stored on the citizen's token. The basic structure of this XAdES signature is illustrated within figure 5.4. The term XAdES stands for XML Advanced Electronic Signature and represents the format used for advanced electronic signatures as defined by [The Council of the European Union, 2000]. The *SignedInfo* element basically contains references to the objects that are signed. In the case of this thesis this is the BDS message (IDL) itself and the XAdES *SignedProperties* that have to include the signing time, as well as advanced information regarding the citizen's public key certificate defined within the *KeyInfo* object. The *SignatureValue* contains the BDS signature value based on the citizen's private key. The last block, the *UnsignedProperties*, may contain additional information for the verification of the IDL, like in this case the SRA's public key certificate. Based on this authentication block a third party, in this case the IdP, is able to verify the IDL by carrying out the BDS message verification algorithm as provided within section 4.3.3.

5.7.2 Citizen Certificate

In the current eID solution the public key provided by the IDL is wrapped within a qualified certificate. Beneath the key this certificate also holds

5.7 Identity Link Details

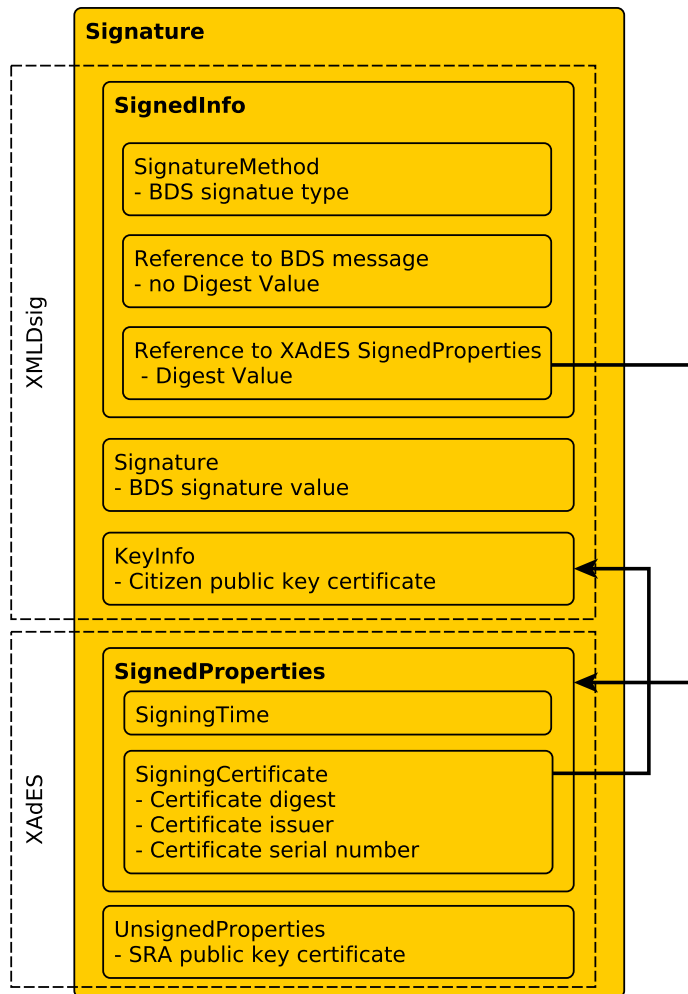


Figure 5.4: XAdES Signature Block

5 Implementation

the first and last name of the citizen the key belongs to. Because the implemented IdM model foresees selective identity disclosure, which also means redacting first and last name from the IDL is possible, including this attributes within the public certificate would be counterproductive because anybody is able and allowed to read the public key certificate. A solution is the use of pseudonyms within the certificate. The following sections from the Austrian Signature Law [Republik Oesterreich, 2010] define some rules regarding pseudonyms:

Section 5: A qualified certificate shall contain at least the following information:

3. the name of the signatory or a pseudonym, which must be indicated as such;

Section 8: Issuing qualified certificates

(4) CSP may use a pseudonym in lieu of the signatory's name according to the certification policy, if the applicant so requests. The pseudonym shall not be offensive or obviously open to confusion with names or signs.

Section 22: Data protection

(2) If a pseudonym is used, the CSP shall transmit data on the signatory's identity insofar as there is prima facie evidence of an overriding legitimate interest in establishing his identity as defined in § 8 para. 1 number 4 and para. 3 of the Data Protection Law. The transmission shall be recorded.

Based on this definitions from the Austrian signature directive the usage of pseudonyms, which is required for the desired model, should be fine according to the legal requirements of advanced electronic identification and authentication.

5.8 Security Layer

The Security Layer specification [Hollosi and Karlinger, 2014] basically defines all the available commands that may be carried out by the Citizen Card,

e.g. *CreateXMLSignatureRequest*, *InfoboxReadRequest* or *EncryptXMLRequest*. One target of this thesis was to reduce the required modifications within the Security Layer standard to a minimum, so the Read IDL command is used for initializing the BDS template instantiation and the PRE key calculation on the card. The additional parameters, like the needed attributes (first name, last name and date of birth), the sector as well as the SP's public key are sent to the CCE as additional form parameter values and thus they are not directly added to the SL command, so no changes have to be implemented within the Security Layer specification.

5.9 Proxy Re-encryption Library

The library used within this thesis has been implemented by [Nuñez, 2013]. As already described within section 4.4.3 it is a very basic implementation directly utilizing the `jpbc`¹ library. To use the library within this thesis a few modification based on the following requirements had to be made:

- Private key storage within Java keystore
- Creation of public key certificates for PRE keys
- Usage of standard JCE API

Therefore a Java cryptographic provider, providing the required algorithms, has been implemented. It also defines the two key types, `PREPrivateKey` and `PREPublic`, that may be created using the implemented `PREKeyFactory`. For storing the keys within keystores or for creating a public key certificate the keys may be represented within ASN.1² structures. A key within the `nics` library is represented by a simple `jpbc` element object which support the `toBytes()` method. Thus the ASN.1 structure of a public key contains three elements:

- the algorithm id as first entry
- the key converted to a byte array and stored within a `BIT_STRING`
- the public PRE parameters within a byte array and also stored within a `BIT_STRING`

¹gas.dia.unisa.it/projects/jpbc/

²<http://www.itu.int/en/ITU-T/asn1/Pages/introduction.aspx>

5 Implementation

Using this simple structure it is possible to store keys within keystores to transfer them and wrap the public key within a X.509 certificate that may be used for public verification or, in the case of PRE, for the calculation of the re-encryption key based on the SP's public key certificate.

5.10 Conclusion

This section provided an overview of the implementation and modifications within the different components participating within the Austrian eID procedure. The first step, described in section 5.2, was to implement an application representing the CSP and the SRA for certifying the required key material (BDS, PRE, DSS) and also for the creation of an IDL based on a citizen's identity data. This whole process basically simulates the registration procedure that has to be carried out by trusted entities within a productive environment.

Based on the identification process, presented within section 5.3, sections 5.4 and 5.5 provide details according to the main applications involved within the Austrian eID solution. The first application that has been modified is the CCE (MOCCA). MOCCA has been modified to accept additional parameters (requested attributes, OA sector and PRE key) within a Security Layer request from MOA. Based on these attributes the commands for instantiation of the BDS template and calculation of the re-encryption key have been implemented. For demonstration purposes a virtual Citizen Card has been implemented. Based on the requested attributes the citizen now is able to check them and, if she wants to, modify the attribute selection based on a simple user interface. The next component that had to be modified was the IdP (MOA-ID). It now is capable of forwarding the sector, required attributes and public key to MOCCA, requesting the IDL and perform the BDS verification based on the received IDL. MOA-ID performs the re-encryption of the attributes and forwards them to the OA.

An OA is able to decrypt the identity attributes directly by using the appropriate algorithm provided by the PRE library. The verification of the AuthBlock was already required for the current eID approach and thus had not to be modified. The last sections of the implementation chapter provided

5.10 Conclusion

a short description of the signature format (XAdES) used for signing the IDL and also a short description of using pseudonyms within this approach to actually provide selective identity disclosure.

6 Evaluation

6.1 Introduction

Based on the implementation that has been done within this thesis this chapter will evaluate the implemented model and provide information regarding different security issues that may arise within the public cloud. Section 6.2 starts with a description of a common architecture that may be used for deploying MOA-ID within a private environment. For moving MOA-ID into the public cloud a concrete cloud provider had to be selected. Therefore section 6.2.2 provides basic knowledge according to cloud computing and the available models while section 6.2.3 describes the cloud environment, namely Jelastic, that has been used within this thesis for the deployment of MOA-ID. This section also provides some details regarding the costs that may arise within such a cloud deployment. Based on the cloud deployment section 6.2.4 describes security threats that have to be considered when deploying applications within the public cloud. Some of the issues are specifically addressed to this master while others have to be considered in general. Based on this security threats 4 different deployment models are presented and are evaluated within section 6.2.4.4.

While the first part of this chapter focuses on the public cloud deployment of the IdP the second part provides information regarding the security token containing the citizen's private data. In the Austrian eID system this concept is known as the Austrian Citizen Card and has been presented in detail within section 3.3. Within this thesis the Citizen Card has been implemented as a virtual smart card only, so the last section 6.3 of this chapter provides a secure hardware based approach for implementing the Citizen Card concept within an Advanced RISC Machines (ARM) powered smart phone using ARM's TrustZone concept.

