Hardware-Software-Codesign of Side-Channel Evaluated Identity-based Encryption

Thomas Unterluggauer
t.unterluggauer@student.tugraz.at

Institute for Applied Information
Processing and Communications (IAIK)
Graz University of Technology
Inffeldgasse 16a
8010 Graz, Austria

Master Thesis

Supervisor: Dipl.-Ing. Erich Wenger
Assessor: Ao.Univ.-Prof. Dipl.-Ing. Dr.techn. Karl-Christian Posch

October, 2013

I hereby certify that the work presented in this thesis is my own work and that to the best of my knowledge it is original except where indicated by reference to other authors.

Ich bestätige hiermit, diese Arbeit selbständig verfasst zu haben. Teile der Diplomarbeit, die auf Arbeiten anderer Autoren beruhen, sind durch Angabe der entsprechenden Referenz gekennzeichnet.

Thomas Unterluggauer

Acknowledgements

I would like to thank my supervisor Erich Wenger for his support and input to successfully accomplish this thesis. Further, I would like to thank Michael Hutter and Thomas Korak for their guidance with respect to performing a side-channel attack practically. Besides, both my parents and friends were a huge support during the challenges involved with this thesis.

Abstract

Providing sufficient security to embedded applications has become increasingly important. The schemes being used rely on the presence of a key for encryption. Contrary to this approach, the promising concept of identity-based encryption (IBE) enables the encryption of data by just knowing the recipient's identity, avoiding the key exchange problem. The goal of this thesis is to provide embedded platforms with identity-based encryption. Besides computational speed and low resource requirements, security against side-channel attacks is considered important.

The presented platform is based on a clone of the ARM Cortex-M0+. It is capable of performing the Boneh-Boyen IBE Key Encapsulation Mechanism for both the 80-bit and 128-bit security level. In this respect, encapsulation is shown to be more expensive than decapsulation. Using a 10 MHz clock, encapsulation is done depending on the security level in 3.7 and 12.1 seconds, while decapsulation performs in 2.4 sec and 6.6 seconds. In an 130 nm UMC process, the respective platforms are as small as 46,000 and 53,800 gate equivalents. Compared to the platforms based on the MSP430 and the ATMega processors, runtime is reduced massively and memory requirements drop by at least 50%. Other dedicated hardware platforms may be faster, but are at the minimum three times larger and have the disadvantage that they can invariably execute their intended function. To speed up the presented platform, two variants of a multiply-accumulate instruction-set extension are introduced. These decrease the runtime and the demand for energy by up to 60%. Another benefit of the platform is, that it is shown to be secure against various types of side-channel attacks, including timing attacks. A differential power analysis attack is performed on the underlying pairing computation and, based on the observations made, an effective countermeasure is deduced.

Evaluation reveals the feasibility of identity-based encryption in embedded environments and clearly suggests that the proposed platform—due to its low memory requirements, its general-purpose design, its adaptability and its side-channel security—is able to fill the momentary gap between existing microcontroller-based platforms and dedicated hardware platforms.

Keywords: Identity-based encryption, Cortex-M0+, ARM, embedded system, optimal Ate pairing, side-channel attack, differential power analysis, hardware-software codesign

Kurzfassung

Die Gewährleistung von einem genügendem Maß an Sicherheit in eingebetteten Systemen ist zusehends wichtig geworden. Die dafür verwendeten Mechanismen basieren auf dem Vorhandensein eines Schlüssels für die Verschlüsselung. Im Gegensatz zu diesem Ansatz, ermöglicht das vielversprechende Konzept der *Identity-Based Encryption* (IBE) das Senden verschlüsselter Daten basierend auf der Identität des Empfängers. Dies vermeidet den Austausch des zu verwendenden Schlüssels. Das Ziel dieser Masterarbeit ist die Ausstattung von eingebetteten Plattformen mit Identity-Based Encryption. Neben der Ausführungsgeschwindigkeit und niedrigen Ressourcenanforderungen war die Seitenkanalsicherheit ein wichtiger Aspekt.

Die hier präsentierte Plattform basiert auf einem Nachbau des ARM Cortex-M0+. Sie ermöglicht die Verwendung des Boneh-Boyen IBE *Key Encapsulation Mechanism* und bietet dabei wahlweise 80 bit oder 128 bit Sicherheit. In diesem Zusammenhang zeigt sich, dass *Key Encapsulation* rechenintensiver ist als *Key Decapsulation*. Unter Verwendung eines 10 MHz Taktgebers benötigt die *Encapsulation* Routine abhängig vom Sicherheitslevel 3,7 und 12,1 Sekunden, während die *Decapsulation* Routine 2,4 und 6,6 Sekunden beansprucht. In einem 130 nm UMC Prozess benötigen die jeweiligen Plattformen lediglich 46.000 bzw. 53.800 Gatter. Verglichen mit ähnlichen Plattformen basierend auf den MSP430 und ATMega Prozessoren reduziert sich die Laufzeit wesentlich und die Speicheranforderungen vermindern sich um mindestens 50%. Andere, dedizierte Hardware Plattformen sind zwar schneller, aber auch mindestens drei mal so groß und besitzen den Nachteil, ausnahmslos die ihr zugedachte Funktion erfüllen zu können. Um die hier präsentierte Plattform zu beschleunigen, werden zwei unterschiedliche Varianten einer Multiplizier-Akkumulier Instruktionssatzerweiterung eingeführt. Diese reduzieren sowohl die Laufzeit als auch den Energiebedarf um bis zu 60%. Ein weiterer Vorteil der Plattform ist, dass sie resistent gegenüber verschiedensten Varianten von Seitenkanalattacken, darunter Timing Attacken, ist. Eine *Differential Power Analysis* Attacke wird an der Berechnung der zugrundeliegenden bilinearen Abbildung durchgeführt und daraus eine effektive Gegenmaßnahme abgeleitet.

Die Evaluierung attestiert die Machbarkeit von Identity-Based Encryption in eingebetteten Systemen und legt nahe, dass die vorgeschlagene Plattform aufgrund ihrer niedrigen Speicheranforderungen, ihrer vielfältigen Einsetzbarkeit, ihrer Adaptierbarkeit, und ihrer Seitenkanalresistenz, in der Lage ist, die gegenwärtige Lücke zwischen existierenden Microcontroller-basierten Plattformen und den dedizierten Hardware Plattformen zu schließen.

Stichwörter: Identity-based Encryption, Cortex-M0+, ARM, eingebettetes System, optimal Ate pairing, Seitenkanalattacke, Differential Power Analysis, Hardware-Software Codesign

Contents

1. Introduction

Year after year tiny microchips pervade our lives even more. Sometimes as little as a grain of sand, they can be found in numerous embedded applications. These range from wireless sensor networks, automotive parts, microcontroller-based automation, to the internet of things. Yet, there are many more of these applications to come. However, acceptance of embedded applications heavily depends on the possibility to preserve people's privacy and to ensure confidentiality of the processed data. Consequently, security in these applications becomes a very important aspect, but security has its cost and it is a demanding task to provide the desired level of security in these environments as lack of resources is a chronic condition: if user interaction is involved, speed is an important criteria as people do not accept long waiting times. On the other hand, a common requirement is to have a small design in order to keep costs in mass production low. Further, the microchips involved are typically powered by battery or passively through electromagnetic fields, leading to a demand for designs that are energy-efficient in the first, and power-efficient in the latter case.

Nowadays, there are two common approaches to keep transmitted data confidential: symmetric cryptography and asymmetric cryptography. Symmetric cryptographic schemes are fast and allow small designs. On the other hand, all parties involved in the communication need to share the same key that is used for encryption and decryption. This leads to the problem of key distribution, which becomes increasingly hard if many small and widespread devices are involved as it is the case for the aforementioned applications. Asymmetric schemes do not have this problem: a publicly known key is used for encryption and decryption is then done using a private key that is linked in some way to the public key. Contrary to symmetric schemes, asymmetric cryptography is slower and more complex. In addition, we cannot be sure about the authenticity of the public key of the communication partner. As a consequence, very complex constructs, namely public key infrastructures, have been introduced. In these structures, trusted authorities certify the authenticity of public keys in a recursive fashion.

Another concept that was proposed by Shamir [Sha84] is identity-based encryption. This asymmetric scheme works without a public key. Instead, the publicly known identity string of the communication partner is used, which solves the problem of authenticity of the public key. The main drawbacks of these schemes are its performance and memory requirements due to the complex elliptic curve and pairing computations involved. Nevertheless, it avoids recursive validation of public keys and seems a promising concept for embedded environments too.

Performance of elliptic curve operations and pairing computation is crucial for implementing identity-based encryption. High-speed software implementations for computing pairings have been presented by Beuchat et al. [Beu+10], Grewal et al. [Gre+13] and Sánchez and Rodríguez-Henríquez [SR13], but are targeted at high-end application processors. On the other hand, the results by Gouvêa, Oliveira, and López [GOL12a], Gouvêa and López [GL09], and Szczechowiak et al. [Szc+08] constitute software implementations of pairings for embedded environments, which either lack sufficient performance or have an

tremendous demand for memory. Moreover, dedicated hardware platforms were created by Kammler et al. [Kam+09] and Fan, Vercauteren, and Verbauwhede [FVV09], which have sizes not necessarily suitable for use in embedded applications. Unfortunately, the various implementation mentioned do not cover security against side-channel attacks, whose possibility was emphasized by Whelan and Scott [WS06]. A practical attack, among others, was shown to be feasible by Ghosh and Roychowdhury [GR11].

In this thesis, a compact, but yet performant platform capable of performing identity-based encryption is presented. The platform is based on a clone of the ARM Cortex-M0+ microprocessor, namely the Xetroc-M0+ by Unterluggauer [Unt13]. This microprocessor-based platform can beneficially be used for any further applications that need to be supported as well. Both the pure software implementation and the overall hardware platform are aimed to be competitive. In the process of hardware-software codesign, two variants of an instruction-set extension to perform multiply-accumulate operations are created, which aid to significantly improve the efficiency of the platform. To underline the significance of the attained results, a comprehensive evaluation with related implementations on hardware and both high-end and low-end microprocessors is done.

Another advantage of the presented platform is, that security against various types of side-channel attacks, for example timing attacks, is shown. In this context, a differential power analysis attack on the Decapsulation routine of the key encapsulation mechanism variant of the BB_1 identity-based encryption scheme by Boyen [Boy06] is carried out. In the scheme's plain implementation, the underlying pairing computation leaks the identity's private key that allows decryption of any ciphertext that is intended for the respective identity. Resulting from this observation, a simple countermeasure to prevent this type of attack is presented.

The thesis is organized as follows. Chapter 2 explains the concept of identity-based encryption. The mathematical background of this work is covered in Chapter 3. Based on this, concrete identity-based encryption schemes are evaluated in Chapter 4. Once a suitable identity-based encryption scheme is chosen, Chapter 5 points out the threats resulting from side-channel attacks and gives an overview on the most relevant ones. Consequently, Chapter 6 details the proposed architecture along with important aspects of both implementation and optimization. A side-channel analysis of the identity-based encryption scheme as well as an side-channel attack is performed in Chapter 7. A comprehensive evaluation is part of Chapter 8. Finally, this thesis is concluded in Chapter 9.

2. Identity-Based Encryption

2.1. Introduction

To this day, two basic approaches to preserve confidentiality of data have emerged: *symmetric* and *asymmetric encryption schemes*. These concepts are used widely, but also suffer from fundamental problems. An alternative concept that is ought to solve the problems involved with conventional cryptography is the so-called Identity-Based Encryption (IBE). In this chapter, conventional symmetric and asymmetric cryptography are reviewed and their problems highlighted in Sections 2.2 and Section 2.3 respectively. Following this, the concept of identity-based encryption is introduced in Section 2.4 and its advantages and disadvantages are discussed in Section 2.5.

2.2. Symmetric Cryptography

Symmetric encryption schemes, such as block ciphers, use a common secret key that is utilized for both encryption and decryption. Figure 2.1 illustrates the concept: both parties, Alice and Bob, have the same key K that is used to perform en- and decryption. Bob encrypts his message M intended for Alice by using the common key K and sends the resulting ciphertext C to Alice. She unveils the original message M by invoking the `Decrypt` method using both the received ciphertext C and the common secret K.

This is a very simple and efficient concept. Typically, block ciphers such as AES, DES and 3DES, but also stream ciphers such as Trivium are used. These ciphers are generally fast and compact. However, the problem that lies within this concept is the establishment of a common secret between the communication partners. For this purpose, *key agreement* mechanisms can be used: each of the involved parties influence the shared secret that is established during the agreement process. Although this is done via an insecure channel, no third party is able to obtain the resulting key. A concrete instance of this concept is the Diffie-Hellman key exchange. [DH76] A significant problem remains though: origin authenticity must be ensured by using an authentic channel in order to avoid man-in-the-middle attacks.

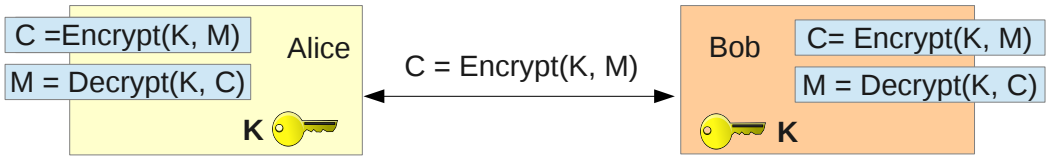

Figure 2.1.: Basic concept of symmetric cryptography: Alice and Bob share a common secret.

Figure 2.2.: Basic concept of asymmetric cryptography: Alice and Bob each have a key pair of their own. Bob uses Alice's public key for encryption, who in turn can decrypt using her private key.

An alternative approach to establishing a common secret is *key distribution*: one of the parties generates a secret and securely distributes it to its desired communication partners. Therefore, a secure channel is needed, which can be established by using asymmetric cryptography.

2.3. Asymmetric Cryptography

In asymmetric cryptography, shown in Figure 2.2, each party has a key pair consisting of a private and a public key. For example, Bob possesses the key pair consisting of $K_{private,Bob}$ and $K_{public,Bob}$. These two keys are related in a way, that if a public key is used to encrypt a message M, only its corresponding private key is able to decrypt the ciphertext C. It is generally hard to calculate the private key from its public key without additional knowledge.

Asymmetric cryptography can be used to securely distribute a shared secret intended for symmetric encryption, that is, the generated common secret is encrypted using the other party's public key. Consequently, no one but the intended recipient is able to unveil the common secret using their private key.

In addition, asymmetric schemes support the creation of signatures. For this purpose, one's private key is used to sign a message. This is equivalent to decryption in the original setting. The resulting signature can then be verified by anyone using the message, the signature, and the appropriate public key. This is, up to comparisons in the verification step, identical to encryption in the original setting. The resulting signatures are authentic as the private key is only known by its creator. This is shown in Figure 2.3: Bob signs the message M using his private key $K_{private,Bob}$ and sends the resulting signature S together with the message M to Alice. She invokes the `Verify` method using the message M, the signature S and Bob's public key $K_{public,Bob}$. The method's result v indicates the validity of the signature. Typically, the signature is calculated from a hash of the message rather than the message itself. The concept of signatures can be used to obtain authenticity in the aforementioned key agreement process.

Asymmetric cryptography also has its drawbacks: a public key used for encryption or verification may be forged, that is, authenticity of the public keys is not assured. The conventional approach to deal with this problem is to certify authenticity of public keys by means of the same public key mechanisms. This concept of *public key infrastructures* is illustrated by Figure 2.4. Certificate Authorities (CA) certify the authenticity of a public key, that is, an identity along with its public key are signed by the authority. This is done

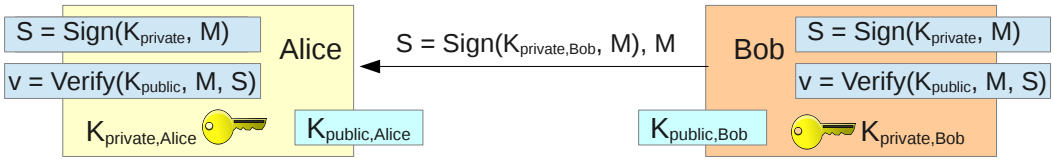

Figure 2.3.: Using asymmetric cryptography for signatures: Alice and Bob each have a key pair of their own. Bob uses his private key for signing a message. Alice can verify the authenticity of the message using the signature and Bob's public key.

in a recursive fashion. The uppermost CA, also called the root CA, certifies itself.

This common procedure implies delegation of trust to the root CA. Authentic keys for root CAs are usually shipped as systems are distributed. Otherwise, there is no reason to trust such a CA. To verify the authenticity of a public key, the whole chain of certificates needs to be validated. This, however, does not seem a satisfying solution. Ellison and Schneier [ES00] point out many good reasons for not relying on these public key infrastructures.

In case a key is compromised, revocation of public key certificates becomes necessary as these should not be used any further. In addition to an expiry date, certificate revocation lists and online services to check the status of a certificate exist to tackle this. Maintaining these lists and services is a huge effort. Considering all these problems, alternatives are of high interest.

2.4. Concept of Identity-Based Encryption

In 1984, Shamir [Sha84] proposed the concept of identity-based encryption with the intent to simplify certificate management in e-mail systems. In this asymmetric concept, the public key is replaced by a unique identity string. One can think of this identity string being an e-mail address, social insurance number, or similar. The advantage of such a concept is obvious: there is no public key that could be faked. Hence, there is no problem of authenticity and the blown up construct of public key infrastructures is avoided. On the other hand, decryption is still done using a private key only known to the intended recipient of a message.

The basic concept of identity-based encryption is illustrated in Figure 2.5. In the following, an exemplary sequence based on Figure 2.5 is shown. Involved are a trusted third party, Alice, and Bob.

1. The trusted third party starts by calling the Setup method. The public parameters P of the scheme are then published, while the master secret is kept hidden.

2. Bob wants to use the identity-based encryption scheme and obtains his private key D_{Bob} from the trusted third party by invoking the Derive method with the public parameters P and his name as arguments. Note that Bob needs his private key for decryption only. If Bob merely wanted to send encrypted data, he would not have to obtain his private key.

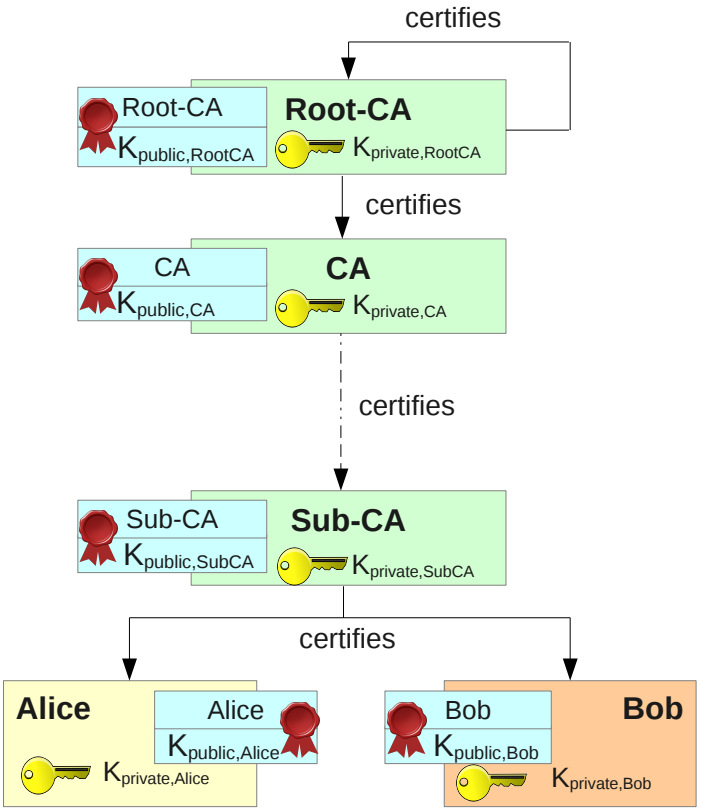

Figure 2.4.: Public key infrastructures: Certificate Authorities (CA) certify authenticity of lower-level parties' public keys. The uppermost CA certifies itself.

3. In the next step, Bob wants to send a secret message M to Alice. Therefore, he starts the Encrypt method with the public parameters P, Alice's name and the confidential data as arguments. The resulting ciphertext C is passed on to Alice.

4. Alice realizes that she received an encrypted message from Bob. Before she can read it, she needs to obtain her private key D_{Alice} via the Derive method from the trusted third party using the public parameters P and her name as arguments.

5. Now that Alice is in possession of her private key, she can reveal the original message from the ciphertext C using the Decrypt method.

From this example it can be seen, that there is a trusted third party that publishes public parameters that clearly define the instance of the identity-based encryption scheme. Further, it is in possession of a master secret. Its confidentiality is crucial as its leakage compromises the whole system. Identities obtain their private keys from that trusted third party, which in turn uses the master secret to derive the private keys for each identity. If a private key is leaked, it does not compromise the rest of the system. Encryption can be done by anyone just knowing the identity string. It is not even necessary that the recipient has already obtained their private key.

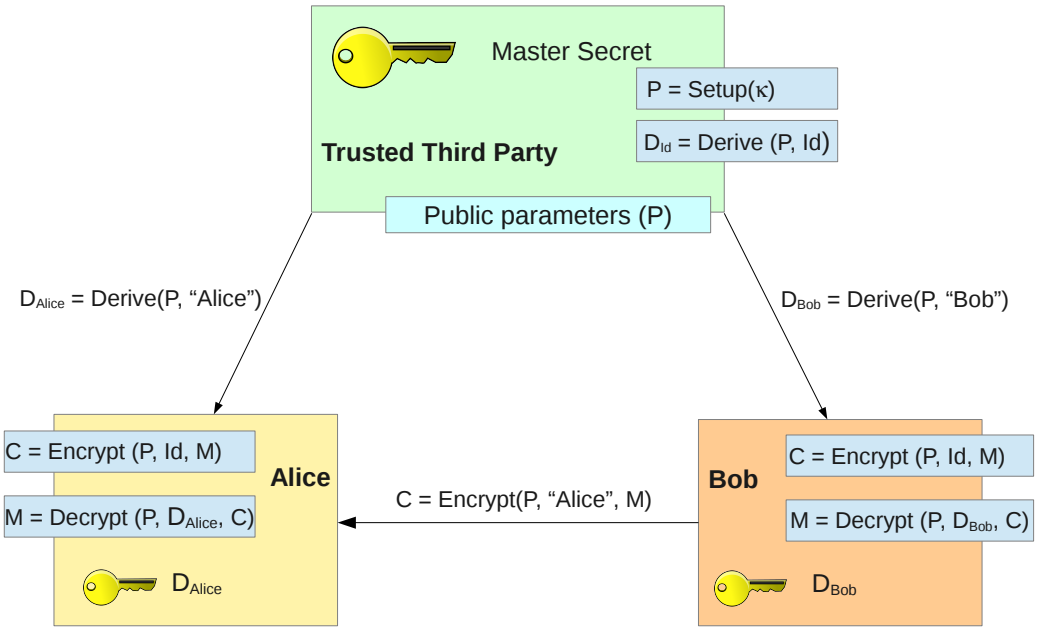

Figure 2.5.: Concept of identity-based encryption (IBE).

Following this, the four basic algorithms of any identity-based encryption scheme can be characterized. Let K_{master} denote the master key and P the public parameters. An identity's private key is labeled D_{id}.

Setup. By this algorithm, the identity-based encryption scheme is set up, that is, given a security parameter κ, the master secret K_{master} and the public parameters P along with the necessary algorithmic descriptions are generated.

$$(K_{master},\ P) = \mathsf{Setup}(\kappa)$$

Derive. Participants in the scheme can obtain their corresponding private keys using this algorithm. It takes the identity string id, the public parameters P and the master secret K_{master} as an argument and returns the private key D_{id}. This algorithm is typically executed at a trusted third party as it ought to be the only facility to keep the master secret.

$$D_{id} = \mathsf{Derive}(P,\ K_{master},\ id)$$

Encrypt. Encryption of messages is done using this algorithm. To compute the correct ciphertext C, the public parameters P and the identity string id of the communication partner are needed in addition to the message M.

$$C = \mathsf{Encrypt}(P,\ id,\ M)$$

Decrypt. This algorithm decrypts the ciphertext C to obtain the original message M. Therefore, the correct private key D_{id}, the public parameters P, and the ciphertext C are needed. If a ciphertext cannot be decrypted because it is invalid, the algorithm returns with a corresponding message.

$$M = \mathsf{Decrypt}(P,\, D_{id},\, C)$$

According to Gentry and Silverberg [GS02], identity-based encryption can be extended hierarchically, where private key derivation can be delegated to intermediate entities. However, complexity of Hierarchical Identity-Based Encryption (HIBE) schemes increases significantly with each additional layer.

2.5. Drawbacks of Identity-Based Encryption

Besides the compelling benefits, identity-based encryption also has its drawbacks as pointed out by Boneh and Franklin [BF01] and Gentry and Silverberg [GS02]. First, proposed schemes are based on bilinear maps or, as presented by Cocks [Coc01], the quadratic residuosity problem. Hence, they have worse performance than conventional asymmetric cryptography. Second, the public parameters need to be authentic and the identities' private keys need to be transmitted securely. This is a very similar problem to that of conventional asymmetric cryptography. Nevertheless, receiving one private key per person seems less effort than obtaining many authentic public keys. Moreover, the public parameters may be distributed as the applications are shipped. The third problem is the inherent key escrow. The trusted third party is able to derive any private key it wants, while in asymmetric cryptography the private key is typically generated on the user's computer. Correspondingly, non-repudiation cannot be assured. Finally, as a consequence of inherent key escrow, the trusted third party poses a security threat as it is an interesting target for attackers.

If a private key is leaked, the remaining system stays secure. Still, some kind of revocation should be available in this case. Identity-based encryption does not generally offer such a mechanism. However, expiry dates can be included in the public identity string: for a user called "Bob", the identity string for the current year could look like "Bob <currentyear>". Anyone who wants to transmit an encrypted message to Bob uses the identity string containing the current year instead. It forces the user to obtain a new private key every year. Hence, if a private key is compromised, the private key remains valid for the current year only. This can also be done for the current date, forcing the user to obtain a new private key every day. For example, if users leaves an organization, they will not receive any more private keys and will not be able to read e-mails past the date of their separation. Along with this comes the problem of having to transmit private keys on a daily basis via a separate secure channel. Ironically, conventional asymmetric cryptography, using a key pair of public and private key for the trusted third party, is a reasonable possibility since there are very likely many more users than trusted third parties.

The previously shown concept of concatenation can also be used for information different from expiry dates, such as permissions. As an additional form of revocation, compromised

identities along with a new identity string to be used could be published on some sort of revocation list.

2.6. Conclusion

Symmetric cryptographic schemes are fast and compact, but suffer from the key distribution problem. Nowadays, it is tackled using conventional asymmetric schemes, but there are some negative aspects about it that leave one with a bad taste. Therefore, the concept of identity-based encryption seems a promising alternative. Public keys are replaced by simple identity strings, avoiding the problem of key authenticity at all. Drawbacks are the secure distribution of the private keys and the public parameters as well as revocation. Apart from distributing these parameters once initially, this problem can, as mentioned before, be overcome with conventional asymmetric cryptography. Using this setup, there are mechanisms that support revocation, but still revocation is the real drawback of identity-based encryption.

3. Mathematical Background

Identity-based encryption schemes commonly make use of bilinear maps, or simply pairings. In this mathematical concept each an element from two groups is taken and then mapped to a third group. The first two of those groups are usually, but not necessarily, defined over elliptic curves. The target group that is being mapped to is usually an extension of a prime field. As one can see from this, it involves many different concepts that need to be brought in line with each other before it is possible to decide on an identity-based encryption scheme and to concretely perform an implementation.

For this purpose, the mathematical basics and concepts behind the work of this thesis are covered in this section. However, it does not pay regard to the concrete implementation aspects. These will be covered in a latter chapter. The concept of groups, fields, and extension fields is the heart of everything in this work. Therefore, Section 3.1 covers the their most important properties that are needed later on. Section 3.2 then introduces elliptic curves. Following, a simple and abstract definition of bilinear maps, as it is usually used for protocol descriptions, is done in Section 3.3. Combining the knowledge gained from the former sections, a concrete instance of a bilinear map based on groups over elliptic curves is presented in Section 3.4. As there are some restrictions to the elliptic curves being used for pairings, Section 3.5 gives a short overview on the methods to generate such curves. Moreover, it describes the type of curves being used in the context of this thesis in more detail. A conclusion is done in Section 3.6.

3.1. Groups, Rings and Fields

The mathematical basis of the whole work are the basic algebraic concepts of groups and fields. For that reason, the aspects relevant to the understanding of this work are reviewed in this part. The definitions in Section 3.1.1 to Section 3.1.3 largely follow Lidl and Niederreiter [LN86].

The well-known concept of multiplication and addition on integers can be generalized to operations on arbitrary sets. Given a set S, a mapping from all ordered pairs $(s,t) \in S \times S$ into S is called a *binary operation* on S. *Closure* of an operation means that the *image* of $(s,t) \in S \times S$ is again in S. An *algebraic structure* is a set S together with one ore more operations on S.

3.1.1. Groups

One of the simplest algebraic structures only having one operation are *groups*.

Definition 3.1. A group is a set G together with a binary operation $\star$ on G that has the following properties:

1. The operation $\star$ is *associative*, that is, for any $a, b, c \in G$,

$$(a \star b) \star c = a \star (b \star c).$$

2. There exists an *identity* element e, such that for any $a \in G$,

$$a \star e = e \star a = a.$$

3. Each element $a \in G$ has an *inverse* element $a^{-1} \in G$, such that

$$a^{-1} \star a = a \star a^{-1} = e.$$

A group that satisfies $a \star b = b \star a$ is called *commutative* or *abelian*. Additionally, $(a \star b)^{-1} = a^{-1} \star b^{-1}$ is valid for all $a, b \in G$. Definition 3.1 uses *multiplicative* notation. However, sometimes additive notation is preferred.

Multiplicative notation	Additive notation
$a^n = a \cdot a \cdot a ... \cdot a$ (n times)	$n \cdot a = a + a + a ... + a$ (n times)
$(a^n)^m = a^{nm}$	$m(na) = (nm)a$
$a^n \cdot a^m = a^{(n+m)}$	$na + ma = (n + m)a$
$a^{-n} = (a^{-1})^n$	$(-n)a = n(-a)$
$a^0 = e$	$0a = e$

Another term to be defined is *equivalence relation*, which is a subset R of $S \times S$ having the properties:

1. Reflexivity: $(s, s) \in R$, $\forall\ s \in S$.
2. Symmetry: $(s, t) \in R \Leftrightarrow (t, s) \in R$.
3. Transitivity: $(s, t), (t, u) \in R \Rightarrow (s, u) \in R$.

An equivalence relation R on a set S partitions the set S into *equivalence classes*. All elements equivalent to a fixed $s \in S$ are denoted by:

$$[s] = \{t \in S : (s, t) \in R\}.$$

Definition 3.2. Given arbitrary integers a, b and a positive integer n, a is said to be *congruent* to b modulo n if the difference $a - b$ is a multiple of n, that is $a = b + kn$, denoted by $a \equiv b \bmod n$.

Congruence modulo n is an equivalence relation on the set of integers $\mathbb{Z}$. The resulting set of equivalence classes is compatible with the original binary operation: it allows modular arithmetic and the equivalence relation is then called a congruence relation.

Definition 3.3. The set $\{[0], [1], ...[n - 1]\}$ of equivalence classes modulo n with the operation of addition $[a] + [b] = [a + b]$ forms a group $\mathbb{Z}_n$ called the *group of integers modulo n*.

Definition 3.4. A group containing finitely many elements is called *finite*. The number of elements in a finite group is called its *order*, denoted by $|G|$.

Definition 3.5. A subgroup of a group G is a subset H of G if it is itself a group with respect to the operation of G. *Trivial* subgroups are $\{e\}$ and G itself.

Definition 3.6. A multiplicative group G is said to be *cyclic* if there is an element $a \in G$ such that for any $b \in G \ \exists \ j \in \mathbb{Z} \ : \ b = a^j$.

Then a is called a *generator* of the cyclic group G. A short notation to define a group using a generator is $G = \langle a \rangle$. A cyclic group may have more than one generator. Cyclic groups are always commutative. For example, $\mathbb{Z}_n$ is an additive cyclic group of order n having the generator [1].

Definition 3.7. A subgroup of G generated by $\langle a \rangle$ consists of all powers of the element a of G. The subgroup is always cyclic. If $\langle a \rangle$ is finite, its order is also called the order of a. Otherwise, a is of *infinite order*.

The finite order k of a is the least positive integer such that $a^k = e$. In terms of subgroups, there is an important theorem on their orders that will be needed for elliptic curves:

Theorem 3.1 (Lagrange's Theorem)**.** *Let H be a subgroup of G. Then, the order of H divides the order of G. Consequently, the order of any element $g \in G$ divides the order of G.*

Definition 3.8. A *homomorphism* is a mapping $f : G \to H$ of the group G into the group H that preserves the operation of G, that is, if $\star$ and $*$ are the operations on G and H respectively, then $f(a \star b) = f(a) * f(b)$ must be true for all $a, b \in G$. A homomorphism of G into G is called an *endomorphism*. A one-to-one homomorphism of G onto H is called an *isomorphism*. In this case, G and H are called *isomorphic*. An isomorphism of G onto G is called *automorphism*.

Another property of a homomorphism $f : G \to H$ is $f(e)f(e) = f(e)$, that is, $f(e) = e'$, the identity element of H. Elements different from $e \in G$ may also map to the identity $e' \in H$, which is covered by the definition of the kernel. Additionally, it holds that $f(a^{-1}) = f(a)^{-1} \ \forall \ a \in G$.

Definition 3.9. The *kernel* of an homomorphism $f : H \to H$ is the set

$$ker(f) = \{a \in G : f(a) = e'\},$$

where e' denotes the identity in H.

3.1.2. Rings and Fields

Very often two distinct binary operations are defined for a specific set of elements, for example, addition and multiplication for integers and rational numbers. This leads to the definition of a *ring* structure.

Definition 3.10. A set R that has two arithmetic operations $+$ and $\cdot$ is said to be a *ring* $(R, +, \cdot)$ if it has the following properties:

1. Abelian group: R is an abelian group with respect to $+$.
2. Associativity: $a \cdot (b \cdot c) = (a \cdot b) \cdot c \ \ \forall \ \ a, b, c \in R$.
3. Distributivity of $+$ with respect to $\cdot$: $(a + b) \cdot c = ac + bc$ and $(b + c) \cdot a = ba + ca \ \ \forall \ a, b, c \in R$.

The identity element of the abelian group $(R, +)$ is denoted 0 (the *zero element*), the inverse of a by $-a$. A ring that has a multiplicative identity e satisfies $ae = ea = a \; \forall \; a \in R$. It is called *ring with identity* or *unitary ring*. If the operation $\cdot$ is commutative, the ring is also called *commutative*. If the elements $R \setminus \{0\}$ form a group with respect to $\cdot$, the ring is called a *division ring*. A *commutative division ring* is called a *field*. Summarizing, the definition of a field is as follows:

Definition 3.11. A set F with two binary operations $(F, +, \cdot)$ and two distinct elements $0 \neq e$, that is an abelian group with respect to $+$ having 0 as an identity element, that also forms an abelian group for the elements $F \setminus \{0\}$ with respect to $\cdot$ having the identity e, is called a *field* if the two operations $+$ and $\cdot$ are linked by the rules of the distributive law, that is, $(a + b) \cdot c = ac + bc$. 0 is called the *zero element* and e is called the multiplicative *identity* element.

The following definitions of *subsets* and *ideals* directly lead to the definition of prime fields that are important in the context of this work.

Definition 3.12. A *subring* of R is a subset S of R that is closed and forms a ring under its operations $+$ and $\cdot$.

Definition 3.13. An *ideal* is a subset J of a ring R that fulfills $ra \in J$, $ar \in J$ for any $a \in J$, $r \in R$.

Definition 3.14. For a given commutative ring R, the smallest ideal that contains a specific element $a \in R$ is denoted by $(a) = \{ra + na : r \in R, n \in \mathbb{Z}\}$ if R does not contain an identity element, and $(a) = \{ra : r \in R\}$ otherwise. Such an ideal is called *principal*. The principal ideal is *generated by a*.

For example, an element $n \in \mathbb{Z}$ generates the subset $J = (n) = \{.., -2n, -n, 0, n, 2n, ..\}$, which is a principal ideal of $\mathbb{Z}$.

An ideal J of a ring R partitions R into *residue classes* modulo J. The residue class of an element a of R modulo J is denoted $[a] = a + J$ as it is consists of all elements of the form $a = a + c$, where $c \in J$. Elements in the same residue class are called *congruent*. Two congruent elements a, b are denoted by $a \equiv b \bmod J$. The residue classes of the ring R modulo J form the the so-called *residue class ring* R/J under the following operations:

1. $(a + J) + (b + J) = (a + b) + J$
2. $(a + J) \cdot (b + J) = ab + J$

An example is the residue class ring $\mathbb{Z}/(n)$, which is spread widely in cryptography. If n is a prime, the the residue class ring becomes a field. The definitions of the diverse variants of *homomorphisms* in Definition 3.8 stay valid so far as both operations $+$ and $\cdot$ are preserved by the mapping. Such mappings can be used to transfer a structure from an algebraic system with a structure, for example a ring R, to a set S without a structure. Then, the resulting structure of S is induced by the mapping.

Definition 3.15. Given a prime p and a set of integers $\mathbb{F}_p = \{0, 1, 2, ..., p-1\}$, the mapping $\phi : \mathbb{Z}/(p) \to \mathbb{F}_p$, which is defined as $\phi([a]) = a$ for $a \in \{0, 1, 2, ..., p - 1\}$, induces a field structure on $\mathbb{F}_p$. $\mathbb{F}_p$ is a *Galois field of order p*.

Definition 3.16. The least positive integer n that fulfills $nr = 0$ for any element r of the ring R is called *characteristic*. If there is no such r, the *characteristic* is 0.

A finite field always has prime characteristic. Computing with elements in the finite field of characteristic p, denoted $\mathbb{F}_p$, is ordinary integer arithmetic modulo p. Its identity is 1, its zero element 0.

A *subfield* of a field F is a subset K of F that itself is a field under the operations of F. It is called a *proper subfield* if $K \neq F$. Conversely, F is an *extension (field)* of K.

Definition 3.17. A *prime field* is a field containing no proper subfields.

The finite field of characteristic p, $\mathbb{F}_p$, is such a prime field.

3.1.3. Extension Fields

A short review on polynomials needs to be done in order to define extension fields. A polynomial over the ring R is defined as

$$f(x) = \sum_{i=0}^{n} a_i x^i = a_n x^n + ... + a_1 x + a_0,$$

where n is a positive integer and the coefficients a_i are elements of R. x is called the *indeterminate* over R. Its degree $deg(f)$ is denoted by n. If all coefficients are zero, the polynomial is called the *zero polynomial*. If the identity of R is 1, a polynomial that has 1 as its leading coefficient is said to be *monic*. If all coefficients but a_0 are zero, it is a *constant polynomial*. Rules for calculating with polynomials is assumed to be known.

Definition 3.18. The polynomials over R form a ring with respect to addition and multiplication. This *ring of polynomials* is denoted by $R[x]$.

The ring of polynomials $R[x]$ inherits commutativity and the identity from the underlying ring R. In the following, R will be replaced by the prime field $\mathbb{F}_p$. A polynomial $g \in \mathbb{F}_p[x]$ divides a polynomial $f \in \mathbb{F}_p[x]$ if there is a polynomial $h \in \mathbb{F}_p[x]$ that fulfills $f = gh$. This leads to the definition of *irreducible polynomials*.

Definition 3.19. A polynomial $f \in \mathbb{F}_p[x]$ *irreducible over* $\mathbb{F}_p$ (or *irreducible in* $\mathbb{F}_p[x]$) is a polynomial of positive degree that only allows trivial factorizations.

In more detail, an irreducible polynomial f implies that $\forall b \,|\, f \Rightarrow b \in \mathbb{F}_p^* \vee b = f$. The residue class ring $\mathbb{F}_p[x]/(f)$ is a field iff $f \in \mathbb{F}_p[x]$ is irreducible over $\mathbb{F}_p$. This works analogously to the residue classes from Section 3.1.2. Two residue classes $[g] = g+(f)$ and $[h] = h+(f)$ are equivalent if $g \equiv h \bmod f$, that is, $g-h$ is divisible by f. Correspondingly, g and h leave the same remainder when dividing by f. This remainder $r \in \mathbb{F}_p[x]$ satisfies $deg(r) < deg(f)$ and is the unique representative of the respective residue class. These residue classes can be explicitly described by $r + (f)$ with r running through all elements in $\mathbb{F}_p[x]$ with $deg(r) < deg(f)$. For a polynomial f with $deg(f) = n \geq 0$, the number of elements in $\mathbb{F}_p[x]/(f)$ equals the number of polynomials in $\mathbb{F}_p[x]$ with degree smaller than n, that is, p^n. Replacing the *indeterminate* x in $f(x)$ by a fixed element from $\mathbb{F}_p$ evaluates to an element in $\mathbb{F}_p$.

Definition 3.20. A *root* (or a *zero*) $r \in \mathbb{F}_p$ of a polynomial $f \in \mathbb{F}_p[x]$ is an element that fulfills $f(r) = 0$. Then, $(x - r)$ divides f.

If a polynomial f has the same root k times, k denotes the roots *multiplicity*. Having the basics of polynomials in mind, extension fields can be defined.

Definition 3.21. Given a subfield K of the field F and any subset M of the field F, the intersection of all subfields of F containing both K and M is called *extension field* of K and is denoted $K(M)$. If M is a single element, the extension is said to be *simple*.

The smallest subfield of F containing both K and M is $K(M)$. Polynomials can be used to define such extensions.

Definition 3.22. Given a subfield K of F and an element $\theta \in F$, θ is said to be *algebraic* over K if it satisfies a nontrivial polynomial equation with coefficients in K, that is, $a_n\theta^n + ... + a_2\theta^2 + a_1\theta + a_0 = 0$. An extension L of K is said to be *algebraic* if every element in L is *algebraic* over K.

The set $J = \{f \in K[x] : f(\theta) = 0\}$ is an ideal of K if $\theta \in F$ is algebraic over K. A monic polynomial $g \in K[x]$ uniquely determines the set J such that it equals the principal ideal $J = (g)$. The polynomial g is irreducible over K. The degree of θ over K is defined as the degree of the irreducible polynomial g.

Theorem 3.2. *The irreducible polynomial g over K for an extension θ that is algebraic over K has the following properties:*

1. *For any $f \in K[x]$, $f(\theta) = 0$ iff g divides f.*
2. *The irreducible polynomial g is the monic polynomial in $K[x]$ of least degree having the root θ.*

Theorem 3.3. *Given an element $\theta \in F$ that is algebraic of degree n over K and g, the irreducible polynomial of θ over K, the following properties apply:*

1. *$K(\theta)$ is isomorphic to $K[x]/(g)$.*
2. *$\{1, \theta, ..., \theta^{n-1}\}$ is a basis of $K(\theta)$ over K.*
3. *Every element $\alpha \in K(\theta)$ is algebraic over K. The degree of α divides n.*

It follows, that, on the one hand, one can use an irreducible polynomial to define an extension field, and, on the other hand, a simple defining element may be used instead. Elements of the simple extension $K(\theta)$ can be represented by polynomials of the form $a_0 + a_1\theta + ... + a_{n-1}\theta^{n-1}$.

For the purpose of this thesis, a field $\mathbb{F}_p$ can be extended using an irreducible polynomial such as to be able to represent solutions to that polynomial equation. The k-th extension of the prime field $\mathbb{F}_p$ is denoted as $\mathbb{F}_{p^k}$. For illustration of *extension fields*, a nice example is the complex number system. It is well known that there is no solution to the equation $x^2 + 1 = 0 \in \mathbb{R}[x]$ in $\mathbb{R}$, that is, it has no roots in $\mathbb{R}$. In other words, the equation is irreducible in $\mathbb{R}[x]$. However, the monic and irreducible polynomial $h(x) = x^2 + 1$ creates a principal ideal $J = (h)$, which splits the ring of polynomials $\mathbb{R}[x]$ into residue classes of the form $a_0 + a_1x$, where x is nothing else than the imaginary part in $\mathbb{C}$. This results from performing a modulo operation on elements in $\mathbb{R}[x]$ using $h(x) = x^2 + 1$.

3.2. Elliptic Curves over Finite Fields

Another fundament of identity-based encryption are elliptic curves. The theory of elliptic curves makes extensive use of the concepts defined before, namely groups and (extension)

fields. It is an interesting field of study since groups formed by elliptic curves defined over some field can be used for different cryptographic applications, and most importantly, for bilinear maps. This introduction to elliptic curves is based in large parts on the publications by Costello [Cos13, Ch. 2] and Silverman [Sil09].

Definition 3.23. An elliptic curve E defined over a field K, denoted E/K, is an affine equation of the form

$$y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6, \tag{3.1}$$

where $a_1, a_2, ..., a_6$ are elements in K.

Equation 3.1 is the *general Weierstrass equation* for elliptic curves. For fields with characteristic not equal to 2 or 3, the equation can be simplified by substitution. This results in the *short Weierstrass equation*, which is defined as

$$y^2 = x^3 + ax + b, \tag{3.2}$$

and covers all isomorphism classes of elliptic curves for large prime fields $\mathbb{F}_p$. The values $a, b \in K$ need to be chosen appropriately in order to obtain a valid elliptic curve, that is, for the choice of $a, b \in K$, both derivatives of the curve $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}$ must not vanish for any point $P = (x_P, y_P)$. Then, the curve is said to be *smooth*, otherwise *singular*. A simple condition that the parameters a,b need to fulfill in order to obtain a valid curve is $4a^2 + 27b^2 \neq 0$ (*discriminant*). For illustrations and examples the interested reader may refer to Costello [Cos13, Chapter 2].

In addition to Equation 3.1 and Equation 3.2, other types of equations for elliptic curves exist, for example Edwards curves [Edw07] and Hessian curves [Sma01]. These sometimes have advantages in terms of computational speed, but their usage is, contrary to the general Weierstrass equation, restricted by some conditions.

3.2.1. Group Laws

Definition 3.24. The group elements of an elliptic curve E/K are the *points* (x, y) with coordinates in $\overline{K}$ (the algebraic closure of K), which satisfy the elliptic curve equation. In addition, the group consists of a *point at infinity*, denoted $\mathcal{O}$, which is the identity element. The operation defined on the group of elliptic curve points is point addition $+$. The group is denoted $E(\overline{K})$.

The operation of *point addition* mentioned in Definition 3.24 still needs to be specified. Two points $P = (x_P, y_P) \in K \times K$ and $Q = (x_Q, y_Q) \in K \times K$ are used to draw a line $\ell : y = \lambda x + \nu$ with gradient $\lambda = (y_Q - y_P)/(x_Q - x_P)$. The line ℓ intersects the curve E in three points. Two of the intersection points are known to be P and Q, and the third intersection point is defined to be $-R$. Drawing a vertical line v through the point $-R$ yields another point of intersection, namely $R = P + Q$. Insertion of the line equations ℓ, v into the curve equation E and comparison of coefficients leads to the specification of the point R as

$$x_R = \lambda^2 - x_P - x_Q \qquad y_R = -(\lambda x_R + \nu), \tag{3.3}$$

where $\nu = y_P - \lambda x_P$. Besides, addition of the identity $\mathcal{O}$ and additions of the form $P + (-P)$ are special cases that result from Section 3.1.

Analogously, *point doubling* is defined for the case $P = Q$: the tangent in the point P has gradient $\lambda = (3{x_P}^2 + a)/(2y_P)$ and is defined to be $\ell_t : y = \lambda x + \nu$. The unknown of the two intersections of ℓ_t and E is defined to be $-R$. Its inverse is the second point of intersection of E and the vertical line v through $-R$, namely $R = P + P = [2]P$. Insertion of the line equations ℓ_t, v in the curve equation E results in the the point R as

$$x_R = \lambda^2 - 2x_P \qquad y_R = -(\lambda x_R + \nu), \tag{3.4}$$

where $\nu = y_P - \lambda x_P$.

From these definitions it becomes clear, why additive notation is used for elliptic curve groups. The two operations are also known as the *chord-and-tangent* rule. Inversion of a point P is defined to be $-P = (x_P, -y_P)$, that is, the y-coordinate is reflected over the x-axis. For a point P, one can observe that

$$[0]P = \mathcal{O},\ [1]P = P,\ [2]P = P + P,\ [3]P = [2]P + P, ...,\ [n]P = [n-1]P + P\,.$$

The operation *point multiplication* follows from this observation and is defined as $[n]P = P + P + ... + P$ (n times) for $n \geq 0$. Note that the concrete computations for any of the elliptic curve operations take place in the underlying field though.

The previous definitions of point addition and point doubling considered the usage of *affine coordinates*. However, *projective coordinates* are often used to speed up the computation as to avoid field inversions, that is, points in the 2-dimensional space are mapped to lines in the 3-dimensional space. Using *homogeneous projective coordinates*, a point $(x, y) \in K^2$ is mapped to the line $(\lambda x, \lambda y, \lambda) \in K^3$, where $\lambda \in \overline{K}^*$. Consequently, the point $(0, 0, 0) \in K^3$ is not mapped at all and all points on the same line in the 3-dimensional space equal the same point in the 2-dimensional space. Hence, a single point has several possible mappings in K^3. The set of all equivalent points is denoted by $(X : Y : Z)$. A point $P = (x, y)$ is usually mapped to $(x : y : 1)$ and a projective point $P_{proj} = (X : Y : Z)$ is mapped back to $(X/Z, Y/Z)$. Using the affine Equation 3.1, the point at infinity $\mathcal{O}$ can not be specified. In projective coordinates, $\mathcal{O}$ is represented by $(0 : 1 : 0)$, which can obviously not be mapped back to affine coordinates. The curve equation in homogeneous projective coordinates changes to

$$Y^2Z = X^3 + aXZ^2 + bZ^3. \tag{3.5}$$

Accordingly, the formulas for point addition and doubling are adapted, avoiding costly field inversion until the projective point is mapped back to affine coordinates. There are also other types of projective coordinates that have different mappings, for example, *Jacobian projective coordinates* perform a mapping $(X : Y : Z) \mapsto (X/Z^2, Y/Z^3)$. A database with explicit formulas for different types of projective coordinates is provided by Bernstein and Lange [BL13]. However, in the context of this thesis, homogeneous projective coordinates are used.

3.2.2. Group Structure

The number of elements for a finite field $\mathbb{F}_q$ with prime characteristic p is obviously q. However, for an elliptic curve $E/\mathbb{F}_q$, *Hasse* gives a bound on the number of elements in the group (see Silverman [Sil09, Ch. 5, Th. 1.1]).

Theorem 3.4 (Hasse's theorem)**.** *Let $E/\mathbb{F}_q$ be an elliptic curve E defined over a finite field $\mathbb{F}_q$, then the number of elements in the group $\#E(\mathbb{F}_q)$ satisfies*

$$\#E(\mathbb{F}_q) = q + 1 - t \qquad |t| \leq 2\sqrt{q}$$

Consequently, the group order can be approximated as $\#E(\mathbb{F}_q) \approx q = \#\mathbb{F}_q$. The concrete number of elements can be counted in polynomial time using Schoof's algorithm [Sch85] though. According to Lagrange's theorem (see Theorem 3.1), there will generally be subgroups of order $n \,|\, \#E(\mathbb{F}_q)$, for example, the (single) point of order 1 is $\mathcal{O}$, and the generator is of order $\#E(\mathbb{F}_q)$. To obtain a point of order n, one has to multiply the generator with the appropriate cofactor $h = \#E(\mathbb{F}_q)/n$. Usually, it is desired for cryptographic applications to have groups of prime order, because it makes hard solving the elliptic curve variant of the discrete logarithm problem. Otherwise, an adversary could solve the problem partially in each of the subgroups and apply the chinese remainder theorem. Over finite fields, E is called *supersingular* if the characteristic p satisfies $p \mid t$, and *ordinary* otherwise.