6.2 Cloud Deployment of the IdP

6.2.1 Private Environment Architecture

The IdP (MOA-ID) is implemented in Java, thus resulting in a web application archive (WAR) containing all the application files like classes, configuration files, libraries, etc. For deploying a WAR file a server providing the required APIs is needed. For this purpose every application server capable of running Java code may be used. Because the default deployment is based on an Apache Tomcat¹ environment this section provides details that focus on this environment. Tomcat provides a servlet container, a connector framework and a Java Server Pages (JSP) engine to enable the creation of dynamic web content. Using dedicated connectors, e.g. the HTTPs connector, Tomcat may be run as a web server on it's own. Attaching a MySQL database to a single tomcat instance basically represents the simplest setup for running MOA-ID. Even if this setup ensures the basic functionality of MOA-ID, it does not provide the same level of scalability and performance as the model provided in figure 6.1. In this architecture a dedicated web server, the default is the Apache webserver² which will be called Apache in the following, is used as a first stage for processing HTTP(s) requests. Apache is responsible for serving static web content and Tomcat provides dynamic content and executes Java servlets. If the incoming request does require additional dynamic processing it is forwarded to one of the Tomcat instances via the Apache JServ protocol (AJP). The Tomcat instances are connected to one or multiple databases using the Java Database Connectivity (JDBC) interface. Of course the provided architecture is more complex than a single standalone Tomcat server but it provides the following advantages [*Apache Web Server*]:

- **Clustering** - different URL namespaces where every Tomcat instance processes only defined URLs, e.g. `www.domain.at/tomcat1/` processed by first Tomcat instance and `www.domain.at/tomcat2/` processed by the second instance.

¹<http://tomcat.apache.org>

²<http://httpd.apache.org>

6.2 Cloud Deployment of the IdP

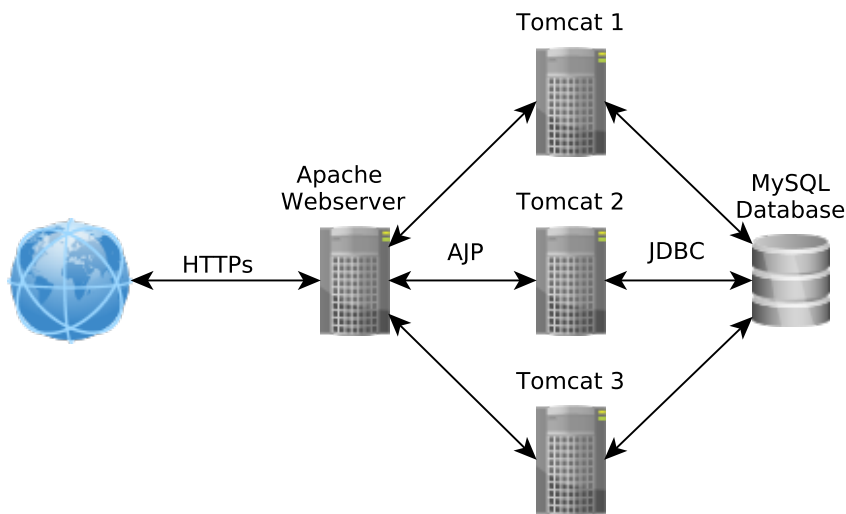


Figure 6.1: Apache Tomcat and Webservice Architecture

- **Load balancing** - a load balancing module provides different algorithms that distribute requests based on defined metrics, e.g. distribute equal number of requests to each Tomcat instance.
- **Redundancy** - if one of the Tomcat instances crashes the user will not even recognize it. Requests will be forwarded to other instances.
- **Stability** - Apache performs better and provides more stability against invalid packets, invalid requests and dropped connections.
- **Add-ons** - add-ons like CGI or PHP are much more efficient and work out of the box within Apache.
- **Speed** - Apache is fast at serving static pages. Tomcat provides the dynamic content.

The architecture shown in figure 6.1 is the common approach for deploying MOA-ID within a private environment. Using an appropriate cloud service provider it is possible to create the same architecture within a cloud environment. Different providers provide different types of service which will be described within the following section 6.2.2.

6.2.2 Cloud Computing

The term cloud computing has been defined by [Mell and Grance, 2011]. The basic idea behind cloud computing is to provide a consumer with computing resources that scale automatically with increasing or decreasing load, thus providing automatic load balancing. The resources are available via a network and the load and resource usage is recorded for information and payment purposes for the provider and the customer. The authors [Mell and Grance, 2011] defined three basic cloud computing service models:

- **Software as a Service (SaaS)** - the software itself is deployed by the provider and the customer may access it via a simple interface, like a web browser. An example is an email client or a web-based calendar application.
- **Platform as a Service (PaaS)** - the cloud provider provides the operating system, the tools and libraries that may be used by the customers to develop and deploy their applications. The customer is not allowed to configure the underlying cloud infrastructure, e.g. server, OS or storage, but she may be able to configure the application-hosting environment.
- **Infrastructure as a Service (IaaS)** - the provider offers the basic infrastructure like hardware, virtual machines (VM) or data storage. The customer is responsible for deploying her applications and may install own operating systems on the VMs but she is not allowed to control the underlying cloud infrastructure.

In addition to the service models the authors also defined four types of deployment models.

- **Private cloud** - the infrastructure is owned and managed by a single company and may only be used by the company's employees. The hardware may be installed within the company or even in another location.
- **Community cloud** - the infrastructure is owned and managed by a subset of customers from companies that want to access/use shared resources.
- **Public cloud** - the infrastructure is owned and managed by a third entity called the cloud provider. It is intended for public usage.

- **Hybrid cloud** - represents a combination of the other three deployment models. E.g. encrypted public cloud storage where the encryption and the data manipulation is handled within a private cloud environment.

Beneath providing a maximum level of scalability and availability the target of selecting an accurate deployment model was also based on the requirement of minimal to zero required code modifications when moving from a private to a cloud environment. So either a self-installed and configured IaaS model or a PaaS model capable of running Java code has to be selected. There are already a number of PaaS providers supporting Java applications, see [Yuan, 2011], providing a huge variety of technology stacks like Tomcat, Java SE and Java EE. Although some of them are limited within their APIs. Google AppEngine³, for example, does not support FileIO via the standard Java API, thus applications developed for local environments have to be adopted to run in the AppEngine. For the public cloud deployment of MOA-ID an appropriate provider, supporting Tomcat and the Java standard APIs, had to be chosen. A developer has many choices regarding Java cloud providers. There are many PaaS providers like Google's AppEngine or Heroku⁴ and also the IaaS providers like Amazon Web Services (AWS)⁵. While IaaS providers provide much more flexibility than PaaS they also require much more effort regarding the initial setup. While most of the PaaS environments provide the same functionalities, like dynamic resource allocation or user interface based configuration, they mainly differ in terms of usability. Based on the usability factor and also the concept of a pre-paying model, where arising costs are directly charged to a pre-paid account, the cloud environment Jelastic has been selected for this thesis. Jelastic is a special type of a PaaS environment and the details will be given in the following section 6.2.3.

³<https://appengine.google.com/>

⁴<https://www.heroku.com/>

⁵<http://aws.amazon.com/>

6.2.3 Jelastic

6.2.3.1 Introduction

Jelastic⁶ defines itself as a platform-as-infrastructure (PaI) provider which is a term created by the combination of PaaS and IaaS. This term is based on the fact that Jelastic offers PaaS tools and libraries (e.g. Apache Tomcat, mysql, etc.) but it also provides the possibility to configure the underlying hardware (basically RAM and CPU frequency). Jelastic supports the standard APIs and does not implement any additional, proprietary APIs so it is possible to deploy locally developed Java applications without any source code modifications.

6.2.3.2 Pricing Model

Jelastic is based on a Pay-per-Use model [Jelastic, 2014], thus only used resources have to be paid. Resources are measured in so-called cloudlets, where one cloudlet equates 128 MB of RAM and 200 MHz of CPU power. Table 6.1 provides an overview of the costs that may occur depending on the used components.

Type	Price per hour	Price per month
Cloudlet	0,0085 Euro	—
Public IP	0,00487 Euro	3,56 Euro
SSL	0,00487 Euro	3,56 Euro
1 GB Storage	0,0001 Euro	0,07 Euro

Table 6.1: Jelastic Pricing Information, see [Jelastic Pricing]

If the applications are disabled no costs arise for the customer providing the application. To provide an idea of the average costs that may arise when deploying an application like MOA-ID in the cloud an environment has been

⁶<http://jelastic.com/>

6.2 Cloud Deployment of the IdP

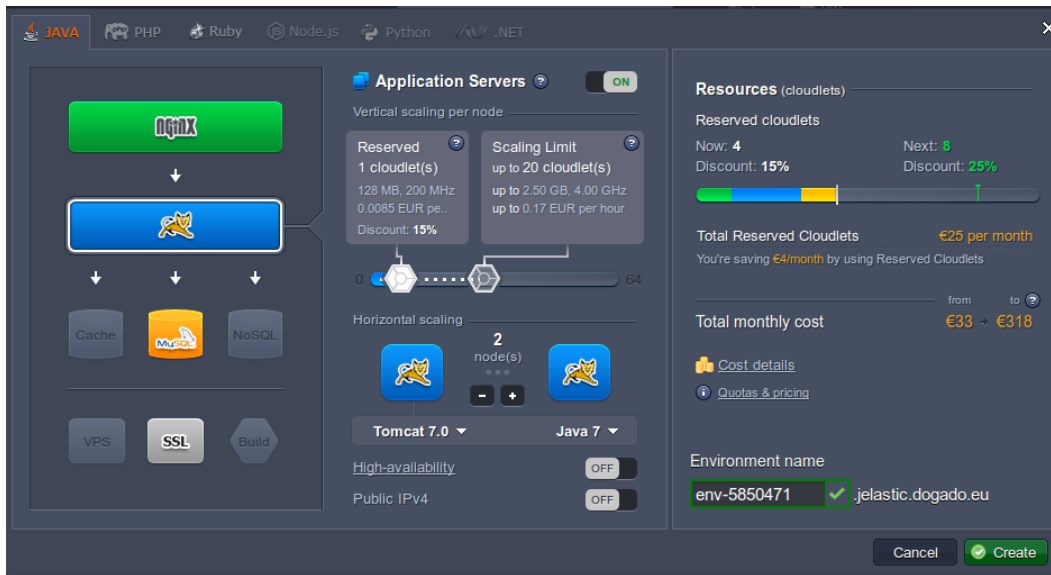
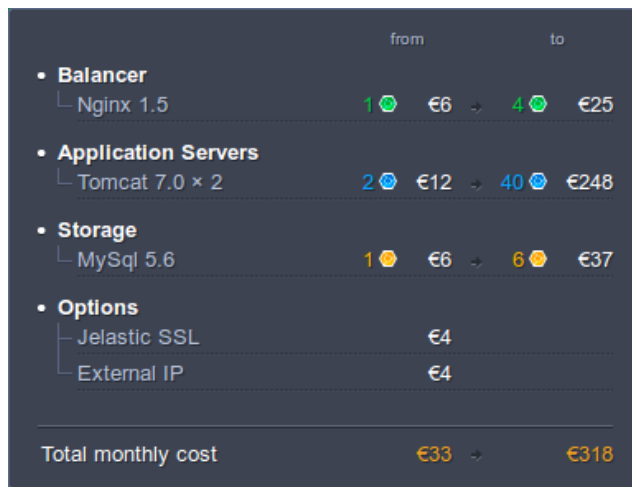


Figure 6.2: The Jelastic Environment Configuration

created as shown in figure 6.2. The environment contains load balancing, two tomcat instances (each using up to 20 cloudlets (2,5GB RAM/4GHz CPU), a MySQL database (up to 6 cloudlets) and provides secure HTTP connections using SSL. Over all 50 cloudlets, which means up to 10 GHz of CPU frequency and about 6 GB of RAM, will be available for MOA-ID and may be automatically activated on demand. This results in average costs between 33 and 318 euros per month depending on the average load. If MOA-ID is idle, meaning no active authorization requests, only some RAM and HD memory is needed and basically no CPU load, so only about 4 cloudlets are active resulting in very low costs per month. Figure 6.3 provides the detailed information regarding the costs per environment-element per month. The figure contains the different installed components, like Tomcat or MySQL database, on the left side and the cost range, depending on the used resources, on the right side. Again, this setup only provides an example to demonstrate the range of the costs that may arise and does not represent a recommendation for a productive environment.

6 Evaluation



	from	to
• Balancer		
└ Nginx 1.5	1 🟢 €6	4 🟢 €25
• Application Servers		
└ Tomcat 7.0 × 2	2 🟢 €12	40 🟢 €248
• Storage		
└ MySQL 5.6	1 🟡 €6	6 🟡 €37
• Options		
└ Jelastix SSL		€4
└ External IP		€4
Total monthly cost	€33	€318

Figure 6.3: The Jelastix Environment Configuration

6.2.3.3 Jelastix Conclusions

Jelastix, as well as other cloud solutions (e.g. Heroku, Google AppEngine, ...), provide a huge variety of services for deploying applications within the cloud. The huge advantage of Jelastix, describing itself as Platform-as-Infrastructure cloud provider, is the modularity of the single components that may be easily connected to each other via a simple user interface. So it provides more flexibility in terms of environment configuration than traditional Platform-as-a-Service providers while it is still be much easier and faster to setup than Infrastructure-as-a-Service environments. Even if such cloud solutions seem to be a perfect replacement for private-environment deployment of MOA-ID there are still some security issues exist that will be addressed within section 6.2.4.

6.2.4 Security Threats

This section provides an overview of security threats that may occur within a public cloud deployment of MOA-ID. Even if some of the threats may be

specific to MOA-ID, other ones may also have to be concerned for general application deployment in the cloud.

6.2.4.1 Binding AuthBlock to BDS Message

After MOA-ID has received the BDS message it has to extract the identity attributes, re-encrypt them for the SP, wrap them into an AuthBlock, sign it using its private key and forward it to the SP. This is the point where the notation of 'honest but curious IdP' gets important. This definition basically means that the IdP is allowed to inspect the incoming packets, because they are encrypted anyway, but it should perform its tasks as expected. In this case the requested task is to actually put the re-encrypted attributes into the AuthBlock and not forged encrypted attributes. The problem here basically is that the SP is not able to differentiate between identity attributes that have been re-encrypted for him by the SP and arbitrary attributes encrypted for him by the IdP, thus both cipher texts have a first-level encryption format.

One approach to mitigate this threat is shown in figure 6.4. The signed IDL (BDS message) is forwarded from the citizen to the IdP. The IdP verifies the BDS message signature, thus ensuring that it represents a valid BDS template instantiation. It extracts the identity attributes (first name, last name, birth day and ssPIN) and re-encrypts them using the included re-encryption key. The attributes are wrapped into a AuthBlock and get signed by the IdP using its private key. In contrast to the implemented model, now the BDS message and the re-encryption key are also added to the AuthBlock which is forwarded to the SP. Depending on the security requirements and the level of trust the IdP operates in from the SP's point of view, the SP is now able to re-perform the re-encryption operation on one, some or all of the attributes within the BDS message and compare the re-encrypted values to the ones contained in the AuthBlock. Only if they match the IdP did his tasks as expected. In terms of efficiency this of course is not a perfect solution but in terms of data integrity and security it is a possible solution. Even a random re-encryption check, e.g. only re-encrypt one randomly chosen attribute, could help to disclose an 'evil' IdP.

6 Evaluation

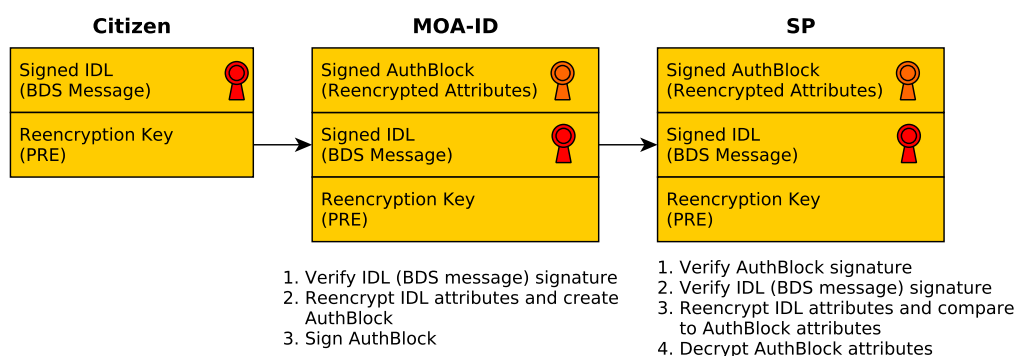


Figure 6.4: Ensure Binding between IDL and AuthBlock

Identifier	Description
AV ₁	the cloud provider may issue signatures on behalf of MOA-ID for arbitrary data
AV ₂	the cloud provider may use any other key to sign identity data
AV ₃	the cloud provider may forward MOA-ID's private key to other entities
AV ₄	a hacker may heist the secret keys

Table 6.2: Cloud Deployment - Attack Vectors

6.2.4.2 Private Keys in the Cloud

Within the context of BDS MOA-ID has to create an AuthBlock containing the citizen's attributes and sign it using its private DSS signing key. In a local deployment of MOA-ID this key resides in a Java keystore and the corresponding password is stored within a configuration file located on the server. Moving this approach to the public cloud opens up new attack vectors, see table 6.2.

Thinking about this attack scenarios some of the points mentioned above may be easily mitigated. Attack vector AV_1 is not a problem if the SP implements the verification algorithm as described within section 6.2.4.1, thus creating an arbitrary AuthBlock leads to a verification error, because the AuthBlock attributes do not match the ones within the IDL represented by the attached BDS message.

Attack vector AV_2 is also easily recognizable because the used keys are certified by the SRA, thus approaches like self signed certificates or certificate chains not ending at a trusted CA (SRA) also lead to a verification error at the SP.

Attack vector AV_3 and AV_4 are not that easily to mitigate because the keys are simple 'files' stored on the cloud provider's server. If an evil cloud provider or an attacker is able to get the file, she may forward them to third entities and even create signature in the name of the IdP. Storing secret keys has emerged to an important field of research within the last years. Thus the following section will provide information regarding so-called hardware security modules (HSM) for storing keys in a secret manner.