Definition 3.25. The i-th *frobenius endomorphism* of the group $E(\mathbb{F}_q)$ is defined as

$$\pi_{q^i} : E(\overline{\mathbb{F}_q}) \mapsto E(\overline{\mathbb{F}_q}), \qquad (x, y) \mapsto (x^{q^i}, y^{q^i})$$

The frobenius endomorphism has the property, that π_{q^i} only acts non-trivially on $E(\overline{\mathbb{F}}_q) \backslash E(\mathbb{F}_{q^i})$, that is, it does not change anything about elements in $E(\mathbb{F}_{q^i})$. Let π_q denote the first frobenius endomorphism. Then, a point $P \in E(\overline{\mathbb{F}}_q)$ is also in $E(\mathbb{F}_q)$ if and only if $\pi_q(P) = P$, which leads to the notation $E(\mathbb{F}_q) = E(\overline{\mathbb{F}}_q) \cap \ker([1] - \pi_q)$. This is important with respect to finding a bound for the number of elements (Hasse) by filtering out elements in extensions of $E(\mathbb{F}_q)$ as shown by Silverman [Sil09, Ch.5, Th. 1.1, Le. 1.2].

The term t in Hasse's theorem is called the *trace of frobenius* as it results from the trace of the first *frobenius endomorphism*. It fulfills the characteristic polynomial [Sil09, Ch. 5, Re. 2.6, Th. 2.3.1]

$$\pi_q^2 + [t] \circ \pi_q + [q] = 0,$$

or equivalently, for all $P \in E(\overline{\mathbb{F}}_q)$ it satisfies

$$(x^{q^2}, y^{q^2}) + [t](x^q, y^q) + [q](x, y) = \mathcal{O}$$

Closely related to group orders is the term of n-torsion.

Definition 3.26. The n-torsion subgroup $E[n]$ of an elliptic curve E, for any $n \in \mathbb{Z}, n \geq 1$, is the set of points of E of order n, that is,

$$E[n] = \{P \in E(\overline{\mathbb{F}}_q) : [n]P = \mathcal{O}\}.$$

The elements of $E[n]$ are called *n-torsion points*. In this context, the following theorem by Cohen et al. [Coh+10, Th. 13.13] gives an idea about the structure of the n-torsion.

Theorem 3.5. *Let E be an elliptic curve defined over the finite field $\mathbb{F}_q$. If its characteristic is either zero or co-prime to n, then*

$$E[n] \cong \mathbb{Z}_n \times \mathbb{Z}_n.$$

3.3. Bilinear Maps

In a simple way, bilinear maps are an additional property for certain types of cyclic groups that allows to map two elements from two cyclic groups to an element of a third cyclic group. Such a map fulfills a bilinearity property with respect to the elements chosen from the former two cyclic groups. Originally, bilinear maps were used by Menezes, Okamoto, and Vanstone [MOV93] and Frey and Rück [FR94] to break cryptosystems by transferring hard problems in cyclic groups to another that provides better attacks. Later it was discovered, that cryptosystems can be built from it, for example, identity-based encryption.

Let $\mathbb{G}$, $\hat{\mathbb{G}}$, and $\mathbb{G}_t$ be cyclic groups of prime order n with their respective generators G, $\hat{G}$, G_t. Lynn [Lyn07, Sections 1.4 and 1.8] defines a bilinear map, also called bilinear pairing, as a map

$$e : \mathbb{G} \times \hat{\mathbb{G}} \longrightarrow \mathbb{G}_t,$$

that has the following properties:

1. Bilinearity: $e(aP, bQ) = e(P, Q)^{ab} \ \forall\ P \in \mathbb{G} \ \ Q \in \hat{\mathbb{G}} \ \ a, b \in \mathbb{Z}$
2. Non-degeneracy: $e(G, \hat{G}) \neq 1$
3. Computable: $e(P, Q)$ can be computed efficiently

As a consequence of bilinearity, the following properties hold:

1. $e(R + S, T) = e(R, T) \cdot e(S, T)$ and $e(R, S + T) = e(R, S) \cdot e(R, T)$,
2. $e(-S, T) = E(S, -T) = e(S, T)^{-1}$.

In some cases, these properties make possible trading operations in $\mathbb{G}$ and $\hat{\mathbb{G}}$ with operations in $\mathbb{G}_t$. Additionally, non-degeneracy also implies that $e(S, T) = 1$ if, and only if $S = \mathcal{O} \ \vee \ T = \mathcal{O}$.

These definitions are rather abstract and are typically used in the context of cryptography so as to define protocols and schemes. As one can see in the above equations, the two groups $\mathbb{G}$, $\hat{\mathbb{G}}$ use additive notation as these are usually elliptic curve groups. Contrary to that, $\mathbb{G}_t$ is a multiplicative group. For the remaining of this thesis, we will stick to this type of definition in the context of pairings.

The type of pairing defined before is called *asymmetric* as $\mathbb{G}$ and $\hat{\mathbb{G}}$ are two different cyclic groups. Accordingly, there is also a *symmetric* definition of a bilinear map, where $\mathbb{G}$ equals $\hat{\mathbb{G}}$. Protocols are often only defined using the symmetric definition of pairings. However, asymmetric pairings are faster and to be preferred practically. Actually, there are four different types of pairings, which are discussed in more detail in Section 3.4.2.

3.3.1. Diffie-Hellman Problem

For the sake of clarity of the security of bilinear maps, a short review on the original Diffie-Hellman problem [DH76] is done. Let $\mathbb{G}$ be a multiplicative cyclic group of order n with generator g.

Computational Diffie-Hellman problem. Given g, g^a, and g^b, compute g^{ab}.

Decisional Diffie-Hellman problem. Boneh [Bon98] defines the problem such as to determine whether $ab = c \bmod n$ provided the values g, g^a, g^b, and g^c.

Both of them rely on the Discrete Logarithm Problem (DLP), namely determining x from g^x, to be hard. Otherwise, anyone could first calculate a from g^a, then b from g^b, and finally compute g^{ab}. As a consequence of pairings, the decisional Diffie-Hellman problem can be solved for certain types of cyclic groups. Consider an additive cyclic group $\mathbb{G}$ in a symmetric bilinear pairing. To solve the decisional Diffie-Hellman problem, provided G, aG, bG, and cG, one could simply compute $e(aG, bG) = e(G, G)^{ab}$ and $e(G, cG) = e(G, G)^c$ and check their equivalence. The computational Diffie-Hellman problem may, however, still be hard.

3.3.2. Bilinear Diffie-Hellman Problem

Similar to the Diffie-Hellman problem for cyclic groups, there exists the Bilinear Diffie-Hellman (BDH) problem for pairings as first described by Boneh and Franklin [BF01]. For a *symmetric* pairing $e : \mathbb{G} \times \mathbb{G} \longrightarrow \mathbb{G}_t$, the bilinear Diffie-Hellman problem is defined as:

> Given G, aG, bG, $cG \in \mathbb{G}^4$, calculate $e(G, G)^{abc} \in \mathbb{G}_t$.

For the more general *asymmetric* pairing $e : \mathbb{G} \times \hat{\mathbb{G}} \longrightarrow \mathbb{G}_t$, the bilinear Diffie-Hellman problem is defined by Boneh and Boyen [BB04b, Section 3.1] in the following variants.

Computational BDH problem Given $G, aG, cG, \hat{G}, a\hat{G}, b\hat{G}$, an algorithm $\mathcal{A}$ has advantage $\mathbf{Adv}_{\mathcal{A}}^{\mathtt{BDH}} = \epsilon$ in solving the computational BDH problem in $(\mathbb{G}, \hat{\mathbb{G}})$ if

$$\mathtt{Pr}\left[\mathcal{A}(G, aG, cG, \hat{G}, a\hat{G}, b\hat{G}) = e(G, \hat{G})^{abc}\right] \geq \epsilon$$

where the coefficients a, b, $c \in \mathbb{Z}_n$ and the generators G, $\hat{G}$ are chosen randomly.

Decisional BDH problem An algorithm $\mathcal{B}$ that, for a tuple $(G, aG, cG, \hat{G}, a\hat{G}, b\hat{G}, T)$, returns a bit $\gamma = 1$ if $T = e(G, \hat{G})^{abc}$ and $\gamma = 0$ otherwise, has an advantage $\mathbf{Adv}_{\mathcal{A}}^{\mathtt{BDDH}} = \epsilon$ in solving the Decisional Bilinear Diffie-Hellman (BDDH) problem in $(\mathbb{G}, \hat{\mathbb{G}})$ if

$$\left|\mathtt{Pr}\left[\mathcal{B}(G, aG, cG, \hat{G}, a\hat{G}, b\hat{G}, e(G, \hat{G})^{abc}) = 0\right] - \mathtt{Pr}\left[\mathcal{B}(G, aG, cG, \hat{G}, a\hat{G}, b\hat{G}, T) = 0\right]\right| \geq \epsilon$$

where the coefficients a, b, $c \in \mathbb{Z}_n$, the generators G, $\hat{G}$, and the value $T \in \mathbb{G}_t$ are chosen randomly.

These formulations lead to the following definition of security with respect to the (decisional-)BDH problem.

Definition 3.27. The (t, ϵ)-(Decisional-)BDH assumption hold in $(\mathbb{G}, \hat{\mathbb{G}})$ if no t-time algorithm has an advantage of at least ϵ in solving the (Decisional-)BDH problem in $(\mathbb{G}, \hat{\mathbb{G}})$.

For negligible advantage ϵ, the parameters are omitted and the assumptions are referred to as BDH and decisional BDH respectively. Again, hardness of the BDH implies hardness of the discrete logarithm problem in $\mathbb{G}$, $\hat{\mathbb{G}}$, and $\mathbb{G}_t$. Assume the DLP could be solved in $\mathbb{G}$. One could easily calculate a from aG, compute a bilinear map $e(bG, c\hat{G}) = e(G, \hat{G})^{bc}$, and exponentiate this result by a, which would solve both the BDH problems. Analogously, the BDH assumption would not hold if the DLP could by solved in $\hat{\mathbb{G}}$. Finally, if the DLP

could be solved in $\mathbb{G}_t$, a could be recovered from $e(aG, \hat{G}) = e(G, \hat{G})^a$, again solving the whole BDH. In addition, the decisional BDH assumption in $(\mathbb{G}, \hat{\mathbb{G}})$ implies that the regular decisional Diffie-Hellman assumption in G_t holds. Accordingly, parameters of appropriate size need to be chosen in any of these groups for a pairing-based cryptosystem to stay secure.

Furthermore, there are several variants of the BDH. Examples for these stronger assumptions are the Bilinear Diffie-Hellman Inversion (BDHI) and the truncated Augmented Bilinear Diffie-Hellman Exponent (q-ABDHE) assumptions. BDHI assumes hardness of calculating $e(G, \hat{G})^{1/x}$ from $(G, xG, \hat{G}, x\hat{G})$. The task in the truncated q-ABDHE problem is to calculate $e(G, \hat{G})^{x^{q+1}}$ from $(G, x^{q+2}G, \hat{G}, x\hat{G}, x^2\hat{G}, ..., x^q\hat{G})$.

3.4. Pairing Definition

Having the concepts of groups, (extension) fields and elliptic curves in mind, a concrete bilinear map that fulfills the properties mentioned in the previous section can be defined. For this, the theory on function divisors is needed and therefore explained in Section 3.4.1. Several ways to specify the concrete groups used for bilinear maps are shown in Section 3.4.2. Then, the Tate pairing over elliptic curves is defined in Section 3.4.3. A method to evaluate the Tate pairing is presented in Section 3.4.4. The optimal Ate pairing that is later on used for implementation is explained in Section 3.4.5.

The definitions follow Costello [Cos13, Ch. 3] and Galbraith [Gal06, Ch. IX.2] for divisors in Section 3.4.1, Costello [Cos13, Ch. 4] for the group definitions in Section 3.4.2, and Costello [Cos13, Ch. 5] and Galbraith [Gal06, Ch. IX.3-5, IX.7] for the Tate pairing and the Miller algorithm in Sections 3.4.3 and 3.4.4.

3.4.1. Divisors

Definition 3.28. Let $E(\overline{\mathbb{F}}_q)$ denote the set of points of an elliptic curve E defined over the algebraic closure of $\mathbb{F}_q$. A *divisor* on E is a formal sum

$$D = \sum_{P \in E(\overline{\mathbb{F}}_q)} n_P(P),$$

where $n_P \in \mathbb{Z}$ and all but finitely many n_P are zero.

Note that the standard parenthesis () is used instead of the square parenthesis [] to distinguish between the sum of a divisor and a point multiplication. The set of divisors on E together with the operation of addition forms a group, denoted $\mathrm{Div}_{\hat{\mathbb{F}}_q}(E)$. Its identity is the *zero divisor*, which is defined by $n_P = 0 \; \forall \; P \in E(\overline{\mathbb{F}}_q)$ and denoted 0. The *support* of a divisor D is defined as the set of points P for which $n_P \neq 0$. A divisors *degree* is defined as $\deg(D) = \sum_P n_P$. Divisors can be used to describe a function f on the elliptic curve E by their points of intersection.

Definition 3.29. Let f be a non-zero function on the curve E and $\mathrm{ord}_P(f)$ denote the multiplicity of f at a point P, which is positive if f has a zero at P, and negative if f has a pole at P. The divisor of f is then defined as

$$(f) = \sum_{P \in E(\overline{\mathbb{F}}_q)} \mathrm{ord}_P(f)(P).$$

Obviously, it follows that $(fg) = (f) + (g)$ and $(f/g) = (f) - (g)$. A divisor (f) is zero if and only if f is a constant. Hence, if $(f/g) = 0$, the two functions f and g are equal up to a non-zero scalar multiple, that is, a divisor defines a function f up to a non-zero scalar multiple. Additionally, the divisor's degree of any function f on E is zero, which results from analyzing the function in projective coordinates.

Considering the chord-and-tangent rule from Section 3.2.1, the line ℓ used for point addition has three intersections with E, namely P, Q and $-(P+Q)$. Moreover, it has three poles at $\mathcal{O}$, hence $(\ell) = (P) + (Q) + (-(P+Q)) - 3(\mathcal{O})$. The mapping of a point P to a divisor $(P) - (\mathcal{O})$ is a group homomorphism. Two divisors are *equivalent*, denoted $D_1 \sim D_2$, if $D_1 = D_2 + (f)$ for some function f. Consequently, Galbraith [Gal06, Th. IX.2] defines the following theorem:

Theorem 3.6. *Let E be an elliptic curve over the field $\mathbb{F}_q$ and $D = \sum_P n_P(P)$ be a divisor with $deg(D) = 0$. Then $D \sim 0$ (i.e., there is a function f such that $D = (f)$) if and only if $\sum_P [n_P]P = \mathcal{O}$ on E. A divisor that satisfies this condition is said to be principal.*

Given a divisor $D = \sum_P n_P(P)$ of zero degree and a function f, such that (f) and D have disjoint support, f is evaluated as

$$f(D) = \prod_P f(P)^{n_P}. \tag{3.6}$$

Note that $f(D) = g(D)$ if $(f) = (g)$. The condition that (f) and D have disjoint support is necessary as the function would otherwise evaluate to zero or infinity. If both (f) and D are defined over a field K, then $f(D) \in K$. To close the section on divisors, the Weil reciprocity theorem is stated as it is an integral part of many proofs for pairings.

Theorem 3.7 (Weil reciprocity)**.** *Given two functions f and g such that (g) and (h) have disjoint support, then*

$$f((g)) = g((f)),$$

that is, evaluating f using divisor (g) equals evaluating g using divisor (f).

Nice illustrations of divisors and evaluations of functions using divisors can be found in the guide by Costello [Cos13, Ch.3].

3.4.2. Group Definition

As indicated in Section 3.3, three groups $\mathbb{G}$, $\hat{\mathbb{G}}$, and $\mathbb{G}_t$ need to be defined for a bilinear map. The focus of this section is the definition of both $\mathbb{G}$ and $\hat{\mathbb{G}}$. Therefore, the n-torsion mentioned in Definition 3.26 is something that needs a closer look first.

The n-torsion basically consists of all points $P \in E(\overline{\mathbb{F}}_q)$ of order n that satisfy the curve equation. Theorem 3.5 gives a hint on its structure. Generally, its size will be $\#E[n] = n^2$. As the identity $\mathcal{O}$ is part of each of the order-n subgroups of the n-torsion, it splits up into $n+1$ cyclic subgroups of order n. Typically, only one of the subgroups of the n-torsion is found in the base field $E(\mathbb{F}_q)$. To find further of them, the field $\mathbb{F}_q$ needs to be extended to $\mathbb{F}_{q^k}$, where $k \geq 1$ is called the *embedding degree*. This embedding degree k is the least positive integer such that another point of order n is found in $E(\mathbb{F}_{q^k}) \backslash E(\mathbb{F}_q)$. There are several meanings of the embedding degree, for example,

- k is the smallest positive integer such that $n \mid (q^k - 1)$,
- k is the smallest positive integer such that $\mathbb{F}_{q^k}$ contains all of the n-th roots of unity of $\overline{\mathbb{F}}_q$ (denoted $\boldsymbol{\mu}_n$),
- k is the smallest positive integer such that $E[n] \subset E(\mathbb{F}_{q^k})$.

In this context, $k > 1$ if $n \mid \#E(\mathbb{F}_q)$ and $n^2 \nmid \#E(\mathbb{F}_q)$. Once another point of order n is found in $E(\mathbb{F}_{q^k}) \backslash E(\mathbb{F}_q)$, the whole n-torsion $E[n] = \mathbb{Z}_n \times \mathbb{Z}_n$ is found. The illustrations of the n-torsion by Costello [Cos13, Ch. 4.1] are really recommended to the interested reader.

Two of the $n + 1$ subgroups of $E[n]$ have special properties. Based on Definition 3.25 (Frobenius endomorphism), the *trace map* for $P \in E(\mathbb{F}_{q^k})$ is defined as

$$\mathrm{Tr}(P) = \sum_{i=0}^{k-1} \pi_{q^i}(P).$$

Let $\mathcal{G}_B$ denote the single subgroup of $E[n]$ which is in $E(\mathbb{F}_q)$, that is, $k > 1$. Then, the trace map maps all elements in the n-torsion to elements in the base-field group $\mathcal{G}_B$. However, there is one subgroup, the *trace zero group* $\mathcal{T}$, where the trace map transfers any element to $\mathcal{O}$. The other way round, the anti-trace map $P' = [k]P - Tr(P)$ can be used to map to $\mathcal{T}$. More formally,

- $\mathcal{G}_B = E[n] \cap Ker(\pi_q - [1])$, and
- $\mathcal{T} = \{P \in E(\mathbb{F}_{q^k})[n] : \mathrm{Tr}(P) = \mathcal{O}\}$.

$\mathbb{G}$ and $\hat{\mathbb{G}}$ can be defined to be any of the order-n subgroups of $E[n]$. Depending on which subgroups are used, different types of pairings are specified. However, $\mathbb{G}$ and $\hat{\mathbb{G}}$ must not be the same subgroup of $E[n]$ as the bilinear map would otherwise become degenerate. The following should give a short overview on the four different types of pairings:

Type-1 Pairing. The only type of pairing that uses supersingular curves is the type-1 pairing. The two groups $\mathbb{G}$, $\hat{\mathbb{G}}$ are set to be $\mathbb{G} = \hat{\mathbb{G}} = \mathcal{G}_B$. In this case, there exists an efficiently computable map to map elements out of $\mathcal{G}_B$, the so-called *distortion map*. This map is applied to the elements of one of these two groups for the pairing not to become degenerate. Obviously, there is an efficient homomorphism $\psi : \hat{\mathbb{G}} \mapsto \mathbb{G}$, namely the identity. Hashing to either of these groups does not state a problem. Drawback of type-1 pairings is the difficulty to find curves that are suitable for efficient pairings.

Type-2 Pairing. As for type-3 and type-4 pairings, the type-2 pairing uses ordinary elliptic curves. The group $\mathbb{G}$ is chosen to be $\mathcal{G}_B$ and $\hat{\mathbb{G}}$ is chosen to be any of the $n - 1$ subgroups of $E[n]$ that is neither $\mathcal{G}_B$ nor $\mathcal{T}$. The trace map constitutes an efficiently computable homomorphism that maps from $\hat{\mathbb{G}}$ to $\mathbb{G}$. However, an efficient way to hash into $\hat{\mathbb{G}}$ is unknown. A point multiplication with the respective generator is the only solution, which is generally not satisfactory.

Type-3 Pairing. In this type of pairing, $\mathbb{G}$ is set to be $\mathcal{G}_B$ and $\mathcal{T}$ is used for $\hat{\mathbb{G}}$. As there is no efficient way to map out of $\mathcal{T}$, this type of pairing lacks an efficient homomorphism that maps from $\hat{\mathbb{G}}$ to $\mathbb{G}$. Instead, hashing to both $\mathbb{G}$ and $\hat{\mathbb{G}}$ is efficiently possible.

Type-4 Pairing. As for the other types, $\mathbb{G}$ is chosen to be $\mathcal{G}_B$. The whole n-torsion $E[n]$ is chosen for $\hat{\mathbb{G}}$, which is a group of order n^2. Hashing in one particular subgroup of $\hat{\mathbb{G}}$ is not possible. Hashing to $\hat{\mathbb{G}}$ is feasible in general, but not very efficient.

Another introduction to the different types of pairings is given by Galbraith, Paterson, and Smart [GPS08].

3.4.3. Tate Pairing

As indicated in the previous section, $E(\mathbb{F}_{q^k})$ contains all elements of the n-torsion $E(\overline{\mathbb{F}}_q)[n]$, where k denotes the embedding degree. The subgroup $nE(\mathbb{F}_{q^k})$ is defined as

$$nE(\mathbb{F}_{q^k}) = \{[n]P : P \in E(\mathbb{F}_{q^k})\}.$$

The subgroup $nE(\mathbb{F}_{q^k})$ has order $h = \#E(\mathbb{F}_{q^k})/n^2$, which is the cofactor that is used to map an element from $E(\mathbb{F}_{q^k})$ to the n-torsion group $E[n]$. Similar to the definition of equivalence classes in Section 3.1.1, a group consisting of n^2 different residue classes of order h can be defined as $E(\mathbb{F}_{q^k})/nE(\mathbb{F}_{q^k})$, shortly denoted E/nE. The quotient group E/nE is simply the set of equivalence classes of points in $E(\mathbb{F}_{q^k})$ under the equivalence relation $P_1 \equiv P_2$ if and only if $P_1 - P_2 \in nE$.

The groups $E[n]$ and E/nE have the same number of points, namely n^2. It is not necessarily the case that the elements from $E[n]$ can be used to represent the residue classes of E/nE. However, in general $E[n] \cap E/nE = \{\mathcal{O}\}$ if $n^2 \,||\, \#E(\mathbb{F}_{q^k})$, which was shown by Galbraith [Gal06, Th. IX.22] for the supersingular case. Therefore, $E[n]$ can be used to represent the cosets of E/nE. If this were not the case, the pairing in Definition 3.30 would become degenerate.

By definition, $[n]P = \mathcal{O}$ for a given point $P \in E(\mathbb{F}_{q^k})[n]$. It results from Theorem 3.6, that there must exist a function that satisfies $(f) = n(P) - n(\mathcal{O})$. Consequently, Costello [Cos13, Def. 5.2] defines the Tate pairing as follows:

Definition 3.30. Let $P \in E(\mathbb{F}_{q^k})[n]$, f be a function with divisor $(f) = n(P) - n(\mathcal{O})$, and $Q = E(\mathbb{F}_{q^k})$ be any representative of any equivalence class in $E(\mathbb{F}_{q^k})/nE(\mathbb{F}_{q^k})$. Let D_Q be a divisor of zero degree defined over $E(\mathbb{F}_{q^k})$ which is equivalent to $(Q) - (\mathcal{O})$ and has disjoint support to that of (f). Then, the Tate pairing is a map

$$t_n : E(\mathbb{F}_{q^k})[n] \times E(\mathbb{F}_{q^k})/nE(\mathbb{F}_{q^k}) \longrightarrow F^*_{q^k}/(F^*_{q^k})^n,$$

defined as

$$t_n(P, Q) = f(D_Q).$$

The definition of the quotient group $\mathbb{F}^*_{q^k}/(\mathbb{F}^*_{q^k})^n$ is similar to the definition of $E(\mathbb{F}_{q^k})/nE(\mathbb{F}_{q^k})$: $(\mathbb{F}^*_{q^k})^n$ is a subgroup of $\mathbb{F}^*_{q^k}$ and defined as $(\mathbb{F}^*_{q^k})^n = \{u^n : u \in \mathbb{F}^*_{q^k}\}$. Consequently, $\mathbb{F}^*_{q^k}/(\mathbb{F}^*_{q^k})^n$ denotes the set of all equivalence classes of elements in $\mathbb{F}^*_{q^k}$ under the equivalence relation $u_1 = u_2$ if and only if $u_1/u_2 \in (\mathbb{F}^*_{q^k})^n$. The quotient group $\mathbb{F}^*_{q^k}/(\mathbb{F}^*_{q^k})^n$ is isomorphic to $\boldsymbol{\mu}_n = \{u \in \mathbb{F}^*_{q^k} : u^n = 1\}$, the set of all n-th roots of unity.

However, the Tate pairing from Definition 3.30 is not practical for cryptography. It maps to an equivalence class rather than to an exact value. The reason for this lies within the choice of D_Q and that fact that Q is merely any representative of the respective equivalence class. To obtain an exact value, the resulting value in $\mathbb{F}^*_{q^k}/(\mathbb{F}^*_{q^k})^n$ needs to be

raised to the power of $(q^k - 1)/n$, which eliminates all n-th powers and maps to an exact n-th root of unity in $\boldsymbol{\mu}_n$. This allows definition of the reduced Tate pairing over finite fields [Cos13, Def. 5.3]:

Definition 3.31. Given P, Q, f and D_Q as in Definition 3.30. The reduced Tate pairing over finite fields is a map

$$T_n : E(\mathbb{F}_{q^k})[n] \times E(\mathbb{F}_{q^k})/nE(\mathbb{F}_{q^k}) \longrightarrow \boldsymbol{\mu}_n,$$

defined as

$$\begin{aligned} T_n(P,Q) &= t_n(P,Q)^{\#\mathbb{F}_{q^k}/n} \\ &= f_{n,P}(D_Q)^{(q^k-1)/n} \end{aligned}$$

Consequently, $\mathbb{G}_t$ is defined to be $\boldsymbol{\mu}_n$.

3.4.4. Miller Algorithm

From Definition 3.31 it is known that computing a pairing is basically evaluating a function f with divisor $(f) = n(P) - n(\mathcal{O})$. For small values of n, it may seem feasible to construct such a function f and evaluate the value of f in D_Q using Equation 3.6. For large n, which is the case for cryptographic applications, this is unpractical though. Miller [Mil04] found an algorithm to evaluate such a function f in polynomial time.