6.2.4.3 Usage of HSMs in the Cloud

A HSM⁷ basically is a hardware device for secure creation, storage and managing cryptographic key material and it provides access to cryptographic primitives. HSMs may implement a huge amount of different algorithms, like

- Asymmetric algorithms, like ECDSA, RSA, ECDH
- Symmetric algorithms, like DES or AES
- Key generation
- Random number generation (hardware-based)
- Hash functions

The most important point regarding keys and HSM's is that a key created within the HSM will actually never leave the HSM, even if there are some secure mechanisms to backup encrypted versions of keys or for secure HSM

⁷http://en.wikipedia.org/wiki/Hardware_security_module

6 Evaluation

cloning to ensure key availability even if the HSM gets destroyed. The secret key will actually only be used for cryptographic algorithms within the HSM. In the case of the proposed model this means that an AuthBlock generated by MOA-ID, based on the IDL, is sent to the HSM and the signature algorithm is applied within the HSM. The signed AuthBlock is then sent back to MOA-ID and forwarded to the SP. At the time this thesis has been written only Amazon was capable of providing a HSM in the cloud, called the AWS CloudHSM [AWS CloudHSM]. Figure 6.5 represents the basic architecture of the AWS CloudHSM. For using a CloudHSM a virtual private cloud

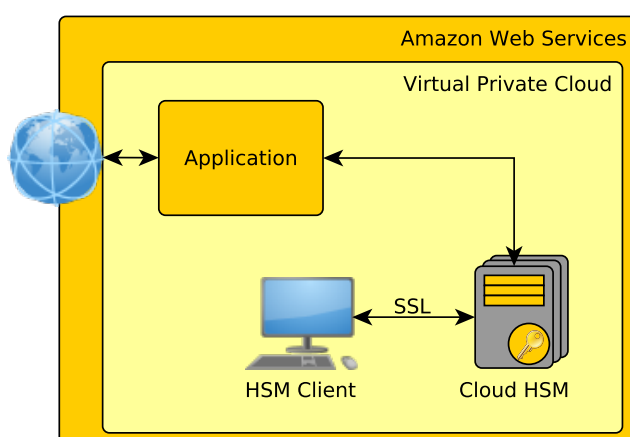


Figure 6.5: Amazon Web Services CloudHSM Architecture

(VPC) has to be deployed within the AWS. A VPC represents a logically isolated area within the AWS cloud. Within this VPC the customer is able to define subnets for public or only private access, define IP-ranges and enable access control mechanisms. A dedicated application, the so-called HSM client, runs within a private-only accessible area. Using this client application only the customer is able to create and manage keys within the CloudHSM. Amazon itself claims to have no control over the keys and thus they are not responsible for key creation and management. Applications running within this VPC may access the cryptographic primitives offered by the CloudHSM and thus perform cryptographic algorithms based on private keys stored within the HSM. Via a defined and access-controlled gateway dedicated applications running within the VPC may be accessed

6.2 Cloud Deployment of the IdP

from the Internet. For using the CloudHSM within the context of MOA-ID there are mainly two possible solutions.

The **first** one is to run MOA-ID as an application within the Amazon VPC and thus having direct access to the HSM. The **second** approach is to run MOA-ID on a different cloud provider, e.g. Jelastic, and use the CloudHSM via an API provided over a secured communication channel (SSL) by another provider. Using one of these approaches one is able to mitigate the attack vectors AV_3 and AV_4 , defined in section 6.2.4.2. The CloudHSMs also support redundancy, thus it is possible to run multiple HSMs within one domain. This infrastructure provides secure key replication over all HSMs and automatic load balancing, thus providing a high level of scalability and availability.

Even if this seems to be the perfect solution to all security relevant issues, another question may arise: Why should someone trust the cloud provider that it really provides a real HSM? Actually state of the art HSMs provide a way to prove that a dedicated private key resides within a 'real' HSM. For example SafeNet HSMs⁸, that are used by Amazon Web Services, contain unique device identity keys that are certified by the SafeNet authority. Based on this certified key the HSM may create key confirmations for keys that have been created and reside within the HSM. A user is able to verify this key confirmation against the public SafeNet CA certificate.

If this still does not provide the accurate level of security a **hybrid model**, as shown in figure 6.6, may be a solution. To mitigate the problem of trusting a HSM within the cloud a HSM could be operated by a trusted public authority within their private computation environment. The key management may only be carried out by a qualified employee at a client within this private environment. Based on access restrictions via a firewall the HSM could be made available to MOA-ID for signing authentication blocks. This solution definitely provides the highest level of data security and privacy. Even the performance of such HSMs should be sufficient. E.g. the Luna SA 7000⁹ is able to perform about 350 to 1200 RSA (2048 bit)

⁸www2.safenet-inc.com/AWS-guides/Remotely_Verifying_an_HSM_is_Real_Tech_Note.pdf

⁹www.safenet-inc.com/data-encryption/hardware-security-modules-hsms/luna-hsms-key-management/luna-sa-network-hsm/

6 Evaluation

operations per second. That definitely will not be the limiting factor in the model.

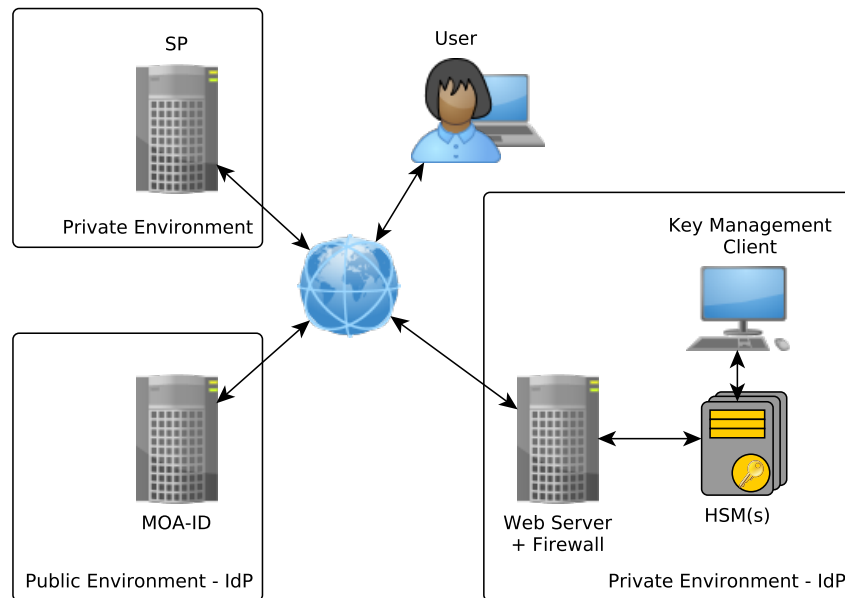


Figure 6.6: Hybrid IdM Model using a HSM

6.2.4.4 Cloud Deployment - Conclusion

This section will provide a short comparison of the different cloud deployment models presented within the former sections. The four different models are:

1. Simple cloud deployment without HSMs
2. Public cloud provider offering own HSM services
3. Public cloud provider running MOA-ID and second cloud provider offering HSM services
4. Public cloud provider running MOA-ID and HSM deployed within private environment

6.2 Cloud Deployment of the IdP

	Model			
	1	2	3	4
Trust Level	-	~	~	+
Availability	+	+	+	~
Scalability	+	+	+	~
Data security (privacy)	-	~	~	+
Costs	+	~	~	~

Table 6.3: Requirement evaluation on different models

Table 6.3 evaluates the different requirements against the different models. A '+' means that the model fits the requirement and a '-' means that it does not. A '~' means a partly fulfilled requirement.

As expected all cloud-based models provide a high level of availability and scalability. Model number 4 was evaluated slightly lower, because it is a huge effort, in terms of costs and development, to reach the same level of availability and scalability as cloud providers nowadays. The simplest model, number 1, fails in terms of required trust into the cloud provider and data security. Because the key is fully available to the cloud provider it basically can do anything with it, e.g. sign arbitrary data, forward the key, etc. Model number 2 and 3, even if they provide a HSM-based key storage, received slightly reduced ratings in terms of trust and data security. Here model number 4, utilizing a private HSM definitely succeeds against the other models. The costs regarding the last 3 models may vary depending on the needed resources, especially on the number of HSMs within the private environment to guarantee the required level of availability and scalability.

Based on this evaluation it is obvious that selecting a dedicated model for a public cloud deployment strongly relates on the application's security requirements. In the case of a productive operated MOA-ID the models 2, 3 or 4 are definitely recommended. To provide the maximum level of security, respectively data integrity model 4 is the model of choice.

The requirements defined within table 6.3 are described in more detail in the following:

6 Evaluation

- **Trust Level** - how much trust is in the cloud provider is required; based on availability of secure key storage
- **Availability** - minimizing the down time of the cloud provider
- **Scalability** - how good does the solution scale based on user numbers; basically every Citizen should be able to use the IdP
- **Data security** - how secure are my data and keys
- **Costs** - including installation costs and running costs for maintenance

6.3 Hardware Platform Evaluation

The first part of this evaluation mainly focused on the IdP and its deployment needs. This section focuses on the other side, namely the electronic identity token. Within this master thesis a virtual card simulating the Citizen Card, which represents the Austrian eID, has been implemented in Java. This virtual Citizen Card is directly integrated within the CCE, respectively MOCCA. One part of this thesis was to evaluate a hardware platform capable of using the concepts implemented within the virtual Citizen Card. The first intention was to use JavaCard. JavaCard offers a comprehensive set of cryptographic APIs capable of calculating standard cryptographic primitives on a low power device like a smart card. The OS is specifically designed for security-relevant applications, e.g. cash cards. The problem is that the JavaCard API does not provide the needed cryptographic primitives needed within this thesis, e.g. pairings are not supported. Another problem is that the access to the underlying co-processors is very restricted and thus it is not easily possible to access them for non-standard calculations as they are needed for pairing-based cryptography required for BDS and PRE. Implementing the required algorithms in software is very inefficient and, in most cases, also impossible regarding memory limits on current JavaCard platforms. The most promising approach within the last years regarding security and also availability, because most ARM-powered smart phones support it, is the ARM TrustZone that will be presented within the next section.

6.3.1 TrustZone

6.3.1.1 Introduction

The concept of the ARM TrustZone¹⁰ is provided within figure 6.7. In this concept a single processor is split up into two environments, a secure and a normal world. Those two environments are hardware-separated from each other. Every one of the two worlds provides a user and a privileged mode just like a common CPU. In addition to these two standard modes a third one, within the secure world, namely the monitor mode has been implemented. It enables the context switch between the normal and the secure world and is basically responsible for saving and restoring the current world's state and it may be entered from the normal world by a defined command¹¹. This architecture provides the possibility to run two operating systems (OS), one rich OS (e.g. Android) in the normal world and one compact and secure one in the secure world. Using this approach it is possible to split applications into a normal and a secure part and perform operations based on keys or private credentials within a secured and isolated environment. The TrustZone provides hardware mechanisms for protecting keys stored within the trusted environment. To enable all the provided security mechanisms a proper OS has to be chosen. One OS supporting the ARM TrustZone has been implemented by [Fitzek, 2014] and is called Andix. Currently Andix supports a quick start board (iMX53QSB) and the Qemu TrustZone. By using a TrustZone-aware OS on a TrustZone-enabled smart phone an evil application installed within the normal world that wants to heist your private data will not be able to access it. In the following section a Citizen Card model based on the TrustZone architecture will be presented.

6.3.1.2 TrustZone-aware Citizen Card

Using a dedicated smart card for secure identification and authentication has always been a barrier for many citizens. Austria's Citizen Card activation

¹⁰<http://www.arm.com/products/processors/technologies/trustzone/index.php>

¹¹Secure Monitor Call (SMC)

6 Evaluation

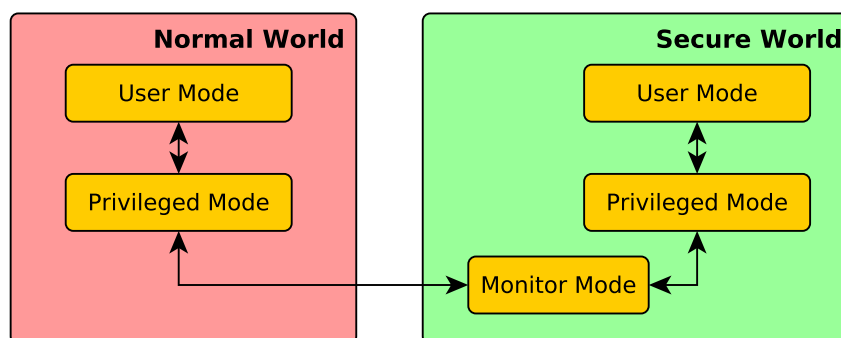


Figure 6.7: The Principle of the ARM TrustZone

statistics within the last years have shown that the smart card variant is still far behind in terms of acceptance compared to the smart phone signature¹². Using a TrustZone-enabled smart phone basically combines the advantages of two worlds, the secure key storage and signature creation within a user-owned security token and the easy access and usability of a smart phone. Figure 6.8 presents the concept providing the following advantages:

- secret keys are stored in a hardware-protected manner within the TrustZone
- no need for dedicated smart cards, smart phone represents the Citizen Card
- wireless connection between smart phone and PC running the CCE

The communication between MOCCA and the smart phone may be carried out either by NFC or Bluetooth. The easiest way to enable the communication between MOCCA and the Citizen Card is to use a NFC reader attached to a PC. Basically nothing would have to be changed within MOCCA because it is just a different physical transport channel. The Citizen Card application on the smart phone is split up into a normal world part and a secure world part. The secure world basically contains the following elements:

- **Secure Storage** - contains all the key material (BDS and PRE) and the IDL issued and signed by the SRA.

¹²<http://www.bka.gv.at/site/6791/default.aspx>

6.3 Hardware Platform Evaluation

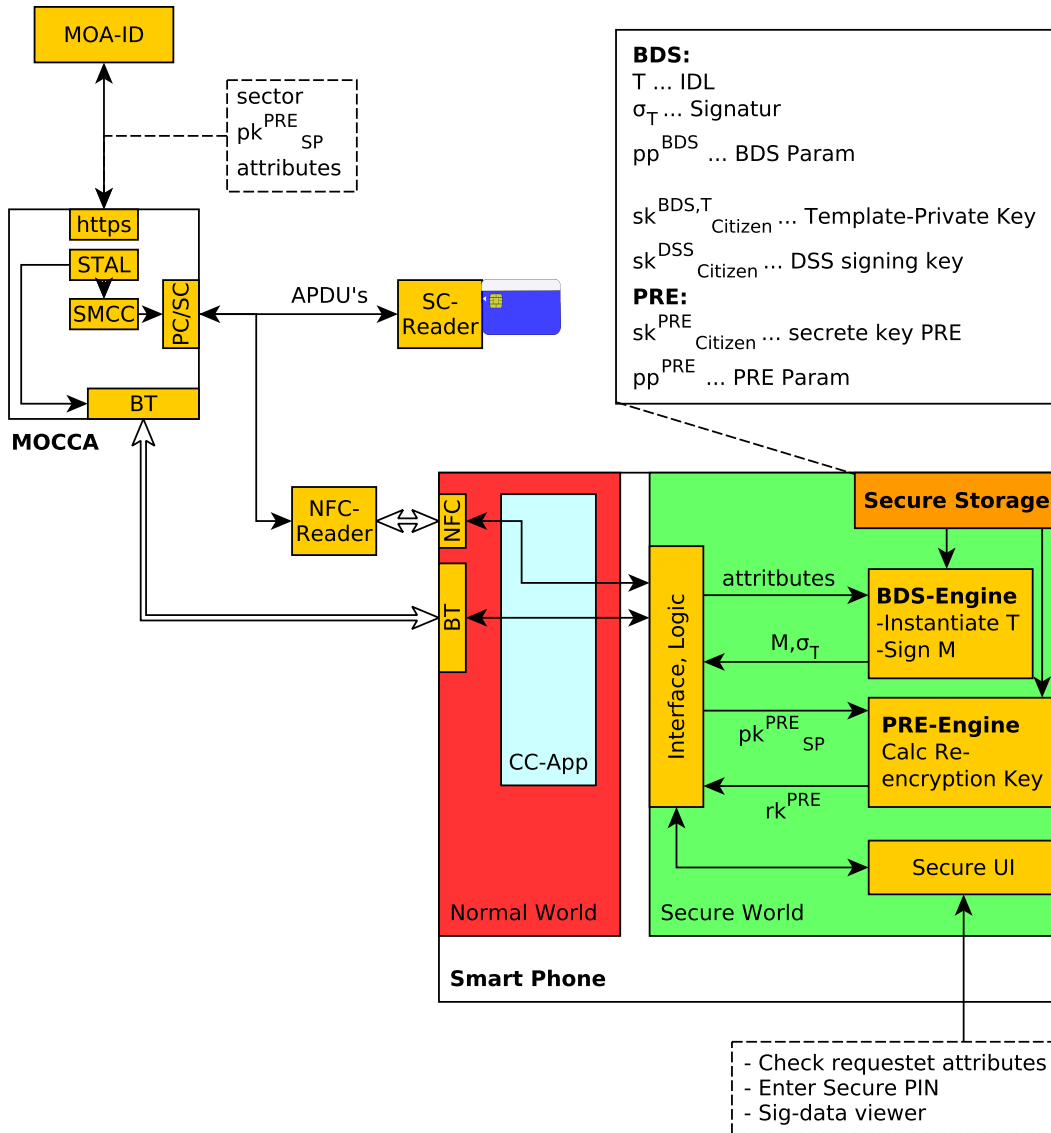


Figure 6.8: Concept CC-App on TrustZone

6 Evaluation

- **BDS-Engine** - gets as input the attributes requested by the SP and based on them it creates the BDS message and signs it.
- **PRE-Engine** - gets the public PRE key and calculates the re-encryption key for the IdP based on the citizen's private PRE key.
- **Secure UI** - is a secure user interface for checking which attributes are requested, entering the secure signature-creation PIN and also for checking the signature data.
- **Interface, Logic** - takes the command packets received via BT or NFC from MOCCA and implements a state machine that controls the required steps to perform the signature creation, key calculation and UI control.