According to Theorem 3.6, there must exist a function $f_{i,P}$ with divisor

$$(f_{i,P}) = i(P) - ([i]P) - (i-1)(\mathcal{O}),$$

that is uniquely defined up to a constant multiple. For a point $P \in E[n]$, the divisor obviously simplifies to $(f_{n,P}) = n(P) - n(\mathcal{O})$, which is exactly what is needed.

The divisor $(f_{i+1,P}) - (f_{i,P}) = (P) + ([i]P) - ([i+1]P) - (\mathcal{O})$ equals the function $\ell_{[i]P,P}/\nu_{[i+1]P}$, where $\ell_{[i]P,P}$ denotes the line through $[i]P$ and P, and $\nu_{[i+1]P}$ denotes the vertical line through $[i+1]P$, that is, the lines used in the computation of $[i]P+P = [i+1]P$ (see Section 3.2.1 and Section 3.4.1). This observation leads to the relations

$$\begin{aligned} f_{i+j,P} &= f_{i,P} f_{j,P} \frac{\ell_{[i]P,[j]P}}{\nu_{[i+j]P}}, \text{ and} \\ f_{2i,P} &= f_{i,P}^2 \frac{\ell_{[i]P,[i]P}}{\nu_{[2i]P}}, \end{aligned}$$

which can in turn be used to express $f_{n,P}$ recursively in a *double-and-add* approach, namely the *Miller algorithm*. Its starting point is $(f_{0,P}) = 0$, that is, a function f defined by an arbitrary constant value not equal to zero. A concrete instance of the Miller algorithm can be found in Section 6.2.4.

3.4.5. (Optimal) Ate Pairing

The Tate pairing can come in different settings as explained in Section 3.4.2. It turned out, that the type-3 pairing is the most efficient, that is, $\mathbb{G} = E[n] \cap \mathrm{Ker}(\pi_q - [1])$ and $\hat{\mathbb{G}} = \mathcal{T} = \{P \in E(\mathbb{F}_{q^k}) : \mathrm{Tr}(P) = \mathcal{O}\} = E[n] \cap \mathrm{Ker}(\pi_q - [q])$. The Ate pairing, which was first defined by Hess, Smart, and Vercauteren [HSV06], evolved from both the Tate pairing

and the Eta pairing over supersingular curves (see Barreto et al. [Bar+04] for details on the Eta pairing). For the Ate pairing, the two groups $\mathbb{G}$ and $\hat{\mathbb{G}}$ as defined above are used, that is, the two eigenspaces of the trace of frobenius. In the pairing definition itself, the two groups are then switched, that is, $\hat{\mathbb{G}} \times \mathbb{G} \longrightarrow \boldsymbol{\mu}_n$.

Vercauteren [Ver10, Ch. 2.2] derives the concrete mapping by raising the fixed result from the reduced Tate pairing by a constant $m = (\lambda^k - 1)/n \in \mathbb{Z}$, which still gives a bilinear pairing, and trying to achieve a small constant $\lambda = q \bmod n$ for the Miller function $f_{\lambda,Q}$ by using the fact that multiplication by q does affect elements from $\hat{\mathbb{G}}$, but not elements in $\mathbb{G}$. Consequently, the Ate pairing, that is given by

$$a_\lambda : \hat{\mathbb{G}} \times \mathbb{G} \longrightarrow \boldsymbol{\mu}_n : (Q, P) \longrightarrow f_{\lambda,Q}(P)^{(q^k-1)/n},$$

results in a value that is a fixed power of the corresponding reduced Tate pairing. Details on how to find an optimal λ to reduce the effort of computing the Miller function as well as concrete examples are given by Vercauteren [Ver10, Ch. 3, Ch.4].

3.5. Pairing-friendly curves

Menezes [Men05] points out that only a small part of all the existing elliptic curves are suitable for pairing-based cryptography as several properties need to be fulfilled:

- The group order n needs to be a prime divisor of $\#E(\mathbb{F}_q)$ that is co-prime to q.
- To avoid Pollard's rho method, n needs to be sufficiently large such that computing the discrete logarithm in an order-n subgroup of $E(\mathbb{F}_q)$ is infeasible.
- The embedding degree k needs to be large enough to avoid index-calculus methods that could solve the DLP in $\mathbb{F}_{q^k}$.
- On the other hand, k should be small enough to have efficient computations.

All current techniques for finding such ordinary elliptic curves rely on the *Complex Multiplication* (CM) method, which evolved from the results by Morain [Mor87] and Atkin and Morain [AM93]. The first procedure to generate ordinary elliptic curves of low embedding degree suitable for pairings was described by Miyaji, Nakabayashi, and Takano [MNT01]. Curves generated by this method are usually called *MNT*-curves and are particularly efficient for fields using 160-bit primes. Galbraith [Gal06, Section IX.15.2.] further states another method, namely the *Cocks-Pinch* method, that allows construction of elliptic curves of any embedding degree. Finally, Barreto and Naehrig [BN06] present a method to construct elliptic curves of prime order n over prime fields $\mathbb{F}_p$ with the fixed embedding degree $k = 12$. These curves are very often just labeled *BN curves*. As these are used in the context of this thesis, the following section shall give an overview on them.

3.5.1. Barreto-Naehrig Curves

Elliptic curves constructed using the method presented by Barreto and Naehrig [BN06] have gained much popularity as they can be computed efficiently for fields defined using 256-bit primes. These elliptic curves, which are defined over a prime field $\mathbb{F}_p$ as $E(\mathbb{F}_p) : y^2 = x^3 + b$, automatically have a group of prime order n that is almost the same size as the prime of the underlying field. The embedding degree with respect to the underlying field $\mathbb{F}_p$ and the group order n is always $k = 12$. The curves are parameterized in u:

- Field prime $p(u) = 36u^4 + 36u^3 + 24u^2 + 6u + 1$.
- Group order: $n(u) = 36u^4 + 36u^3 + 18u^2 + 6u + 1$.
- Trace of frobenius: $t(u) = 6u^2 + 1$.

In Section 3.4.5, the two groups $\mathbb{G}$ and $\hat{\mathbb{G}}$ involved in the Ate pairing $a_\lambda : \hat{\mathbb{G}} \times \mathbb{G} \longrightarrow \boldsymbol{\mu}_n$ were defined to be in the type-3 setting. For BN curves, the two groups are defined as $\mathbb{G} = E(\mathbb{F}_{p^{12}})[n] \cap \mathrm{Ker}(\pi_p - [1]) \subseteq E(\mathbb{F}_p)$ and $\hat{\mathbb{G}} = E(\mathbb{F}_{p^{12}})[n] \cap \mathrm{Ker}(\pi_p - [p]) \subseteq E(\mathbb{F}_{p^{12}})$. Written in a simplified notation, this gives an Ate pairing of the form

$$E(\mathbb{F}_{p^{12}}) \times E(\mathbb{F}_p) \longrightarrow \mathbb{F}_{p^{12}}. \tag{3.7}$$

The large elements in $\hat{\mathbb{G}}$ are not satisfying though. Barreto and Naehrig [BN06, Section 3] propose the compression of points in the group $\hat{\mathbb{G}}$ to both yield better performance and smaller element sizes. This is done by using the curve's sextic twist $E'(\mathbb{F}_{p^2})$. An element $Q \in E(\mathbb{F}_{p^{12}})$ may then be represented by a point $Q' \in E'(\mathbb{F}_{p^2})$ since there exists an efficient injective group homomorphism $\psi : E'(\mathbb{F}_{p^2}) \to E(\mathbb{F}_{p^{12}})$. Consequently, the pairing operation can be done by using this homomorphism. On the other hand, non-pairing operations such as hashing to the group or common curve operations can be done in the smaller field. Using the twist modifies the rather informal notation of the pairing from Equation 3.7 to

$$E'(\mathbb{F}_{p^2}) \times E(\mathbb{F}_p) \longrightarrow \mathbb{F}_{p^{12}}. \tag{3.8}$$

The sextic twist of the curve E is defined as $E'(\mathbb{F}_{p^2}) : y'^2 = x'^3 + b/\xi$, where ξ satisfies that the polynomial $z^6 - \xi$ is irreducible in $\mathbb{F}_{p^2}[z]$. The roots of the polynomial $z^6 - \xi$ are used to both define $\mathbb{F}_{p^{12}}$ over $\mathbb{F}_{p^2}$ and the elliptic curve's sextic twist E'. It must be taken care, that the chosen ξ produces a twist of correct order, that is, $n \mid \#E'(\mathbb{F}_{p^2})$. Otherwise an easily computable alternative is given by ξ^5. For more details on twists in general, the reader may refer to Costello [Cos13, Ch. 4.3].

Given a root $z \in \mathbb{F}_{p^{12}}$ of the polynomial $z^6 - \xi$, the necessary map is defined as $\psi : E'(\mathbb{F}_{p^2}) \to E(\mathbb{F}_{p^{12}}) : (x', y') \mapsto (z^2x', z^3y')$. Since elements in $\mathbb{F}_{p^{12}}$ are represented as a polynomial of the form $a_5z^5 + a_4z^4 + a_3z^3 + a_2z^2 + a_1z + a_0$, the map is indeed efficient and avoids any multiplications.

Vercauteren [Ver10, Section 4] gives an optimal Ate pairing for BN curves. Let $P \in E(\mathbb{F}_p)$ and $Q \in E'(\mathbb{F}_{p^2})$, then the optimal Ate pairing over BN curves is defined as

$$a_{opt} = \Big(f_{6u+2,Q}(P) \cdot \ell_{[6u+2]Q,\pi_p(Q)}(P) \cdot \ell_{[6u+2]Q+\pi_p(Q),-\pi_{p^2}(Q)}(P) \Big)^{(q^{12}-1)/n}, \tag{3.9}$$

where u denotes the BN parameter and π_{p^i} the i-th frobenius endomorphism.

3.6. Conclusion

This chapter dealt with the mathematical concepts behind identity-based encryption schemes and should help to get a better feeling for the schemes presented in the next chapter. Initially, basic algebraic principles such as groups, fields and extension fields were reviewed. Based on these, groups of elliptic curves over finite fields were defined and investigated in terms of their structure. Following the introduction to bilinear pairings and

divisors, the ideas from algebra and elliptic curves were used to define the Tate pairing. A method to evaluate a function that is given by its divisor was given by Miller and optimal pairings were presented by Vercauteren. After a short overview on the construction of suitable elliptic curves, the elliptic curves by Barreto and Naehrig were presented in detail.

4. Identity-Based Encryption Schemes

As introduced in Chapter 2, identity-based encryption seems a promising alternative to conventional asymmetric cryptography. From the day Shamir [Sha84] proposed the concept, it took 17 years for a satisfactory solution to be published. In 2001, Boneh and Franklin [BF01] presented the first practical identity-based encryption scheme. It is based on a bilinear map, a so-called pairing. Following this publication, several more identity-based encryption schemes were invented. However, there are only a few constructions that are the centerpiece of all of them.

Drawbacks of these schemes involve complexity, performance and size. They differ enormously in these terms, and as the goal is to provide identity-based encryption in constrained environments, these criteria are some of the most important. Nevertheless, security must not be overseen. For that reason, a clear definition and understanding of security for IBE schemes is necessary. Since there are many schemes out there, the focus is on two schemes that were shortlisted during the process of research, that is, two schemes with promising properties for usage in constrained environments.

This Chapter starts with a definition of the different types of security in Section 4.1. Consecutively, a historical overview of IBE schemes is given in Section 4.2, leading to the concrete description of two interesting schemes in Section 4.3 and Section 4.4. These two schemes are compared in Section 4.5.

4.1. Security Defintion

In order to be able to assess the quality of a cryptographic scheme, it is important to have clear and precise definitions for their level of security. Especially security proofs make extensive use of these definitions. In the following, the definitions that are of relevance in context of identity-based encryption are explained.

Identity-based encryption schemes are typically proven secure by showing that they are equivalently hard as a problem that is known or thought to be hard. As identity-based encryption schemes usually involve the computation of bilinear maps, these schemes are reduced to the Bilinear Diffie-Hellman (BDH) problem or one of its variants. These problems are thought to be sufficiently hard to solve for an attacker. Depending on the quality of the proof, it introduces an additive or multiplicative factor of security loss, that is, given a certain security level of the underlying problem, the identity-based encryption scheme is by a certain amount less secure. In other words, to attain a desired security level in the identity-based encryption scheme, a security level correspondingly higher has to be chosen for the underlying hard problem. A security proof is called tight if the introduced loss of security is small or can be neglected.

Security proofs can be done using different models. Initially, identity-based encryption schemes were proven secure in the *random oracle model*, which is due to Bennett and Gill [BG81]. According to Chatterjee and Sarkar [CS11], these proofs rely on hash functions being ideal random oracles, that is, all of these functions are independent and have uniformly distributed random output. In reality, there is no hash function that is as ideal as a

random oracle. Therefore, there is clear doubt that schemes proven secure in the random oracle model stay secure in the real world. Hence, the main goal is to create security proofs in the *standard model*, which do not rely on the output of random oracles.

In the context of identity-based encryption, the level of proven security depends on the freedom an attacker is adjudicated. In this matter, we distinguish between security against *Chosen-Plaintext Attacks* (CPA) and security against *Chosen-Ciphertext Attacks* (CCA). Depending on whether the attacker has to confine to a certain identity in advance or not, two different notions of security, namely security in the *selective-identity model* and security in the *adaptive-identity model*, are defined.

4.1.1. Chosen Plaintext Attack (CPA) Security

Chatterjee and Sarkar [CS11, Section 2.3.2] define security against chosen-plaintext attacks by playing a game. Involved in this game are an adversary $\mathcal{A}$, an identity-based encryption scheme $\mathcal{S}_{IBE}$ and a challenger. Further, there exists a key-extraction oracle $\mathcal{O}_k$. The game consists of the following phases:

Setup. Given a security parameter κ, the challenger runs the Setup algorithm of the IBE $\mathcal{S}_{IBE}$, consecutively returns the public parameters to the adversary $\mathcal{A}$ and keeps the master secret to itself.

Query Phase 1. The adversary $\mathcal{A}$ performs a finite number of key-extraction queries to the oracle $\mathcal{O}_k$, which returns the correct private key given a certain identity. The adversary is allowed to adaptively choose the identities to be queried. As the Derive algorithm is typically probabilistic, the algorithm returns different private keys if it is invoked several times for the same identity.

Challenge. The adversary $\mathcal{A}$ creates two messages M_0, M_1 of equal length and decides for an identity id that has not been queried during phase 1. Additionally, prefixes of the chosen identity id must not have been queried. Consecutively, the challenger randomly picks one of the two messages M_0, M_1, encrypts it, and returns the corresponding ciphertext C to the adversary $\mathcal{A}$.

Query Phase 2. Similar as in query phase 1, the adversary $\mathcal{A}$ adaptively performs key-extraction queries for different identities. However, the identity picked during the challenge phase must not be used for such a query.

Guess. At this stage, the adversary makes a guess about which of the two messages M_0, M_1 was used to create the ciphertext C.

Consider $p_{correct}$ being the probability of the adversary making a correct guess. Then, the advantage of the adversary is defined as

$$\mathbf{Adv}_{\mathcal{A}}^{\mathcal{S}_{IBE}} = \left| p_{correct} - \frac{1}{2} \right|.$$

If any t-time adversary $\mathcal{A}$ that makes at most q private key-extraction queries has an advantage $\mathbf{Adv}_{\mathcal{A}}^{\mathcal{S}_{IBE}} \leq \epsilon$, an IBE is said to be (t, q, ϵ)-secure against an adaptive chosen plaintext attack, or simply chosen plaintext attack secure. This is also labeled `IND-CPA` security, which means that the IBE is indistinguishable under chosen plaintext attacks.

4.1.2. Chosen Ciphertext Attack (CCA) Security

Similar to security against chosen plaintext attacks in Section 4.1.1, Chatterjee and Sarkar [CS11, Section 2.3.1] define security against chosen ciphertext attacks playing a game. Again, there is an adversary $\mathcal{A}$, an identity-based encryption scheme $\mathcal{S}_{IBE}$ and a challenger. In addition to the key-extraction oracle $\mathcal{O}_k$, a decryption oracle $\mathcal{O}_d$ exists. The game is defined as follows:

Setup. Given a security parameter κ, the challenger runs the Setup algorithm of the IBE $\mathcal{S}_{IBE}$, consecutively returns the public parameters to the adversary $\mathcal{A}$ and keeps the master secret to itself.

Query Phase 1. The adversary $\mathcal{A}$ performs a finite number of queries to the two oracles $\mathcal{O}_k$ and $\mathcal{O}_d$. Key-extraction queries are placed to the key-extraction oracle $\mathcal{O}_k$, which returns the correct private key given a certain identity. Decryption queries are done using the decryption oracle $\mathcal{O}_d$, which returns the correct plaintext or information that the ciphertext cannot be decrypted. The adversary is allowed to adaptively choose the identities to be queried.

Challenge. The adversary $\mathcal{A}$ creates two messages M_0, M_1 of equal length and decides for an identity id that has not been used in a key-extraction query during phase 1. Additionally, prefixes of the chosen identity id must not have been queried. Consecutively, the challenger randomly picks one of the two messages M_0, M_1, encrypts it, and returns the corresponding ciphertext C to the adversary $\mathcal{A}$.

Query Phase 2. Similar as in query phase 1, the adversary $\mathcal{A}$ adaptively performs key-extraction and decryption queries for different identities. However, the identity picked during the challenge phase must not be used for key-extraction queries and the decryption oracle must not be invoked for the ciphertext C and the previously chosen identity id.

Guess. At this stage, the adversary makes a guess about which of the two messages M_0, M_1 was used to create the ciphertext C.

If the adversary succeeds with probability $p_{correct}$, the advantage of the adversary to successfully attack the scheme is defined as

$$\mathbf{Adv}_{\mathcal{A}}^{\mathcal{S}_{IBE}} = \left| p_{correct} - \frac{1}{2} \right| .$$

Consequently, if any t-time adversary $\mathcal{A}$ that makes at most q_k private key-extraction queries and at most q_d decryption queries has an advantage $\mathbf{Adv}_{\mathcal{A}}^{\mathcal{S}_{IBE}} \leq \epsilon$, an IBE is said to be ($t$, q_k, q_d, ϵ)-secure against an adaptive chosen ciphertext attack, or simply chosen ciphertext attack (`IND-CCA`) secure.

4.1.3. Selective-Identity and Adaptive-Identity Models

Chatterjee and Sarkar [CS11, Section 2.3.3] point out that two different notions of security are relevant in context of identity-based encryption:

Adaptive-identity model This is the standard definition of security. As defined in Sections 4.1.1-4.1.2, the adversary chooses the identity to be attacked during the Challenge stage, that is, the adversary can gain arbitrary much knowledge about the system using key-extraction and decryption queries prior to settling down for an identity to attack.

Selective-identity model In this model, the adversary has to choose the identity to be attacked at the very beginning, even before the system parameters are generated.

Obviously, security in the selective-identity model is a weaker notion of security than it is in the adaptive-identity model. It neglects the fact that an attacker could learn something about the secrets of *any* identity using the knowledge gained from other identities. As the identity needs to be fixed prior to the setup of the system, Chatterjee and Sarkar [CS11, Section 5.3] conclude that the concrete setup might depend on the identity being under attack. Consequently, security for a certain identity might be ensured by choosing an appropriate setup of the system. In other words, the selective-identity security model ensures that one particular identity is secured by the setup of the system, blinding out the many others. The security model does not tell anything about the security of them, meaning, it is not assured. However, it does not say that there is a security problem with these identities.

4.2. (Historic) Overview

The problem of Identity-Based Encryption (IBE) was first posed by Shamir [Sha84] in 1984. It took until 2001 for the first satisfying solution to be presented by Boneh and Franklin [BF01]. It was proven to be adaptive-identity `IND-CCA` secure in the random oracle model. Following the ideas of Katz and Wang [KW03], Attrapadung et al. [Att+05] adapted the Boneh-Franklin scheme, which allowed them to present a tighter security reduction. Gentry and Silverberg [GS02] published the first Hierarchical Identity-Based Encryption (HIBE) scheme by modifying the scheme from Boneh and Franklin.

In 2003, Sakai and Kasahara [SK03] published an identity-based encryption scheme with clear advantages in performance, but it took until 2005 for a proof of security in the random oracle model to appear. Unfortunately, the reduction in the security proof by Chen and Cheng [CC05] is not tight at all. Based on the Sakai-Kasahara scheme, an IBE Key Encapsulation Mechanism (KEM) was proposed by Chen et al. [Che+06]. Unluckily, the proof for adaptive-identity `IND-CCA` security is based on a very strong hardness assumption.

Boneh and Boyen [BB04a] presented efficient identity-based encryption schemes that are selective-identity `IND-CPA` secure in the standard model and adaptive-identity `IND-CCA` secure in the random oracle model. Canetti, Halevi, and Katz [CHK03, Section 2.2] showed a generic method that enabled transformation of the Boneh-Boyen schemes to schemes that are selective-identity `IND-CCA` secure in the standard model. Moreover, several methods were presented that can be used to transform a (l+1)-layer HIBE that is `IND-CPA` secure into a l-layer HIBE that is `IND-CCA` secure: Ran Canetti and Katz [RCK03] achieved this by appending a one-time signature to the ciphertext, while Boneh and Katz [BK04] attached a Message Authentication Code (MAC) instead. The same transformation approach can be pursued using the results from Boyen, Mei, and Waters [BMW05], which exploits features of the ciphertext in the BB_1 scheme by Boneh and Boyen [BB04a]. Contrary to the former two transformations, Boyen, Mei, and Waters avoided using MACs

and signatures. Additionally, the BB_1 scheme was modified to attain a selective-identity IND-CCA secure IBE KEM. Among others, for example the Sakai-Kasahara and the Boneh-Franklin schemes, the BB_1 scheme and its KEM variant were proposed for the future IEEE P1363.3 standard, which will cover identity-based encryption.

Up to that point, there was no adaptive-identity IND-CCA secure IBE in the standard model. Boneh and Boyen [BB04b] were the first to provide a solution to this problem. However, this solution is not practical as it does not perform well as the number of identities increases. This changed with the results by Waters [Wat05]. In this publication, the first efficient IBE that is fully secure in the standard model was presented. To attain this, Waters did a slight modification of the BB_1 scheme by Boneh and Boyen to use a special hash function. The drawback of his solution is the vast memory footprint that results from the modified hash function: if identities are represented by bit strings of length k, it takes k elements from the group $\mathbb{G}$ as public parameters to define the hash function. This is shown in detail in Section 4.4.1. The number of elements can be reduced massively by a factor l as independently presented by Naccache [Nac05] and Chatterjee and Sarkar [CS06]. This reduces the number of public parameters to be stored and speeds up the evaluation of Water's hash function. The price to pay is a security degradation by the same factor as pointed out in Section 4.4.6.

Another practical IBE that was proven to be fully secure was published by Gentry [Gen06]. However, its proof relies on a tight reduction to a strong assumption that depends on the number of queries made by an attacker (truncated q-ABDHE). Therefore, it is not safe to say that this necessarily means more security as already shown by Cheon [Che06].

Based on Water's scheme, Kiltz and Galindo [KG06] presented an adaptive-identity IND-CCA secure IBE KEM. Similarly, Water's and Gentry's schemes were adapted by Kiltz and Vahlis [KV08] using techniques of authenticated symmetric encryption to speed up their schemes as well as to provide full security proofs.

A new approach, called dual system encryption, was presented by Waters [Wat09]. Security proofs for these IBE schemes significantly differ from the previous and allow both tight security reductions and compact public parameters while attaining full adaptive-identity security. However, these schemes are considerably slower.

In consequence of analyzing the world of identity-based encryption schemes, two of them were shortlisted due to their properties: the BB_1 IBE in its KEM variant [IEE08] augurs good performance, is contained in the draft for the future IEEE P1363.3 standard, has good security proofs, and is used in practice. Secondly, the IBE KEM by Kiltz [Kil06] made a good impression due to its full IND-CCA security and its relatively good performance. Section 4.3 and Section 4.4 explain their details.

4.3. Boneh-Boyen IBE-KEM

The IBE by Boneh and Boyen [BB04a] has gained much attention and is clearly one of the most efficient IBE schemes. It is proposed for the future IEEE P1363.3 standard [Boy06]. For this purpose, also a KEM version was published. This is clearly favorable as Chapter 2 suggests: when the involved parties have exchanged their session key using the KEM, a Data Encapsulation Mechanism (DEM) that is based on an efficient block cipher can be used for transferring the real message. In the following, the scheme is described as by Boyen [Boy06, Section 5.2]:

4.3.1. Prerequisites

For the system to work, two groups ($\mathbb{G}$, $\hat{\mathbb{G}}$) of prime order n, suitable for a bilinear map $e : \mathbb{G} \times \hat{\mathbb{G}} \longrightarrow \mathbb{G}_t$, are needed. Their corresponding generators are G and $\hat{G}$. Additionally, two cryptographic hash functions are needed:

1. $H : \{0,1\}^* \longrightarrow \mathbb{Z}_n$ to hash the recipients identity, a string of arbitrary length, to a fixed-range number.
2. $H' : \mathbb{G}_t \longrightarrow \{0,1\}^\ell$ to hash an element of the target group to fixed size (ℓ bits) to obtain a session key.

4.3.2. Setup

The system is set up by the following procedure:

1. Randomly pick three integers α, β, and $\gamma \in \mathbb{Z}_n$.
2. Calculate $G_1 = \alpha \cdot G$ and $G_3 = \gamma \cdot G \in \mathbb{G}$.
3. Compute $\hat{G}_0 = \alpha \cdot \beta \cdot \hat{G} \in \hat{\mathbb{G}}$ and then $v_0 = e(G, \hat{G}_0) \in \mathbb{G}_t$.

When these computations are done, the public parameters and the master secret are defined as follows:

- Public parameters: $P = (G, G_1, G_3, v_0) \in \mathbb{G}^3 \times \mathbb{G}_t$.
- Master secret: $K_{master} = (\hat{G}, \alpha, \beta, \gamma) \in \hat{\mathbb{G}} \times \mathbb{Z}_n^3$.

The identity-based encryption scheme is then ready to be used.

4.3.3. Derive

To derive a private key $D_{id} = (D_{id,0}, D_{id,1}) \in \hat{\mathbb{G}} \times \hat{\mathbb{G}}$ for a certain identity *id*, the following routine needs to be done:

1. Choose a random value $r \in \mathbb{Z}_n$.
2. Calculate $D_{id,0} = (\alpha\beta + (\alpha H(id) + \gamma) \cdot r) \cdot \hat{G}$.
3. Compute $D_{id,1} = r \cdot \hat{G}$.