Details regarding the general BDS signature creation process and proxy re-encryption have already been given within section 4.1, so only a short description will be given here. MOA-ID forwards the sector, the requested attributes and the SP's public PRE key to MOCCA. MOCCA forwards this parameters to the smart phone and requests the BDS instantiation of the template. The normal world application receives the command and the parameters and provides them to the secure world. After switching to the secure world, the citizen is able to verify the requested attributes, enter the PIN in a secure manner (isolated from the normal world) and check the signature data again via the secure UI provided by the secure world. Then the BDS message is created and the re-encryption key is calculated. The data is packed and sent back via MOCCA to MOA-ID where it is further processed. All in all this approach seems to be very promising, the problem here is that currently even TrustZone-enabled consumer smart phones do not provide the possibility to deploy own OSs neither their OS is able to run self-developed secure applications within the secure world. So right at the moment the model represents only a theoretical approach that may be available within a few years on consumer devices.

6.3.2 Conclusion Hardware Platform

Currently available smart cards, especially based on JavaCard, do not provide the hardware resources or cryptographic primitives to efficiently implement the model presented within this thesis. A pretty innovative concept

6.3 Hardware Platform Evaluation

is the ARM TrustZone providing an environment on a smart phone for secure data storage and secure code execution. A model representing a Citizen Card on a smart phone has been proposed to demonstrate the basic idea. Although this model may be promising, at the moment it may only be deployed on development hardware or software emulators. Based on the increasing need for secure identification and authentication services consumer hardware providing the required functionalities will hopefully be made available within a few years.

7 Summary and Conclusions

7.1 Summary

This thesis targets on a modification of the current Austrian eID concept to enable secure and qualified identification and authentication utilizing an identity provider deployed within a public cloud environment. The model, this thesis is based on, provides selective identity disclosure by using the concept of blank digital signatures as well as identity data privacy by using proxy re-encryption. Thus the identity provider only operates on encrypted data. To provide a basis for this thesis chapter 2 provides information about electronic signatures, respectively conventional digital signature schemes and modifiable signature which enable specified modifications of signed data without breaking the signature. While modifiable signatures are mainly used within very specific applications the current eID solution uses conventional digital signatures as a main building block. Details regarding the current Austrian eID solution are provided within chapter 3. Beneath the details regarding the Austrian eID, including the Citizen Card as well as legal framework details, this chapter also provides information regarding state of the art identity management models.

Based on these identity management models the model implemented within this thesis is presented within chapter 4. This chapter also presents the two main cryptographic schemes that are used within this thesis. Blank digital signatures are used to provide selective identity data disclosure and proxy re-encryption is used for providing an encrypted identity data flow from the citizen via the identity provider up to the service provider. Combining these two schemes enables the secure deployment of the identity provider within the public cloud. The concrete implementation within the Austrian eID components is described in detail within chapter 5. This chapter describes

7 Summary and Conclusions

every component's modifications, from the Citizen Card environment, over the identity provider up to the service provider, that hat to be carried out to implement the proposed model.

The last chapter 6 describes and evaluates the deployment of the implemented model, respectively the identity provider, within a concrete public cloud environment. The chapter describes security relevant issues regarding secure key storage within a public cloud provider's domain. The second part of chapter 6 focuses on the evaluation of an accurate hardware-based token for storing a citizen's identity data in a secure manner.

7.2 Conclusions

The identity management model implemented within this thesis, based on the Austrian eID software components, provides a proof-of-concept of the model developed by [Slamanig, Stranacher, and Zwattendorfer, 2014]. The required changes within the applications, e.g. MOA-ID or MOCCA, were a manageable effort. Although the legal and organizational steps that have to be carried out to use the implemented model within a real environment have to be addressed. The BDS scheme and the PRE scheme have to be implemented by the affected authorities (SRA, CSP) and it has to be verified that this schemes fulfill the requirements for advanced electronic signatures defined by the signature directive, respectively the Austrian signature law.

The next point that has to be addressed focuses on the selection of a cloud provider. Even if the implemented identity provider may be deployed within the cloud without risking that the cloud provider may accesses plain identity data, there have still be some actions regarding secure key storage in the cloud to be carried out by cloud providers and public authorities. The main problem is that at the moment no trusted third entities providing certification services for cloud providers, based on legal frameworks, are available. As long as no legal basis for cloud providers exist, it will be hard to determine or control what happens with privacy-sensitive data.

The last point of this conclusion handles the Citizen Card itself. In this thesis a virtual Citizen Card has been implemented within the context of

7.2 Conclusions

MOCCA. The problem in using a real smart card is that current smart card operating systems mainly provide APIs to standard cryptographic primitives and no direct access to the underlying cryptographic processors. So implementing e.g. pairing based cryptography was not achievable within this thesis. So an alternative model based on a secure hardware platform (ARM TrustZone) has been proposed. Most of the modern smart phones using TrustZone-enabled ARM processors so the proposed model may be implemented within the context of another thesis in the near future.

8 Acronyms

AJP	Apache JServ protocol
ARM	Advanced RISC Machines
AWS	Amazon Web Services
BDS	Blank Digital Signatures
CA	Certificate Authority
CCE	Citizen Card Environment
CCR	Central Register of Residences
CRL	Certificate Revocation List
CSP	Certification Service Provider
eID	Electronic Identity
HSM	Hardware Security Module
IaaS	Infrastructure as a Service
IaaS	Infrastructure as a Service
ICT	Information and Communication Technology
IDL	Identity Link
IdM	Identity Management
IdP	Identity Provider
JDBC	Java Database Connectivity
MOA-ID	Module for Online Applications - Identification

8 Acronyms

MOCCA	Modular Open Citizen Card Architecture
OA	Online Application
OCSP	Online Certificate Status Protocol
PaaS	Platform as a Service
PaaS	Platform as a Service
PaI	Platform as Infrastructure
PKI	Public Key Infrastructure
PRE	Proxy Re-encryption
SaaS	Software as a Service
SaaS	Software as a Service
SAML	Security Assertion Markup Language
SL	Security Layer
SigD	Signature Directive
SigG	Signatur Gesetz
SMC	Secure Monitor Call
sPIN	Source PIN
SP	Service Provider
SRA	sPIN Register Authority
ssPIN	Sector-specific Source Pin
STAL	Security Token Abstraction Layer
TTP	Trusted Third Party
VM	Virtual Machine
VPC	Virtual Private Cloud
WAR	Web Application Archive

9 Appendix

9.1 BDS-Message Signature Format

Listing 9.1 shows a signature element from a BDS message.

```
1 <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
2   <ds:SignedInfo>
3     <ds:CanonicalizationMethod
4       Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
5     <ds:SignatureMethod
6       Algorithm="http://www.iaik.tugraz.at/bdss#ECDSAwithSHA256" />
7     <ds:Reference URI="">
8       <ds:Transforms>
9         <ds:Transform
10          Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature" />
11         </ds:Transforms>
12       </ds:Reference>
13     <ds:DigestMethod
14       Algorithm="http://www.iaik.tugraz.at/bdss#ECDSAwithSHA256" />
15     <ds:DigestValue></ds:DigestValue>
16   </ds:SignedInfo>
17   <ds:Reference
18     Id="Reference-c7b5c8fc-1b"
19     URI="#SignedProperties-c7b5c8fc-1" />
20   <ds:DigestMethod
21     Algorithm="http://www.w3.org/2001/04/xmenc#sha256" />
22   <ds:DigestValue>UXGhCgHW1Bn56md37uffjC3q6MV+77719WpOrQz/fw=</ds:DigestValue>
23 </ds:Reference>
24 </ds:SignatureInfo>
25 </ds:SignatureValue>AwAAAwgCAAAABHRLc3QAAAABOAAAAACEA7FCnzfdCBj6rHHz9LEN/1V1Sz/n
26 mps
27 d/PKxEGgvQAAAAmcQAAAACEClBsBDVjs5Sbq4VwKcvkg6rH0o5hgOo/ZsunYxo
28 QN8AAAAZPgAAAABRSKdSYTYCPjKoyIMpcubz3gQAAAOpYAAAAAQMIwGK6sD8
29 sLCewmGvLrL/OJoolkVTw99GZPntQzzEywAAAJtiAAAAASWEAAAAAGUYhc+Alai96A
30
31
```

```
32 mB/4JlJkUjA VZRYFT8+r8keoTBoeFAAAAgE=xiIdFUZzOXgTYpIMwaXYfaI1
33 xLu/TZnGmMfRAAAABYQAAAC4UBYY9IKKriHhUtu/AObJQITTHZpHalNYcnZH
34 w86WVQAAACBhRLIQBBHvsy9zJDFGGRkRfVg4ENySwpGrW5MfmxwAAACEDSley
35 /Zf8FDgDhntFecjMalpwTo63dyAI58aqYxwwAAAAMDDCCBYDCUdHn+zvALED
36 ZqdqWHhu8uAm3W5DlQYwZa197Lc0J16CqcvGXW7rDHQCDUSBhF5</ds:SignatureValue>
37 <ds:KeyInfo>
38 <ds:X509Data>
39 <ds:X509Certificate>MIIBAjCCASACjG16NzZmMjM3LWQzNDYhNDY0OQYyY4YWQ6YTYzNDk5
40 NTAJ
41 BgcqhskQPBROAwPZELMAGAtUERBhMCQVQEDAOBgNVBAoTBhRVIEqYyXoxDTAL
42 BgNVBAsTBREBSUszDzANBgNVBAMTBTBhbkZlc1N0TAAeFwoxMAZAMTIsNDM1MThtafwox
43 NTAAITRNDAAVtthAMDBKCAAJBgNVBAYAUFMRAwIgdjDjVQKewdUySfHemfM/Qww
44 CwYDQQLERjQUiLMDwDQzDVQ0DEwZlc3N0UkewSTA TBgcqhskQPBgqgqKqO
45 PQMBAQMYAAS7onIew78Y7x/QHorA51v1fIA19AZaHk4gbaekKw6LAmDrzqf
46 pDKBZ+rsD3AwCwYHkoZJpCAQUAAZAMDDCCGHN76EpbcCvswAzdirKRoYBKij
47 7WifWAIYJ2ahexym1U:48X529no0q8fDuXNzoD</ds:X509Certificate>
48 </ds:X509Data>
49 </ds:KeyInfo>
50 <ds:Object Id="Object-c7b5c8fc-2">
51 <xades:QualifyingProperties
52 xmlns:xades="http://uri.etsi.org/01903/v1.3.2#" Target="#signature-1-1">
53 <xades:SignedProperties
54 Id="SignedProperties-c7b5c8fc-1">
55 <xades:SignedSignatureProperties>
56 <xades:SigningTime>2014-06-12T16:05:48Z</xades:SigningTime>
57 <xades:SigningCertificate>
58 <xades:Cert>
59 <xades:Certificate
60 <xades:CertDigest>
61 <xades:DigestMethod Algorithm="http://www.w3.org/2001/04/xmlenc#sha256"/>
62 <xades:DigestValue>Nrxp18LsVIEZqSGoXoDjx/3/0SP98OgmFuWuJS=</xades:DigestVal
63 ue>
64 </ds:X509IssuerName>CN=dssSRA,O=ITU,Graz,C=AT</ds:X509IssuerName>
65 </ds:X509SerialNumber>1907779826648900468002611046824934450227935370979681199526
66 5519399396929293707158042933</ds:X509SerialNumber>
67 </xades:IssuerSerial>
68 </xades:Cert>
69 </xades:Certificate>
70 </xades:SigningCertificate>
71 <xades:SignaturePolicyIdentifier>
72 <xades:SignaturePolicyImplied/>
73 </xades:SignaturePolicyIdentifier>
74 </xades:SignedSignatureProperties>
75 <xades:SignedDataObjectProperties>
76 <xades:DataObjectFormat
77 ObjectReference="#Reference-c7b5c8fc-1a">
78 <xades:MimeType>text/xml</xades:MimeType>
79 </xades:DataObjectFormat>
80 </xades:SignedDataObjectProperties>
81 </xades:MimeFormat>
82 </xades:SignedDataObjectProperties>
83 </xades:SignedSignatureProperties>
84 </ds:SignedSignatureProperties>
85 </ds:SignedSignatureProperties>
86 </ds:SignedSignatureProperties>
87 </xades:SignedDataObjectProperties>
```

```

88 </xades:SignedProperties>
89 <xades:UnsignedProperties
90   Id="UnsignedProperties-c7b5c8fc-1">
91   <xades:UnsignedSignatureProperties>
92     <xades:CertificateValues>
93       <xades:OtherCertificate
94         type="proxy"/>
95     </xades:CertificateValues>
96   </xades:UnsignedSignatureProperties>
97 </xades:SignedProperties>
98 <xades:UnsignedProperties-c7b5c8fc-1">
99   <xades:UnsignedSignatureProperties>
100     <xades:CertificateValues>
101       <xades:OtherCertificate
102         type="proxy"/>
103     </xades:CertificateValues>
104   </xades:UnsignedSignatureProperties>
105 </xades:SignedProperties>
106 </xades:UnsignedProperties>
107 </ds:Object>
108 </ds:Signature>
109 </ds:Signature>
110 </ds:Signature>
111 </ds:Signature>

```

Listing 9.1: Identity Link

9.2 Identity Link of the current eID Solution

This section contains a sample Identity Link (IDL) based on the current implementation of the Austrian eID.

```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <saml:Assertion xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion"
3   xmlns:dsig="http://www.w3.org/2000/09/xmldsig#"
4   xmlns:ecdsa="http://www.w3.org/2001/04/xmldsig-more#"
5   xmlns:pr="http://reference.e-government.gv.at/namespaces/persondata/
6     _-20020228#"
7   xmlns:si="http://www.w3.org/2001/XMLSchema-instance"
8   AssertionID="s2r-bmi.gv.at-AssertionID13815189324521476"
9   IssueInstant="2013-10-11T21:15:32+01:00"
10  Issuer="http://portal.bmi.gv.at/ref/s2r/issuer" MajorVersion="1"
11  MinorVersion="0">
12   <saml:AttributeStatement>
13     <saml:Subject

```

```

14 <saml:SubjectConfirmation>
15 <saml:ConfirmationMethod>urn:oasis:names:tc:SAML:1.0:
16 cm:sender-vouches</saml:ConfirmationMethod>
17 <saml:SubjectConfirmationData>
18 <pr:Person si:type="pr:PhysicalPersonType">
19 <pr:Identification>
20 <pr:Value>eAMBNedeafvhtPBrVw=</pr:Value>
21 <pr:Type>urn:publicid:gv.at:baseid</pr:Type>
22 </pr:Identification>
23 <pr:Name>
24 <pr:GivenName>Max</pr:GivenName>
25 <pr:FamilyName primary="undefined">Muster</pr:FamilyName>
26 </pr:Name>
27 <pr:DateOfBirth>1980-10-10</pr:DateOfBirth>
28 </pr:Person>
29 </saml:SubjectConfirmationData>
30 </saml:SubjectConfirmation>
31 </saml:Subject>
32 <saml:Attribute AttributeName="CitizenPublicKey"
33 AttributeNamespace="urn:publicid:gv.at:namespaces:identitylink:1.2">
34 <saml:AttributeValue>
35 <ecdsa:ECDSAPublicKey>
36 <ecdsa:DomainParameters>
37 <ecdsa:NamedCurve URN="urn:oid:1.2.840.10045.3.1.7" />
38 </ecdsa:DomainParameters>
39 <ecdsa:PublicKey>
40 <ecdsa:X Value="1...3" si:type="ecdsa:PrimeFieldElementType" />
41 <ecdsa:Y Value="5...1" si:type="ecdsa:PrimeFieldElementType" />
42 </ecdsa:PublicKey>
43 </ecdsa:ECDSAPublicKey>
44 </saml:AttributeValue>
45 </saml:Attribute>
46 <saml:Attribute AttributeName="CitizenPublicKey"
47 AttributeNamespace="urn:publicid:gv.at:namespaces:identitylink:1.2">
48 <saml:AttributeValue>
49 <dsig:RSAPublicKey>
50 <dsig:Modulus>214 + 3.....nmb</dsig:Modulus>
51 <dsig:Exponent>AQAB</dsig:Exponent>
52 </dsig:RSAPublicKey>
53 </saml:AttributeValue>
54 </saml:Attribute>
55 </saml:AttributeStatement>
56 </dsig:Signature>
57 <dsig:SignedInfo>
58 <dsig:CanonicalizationMethod
59 Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
60 <dsig:SignatureMethod
61 Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1" />
62 <dsig:Reference URI="">
63 <dsig:Transforms>
64 <dsig:Transform
65 Algorithm="http://www.w3.org/TR/1999/REC-xpath-19991116">
66 <dsig:XPath>not(ancestor-or-self::pr:Identification)
67 </dsig:XPath>
68 </dsig:Transform>
69 </dsig:Transform>

```


9.2 Identity Link of the current eID Solution

```

70 Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature" />
71 </dsig:Transforms>
72 <dsig:DigestMethod
73 Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />
74 <dsig:DigestValue>0i6odc....APA=</dsig:DigestValue>
75 </dsig:Reference>
76 <dsig:Reference Type="http://www.w3.org/2000/09/xmldsig#Manifest"
77 URI="#manifest">
78 <dsig:DigestMethod
79 Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />
80 <dsig:DigestValue>0l644yvXuqKYKhGtN/8CdOoyC4=
81 </dsig:DigestValue>
82 </dsig:Reference>
83 </dsig:SignedInfo>
84 <dsig:SignatureValue>U3edg.....oA=</dsig:SignatureValue>
85 <dsig:KeyInfo>
86 <dsig:X509Data>
87 <dsig:X509Certificate>MIIF.....ocIBzYp4=</dsig:X509Certificate>
88 </dsig:X509Data>
89 </dsig:KeyInfo>
90 <dsig:Object>
91 <dsig:Manifest Id="manifest">
92 <dsig:Reference URI="">
93 </dsig:Reference>
94 </dsig:Transform>
95 Algorithm="http://www.w3.org/TR/1999/REC-xpath-19991116">
96 <dsig:XPath>not(ancestor-or-self::dsig:Signature
97 </dsig:XPath>
98 </dsig:Transform>
99 </dsig:Transforms>
100 <dsig:DigestMethod
101 Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />
102 <dsig:DigestValue>X9LBF...0J4E=</dsig:DigestValue>
103 </dsig:Reference>
104 </dsig:Manifest>
105 </dsig:Object>
106 </dsig:Signature>
107 </saml:Assertion>