As it can be seen, the private key is randomized. Hence, if an identity invokes Derive several times, different private keys that can be used equivalently for decapsulation are returned.

4.3.4. Encapsulate

Encapsulation pursues two different goals: creating a random session key and encapsulating it in a ciphertext $C = (C_0, C_1) \in \mathbb{G} \times \mathbb{G}$ decryptable by a certain identity *id*. Boyen [Boy06] states this process as follows:

1. Pick a random value $s \in \mathbb{Z}_n$.

Table 4.1.: Configurations of the IBE scheme by Kiltz [Kil06] for asymmetric pairings. A trade-off between speed and ciphertext size can be done.

Variant	Ciphertext space	Ciphertext size	Encryption	Decryption
V1	$\hat{\mathbb{G}} \times \hat{\mathbb{G}}$	big	slow	fast
V2	$\mathbb{G} \times \mathbb{G}$	small	fast	slow
V3	$\hat{\mathbb{G}} \times \mathbb{G}$	big	medium	fast

2. Generate the random session key $K = H'(v_0^s) \in \{0,1\}^\ell$.
3. Compute $C_0 = s \cdot G$.
4. Calculate $C_1 = s \cdot G_3 + (s \cdot H(id)) \cdot G_1$.

4.3.5. Decapsulate

Decapsulating the ciphertext $C = (C_0, C_1) \in \mathbb{G} \times \mathbb{G}$ given the private key $D_{id} = (D_{id,0}, D_{id,1}) \in \hat{\mathbb{G}} \times \hat{\mathbb{G}}$ of the identity id consists of solely calculating

$$K = H'(e(C_0, D_{id,0})/e(C_1, D_{id,1})) \in \{0,1\}^\ell.$$

4.3.6. Security

The scheme by Boyen [Boy06] is proven to be secure against selective-identity chosen-ciphertext attacks in the standard model (`sID-IND-CCA` security). At the same time, there is a proof for security against adaptive-identity chosen-ciphertext attacks using random oracles (`IND-CCA` security). The scheme is based on "commutative blinding". According to Boyen [Boy06, A.2.1], security proofs for schemes like this make use of random oracles, but the schemes can also achieve full security in the standard model.

Security is proven by assuming security of the decisional Bilinear Diffie-Hellman (BDH) problem, which is the weakest of all BDH assumptions. Consequently, the attack by Cheon [Che06] does not pose a problem for the scheme. Considering that at most q private key-extraction queries can be made by an adversary, security of the scheme reduces by a factor of q, respectively $log_2\, q$ bits, relative to the security of underlying BDH problem. A typical assumption for q is 2^{30}. Additionally, the security of this scheme relies on a data encapsulation mechanism with integrity checks. For that purpose, a MAC can be used.

4.4. Kiltz IBE-KEM

Analogously to Boyen [Boy06] in Section 4.3, Kiltz [Kil06] presents an identity-based encryption KEM-DEM construction that is based on the previous work by Kiltz and Galindo [KG06], which merely comprises a KEM part. Viewing the system as a full KEM-DEM has some advantages with respect to performance as the integrity check is delegated to the DEM part. The original paper states the scheme in a setting usable for symmetric pairings. However, Kiltz [Kil06, Section 7.2] points out the advantage in performance and ciphertext size when utilizing asymmetric pairings and proposes three different configurations in the asymmetric setting, namely V1,V2 and V3, shown in Table 4.1.

As small ciphertexts and fast encryption seem favorable for constrained environments, setting V2 was chosen and is presented in the following.

4.4.1. Prerequisites

Two groups $(\mathbb{G}, \hat{\mathbb{G}})$ of prime order n that are suitable for a bilinear map $e : \mathbb{G} \times \hat{\mathbb{G}} \longrightarrow \mathbb{G}_t$ are needed. Their corresponding generators are G and $\hat{G}$.

For the system to work, a special hash function, first introduced by Waters [Wat05], is needed. Let this hash function be denoted H_{id} as it is used for hashing an identity string id of fixed length k to $\mathbb{G}$. Given $k+1$ random elements $\mathbf{H} = \{H_0, H_1, ..., H_{k-1}, H_k\} \in \mathbb{G}^{k+1}$, the hash function is defined as

$$H(id) = H_0 + \sum_{i=1}^{k} id_i \cdot H_i,$$

where id_i denotes the i-th bit of the identity string id. Accordingly, the elements $\hat{\mathbf{H}} = \{\hat{H}_0, \hat{H}_1, ..., \hat{H}_{k-1}, \hat{H}_k\} \in \hat{\mathbb{G}}^{k+1}$ are used to define a second hash function $\hat{H}(id)$ of the same type.

Additionally, a target collision resistant hash function $TCR : \mathbb{G} \to \mathbb{Z}_n$ is needed. Boyen, Mei, and Waters [BMW05, Appendix C] give an efficient way to built such a function for bilinear maps defined over elliptic curves. However, a common cryptographic hash function may be used instead.

4.4.2. Setup

The setup of the system is done by the following routine:

1. Randomly pick two integers α and $u \in \mathbb{Z}_n$.
2. Calculate $\hat{A} = \alpha \cdot \hat{G}$, $U = u \cdot G$, and $\hat{U} = u \cdot \hat{G}$.
3. Prepare the constants for Waters' hash:
   a) Generate $k+1$ random numbers $t_0, t_1, ..., t_{k-1}, t_k \in \mathbb{Z}_n$.
   b) Calculate $\mathbf{H} = \{H_0, H_1, ..., H_{k-1}, H_k\} = \{t_0 G, t_1 G, ..., t_{k-1} G, t_k G\} \in \mathbb{G}^{k+1}$.
   c) Compute $\hat{\mathbf{H}} = \{\hat{H}_0, \hat{H}_1, ..., \hat{H}_{k-1}, \hat{H}_k\} = \{t_0 \hat{G}, t_1 \hat{G}, ..., t_{k-1} \hat{G}, t_k \hat{G}\} \in \hat{\mathbb{G}}^{k+1}$.
4. Prepare the parameter $z = e(G, \hat{A})$.

Consequently, the IBE scheme is ready-to-use and comprises the following parameters:

- Public parameters: $P = (\mathbf{H}, \hat{\mathbf{H}}, U, \hat{U}, z) \in \mathbb{G}^{k+1} \times \hat{\mathbb{G}}^{k+1} \times \mathbb{G} \times \hat{\mathbb{G}} \times \mathbb{G}_t$.
- Master secret: $K_{master} = (\hat{A}) \in \hat{\mathbb{G}}$

4.4.3. Derive

Given a certain identity id, the subsequent steps need to be performed to derive its private key $D_{id} = (D_{id,0}, D_{id,1}, D_{id,2}) \in \hat{\mathbb{G}}^3$.

1. Randomly pick a number $s \in \mathbb{Z}_n$.

2. Compute $D_{id,0} = \hat{A} + s \cdot \hat{H}_{id}(id)$.
3. Calculate $D_{id,1} = s \cdot \hat{G}$.
4. Compute $D_{id,2} = s \cdot \hat{U}$.

As also stated for the IBE by Boneh and Boyen [BB04a], the process of key derivation is randomized.

4.4.4. Encapsulate

The encapsulation routine creates a random session key and encapsulates it in a ciphertext $C = (C_0, C_1) \in \mathbb{G} \times \mathbb{G}$ that is only decryptable by a certain identity *id*. According to Kiltz [Kil06], this is done as follows:

1. Pick a random number $r \in \mathbb{Z}_n$.
2. Generate the random session key $K = z^r \in \mathbb{G}_t$.
3. Calculate $C_0 = r \cdot G$.
4. Derive a parameter $t = TCR(C_0)$.
5. Compute $C_1 = r \cdot (t \cdot U + H_{id}(id))$.

4.4.5. Decapsulate

Decapsulating the ciphertext $C = (C_0, C_1) \in \mathbb{G} \times \mathbb{G}$ given the private key $D_{id} = (D_{id,0}, D_{id,1}, D_{id,2}) \in \hat{\mathbb{G}}^3$ of the identity *id* is done by the following steps:

1. Derive the parameter $t = TCR(C_0)$.
2. Randomly pick a number $v \in \mathbb{Z}_n$.
3. Compute $\hat{X}_0 = D_{id,0} + t \cdot D_{id,2} + v \cdot (\hat{H}_{id}(id) + t \cdot \hat{U})$.
4. Calculate $\hat{X}_1 = v \cdot \hat{G} + D_{id,1}$.
5. Compute the session key $K = e(C_0, \hat{X}_0)/e(C_1, \hat{X}_1) \in \mathbb{G}_t$.

4.4.6. Security

For the scheme by Kiltz [Kil06], security against adaptive-identity chosen-ciphertext attacks is proven in the standard model (`IND-CCA` security). Security reduces to a modified Decisional Bilinear Diffie-Hellman (mBDDH) problem, which is almost the same as the original decisional BDH. In contrast to the latter, the mBDDH offers an additional term $b^2 \cdot G$ to be used to solve the decisional BDH. This term is generally hard to compute given $b \cdot G$. As this assumption is indeed very similar, the attack by Cheon [Che06] should not become viable. Considering that at most q private key-extraction queries can be made by an adversary and the length of the identity strings is k bits, security of the scheme reduces by a factor of $q \cdot 2^k$ relative to the security of underlying mBDDH problem, which is equivalent to $log_2\, q + k$ bits.

As this scheme obviously comes with a large set of public parameters, it might be desirable to reduce its size. As mentioned in Section 4.2, Naccache [Nac05] presented a solution to this problem, which reduces the number of elements needed for Waters' hash by a factor of ℓ. For that purpose, the k-bit long identity strings are reinterpreted as $v = (v_1, ...v_{k'})$. Each v_i is ℓ bits long. Then, the hash is computed as follows:

$$H(ID) = H_0 + \sum_{i=1}^{k'} v_i \cdot H_i$$

Consequently, less elements are needed as public parameters and the evaluation of the hash function becomes faster. However, security also reduces by another factor of 2^ℓ, leading to an overall loss of $2^\ell q 2^k$.

According to Bentahar et al. [Ben+08], a hybrid encryption scheme is secure against chosen ciphertext attacks if both KEM and DEM are secure against chosen-ciphertext attacks. Secure DEMs can be built from a symmetric encryption scheme by adding a MAC. Kiltz [Kil06, Section 5] states that the overhead of a MAC can be avoided by using the CMC [HR03] or EMC [HR04] modes of operation. Deriving random session keys of appropriate size as well as using identity strings of arbitrary length need additional hash functions, which may have impact on the proof and type of security.

4.5. Comparison

Sections 4.3 and 4.4 introduced the KEM-DEM configuration two different identity-based encryption schemes in more detail. To be able to decide on which scheme to use, a comparison of these two schemes based on several characteristics seems suitable: performance, memory, and security.

Table 4.2 gives an overview on the number of operations involved in the four basic algorithms Setup, Derive, Encapsulate, and Decapsulate for the two mentioned schemes. Typically, operations in $\mathbb{G}$ are the fastest, followed by operations in $\hat{\mathbb{G}}$. Also, point additions in these groups are generally significantly faster than their multiplication counterparts. Depending on the concrete definition of $\mathbb{G}$, $\hat{\mathbb{G}}$ and the bilinear map e, exponentiations in $\mathbb{G}_t$ are often more expensive than the computation of a pairing. In addition, pairings are typically slower than a multiplication in $\hat{\mathbb{G}}$. Ratios of pairings are assumed to take 50% longer than a simple pairing as the algorithm allows some optimizations in this case.

As it can be seen, the routines Derive and especially Setup involve more operations for the scheme by Kiltz, that is, the two routines are slower in this case. However, these are performed at the trusted third party. There, performance is not as critical as it is for the Encapsulate and Decapsulate routines that are executed in the constrained environment. Additionally, Setup is only executed once. Hence, these performance-related drawbacks are not of great importance for our purpose. Considering Encapsulate, the two schemes differ only in a multiplication in $\mathbb{G}$, giving the scheme by Boyen a small advantage. However, the dominant factor for Encapsulate seems the exponentiation in $\mathbb{G}_t$. A different view reveals when investigating the Decapsulate routine. Boyen's scheme simply calculates a ratio of two pairings, while Kiltz's scheme additionally takes several additions and multiplications in $\hat{\mathbb{G}}$. This has significant impact on runtime which might not be tolerable on devices already lacking performance.

In terms of memory footprint, a look at the number of elements in each of the groups $\mathbb{G}$, $\hat{\mathbb{G}}$, and $\mathbb{G}_t$ is necessary. Elements in $\mathbb{G}$ are the smallest, while elements in $\hat{\mathbb{G}}$ and $\mathbb{G}_t$

Table 4.2.: Overview of the relevant operation counts involved in the two considered schemes.

	$\mathbb{G}$		$\hat{\mathbb{G}}$		$\mathbb{G}_t$	$\mathbb{G} \times \hat{\mathbb{G}}$
	Add	Mul	Add	Mul	Exp	Pairing
Boyen [Boy06]						
Setup		2		1		1
Derive				2[a]		
Encapsulate	1	3			1	
Decapsulate						1.5
Kiltz [Kil06]						
Setup		k+1[b]		k+3 [b]		1
Derive			1	4[c]		
Encapsulate	1	4[c]			1	
Decapsulate			4	5[c]		1.5

[a] Two consecutive point multiplications are simplified to one by multiplying the scalar values modulo n.
[b] k denotes the fixed length of the identity strings, for example, k=8 if 256 identities should be supported.
[c] Hashing the identity is counted as one multiplication.

are significantly larger. Their actual sizes depend heavily on the concrete definition of the groups and the bilinear map being used. For the BN curves from Section 3.5.1, elements in $\hat{\mathbb{G}}$ and $\mathbb{G}_t$ are respectively twice and six times larger than elements in $\mathbb{G}$. An overview of the number of elements needed in the considered schemes is given in Table 4.3. The ciphertext is small in both cases, which is good to keep communication overhead low. The private keys of the scheme by Kiltz are larger by a third, which does not seem a big issue. The master secret, however, is clearly bigger for the scheme by Boyen. The main drawback of Kiltz's scheme is the size of its public parameters. While this may not be a problem for a high-level application, it is clearly unfavorable for systems designed to be as small as possible.

The last important aspect to look at is security. Both schemes are designed as hybrid KEM-DEM construction and rely on an integrity check by the DEM, which seems legitimate. Security proofs assume hardness of (a slight variation of) the decisional bilinear diffie-hellman problem for both of the schemes, avoiding exposure to the attack presented by Cheon [Che06]. However, the type of security assured differs: Kiltz [Kil06] offers full adaptive-identity `IND-CCA` security in the standard model, while Boyen [Boy06] can offer this only using random oracles. Nevertheless, in the standard model it still achieves selective-identity `IND-CCA` security. The two schemes have a common basis as the scheme by Kiltz has evolved from Boyen's scheme. Their difference that is reflected in their types of security is also seen in the security loss introduced by the schemes. Relative to the underlying hard problem, Kiltz's scheme introduces an additional loss of k bits compared to Boyen, where k denotes the length of the identity strings involved. Chatterjee and Sarkar [CS11, Section 5.3] points out, that this seems to be the price to be paid for assuring security for adaptive identities. Accordingly, this factor further suggests that even the weakest notion of security in the adaptive-identity model offers more security than strongest notion of security in the selective-identity model. The security loss in Kiltz's scheme is even increased if the hash function being used is adapted to reduce the number of elements in the public parameters as shown by Naccache [Nac05].

Table 4.3.: Overview of the sizes of keys, parameters and ciphertexts in the two considered schemes.

	$\mathbb{Z}_n$	$\mathbb{G}$	$\hat{\mathbb{G}}$	$\mathbb{G}_t$
Boyen [Boy06]				
Ciphertext		2		
Public parameters		3		1
Private key			2	
Master secret	3		1	
Kiltz [Kil06]				
Ciphertext		2		
Public parameters		k+2[a]	k+2[a]	1
Private key			3	
Master secret			1	

[a] k denotes the fixed length of the identity strings.

For our goal of establishing identity-based encryption in constrained environments, runtime and memory footprint of both constants and variables seem the most important aspects. In addition, the previous discussion on the security of these schemes suggests that their difference in these terms is merely a trade-off between assured security, memory, and runtime. As a consequence of both, we decided to implement the scheme by Boyen. Additionally, this scheme is contained by the draft of the future IEEE P1363.3 standard on identity-based encryption and is used in practice by Voltage Security [Vol13], which also adds a slight advantage to the scheme.

4.6. Conclusion

This chapter focused on the various forms of identity-based encryption schemes. For understanding their differences, the several forms of security for identity-based encryption schemes were defined. In a chosen-plaintext attack, an adversary is allowed to do key-extraction queries, but cannot make decryption queries to a corresponding oracle prior to making a guess about the message being encrypted. In a chosen-ciphertext attack, an adversary can additionally make decryption queries. In a selective-identity model, the adversary has to define the identity to be attacked prior to system setup, while this is done in an adaptive-identity model when the challenge is issued. Further, some of the schemes rely on random oracles, which do not exist in reality. For that reason, security proofs in the standard model are preferred.

Following this, an overview on the existing IBEs was given. Two of them [Boy06; Kil06] were discussed in more detail due to their interesting properties. Favorably, both offer small ciphertext sizes. Their sizes are according to Kiltz [Kil06, Section 5] comparable to the most efficient public-key encryption schemes secure in the standard model. However, their detailed comparison shows clear advantages for the scheme by Boyen [Boy06] in constrained environments as a consequence of expected runtime and public parameter size. For that reason, and despite its weaker notion of security, it is chosen for the target platform to be implemented.

5. Side-Channel Attacks

The main goal of this thesis is to deploy an identity-based encryption scheme to an embedded platform. Whenever cryptographic applications are established in such an environment, one has to consider physical attacks too. In the embedded scenario, attackers may have full access to the device they wish to attack, and consequently may do *anything* they like with it in order to compromise its security. In this respect, security of the cryptographic algorithm is widened by the facet of secrecy of the used keys.

To be able to protect the used keys in the identity-based encryption scheme, the most relevant types of physical attacks are reviewed in this Chapter. Based on this, the goals for physical security can be defined. Section 5.1 gives a categorization of physical attacks. Passive attacks are covered in Section 5.2 and one particularly interesting active attack is shown in Section 5.3.

5.1. Overview

Physical attacks can be distinguished by two criteria. Based on the first criteria, Mangard, Oswald, and Popp [MOP07] state the following types of attacks:

- Passive Attacks: The device is operated within its specification, for example, analysis of power consumption, electromagnetic emission, or execution time.
- Active Attacks: The device is forced to behave abnormally by manipulating the device itself, its input, or its environment. The goal is to exploit the abnormal behavior to reveal the key.

Based on the second criteria, each attack can be considered being one of the following:

- Invasive Attacks: In this case, there are no limits to what is done with the device in order to reveal its key. This includes depackaging and directly accessing single components of the device using for example a probing station.
- Semi-Invasive Attacks: As in invasive attacks, the device is depackaged. However, there is no direct electrical contact made during the attack, for example, optical attacks.
- Non-Invasive Attacks: In this type of attack, the device is kept as it is and only the directly accessible interfaces are used.

Combining the two criteria, *side-channel attacks* are defined as passive, non-invasive attacks. This means that the device is kept as it is and is only accessed through external interfaces. Additionally, no faults are induced to reveal the key. In a similar manner, *fault analysis* is defined. This type of attack is always active as the goal is to exploit the induced abnormal behavior of the cryptographic device. These attacks can be invasive, semi-invasive, or non-invasive. For example, if faults are induced by clock or power

glitches, the attack is non-invasive. The attack is semi-invasive if depackaging is necessary, for example, if x-rays or light flashes are used to alter single bits in memory cells. If faults are induced using an electric probe, the attack is invasive.

In the context of this thesis, the one goal is to secure the application against side-channel attacks. The remaining of this chapter shall give an overview on the most common. For a more comprehensive overview the reader may have a look at the survey by Fan and Verbauwhede [FV12].

5.2. Passive Attacks

5.2.1. Timing Attacks

The concept of *timing attacks* was first presented by Kocher [Koc96]. Here, the fact that the runtime of the implementation of the cryptographic algorithm depends on data and key is exploited. The reasons for this are branching, conditional statements, and performance optimizations on a higher level. On a lower level, instructions performed by the CPU, for example, multiplications, may have a data-dependent runtime.

For example, if a certain bit of the key is set, the cryptographic algorithm's runtime may be longer than otherwise. Similarly, the algorithm's runtime may vary for different data inputs. Consequently, for a given key hypothesis, the measured runtimes for different inputs can be separated into two sets according to the expected runtime. If the means of those two sets are very close, the key hypothesis was very probably wrong, otherwise right. Such an attack can be easily counteracted by adapting all algorithms to have constant runtime. An example for a routine that has runtime dependent on a secret value k is given with Algorithm 1. A simple countermeasure that yields constant runtime is shown in Algorithm 2.

5.2.2. Simple Power Analysis

In *Simple Power Analysis* (SPA) attacks, the goal is to reveal the key of a cryptographic device by measuring its power consumption during one or a small number of runs. The cryptographic algorithm might be performed with the same input or with different inputs. When performing several runs using the same input, noise can be reduced by calculating the mean. The basic idea behind SPA attacks is to find key-dependent patterns within the power traces.

One tries is to derive the key from the power trace by analyzing it visually. It is based on the fact, that different operations and instructions during the execution of the cryptographic algorithm cause different patterns in the power trace, for example, instructions involving memory may take more clock cycles than others. If the sequence of instructions depends on the key and if the sequence of instructions can be derived from the power trace, the key is leaked. This can be counteracted by yielding constant runtime, since such algorithms have regular power traces that do not leak information when inspecting them visually.

An example is given by the basic *square-and-multiply* exponentiation in Algorithm 1. The multiplication in Line 4 is only executed if the i-th bit of the secret exponent k is set. If the attacker is able to distinguish between squarings and multiplications in the power trace, the secret key is easily revealed. One simple countermeasure is to introduce a dummy operation as shown in Algorithm 2. This algorithm is simply called *square-and-*

Algorithm 1 Square-and-multiply exponentiation in $\mathbb{F}_p$.

Input: $g \in \mathbb{F}_p, k = (k_{t-1}, ..., k_1, k_0)_2 \in \mathbb{N}$
Output: $y = g^k \in \mathbb{F}_p$
1: $r \leftarrow 1$
2: **for** $i = t - 1$ downto 0 **do**
3: $\quad r \leftarrow r^2$
4: $\quad$ **if** $k_i = 1$ **then** $r \leftarrow r \cdot g$
5: **end for**
6: **return** r

multiply-always. Due to the multiplication on Line 7, runtime is constant if and only if $k_{t-1} = 1$. Execution of the algorithm always consists of the same sequence of squarings and multiplications, hence being not vulnerable to SPA attacks.

Algorithm 2 Square-and-multiply-always exponentiation in $\mathbb{F}_p$.

Input: $g \in \mathbb{F}_p, k = (k_{t-1}, ..., k_1, k_0)_2 \in \mathbb{N}$
Output: $y = g^k \in \mathbb{F}_p$
1: $r \leftarrow 1$
2: **for** $i = t - 1$ downto 0 **do**
3: $\quad r \leftarrow r^2$
4: $\quad$ **if** $k_i = 1$ **then**
5: $\quad\quad r \leftarrow r \cdot g$
6: $\quad$ **else**
7: $\quad\quad d \leftarrow r \cdot g$
8: $\quad$ **end if**
9: **end for**
10: **return** r

5.2.3. Template Attacks

Similar to SPA attacks, *template attacks* try to derive the secret key by creating only one or a small number of power traces from the cryptographic device that is under attack. Contrary to SPA attacks, the process consists of two steps. In the first step, templates are created. These templates are simply power traces created for each pair of key and data. In order to create these, the attacker must possess a device of the same type that is under the attackers full control. Otherwise, the attacker is not able to create power traces for all pairs of key and data. In the second step, the templates are used to figure out the key being used on the attacked device. For this purpose, a power trace is measured from the device under attack. For each of the templates the probability that the recorded power trace belongs to this template is calculated. The maximum probability then reveals the used key. The number of pairs of possible keys and data is huge. Therefore, usually templates for intermediate values, which are calculated from pairs of keys and data, or templates for power models are used. Note that template attacks are very powerful attacks, as the results by Herbst and Medwed [HM09] and Medwed and Oswald [MO09] show.

5.2.4. Differential Power Analysis

In *Differential Power Analysis* (DPA), the goal is to reveal the key of a cryptographic device by measuring a large number of power traces using different inputs. Its basic idea is to analyze the data-dependent power consumption of the device at a fixed moment of time based on a large number of power traces. For this purpose, a concrete intermediate value of the executed algorithm is chosen to be attacked. This value should both depend on the processed input data and the unknown key. For every combination of k possible keys and n different data inputs, the hypothetical intermediate value is computed and mapped to a power model, for example, the hamming weight model, which simply counts the number of bits set to one. Next, power traces of length t for the n different data inputs are recorded. In the last step, the hypothetical power-consumption is compared to the real power-consumption using statistical methods, for example, Pearson correlation. For each of the k keys, the correlation with each of the t samples is computed. High correlation results reveal the used key and the points of time, where the intermediate value is processed. The typical countermeasure is to make statistical analysis impossible by introducing randomness into the computation as shown by Coron [Cor99].

The *address-bit DPA* constitutes a special form of the DPA attack and was first shown by Itoh, Izu, and Takenaka [IIT03]. In many implementations, the register or memory addresses used depend on a bit of the secret k. This, for example, is a side-effect of masking addresses in order to avoid branches. Their attack showed that one can mount a successful DPA to distinguish whether data is read from or written to either of two memory locations, hence leaking information on the secret k. The attack still applies for implementations that randomize input data. One possible countermeasure is to randomize the secret k.

5.2.5. Comparative Side-Channel Attacks

Comparative side-channel attacks are a mixture of SPA and DPA attacks. Using a single or a few power traces, correlations between related values reveal the secret key. In particular, one can decide whether the same value is processed twice without knowing the actual value. One such attack was presented by Fouque and Valette [FV03], namely the doubling attack. In this attack, two elliptic curve point multiplications with a secret scalar k are done as $[k]P$ and $[k](2P)$. By comparing the patterns resulting from the doublings of the intermediate values in two point multiplication's power traces, the whole secret k can be recovered. This is possible, since it depends on the secret key whether the same doublings are performed for a simple *double-and-add* implementation (cf. Algorithm 1).

5.2.6. Refined Power Analysis

Goubin [Gou02] introduced the term of *refined power analysis*, which works for addition-chains in general, but was presented for elliptic curve point multiplications. In this attack, the point multiplication $[k]P$ is started with a point P that, under the correct hypothesis for the key bit k_i, leads to one of the special points $(x, 0)$ or $(0, y)$ in step i of the algorithm. The mean of several traces reveals whether the guess for the key bit k_i was correct. Since the respective coordinate also becomes zero at step i when the algorithm is randomized as originally suggested by Coron [Cor99], this attack also works for many secured implementations. The attack was extended to exploit zero values in arbitrary registers by Akishita and Takagi [AT03].

5.2.7. Electromagnetic Attacks

Electromagnetic attacks try to reveal the key by analyzing the electromagnetic emissions of the attacked device. They are basically equal to SPA and DPA attacks, but are referred to as *Simple Electromangetic Analysis* (SEMA) and *Differential Electromagnetic Analysis* (DEMA). In contrast to the former two attacks, not the power consumption is measured, but the electromagnetic emission, which is proportional to the power consumption of the device. An advantage of this attack is that the electromagnetic emission can also be measured from the distance. The attacker does not necessarily have to possess the device. Moreover, electromagnetic attacks can be mounted such that the origin of the electromagnetic emanation on the chip is additionally used to extract information, for example, one could find out about memory accesses if the radiation originates from the memory.

5.3. Active Attacks

5.3.1. Safe-Error Analysis

This particularly interesting active attack is both simple and powerful. Yen and Joye [YJ00][JY03] introduced the concept of *safe-error* analysis, which exploits the existence of unnecessary operations that were introduced in order to counteract SPA and timing attacks. There are two different types of safe-error analysis.

C safe-error analysis. An attacker can learn about the secret value that is being processed by inducing a fault during a specific operation and checking whether the final result is correct or not. Assume an attacker can induce a temporary fault in iteration i during the multiplication in Algorithm 2. If the final result is correct, the dummy operation on Line 7 was executed. If the final result is wrong, obviously the real operation on Line 5 was performed. Consequently, the attacker can learn one bit of the secret k after another simply by checking the correctness of the final result. Joye and Yen [JY03] showed that a *Montgomery ladder* can be used as a countermeasure. The algorithm presented in Algorithm 3 is based on a result by Montgomery [Mon87] and yields constant runtime. Instead of using dummy operations, it interleaves the computations of r_0 and r_1.

M safe-error analysis. Similarly to the C safe-error analysis, an attacker can learn about the secret value by inducing a fault into a specific memory location and checking whether the final result is correct or not. The idea behind this is, that memory locations with induced faults may be cleared and overwritten under conditions, that depend on a bit of the secret k. This kind of attack is more difficult to perform than the C safe-error attack.

5.4. Conclusion

An attacker in possession of a device performing cryptographic algorithms has many options to find out about the secret key being processed. The different types of attacks were initially categorized, which lead to the definition of side-channel attacks as passive, non-invasive attacks. Similarly, active attacks that try to exploit abnormal behavior of devices

Algorithm 3 Simple Montgomery ladder exponentiation in $\mathbb{F}_p$.

Input: $g \in \mathbb{F}_p, k = (k_{t-1}, ..., k_1, k_0)_2 \in \mathbb{N}$
Output: $y = g^k \in \mathbb{F}_p$
1: $r_0 \leftarrow 1$; $r_1 \leftarrow g$
2: **for** $i = t - 1$ downto 0 **do**
3: $\quad r_{\neg k_i} \leftarrow r_{k_i} \cdot r_{\neg k_i}$; $r_{k_i} \leftarrow r_{k_i}^2$
4: **end for**
5: **return** r_0

were named fault attacks. Several side-channel attacks, such as timing attacks, simple and differential power analysis attacks, were explained. These attacks are the easiest and hence the most important to secure an application against. Fault attacks, such as the presented safe-error attack, are in general very powerful, but also more difficult to perform than simple side-channel attacks. For this reason, we decided to protect the identity-based encryption scheme against the aforementioned variants of side-channel attacks.

6. Implementation in an Embedded Environment

In the previous chapters, an introduction to the concept of identity-based encryption and the underlying mathematical model was given. As indicated in the very beginning, these concepts were aimed to be brought into the embedded world. Concerning the implementation, several goals were initially defined:

- An efficient identity-based encryption scheme should become available for systems working in embedded environments.
- The identity-based encryption system should be resistant to the most common side-channel attacks, namely timing, simple power analysis and differential power analysis attacks.
- Realization of the underlying pairing computation should be state-of-the-art, that is, it should be competitive with regard to other good implementations.
- Correctness of the implementation shall be assured.

In terms of efficiency, the demand for resources, for example memory, should be low, because we interpret security as a feature rather than an application. On the other hand, short runtime is expected. These goals shall be achieved for both the hardware platform and the software implementation. However, and as it will show in the following, it is a demanding task to fulfill as the complexity is rather high for an additional feature in an embedded environment.

The architecture of the implemented system is covered in Section 6.1. The most important implementation aspects are part of Section 6.2. The approach to testing is pointed out in Section 6.3 and the various optimizations done are detailed in Section 6.4.

6.1. Architecture

For the realization of the identity-based encryption system, we decided to use a microprocessor-based architecture. The main reason for this is that identity-based encryption, and pairings in general, comprise computations that are very irregular. Even though the algorithms spend most of the time doing the same computations, the high-level algorithms would be hard to implement as a state machine in hardware. Such a pure-hardware implementation would result in a complex structure, very probably with bad critical path, that is hard to maintain and almost impossible to reuse for other systems that implement other types of pairings or other pairing-based schemes.

Using a microprocessor, on the other hand, has several advantages. The same microprocessor can be used for the actual application that is to be deployed in the embedded environment, hence viewing the security added as a feature which can also be equipped at a later time. In addition, and this will also be done later on, one can improve such a system

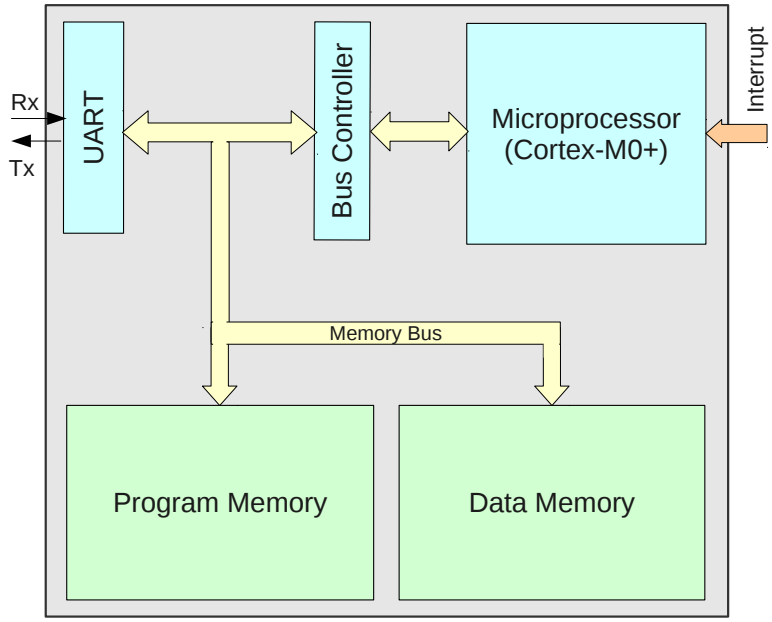

Figure 6.1.: Architecture of the implemented hardware platform.

by adding simple instructions that increase performance of heavily used routines, therefore creating an Application-Specific Instruction-set Processor (ASIP). Further, the software implementation for the security feature is portable to other systems as well. However, very often pure hardware implementations are faster than such using a microprocessor.

6.1.1. Hardware

The microprocessor-based hardware platform used for our purpose is illustrated by Figure 6.1. The design consists of a *microprocessor*, a *program memory*, a *data memory*, a *bus arbiter*, and an *UART interface* for communication. Of course, the communication interface may be replaced by any other interface suitable for the concrete purpose. The program memory is built using ROM macros, yielding low power consumption. Analogously, the data memory is created from RAM macros. As a microprocessor, the Cortex-M0+ by ARM was chosen. A clone of this processor was created in previous work by Unterluggauer [Unt13], which allows to give concrete power measures and reliable area figures.

Cortex-M0+

The Cortex-M0+ is the most energy-efficient microprocessor offered by ARM [Ltd13]. Its main areas of application are microcontrollers, wireless sensor nodes, automotive, and mixed-signal processing. The 32-bit microprocessor supports a Von Neumann architecture and a mixed 16-bit and 32-bit instruction-set, namely *Thumb* and small parts of *Thumb-2*. The processor has 16 registers `r0-r15`, which are 32 bits in size. Three of those registers have predefined meaning, namely the program counter `pc` (`r15`), the stack pointer `sp` (`r14`), and the link register `lr` (`r13`). From the remaining 13 registers, merely 8 can be efficiently used as many instructions are confined to the lower 8 registers `r0-r7`. In

addition, many of the instructions require the destination register to be equal to one of the source registers. The upper 8 registers `r8-r15`, apart from the stack pointer, are most commonly only accessible by a `mov` instruction.

For efficiency reasons, the calling convention requires the first four parameters of a function to be passed via the registers `r0-r3`. Similarly, the return value is also passed via the first register `r0`. In case the number of available registers is insufficient for the function call, the stack is used for any additional parameters.

Several options for customizations are offered by the processor. For example, one can choose between a 32-cycle bit-serial or a 1-cycle bit-parallel multiplier. In the context of this thesis, always the 1-cycle bit-parallel multiplier was used as multiplications are significant for pairing computations. Up to 32 interrupts can be served by the processors internal interrupt controller. In the configuration used, four interrupts are supported. The processor offers advanced mechanisms for interrupt handling, that is, it automatically takes care of pushing the current register values to the stack prior to invocation of the interrupt service routine. Also, four different priorities may be defined for the interrupts. However, the identity-based encryption scheme implemented does not make use of them, but the powerfulness of interrupts on the Cortex-M0+ is clearly an advantage for any embedded application that is deployed. At the same time, one can resort to state-of-the-art security features.

The processor, which only has about 12,000 Gate Equivalents (GE) at 90 nm in its minimal configuration, supports addressing of up to 4 GB of memory due to its 32-bit address bus. The address space is split into areas that are dedicated to certain types of memory, that is, constant program memory, data memory, and memory-mapped devices. Therefore, the Cortex-M0+ seems adequate for any future application.

6.1.2. Software

The identity-based encryption scheme and the underlying pairing operations were implemented on the hardware architecture arising from the Cortex-M0+ microprocessor. The architecture of the software implementation is depicted by Figure 6.2. As it shows, the implementation consists of several layers. The identity-based encryption scheme relies on a Pseudo Random Number Generator (PRNG) and a hash function, that stand outside of this layered implementation. As a PRNG, the random number generator stated in the FIPS 186-2 standard [NIS01, Appendix 3] was used. SHA-256 [NIS12] was taken as the hash function.

Besides these, the identity-based encryption scheme requires the computation of pairings and operations in groups of elliptic curves over simple prime fields and its quadratic extension. The pairing itself again relies on the computation of the elliptic curve operations, but additionally needs to perform operations in the 12th extension of the prime field being used. The extension field arithmetic as well as elliptic curve arithmetic is based on the simpler finite field arithmetic, which in turn uses simple multi-precision integer arithmetic. The latter is necessary to operate with integers of large width on a processor architecture with fixed data width.

Contrary to this view, the architecture may be looked at in a bottom-up style, where one starts with the multi-precision integer arithmetic, builds finite field arithmetic on top of it, and creates a tower of extensions. With this arithmetic, operations in groups of elliptic curves and further pairings can be implemented. Finally, everything needed for the identity-based encryption scheme is ready to use.

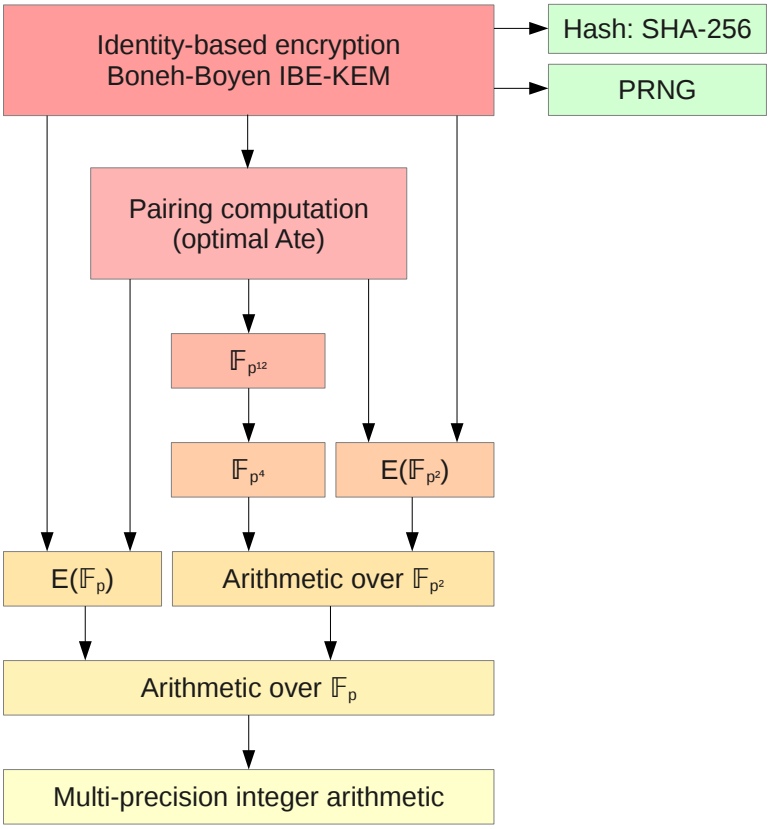

Figure 6.2.: Software architecture of the implemented identity-based encryption scheme.

6.2. Implementation Aspects

As pointed out in the comparison of Chapter 4, the BB_1 scheme by Boyen in its KEM variant was implemented. As underlying curve, three different BN curves (see Section 3.5.1) were used to accomplish different levels of security. For the sake of transparency and reproducibility, the various parameters found by Pereira et al. [Per+11] are stated as follows:

BN256 $(y^2 = x^3 + 3)$

u : `0x60800000 00040043`
p : `0xBA139EC2 401EDC28 FB605C6B 53E289B5 1311ACA0 D477DF46 FEEE89B1 622C349B`
n : `0xBA139EC2 401EDC28 FB605C6B 53E289B4 38D02CA0 D465C617 ECEE8951 559BCB65`

BN254 $(y^2 = x^3 + 2)$

u : `-0x40800000 00000001`
p : `0x25236482 40000001 BA344D80 00000008 61210000 00000013 A7000000 00000013`
n : `0x25236482 40000001 BA344D80 00000007 FF9F8000 00000010 A1000000 0000000D`

BN158 ($y^2 = x^3 + 2$)

u : 0x00000040 00800023
p : 0x24012003 AF565BE6 394AC09E 1E2D54A0 4C50525B
n : 0x24012003 AF565BE6 394A609C 9E2B6B9F 7A5035A5

As BN curves have embedding degree $k = 12$, appropriate extension field arithmetic needs to be implemented, which is already indicated by Figure 6.2. The following sections shall give an overview on the most important implementation aspects, that is, the structure and the algorithms used.

6.2.1. Prime Field Arithmetic

Prime field arithmetic comprises addition, subtraction, doubling, halving, inversion, multiplication and exponentiation. Addition, subtraction, doubling and halving are trivial modular operations and bascially do not need any further explanation. However, one of the goals was to attain security against Simple Power Analysis (SPA) attacks, that is, the routines should perform independent of the data being processed. Therefore, it is necessary for these operations to always consume the same amount of time. This can be achieved by always doing the reduction step. Since branches must be avoided, masking of operands is used instead. For example, to avoid branches in modular subtraction, one can use the borrow to build a suitable mask to avoid branches:

Input: a, b
Output: $a - b \bmod p$
1: $(borrow, s) \leftarrow a - b$
2: **if** $borrow$ **then**
3: **return** $s + p$
4: **else**
5: **return** s
6: **end if**

Input: a, b
Output: $a - b \bmod p$
1: $(borrow, s) \leftarrow a - b$
2: $sr \leftarrow s + p$
3: $mask_i \leftarrow borrow \quad \forall \quad 0 \leq i < bitwidth$
4: **return** $(sr \wedge mask) \vee (s \wedge \overline{mask})$

Modular multiplication consists of a multiplication and a reduction step. The reduction step can be done in several ways: a simple loop subtracting the modulus as long as the result is too large, trial division known as Barrett reduction [Bar87], Montgomery's method [Mon85], and fast-reduction techniques. The loop-based approach is clearly inadmissibly slow. For primes used in the context of BN curves, there are no fast-reduction techniques, which is why this approach cannot be used. From the remaining two methods, Montgomery reduction was examined to be the most efficient.

According to Koç, Acar, and Kaliski [KAK96, Section 2], the Montgomery reduction is very often faster than other methods as it does not involve a division by the prime p. To perform a Montgomery reduction, for the prime $2^{k-1} < p < 2^k$ a number r, that is co-prime to p, is chosen to be 2^k. Further, the operands are replaced by their p-residues, that is, for a given integer $a < p$, the corresponding p-residue is defined as $\bar{a} = a \cdot r \bmod p$. The set $\{a \cdot r \bmod p : 0 \leq a \leq p-1\}$ is a complete p-residue system that is bijective to the numbers in range 0 to $p - 1$. The Montgomery product for two p-residues is then defined to be

$$\bar{c} = \bar{a} \cdot \bar{b} \cdot r^{-1} \bmod p,$$

where r^{-1} is the inverse of r such that $r^{-1} \cdot r = 1 \bmod p$. The result $\bar{c}$ is then the p-residue of the product of a and b:

$$\begin{aligned}\bar{c} &= \bar{a} \cdot \bar{b} \cdot r^{-1} \bmod p \\ &= a \cdot r \cdot b \cdot r \cdot r^{-1} \bmod p \\ &= c \cdot r \bmod p\end{aligned}$$

The inverse r^{-1} and a value p_0 are precomputed from $r \cdot r^{-1} + p \cdot p_0 = 1$ using the extended euclidean algorithm and then stored as constants in memory. The Montgomery multiplication can then be expressed as shown in Algorithm 4.

Algorithm 4 Computation of the Montgomery product.

Input: $\bar{a}, \bar{b}, p, p_0, r$
Output: $\bar{c}$
1: $t \leftarrow a \cdot b$
2: $u \leftarrow (t + (t \cdot p_0 \bmod r) \cdot p)/r$
3: **if** $u \geq p$ **then**
4: **return** $u - p$
5: **else**
6: **return** u
7: **end if**

As divisions and modulo operations in Algorithm 4 are performed using a power of 2, calculation of the Montgomery product is faster than other modular multiplication techniques on the Cortex-M0+. However, conversion to the Montgomery domain, that is, calculating $\bar{a} = a \cdot r \bmod p$, and conversion back, that is, calculating $a = \bar{a} \cdot r^{-1} \bmod p$, introduce an overhead that does not amortize in a single multiplication. Therefore, all operations that are not exposed via external interfaces, that is, anything, but the identity-based encryption scheme, operate on parameters in the Montgomery domain. Consequently, constants being used are stored in Montgomery form as well. In practical scenarios and in this work, r is set to fit the underlying architecture's word size, that is, if s words of size w are needed to represent a multi-precision integer, then $r = 2^{sw}$.

Inversion can be done by either using the extended euclidean algorithm or Fermat's little theorem. In general, the extended euclidean algorithm is faster. However, constant runtime is something that cannot be achieved using the extended euclidean algorithm. Therefore, inversion based on Fermat's little theorem was used, that is,

$$a^{p-2} \equiv a^{-1} \bmod p\ .$$

This theorem leads to inversion by exponentiation. As inversion uses a fixed exponent, namely $p - 2$, it always consumes the same amount of time. Any other modular exponentiation in the prime field is not needed by the identity-based encryption scheme. If it is implemented though, runtime that is independent of a secret exponent is necessary. This can be achieved using a Montgomery powering ladder. Similar to the subtraction example mentioned before, its implementation must not use branches.

6.2.2. Extension Field Arithmetic

For the pairing computation over BN curves, several extension fields are needed: $\mathbb{F}_{p^2}$ is needed to represent one of the input parameters and $\mathbb{F}_{p^{12}}$ is the smallest extension of the

prime field $\mathbb{F}_p$ where the result lies. Therefore, a tower of extensions was implemented as suggested by Koblitz and Menezes [KM05, Section 5]. For this thesis, the tower was built as $\mathbb{F}_p \leftarrow \mathbb{F}_{p^2} \leftarrow \mathbb{F}_{p^4} \leftarrow \mathbb{F}_{p^{12}}$. Note that there are also other constructions possible that are isomorphic to that. However, this construction seems suitable for our purpose.

Fast operations over the quadratic extension $\mathbb{F}_{p^2}$ seem particularly important as both elliptic curve operations in $E'(\mathbb{F}_{p^2})$ and the remaining tower $\mathbb{F}_{p^2} \leftarrow \mathbb{F}_{p^4} \leftarrow \mathbb{F}_{p^{12}}$ rely on it. Therefore, and this is always possible if $p \equiv 3 \bmod 4$, $\mathbb{F}_{p^2}$ was built by adjoining the root of -1, that is, $\mathbb{F}_{p^2} = \mathbb{F}_p[i]/(i^2+1)$. Elements in $\mathbb{F}_{p^2}$ may hence be represented as $b_0 + b_1 i$, where i denotes a root of $i^2 = -1$ and $b_0, b_1 \in \mathbb{F}_p$. This allows fast computations in the quadratic extension as multiplying by i does not involve any multiplication overhead. Instead, reducing terms of i^2 corresponds to simple negation.

As indicated by Barreto and Naehrig [BN06, Le. 1 and following], the residue used to construct $\mathbb{F}_{p^{12}}$ needs to be chosen carefully so as to produce a sextic twist of correct order. To satisfy this condition, the residue was chosen as $\xi = (1+i)$ in case of the BN256 curve, and $\xi = 1/(1+i)$ for the curves BN254 and BN158. The sextic residue ξ over $\mathbb{F}_{p^2}$ is used to create $\mathbb{F}_{p^4} = \mathbb{F}_{p^2}[u]/(u^2 - \xi)$ and $\mathbb{F}_{p^{12}} = \mathbb{F}_{p^4}[v]/(v^3 - \zeta)$, where $\zeta = \sqrt{\xi}$. Note that $\mathbb{F}_{p^{12}}$ can also be expressed as $\mathbb{F}_{p^{12}} = \mathbb{F}_{p^2}[z]/(z^6 - \xi)$.

Analog to the quadratic extension $\mathbb{F}_{p^2}$, elements in $\mathbb{F}_{p^4}$ can be represented as $c_0 + c_1 u$, where $u = \sqrt{\xi}$ and $c_0, c_1 \in \mathbb{F}_{p^2}$. Elements in $\mathbb{F}_{p^{12}}$ are similarly represented as $d_0 + d_1 v + d_2 v^2$, where $v = \sqrt[3]{\zeta} = \sqrt[6]{\xi}$ and $d_0, d_1, d_2 \in \mathbb{F}_{p^4}$. However, sometimes it is more convenient to represent elements in $\mathbb{F}_{p^{12}}$ as $a_0 + a_1 z + a_2 z^2 + a_3 z^3 + a_4 z^4 + a_5 z^5$, where $z = \sqrt[6]{\xi}$ and $a_0, ..., a_5 \in \mathbb{F}_{p^2}$. One can easily switch between those two representations:

$$\begin{aligned}(c_{0,0} + c_{0,1}u) + (c_{1,0} + c_{1,1}u)v + (c_{2,0} + c_{2,1}u)v^2 &= \\ c_{0,0} + c_{0,1}u + c_{1,0}v + c_{1,1}uv + c_{2,0}v^2 + c_{2,1}uv^2 &= \\ c_{0,0} + c_{1,0}z + c_{2,0}z^2 + c_{0,1}z^3 + c_{1,1}z^4 + c_{2,1}z^5.\end{aligned}$$

It shall be noted, that the tower construction $\mathbb{F}_{p^2} \leftarrow \mathbb{F}_{p^6} \leftarrow \mathbb{F}_{p^{12}}$ is equivalent by the same means. Further, the residue ξ constitutes an efficient solution since multiplications and divisions by $i+1$ can be efficiently computed merely using additions and subtractions. This is an important aspect as this type of computation occurs regularly in the tower of extension fields.

Additions, subtractions, doublings, negations and halvings are easy to implement in extension fields: each coefficient in the polynomial is separately processed in the underlying field. Crucial, however, are multiplications and squarings in extension fields. Different approaches for quadratic and cubic extensions are needed. All of these trade additions and subtractions in the underlying field against the respective multiplications. This is clearly favorable on platforms, where multiplications are significantly more expensive than the former.

Multiplication and Squaring in Quadratic Extensions

Multiplication in $\mathbb{F}_{p^2}$ can be done using Karatsuba's [KO63; Dev+06] algorithm. Given two values $a, b \in \mathbb{F}_{p^2}$, represented as $a_0 + a_1 x \in \mathbb{F}_p[x]/(x^2 - \beta)$, the Karatsuba algorithm to compute $c = a \cdot b \in \mathbb{F}_{p^2}$ is

$$\begin{aligned}v_0 &= a_0 b_0 & v_1 &= a_1 b_1 \\ c_0 &= v_0 + \beta v_1 & c_1 &= (a_0 + a_1)(b_0 + b_1) - v_0 - v_1.\end{aligned}$$

Consequently, it takes three multiplications instead of four in the schoolbook method. On the other hand, it takes two additions and three subtractions compared to two additions in the schoolbook method, but, as already mentioned, it amortizes when multiplications are significantly more expensive. For squaring in the quadratic extension Devegili et al. [Dev+06, Section 3] showed even better formulas based on Karatsuba. The square of a value $c = a^2 \in \mathbb{F}_{p^2}$ is computed as

$$\begin{aligned} v_0 &= a_0 a_1 \\ c_0 &= (a_0 + a_1)(a_0 + \beta a_1) - v_0 - \beta v_0 \qquad c_1 = 2v_0. \end{aligned}$$

The same multiplication and squaring formulas are used for $\mathbb{F}_{p^4}$, which is the quadratic extension over $\mathbb{F}_{p^2}$.