```

Listing 9.2: Identity Link

Bibliography

- Amazon. *AWS CloudHSM*. URL: <http://aws.amazon.com/de/cloudhsm/details/> (cit. on p. 78).
- Apache. *Apache Web Server*. URL: <http://wiki.apache.org/tomcat/FAQ/Connectors#Q3> (cit. on p. 68).
- Ateniese, Giuseppe et al. (2005). "Sanitizable signatures." In: *Computer Security—ESORICS 2005*. Springer, pp. 159–177 (cit. on p. 13).
- Ateniese, Giuseppe et al. (Feb. 2006). "Improved Proxy Re-encryption Schemes with Applications to Secure Distributed Storage." In: *ACM Trans. Inf. Syst. Secur.* 9.1, pp. 1–30. ISSN: 1094-9224. DOI: 10.1145/1127345.1127346. URL: <http://doi.acm.org/10.1145/1127345.1127346> (cit. on p. 42).
- Bertoni, G. et al. (2011). *The Keccak reference*. Submission to NIST (Round 3). URL: <http://keccak.noekeon.org/Keccak-reference-3.0.pdf> (cit. on p. 4).
- Cooper, D. et al. (May 2008). *Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile*. RFC 5280 (Proposed Standard). Updated by RFC 6818. Internet Engineering Task Force. URL: <http://www.ietf.org/rfc/rfc5280.txt> (cit. on p. 8).
- Cruellas, Juan Carlos et al. (Feb. 2003). *XML Advanced Electronic Signatures (XAdES)*. W3C Note 20 February 2003. <http://www.w3.org/TR/2003/NOTE-XAdES-20030220>. W3C (cit. on p. 60).
- De Caro, Angelo and Vincenzo Iovino (2011). "jPBC: Java pairing based cryptography." In: *Proceedings of the 16th IEEE Symposium on Computers and Communications, ISCC 2011*. Kerkyra, Corfu, Greece: IEEE, pp. 850–855. URL: <http://gas.dia.unisa.it/projects/jpbc/> (cit. on p. 43).
- Derler, David (2013). *On the Optimization of two Recent Proxy-Type Digital Signature Schemes and their Efficient Implementation in Java* (cit. on p. 37).

Bibliography

- Fillingham, David (1997). *A Comparison of Digital and Handwritten Signatures*. URL: <http://groups.csail.mit.edu/mac/classes/6.805/student-papers/fall97-papers/fillingham-sig.html> (cit. on p. 3).
- Fitzek, Andreas Gregor (2014). *Development of an ARM TrustZone aware operating system ANDIX OS* (cit. on p. 83).
- Hanser, Christian and Daniel Slamanig (2013). "Blank Digital Signatures." In: *8th ACM Symposium on Information, Computer and Communications Security (ASIACCS 2013), Hangzhou, China, 7-10 May 2013*. Note: This is the more actual version which is available as *Cryptology ePrint Archive Report 2013/130*. ACM, pp. 95–106 (cit. on pp. 37, 39).
- help.gv.at (Apr. 2014). *help.gv.at*. URL: <https://help.gv.at/Portal.Node/hlpd/public/content/impressum/Seite.731300.html> (cit. on p. 19).
- Hollosi, Arno and Gregor Karlinger (Feb. 2005). *XML Definition der Personenbindung*. URL: www.buergerkarte.at/konzept/personenbindung/spezifikation/20050214/Personenbindung-20050214.pdf (cit. on p. 26).
- Hollosi, Arno and Gregor Karlinger (Jan. 2014). *Die österreichische Bürgerkarte*. URL: <http://www.buergerkarte.at/konzept/securitylayer/spezifikation/20140114/> (cit. on pp. 28, 62).
- Jelastic. *Jelastic Pricing*. URL: <http://www.dogado.de/cloud-services/jelastic-cloud-hosting/preisuebersicht.html> (cit. on p. 72).
- Jelastic (Aug. 2014). *Jelastic Pricing*. URL: <http://jelastic.com/features/pricing/> (cit. on p. 72).
- Johnson, Robert et al. (2002). "Homomorphic Signature Schemes." English. In: *Topics in Cryptology — CT-RSA 2002*. Ed. by Bart Preneel. Vol. 2271. Lecture Notes in Computer Science. Springer Berlin Heidelberg, pp. 244–262. ISBN: 978-3-540-43224-1. DOI: 10.1007/3-540-45760-7_17. URL: http://dx.doi.org/10.1007/3-540-45760-7_17 (cit. on p. 11).
- Krawczyk, Hugo and Tal Rabin (1998). *Chameleon Hashing and Signatures*. Cryptology ePrint Archive, Report 1998/010. <http://eprint.iacr.org/> (cit. on p. 15).
- Mell, Peter and Timothy Grance (Sept. 2011). *The NIST Definition of Cloud Computing*. Tech. rep. 800-145. Gaithersburg, MD: National Institute of Standards and Technology (NIST). URL: <http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf> (cit. on p. 70).

- Miyazaki, Kunihiro et al. (2003). "Digital Documents Sanitizing Problem." In: *IEIC Technical Report (Institute of Electronics, Information and Communication Engineers)*, pp. 61–67 (cit. on p. 9).
- Myers, M. et al. (June 1999). *X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP*. RFC 2560 (Proposed Standard). Obsoleted by RFC 6960, updated by RFC 6277. Internet Engineering Task Force. URL: <http://www.ietf.org/rfc/rfc2560.txt> (cit. on p. 8).
- Nuñez, David (Jan. 2013). *NICS Crypto Library*. URL: <https://www.nics.uma.es/dnunez/nics-crypto> (cit. on pp. 43, 63).
- OASIS Security Services (Nov. 2002). *Security Assertion Markup Language (SAML) v1.0*. URL: <https://www.oasis-open.org/standards> (cit. on p. 30).
- Republik Oesterreich (2010). *Bundesgesetz über elektronische Signaturen (Signaturgesetz - SigG)* (cit. on pp. 24, 25, 62).
- Republik Oesterreich (2014). *E-Government-Bereichsabgrenzungsverordnung - E-Gov-BerAbgrV* (cit. on p. 49).
- Rivest, R. L., A. Shamir, and L. Adleman (Feb. 1978). "A Method for Obtaining Digital Signatures and Public-key Cryptosystems." In: *Commun. ACM* 21.2, pp. 120–126. ISSN: 0001-0782. DOI: 10.1145/359340.359342. URL: <http://doi.acm.org/10.1145/359340.359342> (cit. on pp. 5–7).
- Slamanig, Daniel and Stefan Rass (2010). "Generalizations and Extensions of Redactable Signatures with Applications to Electronic Healthcare." English. In: *Communications and Multimedia Security*. Ed. by Bart De Decker and Ingrid Schaumüller-Bichl. Vol. 6109. Lecture Notes in Computer Science. Springer Berlin Heidelberg, pp. 201–213. ISBN: 978-3-642-13240-7. DOI: 10.1007/978-3-642-13241-4_19. URL: http://dx.doi.org/10.1007/978-3-642-13241-4_19 (cit. on p. 18).
- Slamanig, Daniel and Stefan Rass (2011). "Redigierbare Digitale Signaturen: Theorie und Praxis." In: *Datenschutz und Datensicherheit* 35, pp. 757–762 (cit. on pp. 12, 13).
- Slamanig, Daniel, Klaus Stranacher, and Bernd Zwattendorfer (2014). "User-Centric Identity as a Service-Architecture for eIDs with Selective Attribute Disclosure." In: *19th ACM Symposium on Access Control Models and Technologies (SACMAT 2014)*. ACM, pp. 153–163 (cit. on pp. 35, 48, 90).
- Steinfeld, Ron, Laurence Bull, and Yuliang Zheng (2002). "Content Extraction Signatures." English. In: *Information Security and Cryptology — ICISC*

Bibliography

2001. Ed. by Kwangjo Kim. Vol. 2288. Lecture Notes in Computer Science. Springer Berlin Heidelberg, pp. 285–304. ISBN: 978-3-540-43319-4. DOI: 10.1007/3-540-45861-1_22. URL: http://dx.doi.org/10.1007/3-540-45861-1_22 (cit. on pp. 11, 12).
- Stranacher, Klaus et al. (2013). “The Austrian Identity Ecosystem – An e-Government Experience book title: Architectures and Protocols for Secure Information Technology.” In: *Advances in Information Security, Privacy, and Ethics (AISPE)*. Antonio Ruiz Martínez, Rafael Marin-Lopez, Fernando Pereniguez-Garcia, pp. 288 –309 (cit. on pp. 19, 29).
- The Council of the European Union (2000). *Directive 1999/93/EC the European Parliament and of the Council of 13 December 1999 on a Community framework for electronic signatures* (cit. on pp. 24, 26, 60).
- Yuan, Michael (Dec. 2011). *A Java Developer’s Guide to PaaS*. URL: http://www.infoq.com/articles/paas_comparison (cit. on p. 71).
- Zwattendorfer, Bernd, Thomas Zefferer, and Klaus Stranacher (2014). “An Overview of Cloud Identity Management-Models.” In: *Proceedings of the 10th International Conference on Web Information Systems and Technologies*. Ed. by INSTICC, pp. 82 –92 (cit. on p. 20).