Multiplication and Squaring in Cubic Extensions

In a cubic extension, multiplication and squaring is more difficult as it consists of 3×3 partial products. As in the quadratic case, a variant of Karatsuba can be used. According to Knuth [Knu97, Section 4.3.3.A], the method by Toom-Cook constitutes a more general alternative. If the cubic extension is utilized only to compute the pairing in $\mathbb{F}_{p^{12}}$, divisions involved in the Toom-Cook method can be avoided, because constant multiples are filtered out in the final exponentiation step. However, the advantage in runtime was too small in order to justify the additional demand for memory. Therefore, the Karatsuba method for cubic extensions was used. [Dev+06, Section 4]

Let a, b denote two elements in $\mathbb{F}_{p^{12}}$, represented as $a_0 + a_1 x + a_2 x^2 \in \mathbb{F}_{p^4}[x]/(x^3 - \beta)$, then the product $c = a \cdot b \in \mathbb{F}_{p^{12}}$ is calculated as follows:

$$\begin{aligned} v_0 &= a_0 b_0 \quad v_1 = a_1 b_1 \quad v_2 = a_2 b_2 \\ c_0 &= v_0 + \beta((a_1 + a_2)(b_1 + b_2) - v_1 - v_2) \\ c_1 &= (a_0 + a_1)(b_0 + b_1) - v_0 - v_1 + \beta v_2 \\ c_2 &= (a_0 + a_2)(b_0 + b_2) - v_0 + v_1 - v_2. \end{aligned}$$

Compared to the schoolbook method with 9 multiplications and 6 additions, the Karatsuba method takes 6 multiplications and 13 additions. Fast formulas for squaring in cubic extensions were presented by Chung and Hasan [CH07]. In three different variants, a trade-off between multiplications, squarings, additions and halvings is done. Nevertheless, the sum of multiplications and squarings in the underlying field stays constant. As squaring in the underlying field $\mathbb{F}_{p^4}$ is cheaper than multiplication, the formulas with the least number of multiplications, namely CH-SQR3, were used. Devegili et al. [Dev+06, Section 4] square a value $c = a^2 \in \mathbb{F}_{p^4}$ as

$$\begin{aligned} s_0 &= a_0^2 \qquad s_1 = (a_0 + a_1 + a_2)^2 \\ s_2 &= (a_0 - a_1 + a_2)^2 \qquad s_3 = 2a_1 a_2 \\ s_4 &= a_2^2 \qquad t_1 = (s_1 + s_2)/2 \\ c_0 &= s_0 + \beta s_3 \\ c_1 &= s_1 - s_3 - t_1 + \beta s_4 \\ c_2 &= t_1 - s_0 - s_4. \end{aligned}$$

Contrary to the schoolbok method, which takes three multiplications, three squarings, and 6 additions, the formulas presented make possible squaring in one multiplication, four

squarings, 11 additions and one halving. Note that halving a value is also cheap compared to multiplication.

Inversion

Inversion in the prime field was done by exploiting Fermat's little theorem, which is quite expensive. In extension fields, Davida [Dav72] shows that direct inversion techniques are more suitable. Let $p(x)$ denote the irreducible polynomial defining the extension field. The product of two polynomials

$$c(x) = a(x)b(x) \bmod p(x) = 1$$

forms a set of equations in the coefficients of $a(x)$ and $b(x)$ that is solved in order to transfer the inversion problem to the underlying base field. In this manner, Baktir et al. [Bak+04, Ex. 2,3] state formulas for direct inversion in quadratic and cubic extension fields. Given the value $a \in \mathbb{F}_{p^2}$ represented as $a_0 + a_1 x \in \mathbb{F}_p[x]/(x^2 - \beta)$, the inverse $b = a^{-1} \in \mathbb{F}_{p^2}$ is computed as

$$\begin{aligned} \Delta &= a_0^2 - \beta a_1^2, \\ b_0 &= a_0 \Delta^{-1} \quad b_1 = -a_1 \Delta^{-1}. \end{aligned}$$

Note that the inversion of Δ takes place in the underlying base field, namely $\mathbb{F}_p$. Consequently, inversion in extension fields introduces only negligible overhead. Analogously, inversion is done in $\mathbb{F}_{p^4}$.

For inversion in the cubic extension of $\mathbb{F}_{p^4}$, namely $\mathbb{F}_{p^{12}}$, different formulas are needed: let $a \in \mathbb{F}_{p^{12}}$ be represented as $a_0 + a_1 x + a_2 x^2 \in \mathbb{F}_{p^4}[x]/(x^3 - \beta)$, then the inverse $b = a^{-1} \in \mathbb{F}_{p^{12}}$ is determined by computing

$$\begin{aligned} \Delta &= a_0^3 - 3a_0 a_1 a_2 \beta + a_1^3 \beta + a_2^3 \beta^2, \\ b_0 &= (a_0^2 - a_1 a_2 \beta)\Delta^{-1}, \\ b_1 &= (a_2^2 \beta - a_0 a_1)\Delta^{-1}, \\ b_2 &= (a_1^2 - a_0 a_2)\Delta^{-1}. \end{aligned}$$

Frobenius Endomorphism

Similar to Definition 3.25 for elliptic curves, the frobenius endomorphism is defined for extension fields. It can help to massively reduce runtime of large exponentiations and is specified as:

$$\pi_{p^i} : \mathbb{F}_{p^k} \mapsto \mathbb{F}_{p^k}, \quad x \mapsto x^{p^i}.$$

For the base field, that is, the case $k = 0$, it is the identity morphism. For extension fields, it depends on the residue that defined the extension field whether it is trivial to calculate the frobenius endomorphism. Considering the quadratic extension of a field $\mathbb{F}_p$, its elements are represented as $a_0 + a_1 x \in \mathbb{F}_p[x]/(x^2 - \beta)$. According to Gouvêa [Gou13], the i-th frobenius endomorphism is calculated as

$$\begin{aligned} (a_0 + a_1)^{p^i} &= a_0^{p^i} + a_1^{p^i} x^{p^i} && (6.1) \\ &= a_0 + a_1 x^{p^i} && (6.2) \\ &= a_0 + a_1 (x^2)^{(p^i-1)/2} x \\ &= a_0 + a_1 \beta^{(p^i-1)/2} x. \end{aligned}$$

The two marked equations are particularly interesting. Equation 6.1 is allowed since any binomial coefficient $\binom{p}{k} = \frac{p!}{k!(p-k)!}$ vanishes for $k \notin \{0, p\}$ (Freshman's dream). Further, the two coefficients a_0 and a_1 are in the base field, which is why the frobenius endomorphism acts trivially on them in Equation 6.2. The remaining consists of simple substitutions.

In the context of this thesis, $\mathbb{F}_{p^2}$ was defined by adjoining the root of $\beta = -1$. This is particularly advantageous since the frobenius map simplifies to

$$\begin{aligned} a_0 + a_1\beta^{(p^i-1)/2}x &= a_0 + a_1(-1)^{(p^i-1)/2}x \\ &= a_0 + a_1(-1)^i \qquad \text{iff } p \equiv 3 \bmod 4, \end{aligned}$$

which holds for BN curves. Consequently, frobenius endomorphisms in $\mathbb{F}_{p^2}$ are simply computed as conjugations. For the quadratic extension $\mathbb{F}_{p^4}$ over $\mathbb{F}_{p^2}$, which is defined as $\mathbb{F}_{p^2}[u]/(u^2 - \xi)$, the frobenius action looks as follows:

$$(a_0 + a_1x)^{p^i} = a_0^{p^i} + a_1^{p^i}\xi^{(p^i-1)/2}x. \tag{6.3}$$

Since $a_0, a_1 \in \mathbb{F}_{p^2}$, the frobenius endomorphism on the two coefficients a_0, a_1 is not the identity any more. Instead, the frobenius map is applied recursively on the coefficients a_0, a_1 in $\mathbb{F}_{p^2}$.

Similarly, the frobenius endomorphism can be computed in $\mathbb{F}_{p^{12}}$. Let $a_0 + a_1v + a_2v^2$ denote an element in $\mathbb{F}_{p^{12}} = \mathbb{F}_{p^4}[v]/(v^3 - \sqrt{\xi})$. Then, the frobenius map evaluates as

$$\begin{aligned} (a_0 + a_1v + a_2v^2)^{p^i} &= a_0^{p^i} + a_1^{p^i}v^{p^i} + a_2^{p^i}v^{2p^i} \\ &= a_0^{p^i} + a_1^{p^i}(v^3)^{\lfloor p^i/3 \rfloor}v^{p^i \bmod 3} + a_2^{p^i}(v^3)^{\lfloor 2p^i/3 \rfloor}v^{2p^i \bmod 3} \\ &= a_0^{p^i} + a_1^{p^i}\sqrt{\xi}^{\lfloor p^i/3 \rfloor}v^{p^i \bmod 3} + a_2^{p^i}\sqrt{\xi}^{\lfloor 2p^i/3 \rfloor}v^{2p^i \bmod 3} \\ &= a_0^{p^i} + a_1^{p^i}\sqrt{\xi}^{\frac{p^i-1}{3}}v + a_2^{p^i}\sqrt{\xi}^{2\frac{p^i-1}{3}}v^2. \end{aligned} \tag{6.4}$$

The derivation of the formulas to compute the map is basically the same as for the quadratic extension. However, in Equation 6.4 the fact that BN curves satisfy $p = 1 \bmod 3$ is used to simplify the result. Again, the frobenius map on the coefficients a_0, a_1 and a_2 has to be computed in the underlying field, namely $\mathbb{F}_{p^4}$.

For computing the first frobenius endomorphism π_p in $\mathbb{F}_{p^4}$ and $\mathbb{F}_{p^{12}}$, several roots of the exponentiated residue ξ^{p-1} are needed as one can see in Equation 6.3 and Equation 6.4. It would take too much effort to compute these on an embedded platform. For this reason, the powers of the 6th root of ξ^{p-1}, namely $\sqrt[6]{\xi}$, $\sqrt[6]{\xi^2}$, ..., $\sqrt[6]{\xi^5}$, are precomputed and stored as constants in memory. To evaluate higher orders of the frobenius endomorphism π_{p^i}, precomputation of even more constants would be necessary. In order to avoid this, higher orders of frobenius operation are simply done iteratively, that is, $\pi_{p^i} = \pi_p \circ \pi_p \circ ... \circ \pi_p$ (i times).

Exponentiation

The encryption routine of the identity-based encryption scheme needs an exponentiation by a secret value in $\mathbb{F}_{p^{12}}$. Therefore, one could use a simple square-and-multiply approach as shown in Algorithm 1. However, this algorithm has runtime that depends on the secret value k, which constitutes a vulnerability. Hence, the *Montgomery powering ladder* from

Algorithm 5 Elliptic curve point multiplication using homogeneous projective co-Z coordinates and a Montgomery ladder.

Input: $P \in E(\mathbb{F}_p), k = (k_{t-1}, ..., k_1, k_0)_2 \in \mathbb{N},$ with $k_{t-1} \neq 0$
Output: $Q = kP$
1: $\{X_1, X_2, Z\} \leftarrow \texttt{AddDblCoZ}(\{\texttt{0}, \texttt{x}_\texttt{P}, \texttt{1}\})$
2: $X_1 \leftarrow x_P \cdot Z$
3: **for** $i = t - 2$ downto 0 **do**
4: $\quad b \leftarrow k_i$
5: $\quad \{X_{2-b}, X_{1+b}, Z\} \leftarrow \texttt{AddDblCoZ}(\{X_{2-b}, X_{1+b}, Z\})$
6: **end for**
7: $\{X, Y, Z\} \leftarrow \texttt{RecoverFullCoordinatesCoZ}(\{X_1, X_2, Z\})$
8: Verify that $Z(Y^2 - bZ^2) = X(X^2 + aZ^2)$
9: **return** $\{X, Y, Z\}$

Algorithm 3 is used instead. The algorithm has constant runtime and does not involve any branches. In addition, the two values r_0 and r_1 are interleaved which avoids the C safe-error attacks from Section 5.3.1. Nevertheless, Algorithm 3 has a higher overall runtime than Algorithm 1 on average.

6.2.3. Elliptic Curve Arithmetic

Arithmetic over the elliptic curve basically splits into two parts. On the one hand, basic arithmetic curve arithmetic such as point addition, point doubling and point multiplication is needed. On the other hand, arithmetic over elliptic curves is needed for the evaluation of pairings. For the latter, very specialized formulas exist that perform interleaved elliptic curve arithmetic and function evaluation. These are the topic of Section 6.2.4. This section covers the basic elliptic curve arithmetic though.

For isolated point additions and point doublings, the formulas from Equation 3.3 and Equation 3.4 in Section 3.2.1 are used. These, however, are not efficient to use in point multiplications as they involve many field inversions, which are significantly more expensive than field multiplications. Note that inversions become cheaper relatively to multiplications in large extension fields, but this is not the case for prime fields and their quadratic extension, where elliptic curve arithmetic is performed in the context of this thesis.

The identity-based encryption scheme chosen involves several point multiplications over the prime field $\mathbb{F}_p$ during encryption. A secret value is used as the scalar. The common *double-and-add* approach cannot be used in such a case, because both power consumption and timing would clearly relate to the secret value as it does in its square-and-multiply counterpart in Algorithm 1. An efficient method that computes the product of an elliptic curve point with a scalar secret in constant time was shown by Hutter, Joye, and Sierra [HJS11]. The method is based on a Montgomery ladder, whose square-and-multiply counterpart was already shown in Algorithm 3. Further, it utilizes the homogeneous projective coordinates from Section 3.2.1 and the so-called *co-Z* coordinate system, that is, both points involved in the Montgomery ladder share a common z-coordinate. In addition, each of the points is only represented by their x-coordinates during computation of the ladder. To obtain the final result, its full coordinates are recovered from both the x- and the z-coordinates. This is illustrated by Algorithm 5.

The routine $\texttt{AddDblCoZ}(\{\texttt{X}_\texttt{1}, \texttt{X}_\texttt{2}, \texttt{Z}\})$ interleaves the double step with the add step of the

Algorithm 6 Optimal Ate pairing over BN curves.

Input: $P \in E(\mathbb{F}_p), Q \in E(\mathbb{F}_{p^2}), s = |6u + 2|$
Output: $a_{opt}(Q, P)$
1: $T \leftarrow Q, f \leftarrow 1$
2: **for** $i = \lfloor ld(s) \rfloor - 2$ downto 0 **do**
3: $f \leftarrow f^2 \cdot \ell_{T,T}(P)$
4: $T \leftarrow [2]T$
5: **if** $s_i = 1$ **then**
6: $f \leftarrow f^2 \cdot \ell_{T,Q}(P)$
7: $T \leftarrow T + Q$
8: **end if**
9: **end for**
10: **if** $u < 0$ **then**
11: $T \leftarrow -T, \ f \leftarrow f^{-1}$
12: **end if**
13: $Q_1 \leftarrow \pi_p(Q); Q_2 \leftarrow \pi_{p^2}(Q)$
14: $f \leftarrow f \cdot \ell_{T,Q_1}(P); T \leftarrow T + Q_1$
15: $f \leftarrow f \cdot \ell_{T,-Q_2}(P); T \leftarrow T - Q_2$
16: $f \leftarrow f^{(p^{12}-1)/n}$
17: **return** f

Montgomery ladder in order to obtain more efficient formulas. The algorithm can be randomized by introducing a random number in the initial double-and-add step. However, this is not done as the secret parameter in the scheme is already random. The full coordinates are recovered by the routine `RecoverFullCoordinatesCoZ`($\{\mathtt{X}_1, \mathtt{X}_2, \mathtt{Z}\}$). By verifying that $Z(Y^2 - bZ^2)$ equals $X(X^2 + aZ^2)$, fault attacks applied during the multiplication can be detected. Several variants were presented for the two routines `AddDblCoZ` and `RecoverFullCoordinatesCoZ`. One can choose between in-place and out-of-place computations on the one hand, and do a trade-off between memory footprint and runtime on the other hand. The concrete algorithms were chosen to have low memory requirements and little computational effort considering that the equations of BN curves do not have a term in x. They are listed in Appendix A.

6.2.4. Pairing Realization

The optimal Ate pairing from Equation 3.9 was chosen to be implemented due to its good performance. As for every pairing, the computation shown in Algorithm 6 consists of two steps: evaluation of the Miller function in Lines 1-15, and the final exponentiation in Line 16.

By definition of the pairing, the former is split into a loop part, namely the Miller loop in Lines 1-9, and some additional computations in Lines 10- 15 to obtain the final result of the Miller function. The Miller loop is based on a double-and-add approach that results from the observations made by Miller as presented in Section 3.4.4. The Miller function f is evaluated using line functions of the form $\ell_{T,Q}(P)$, which give the result of the straight line from point T to point Q at point P. In addition, the first and the second frobenius endomorphism π_p, π_{p^2} are needed.

The Miller loop is intended to calculate a function $f_{6u+2,Q}(P)$. For negative BN pa-

rameters u, the term $6u + 2$ that controls the Miller loop would also become negative. To avoid this situation, the function $f_{|6u+2|,Q}(P)$ is evaluated first by setting the loop parameter $s = |6u+2|$. Afterwards, the correct function $f_{6u+2,Q}(P)$ is computed by doing an inversion of f in Lines 10-12. Additionally, the point T is negated for the remaining computations.

In contrast to the formulas shown Section 3.4.4, the algorithm does not involve any vertical line functions ν. The reason for this is, that under certain circumstances, the vertical lines through the result of point additions of the form $P_3 = P_1 + P_2$ lie in $\mathbb{F}_p$ and become 1 in the final exponentiation step due to Fermat's little theorem. This was first shown by Barreto et al. [Bar+02], titled *denominator eliminiation*, and generalized for twisted curves by Barreto, Lynn, and Scott [BLS04]. In addition, speaking in terms of the Ate pairing in Algorithm 6, Barreto et al. showed that the Miller function f does not have to be evaluated on a divisior D_P equivalent to $(P) - (\mathcal{O})$, but to simply use the second argument P if the points P and Q are linearly independent, that is, they are from different subgroups of $E(\mathbb{F}_{p^{12}})$, which is clearly the case for $\mathbb{G} = E(\mathbb{F}_{p^{12}})[n] \cap \mathrm{Ker}(\pi_p - [1]) \subseteq E(\mathbb{F}_p)$ and $\hat{\mathbb{G}} = E'(\mathbb{F}_{p^2})$ isomorphic to $E(\mathbb{F}_{p^{12}})$. Scott [Sco05] introduced the term *BKLS algorithm* for the Miller algorithm that incorporates the optimizations by Barreto et al.

It seems clear that the evaluation of a line function $\ell_{T,Q}$ between two points T and Q has much in common with adding these two points. The same observation can be made for the tangent function $\ell_{T,T}$ in a point T and the doubling of the same point T. Therefore, the operations in Lines 3-4 as well as in Lines 6-7 can be interleaved. Costello, Lange, and Naehrig [CLN10] published highly optimized formulas for interleaved point addition and line evaluation as well as interleaved point doubling and tangent line evaluation. These formulas, which were used for the implementation in this thesis, use homogeneous projective coordinates and are listed in Appendix B.

The final exponentiation in Line 16 of Algorithm 6 has a very large exponent and is in this form far from being efficiently computable. Therefore, the exponent of a pairing's final exponentiation is generally split as

$$(p^k - 1)/n = (p^{k/2} - 1) \cdot [(p^{k/2} + 1)/\phi_k(p)] \cdot [\phi_k(p)/n], \tag{6.5}$$

where ϕ_k denotes the k-th cyclotomic polynomial. As Scott et al. [Sco+09] show, for BN curves this translates into

$$(p^{12} - 1)/n = (p^6 - 1) \cdot (p^2 + 1) \cdot [(p^4 - p^2 + 1)/n].$$

Exponentiations by $(p^6 - 1)$ and $(p^2 + 1)$ are easy as they can be computed using the frobenius endomorphism, an inversion and two multiplications. For the hard part, namely the exponentiation by $(p^4 - p^2 + 1)/n$, several methods that exploit the parameterized definition of the group order n were presented by Devegili, Scott, and Dahab [DSD07], Scott et al. [Sco+09], and Fuentes-Castañeda, Knapp, and Rodríguez-Henríquez [FKR12]. Fuentes-Castañeda, Knapp, and Rodríguez-Henríquez uses a lattice-based approach to obtain the fastest currently known formulas for the hard part of exponentiation. The approach by Scott et al. uses addition chains, while the method by Devegili, Scott, and Dahab is rather random. For this implementation, the formulas by Fuentes-Castañeda, Knapp, and Rodríguez-Henríquez are used.

Using the definitions of $p(u)$ and $n(u)$ from Section 3.5.1, Fuentes-Castañeda, Knapp, and Rodríguez-Henríquez express the term $(p^4 - p^2 + 1)/n$ via the parameter u, from which they derive the following chain of computations:

$$f \to f^u \to f^{2u} \to f^{4u} \to f^{6u} \to f^{6u^2} \to f^{12u^2} \to f^{12u^3}.$$

After computing $a = f^{12u^3} \cdot f^{6u^2} \cdot f^{6u}$ and $b = a \cdot (f^{2u})^{-1}$, the hard part and hence the full exponentiation is finished by

$$[a \cdot f^{u^2} \cdot f] \cdot [b]^p \cdot [a]^{p^2} \cdot [b \cdot f^{-1}]^{p^3}.$$

Inversions become simple conjugations after the easy part of the final exponentiation, which equals the 6th frobenius endomorphism. This can be cheaply computed by several negations in $\mathbb{F}_p$. Hence, the overall effort in $\mathbb{F}_{p^{12}}$ sums up to 3 exponentiations by u, 3 squarings, and 10 multiplications.

6.3. Testing

Assuring the correctness of implementations is a very important aspect that must not be overseen. Once a platform is deployed, faulty implementations may raise high costs. Especially on an embedded hardware platform such as the one presented, fixing errors after deployment may be impossible. Therefore, comprehensive tests of the platform are necessary. Evaluating the correctness of any possible input is impossible though. Therefore, a meaningful trade-off has to be done. This section shall give an overview on the approach to testing the platform created.

In terms of the software implementation, a high-level reference was initially created in Java. This implementation is self-contained and comes with several self-tests that check its correctness, for example, the group orders of the sextic twists, and the bilinearity and non-degeneracy of the pairing. These self-tests already give a good indication of correctness. However, this code was additionally tested against the implementation by Pereira et al. [Per+11].

Consequently, an optimized software implementation for use in embedded environments was done in C. It was at first tested on a common x86-64 platform using test cases generated from the verified high-level reference in Java. In general, these test-cases do not involve self-tests, but consist of concrete input samples for each of the routines of all layers along with their expected result. Consequently, the output of a routine given a concrete input is tested against its corresponding reference result by a simple test program. These pairs of input and reference result involve both random samples and border cases that are expected to lead to problems, for example, overflows. Concerning the identity-based encryption scheme, the decapsulation routine is tested the same way, but there exists a self-test as encapsulation is randomized and cannot be covered using the just mentioned approach.

At this point, there is a high degree of certainty that the software part is correct. This implementation was then transferred to the Cortex-M0+ architecture and its instruction set. To avoid repeating hardware simulations, the more efficient instruction-set simulator for the Cortex-M0+ by Winter and Hein [WH12] was used. Consequently, both the cross-compiled C implementation and the optimized assembler routines could be tested for the target architecture.

As mentioned before, the processor itself is a clone of the Cortex-M0+ that was created in previous work by Unterluggauer [Unt13]. Its instruction-set has extensively been tested during that work using both hand-crafted test cases for each instruction and several cryptographic routines. To verify the correctness of the overall hardware platform, it was deployed to an FPGA and run successfully with 15,000 different, randomly chosen input values.

6.4. Optimization

Based on the implementation described in Section 6.2, several optimizations have been applied to obtain a fast and resource-efficient solution for the Cortex-M0+ hardware platform.

6.4.1. Prime Field Arithmetic

Besides multi-precision integer routines, the basic operations in the prime field $\mathbb{F}_p$ were optimized in assembler to make the best possible use of the instruction-set provided by the Cortex-M0+. As heavy use of loop unrolling is made, the routines `add`, `subtract`, `negate`, `halve`, and `multiply` have to be adapted for each of the curves being used, that is, BN158, BN254, and BN256. Further, each of the implemented assembler routines operates in constant time to avoid timing attacks and simple power analysis. Therefore, they do not use any branches, but use masking of operands instead, which was already shown in Section 6.2.1.

Particularly important is the optimization of the prime field multiplication as most of the overall runtime of elliptic curve arithmetic is due to it. For this reason, the Finely Integrated Product Scanning (FIPS) method of the Montgomery multiplication by Koç, Acar, and Kaliski [KAK96] was chosen for assembler optimization. The method avoids slow memory accesses for intermediate results by using an accumulator instead. Three of the registers are dedicated to the accumulator, hence achieving better performance. Besides, the multiplication and the reduction step of the Montgomery multiplication are interleaved.

Algorithm 7 shows the FIPS method: the two multi-precision integers $a, b \in \mathbb{F}_p$ are multiplied modulo p. Additionally the constant p_0, mentioned in Section 6.2.1, is needed. Each element is represented by s words of size W. As the Cortex-M0+ is a 32-bit processor, the value of W is 32. Consequently, $s = 8$ for the curves BN254 and BN256. The routine `MultiplyAccumulate`(`t`[0..2], `a`[`i`], `b`[`j`]) simply multiplies the i-th words of a with the j-th word of b and accumulates the result in the accumulator t, which consists of three words. Subtraction of two multi-precision integer values a, b is done by the routine `Subtract`(`a`, `b`), which besides the result also returns the borrow flag. Shifting in Lines 10 and 18 is possible since the least significant word of the accumulator becomes zero at the end of each iteration. After finishing the two loops, the result is found in m. However, $t[0]$ contains the most significant bit of the result. Consequently, if the result in m is larger than the modulus p or the least significant bit in $t[0]$ is set, an additionally necessary subtraction of the modulus p is performend in Lines 20-22. Note that this subtraction is done without branches, merely using masking of the operands. For performance reasons, the loops are all unrolled applying appropriate assembler macros.

As one can see from Algorithm 7, the `MultiplyAccumulate` routine is the most important. Therefore, Wenger, Unterluggauer, and Werner [WUW13] have put much effort into optimizing this routine, which is shown in Algorithm 8. The two registers `r8`, `r9` contain pointers to the two multi-precision values in $\mathbb{F}_p$. The accumulator t is represented by the registers `r5`-`r3`, where `r5` and `r3` denote the most and least significant words respectively. For each of the two operands, a single word, which is specified by `offset`, is loaded in Lines 1-4 and then multiplied in Lines 10-13. The Cortex-M0+ merely supports a 32 bit $\times$ 32 bit $\rightarrow$ 32 bit multiplication. As one half of the 64-bit result is missing in this case, the multiplier is used to perform 16 bit $\times$ 16 bit $\rightarrow$ 32 bit multiplications. Therefore,

Algorithm 7 Finely Integrated Product Scanning (FIPS) Montgomery multiplication.

Input: $a,\ b,\ p,\ p_0$
Output: $a \cdot b \bmod p$
1: $t[0..2] \leftarrow 0$
2: **for** $i = 0$ to s **do**
3: **for** $j = 0$ to $i + 1$ **do**
4: `MultiplyAccumulate`$(t[0..2],\ a[j],\ b[i-j])$
5: `MultiplyAccumulate`$(t[0..2],\ m[j],\ p[i-j])$
6: **end for**
7: `MultiplyAccumulate`$(t[0..2],\ a[i],\ b[0])$
8: $m[i] \leftarrow t[0] \cdot p_0[0] \bmod W$
9: `MultiplyAccumulate`$(t[0..2],\ m[i],\ p[0])$
10: $t[0] \leftarrow t[1];\ t[1] \leftarrow t[2];\ t[2] \leftarrow 0$
11: **end for**
12: **for** $i = s$ to $2s - 1$ **do**
13: **for** $j = i - s + 1$ to $s - 1$ **do**
14: `MultiplyAccumulate`$(t[0..2],\ a[j],\ b[i-j])$
15: `MultiplyAccumulate`$(t[0..2],\ m[j],\ p[i-j])$
16: **end for**
17: $m[i-s] \leftarrow t[0]$
18: $t[0] \leftarrow t[1]; t[1] \leftarrow t[2]; t[2] \leftarrow 0$
19: **end for**
20: $(borrow, tmp) \leftarrow$ `Subtract`$(m[0..s-1], p[0..s-1])$
21: $mask_i \leftarrow \overline{borrow} \vee (t[0])_0 \quad \forall \quad 0 \leq i < bitwidth$
22: **return** $(tmp \wedge mask) \vee (m \wedge \overline{mask})$

the shifting and masking routines in Lines 5-8 are needed. After multiplication, the four partial products are summed up in the accumulator. The intermediate result is stored on the stack. Therefore, the single words of the intermediate value can be loaded and stored using stack-relative addressing, and an additional `mov` instruction, such as the one in Line 1, is avoided in this case.

For the curve BN254, Gouvêa, Oliveira, and López [GOL12b] suggested exploiting the sparse form of the prime in the field multiplication. The prime used for this curve has five 16-bit parts being all zero. They proposed this optimization for the MSP430 that operates on 16-bit words. However, due to the fact that on the 32-bit Cortex-M0+ a full 32-bit multiplication has to be split into four 16-bit multiplications, this optimization can also be applied to this work. It reduces both the number of multiply-accumulate operations to be performed and the size of the unrolled loops.

The optimizations done for the prime field multiplication automatically speed up the exponentiation-based inversion as well. Additionally, inversion can be further optimized when using BN curves with positive parameters u. Therefore, the computation of the inverse $a^{-1} \in \mathbb{F}_p$ is split as

$$\begin{aligned} a^{-1} \bmod p = a^{p-2} \bmod p &= a^{36u^4+36u^3+24u^2+6u-1} \bmod p \\ &= a^{6u(4u+6u^2(1+u))} \cdot a^{6u-1} \bmod p. \end{aligned}$$

Algorithm 8 Multiply-Accumulate routine on the Cortex-M0+ processor.

Input: `r8, r9` are pointers to the operands
Output: {`r5, r4, r3`} is the accumulator

```
 1: mov   r1, r8
 2: ldr   r1, [r1, #offset1]
 3: mov   r2, r9
 4: ldr   r2, [r2, #offset2]
 5: uxth  r6, r1
 6: uxth  r7, r2
 7: lsr   r1, r1, #16
 8: lsr   r2, r2, #16
 9: mov   r0, r6
10: mul   r0, r0, r7     ▷ low × low
11: mul   r6, r6, r2     ▷ low × high
12: mul   r2, r2, r1     ▷ high × high
13: mul   r1, r1, r7     ▷ high × low
14: mov   r7, #0
15: add   r5, r5, r0     ▷ low × low
16: adc   r4, r4, r2     ▷ high × high
17: adc   r3, r3, r7
18: lsl   r0, r6, #16
19: lsr   r2, r6, #16
20: add   r5, r5, r0     ▷ low × high
21: adc   r4, r4, r2
22: adc   r3, r3, r7
23: lsl   r0, r1, #16
24: lsr   r2, r1, #16
25: add   r5, r5, r0     ▷ high × low
26: adc   r4, r4, r2
27: adc   r3, r3, r7
```

Precomputing $6u - 1$ as a constant leads to the following chain of computations:

$$a^{6u-1} \to a^{6u} \to a^{12u^2} \to a^{24u^2} \to a^{36u^2} \to a^{36u^3} \to a^{36u^4}.$$

Afterwards, the inverse of an element $a \in \mathbb{F}_p$ is simply computed as the product

$$a^{-1} \bmod p = a^{6u-1} \cdot a^{24u^2} \cdot a^{36u^3} \cdot a^{36u^4} \bmod p.$$

Therefore, the exponentiation by a large exponent is done using three exponentiations by u, one exponentiation by $6u - 1$, five multiplications and two squarings. Note that the exponents are fixed and publicly known. Hence, runtime is constant any way and branches may be used, which leads to reduced runtime due to the usually sparse form of u.

Besides optimizing the standard implementation of the prime field operations using assembler, the effect of lazy reduction has also been evaluated, that is, elements in $\mathbb{F}_p$ must not necessarily be represented by a value smaller than p. Consequently, reduction steps in modular additions only become necessary when a carry flag is raised. Algorithm 9 shows the lazy reduction technique applied to the addition of a and $b \in \mathbb{F}_p$. Note that in regular modular addition, that is, when elements are represented by values smaller than the modulus p, the reduction in Line 3 must also take place if $s \geq p$, leading to an additional comparison. On the other hand, a second reduction may become necessary if the two operands a and b are larger than the modulus p, which becomes visible in Line 7. Analogously, an additional reduction step may be necessary for subtraction too. The savings during the final reduction of the Montgomery multiplication are negligible.

Let each multi-precision integer be represented by s words of each W bits. If the prime satisfies $p < 2^{Ws-1}$, even precomputed multiples of p may be subtracted in the reduction step in Line 3. This, for example, is the case for the curve BN254. Then, the speedup is even higher if lazy reduction is implemented as in Algorithm 9. However, each routine

Algorithm 9 Lazy reduction technique applied to modular addition.

Input: $a, b \in \mathbb{F}_p$
Output: $a + b \bmod p$
1: $(carry, s) \leftarrow a + b$
2: **if** $carry = 1$ **then**
3: $\quad (carry_t, t) \leftarrow s - p$
4: $\quad$ **if** $carry_t = 1$ **then**
5: $\quad\quad$ **return** t
6: $\quad$ **else**
7: $\quad\quad$ **return** $t - p$
8: $\quad$ **end if**
9: **else**
10: $\quad$ **return** s
11: **end if**

should have a constant-runtime characteristic. To achieve this, addition and subtraction have to perform two reductions every time (such as in Lines 3, 7). Consequently, lazy reduction trades a comparison against a normal subtraction during modular addition. In modular subtraction, lazy reduction introduces another addition. The impact on the Montgomery multiplication is, however, very small. For a fair comparison, these lazy reduction routines were also optimized in assembly language.

6.4.2. Extension Field Arithmetic

The computation of pairings over BN curves relies heavily on operations in $\mathbb{F}_{p^2}$. Out of this context, Sánchez and Rodríguez-Henríquez [SR13] presented faster formulas for multiplication in $\mathbb{F}_{p^2}$. These formulas make use of the lazy reduction technique and are thus able to save a modular reduction. They are particularly interesting for the two curves BN254 and BN158, because their corresponding primes have two leading zeros in their computational representation as a multi-precision integer. Algorithm 10 illustrates these formulas.

The operations in Lines 1 - 7 are done using plain multi-precision integer arithmetic.

Algorithm 10 Optimized multiplication in $\mathbb{F}_{p^2}$.

Input: $a = a_0 + a_1 i, \;\; b = b_0 + b_1 i \in \mathbb{F}_{p^2}$
Output: $c = c_0 + c_1 i \in \mathbb{F}_{p^2}$
1: $s \leftarrow a_0 + a_1$
2: $t \leftarrow b_0 + b_1$
3: $d_0 \leftarrow s \cdot t$
4: $d_1 \leftarrow a_0 \cdot b_0$
5: $d_2 \leftarrow a_1 \cdot b_1$
6: $d_0 \leftarrow d_0 - d_1 - d_2$
7: $d_0 \leftarrow d_1 - d_2$
8: $c_0 = d_0 \bmod p$
9: $c_1 = d_1 \bmod p$
10: **return** $c = c_0 + c_1 i$

Since for BN158 and BN254 all elements are represented with two leading zero bits, the additions in Lines 1 - 2 do not produce a carry flag. The multiplications in Lines 3 - 5 have results that are twice the size of their inputs. Correspondingly, the further calculations are performed on elements which are also twice as large, but do not involve any modular reductions. There are only two reductions modulo p that take place in Lines 8 - 9. Consequently, there are three multiplications, but only two full reductions, which leads to an improved performance.

Contrary to the original multiplication in $\mathbb{F}_{p^2}$, this algorithm does not use a Montgomery multiplication that has an integrated reduction, but performs three full multi-precision integer multiplications and two separate Montgomery reductions in the end. Since this is also optimized using assembly language, the size of the program memory increases significantly, because both the Montgomery multiplication with integrated reduction and the separated multiplication and Montgomery reduction have their loops unrolled. Therefore, the FIPS method is replaced by a sequence of multi-precision integer multiplication and Montgomery reduction as the multiplication algorithm in $\mathbb{F}_p$, which in turn reduces the impact on the size of the program memory. The drawback of this solution is, that even though pairings and operations in $\mathbb{F}_{p^2}$ become faster, operations in elliptic curve groups over $\mathbb{F}_p$ are a bit slower, because the separated multiplication method involves a higher amount of memory accesses.

6.4.3. Pairing Computation

In Algorithm 6, negative BN parameters involved an additional inversion in $\mathbb{F}_{p^{12}}$, namely f^{-1} following the Miller loop. Aranha et al. [Ara+11, Section 5.1] showed how to avoid this expensive inversion and replace it by a simple conjugation instead. Consequently, an optimal Ate pairing can be computed as

$$a_{opt}(Q,P) = [g^{-1} \cdot h]^{\frac{p^{12}-1}{n}} = [g^{p^6} \cdot h]^{\frac{p^{12}-1}{n}},$$

where $s = 6u + 2$, $g = f_{|s|,Q}(P)$ and $h = \ell_{[s]Q,\pi_p(Q)}(P) \cdot \ell_{[s]Q+\pi_p(Q),-\pi_{p^2}(Q)}(P)$.

There is another optimization that can be done in Algorithm 6: the evaluated line functions $\ell_{T,Q}(P)$ and $\ell_{T,T}(P)$ give results of the form $a = a_0 + a_1 z + a_3 z^3 \in \mathbb{F}_{p^{12}} = \mathbb{F}_{p^2}[z]/(z^6 - \xi)$, that is, the coefficients a_2, a_4, a_5 are zero. Therefore, multiplication of the intermediate Miller function f with the result of ℓ can be done using a more efficient multiplication method in $\mathbb{F}_{p^{12}}$, which exploits the sparse form of one of the operands. Recalling the equivalence of $a = a_0 + a_1 z + a_2 z^2 + a_3 z^3 + a_4 z^4 + a_5 z^5 \in \mathbb{F}_{p^{12}}$ and $(a_0 + a_3 z^3) + (a_1 + a_4 z^3)z + (a_2 + a_5 z^3)z^2$, the sparse result of the line function may be represented by the two elements $b_0 = a_0 + a_3 z^3 \in \mathbb{F}_{p^4}$ and $b_{1,0} = a_1 \in \mathbb{F}_{p^2}$. Then, the sparse outcome $b = b_0 + b_{1,0} z$ of the line function is multiplied with the intermediate Miller variable f according to Algorithm 11. It takes merely three multiplications in $\mathbb{F}_{p^4}$ and two multiplications of the form $\mathbb{F}_{p^4} \times \mathbb{F}_{p^2}$ compared to 9 multiplications in $\mathbb{F}_{p^4}$ using the common formulas for multiplication in cubic extensions.

Besides these optimizations during the computation of the Miller function, the final exponentiation also offers several possibilities to increase performance. As covered in Section 6.2.4, the final exponentiation consists of an easy and a hard part. The easy part is computed efficiently due to Frobenius. However, the hard part of the final exponentiation is definitely worth optimizing: both Karabina [Kar10] and Granger and Scott [GS10] showed how squarings, and hence exponentiations, in the hard part of the final exponentiation can be made more efficient. Both present formulas that take advantage of the structure

Algorithm 11 Sparse multiplication of the result b of the line function ℓ with the intermediate Miller variable f.

Input: $f, b \in \mathbb{F}_{p^{12}}$, where $f = f_0 + f_1 z + f_2 z^2$, $b = b_0 + b_{1,0} z$: $f_{0-2}, b_0 \in \mathbb{F}_{p^4}$, $b_{1,0} \in \mathbb{F}_{p^2}$
Output: $f \cdot b \in \mathbb{F}_{p^{12}}$

1: $d_0 \leftarrow f_0 \cdot b_0$
2: $d_1 \leftarrow f_1 \cdot b_{1,0}$
3: $v_0 \leftarrow f_1 + f_2$
4: $v_0 \leftarrow v_0 \cdot b_{1,0}$
5: $v_0 \leftarrow v_0 - d_1$
6: $v_0 \leftarrow v_0 \cdot z^3$
7: $v_0 \leftarrow d_0 + v_0$
8: $t_0 \leftarrow b_0 + b_{1,0}$
9: $f_1 \leftarrow f_0 + f_1$
10: $f_1 \leftarrow f_1 \cdot t_0$
11: $f_1 \leftarrow f_1 - d_0 - d_1$
12: $f_2 \leftarrow f_0 + f_2$
13: $f_2 \leftarrow f_2 \cdot b_0$
14: $f_2 \leftarrow f_2 - d_0 - d_1$
15: $f_0 \leftarrow v_0$
16: **return** f

of the so-called *cyclotomic subgroup* of $\mathbb{F}_{p^k}$, where k is the embedding degree. Therefore, Karabina does a compressed squaring, which uses a compressed representation of elements in the cyclotomic subgroup. Contrary to that, Granger and Scott simply provide more efficient formulas for squaring elements in the cyclotomic subgroup.

The cyclotomic subgroup $G_{\phi_k(p)}$ can be defined as

$$G_{\phi_k(p)} = \{\alpha \in \mathbb{F}_{p^k} \mid \alpha^{\phi_k(p)} = 1\},$$

where $\phi_k(p)$ denotes the k-th cyclotomic polynomial. It follows from the partition of the final exponentiation into an easy and a hart part in Equation 6.5, that any element $a \in \mathbb{F}_{p^k}$ is an element of the cyclotomic subgroup after computing the easy part of the final exponentiation, that is, $a^{(p^k-1)/\phi_k(p)} \in G_{\phi_k(p)} \subseteq \mathbb{F}_{p^k}$, because the order of $\mathbb{F}_{p^k}$ is $p^k - 1$.

In this thesis, the faster formulas from Granger and Scott are used for the hard part of the final exponentiation: let $\mathbb{F}_{p^4} = \mathbb{F}_{p^2}[u](u^2 - \xi)$ and $\mathbb{F}_{p^{12}} = \mathbb{F}_{p^4}[v]/(v^3 - \sqrt{\xi})$, the square $b = b_0 + b_1 v + b_2 v^2$ of an element $a = a_0 + a_1 v + a_2 v^2 \in G_{\phi_{12}(p)}$ is computed as

$$\begin{aligned} b_0 &= 3a_0^2 - 2\overline{a_0}, \\ b_1 &= 3\sqrt{\xi}a_2^2 + 2\overline{a_1}, \\ b_2 &= 3a_1^2 - 2\overline{a_2}, \end{aligned}$$

where $\overline{a}$ denotes the conjugate of a. Note that these formulas only need three squarings in $\mathbb{F}_{p^4}$ compared to four in the formulas by Chung and Hasan on page 56. Further, the squaring formulas presented are also used when operating on values that result from pairings, that is, elements in $\mathbb{G}_t$, since these are contained by the cyclotomic subgroup.

Analysis of the stack trace of the pairing computation revealed that the majority of the memory needed results from the final exponentiation. Consequently, optimizing this computation in terms of memory turned out to be important. One thing that immediately helped to reduce the memory footprint is splitting the Miller loop and the final exponentiation into two functions. By applying this trick, variables that become unnecessary after the Miller algorithm are removed from the stack when leaving the function. However, the biggest impact on memory is due to the large elements in $\mathbb{F}_{p^{12}}$. To avoid repeated exponentiations by the BN parameter u, several elements in $\mathbb{F}_{p^{12}}$ are held in memory during the hard part of the final exponentiation. While keeping the computation similarly fast, the number of those elements in memory was reduced by adapting the formulas by

Algorithm 12 Memory-optimized hard part of the final exponentiation for pairings over BN curves.

Input: $f \in \mathbb{F}_{p^{12}}$
Output: $f^{\phi_{12}(p)/n} \in \mathbb{F}_{p^{12}}$
1: $t_0 \leftarrow f^p$
2: $b \leftarrow f^u$
3: **if** $u < 0$ **then** $b \leftarrow \overline{b}$
4: $b \leftarrow b^2$
5: $a \leftarrow b^2$
6: $a \leftarrow a \cdot b$
7: $b \leftarrow b \cdot f$
8: $b \leftarrow \overline{b}$
9: $f \leftarrow f \cdot t_0$
10: $t_0 \leftarrow a^u$
11: **if** $u < 0$ **then** $t_0 \leftarrow \overline{t_0}$
12: $f \leftarrow f \cdot t_0$
13: $a \leftarrow a \cdot t_0$
14: $t_0 \leftarrow t_0^2$
15: **if** $u < 0$ **then** $t_0 \leftarrow \overline{t_0}$
16: $a \leftarrow a \cdot t_0^u$ ▷ Interleaved
17: $b \leftarrow b \cdot a$
18: $t_0 \leftarrow b^p$
19: $t_0 \leftarrow t_0 \cdot a$
20: $t_0 \leftarrow t_0^p$
21: $t_0 \leftarrow t_0 \cdot b$
22: $t_0 \leftarrow t_0^p$
23: $t_0 \leftarrow t_0 \cdot f$
24: $f \leftarrow t_0 \cdot a$
25: **return** f

Fuentes-Castañeda, Knapp, and Rodríguez-Henríquez [FKR12]. Initially $t_0 = f^p$ and the following chain are computed:

$$f^u \rightarrow f^{2u} \rightarrow f^{4u} \rightarrow f^{6u} \rightarrow f^{6u^2} \rightarrow f^{12u^2} \rightarrow f^{12u^3}.$$

Consequently, a and b are set to $a = f^{6u} \cdot f^{6u^2} \cdot f^{12u^3}$ and $b = a \cdot (f^{2u} \cdot f)^{-1}$. Afterwards, one can compute the result

$$f = f^{6u^2} \cdot f \cdot f^p,$$
$$f = [f \cdot a][b]^p[a]^{p^2}[b]^{p^3}.$$

This computation takes one multiplication and one frobenius action more than the original version by Fuentes-Castañeda, Knapp, and Rodríguez-Henríquez. However, by performing Algorithm 12, only three temporary variables are needed instead of four. Note that the exponentiation and multiplication on Line 16 is done in one single step using an especially crafted function. Since the operand being exponentiated is not needed any more, this can be done without any further memory needs. The newly introduced function has only little impact on the size of the program memory: merely additional 50 byte are needed. Note that on the other hand, this method saves 384 byte on the stack for BN254.

6.5. Instruction-Set Extension

Most of the runtime of cryptography based on elliptic curves is spent performing multiplications in the prime field $\mathbb{F}_p$. Although inversions in $\mathbb{F}_p$ are more expensive, their effect on runtime is relatively small since these can mostly be avoided at the cost of additional multiplications. The remaining basic operations, such as modular addition, subtraction and negation, are significantly faster than multiplication and have comparably little impact on overall performance. Therefore, optimization of the multiplication in $\mathbb{F}_p$ can speed up the implemented system a lot.

Much effort has already been put into assembly optimization of the Montgomery multiplication as shown in Section 6.4.1. However, as Algorithm 8 indicates, the multiplier that comes with the Cortex-M0+ is not ideal. Even though a single `mul` instruction is done in one cycle using the bit-parallel multiplier, performance is rather mediocre: for a full 32 bit $\times$ 32 bit $\rightarrow$ 64 bit multiplication, four of these `mul` instructions have to be performed since the resulting product of the two 32-bit operands is only 32 bits. In addition, the accumulation and shifting overhead is tremendous. The multiply-accumulate routine listed in Algorithm 8 constitutes the real bottleneck of the prime field computation, which in turn limits the performance of pairings, elliptic curve operations, and identity-based encryption.

In order to further improve the prime field multiplication and consequently everything else, two versions of a multiply-accumulate instruction-set extension were designed for the Cortex-M0+. The first instruction-set extension, *MAC-1*, performs both 32 bit $\times$ 32 bit $\rightarrow$ 64 bit multiplication and accumulation in a single cycle. Using a 130 nm process technology, it increases the processor's regular size of 15,850 GE by 3,500 GE. The second extension, *MAC-2*, does the same as *MAC-1* in four cycles, which reduces the amount of additional hardware needed to merely 1,300 GE. Nevertheless, it achieves comparable performance since the multiply-accumulation step can be done parallel to loading new operands.

6.5.1. MAC-1

The one-cycle multiply-accumulate extension *MAC-1* is illustrated by Figure 6.3. Basically, the existing multiplier is extended to perform full 32 bit $\times$ 32 bit $\rightarrow$ 64 bit multiplications. The result may then be added to an accumulator. By default, the `mul` instruction performs the standard 32 bit $\times$ 32 bit $\rightarrow$ 32 bit multiplication. Therefore, the two operands `opA` and `opB` are selected from the range of registers using the corresponding select signals. Following, the multiplication is done and the result is stored in the destination register $\texttt{r}_{\texttt{dest}}$, which is defined by the instruction. The signal `doMultiply` triggers such a multiplication and is also used for operand isolation.

The multiply-accumulate unit can be activated by setting a bit in the processors control register using the `msr` instruction. The respective bit is represented by the control signal `doMAC` in Figure 6.3. When this signal is set, the `mul` instruction is interpreted as a multiply-accumulate. At first, the two operands `opA` and `opB` are selected. Subsequently, the result is not stored in the destination register $\texttt{r}_{\texttt{dest}}$, but added to the accumulator that is represented by the registers `r8-10`. The least significant word of the accumulator is `r8`, and the most significant word `r10`. The result of the addition is stored back to the accumulator.

Using this extension, the routine from Algorithm 8 simplifies to Algorithm 13. Note that the destination register used in the `mul` instruction is ignored in that case. In addition, the pointers to the operands can be stored in one of the low registers, avoiding the additional `mov` instructions. Typically, the multiply-accumulate unit is activated once at the beginning of the multiplication routine, and deactivated once at the end of it. However, the Montgomery multiplication also needs a standard multiplication modulo the word size when performing the multiplication by the value $p_0[0]$. At this point, the multiply-accumulate unit has to be temporarily deactivated. This may seem a bad compromise, but on the other hand, any standard compiler can be used in such case.

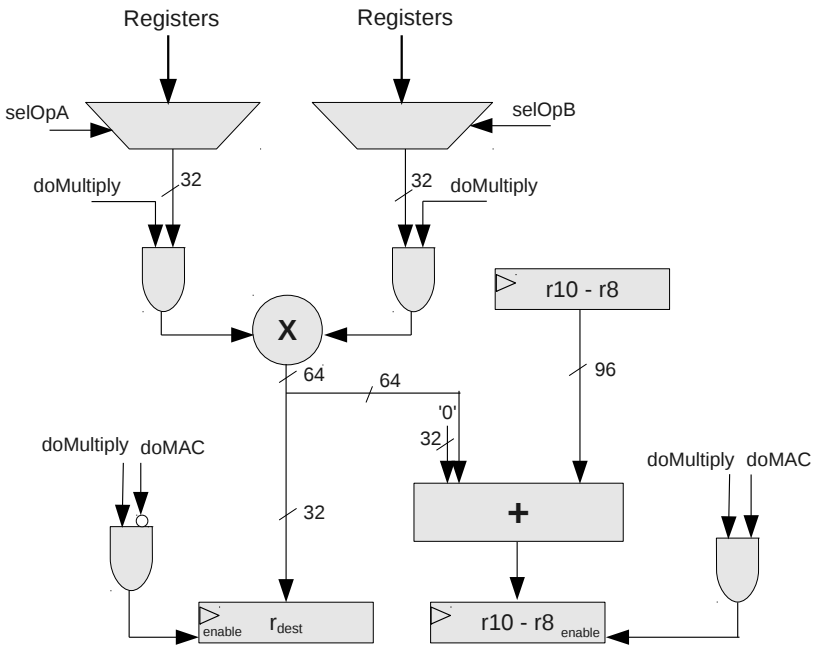

Figure 6.3.: Hardware design of the multiply-accumulate extension *MAC-1*.

Algorithm 13 Multiply-Accumulate routine on the Cortex-M0+ processor using the multiply-accumulate extension.

Input: `r1, r2` are pointers to the operands
Output: {`r10, r9, r8`} is the accumulator

```
1: ldr  r5, [r1, #offset1]
2: ldr  r6, [r2, #offset2]
3: mul  r5, r5, r6
```

6.5.2. MAC-2

Figure 6.4 shows the four-cycle multiply-accumulate extension *MAC-2*. Just as the standard processor, it incorporates merely a 32 bit × 32 bit → 32 bit multiplier. As for the extension *MAC-1*, a normal multiplication can be performed. Therefore, the two operands `opA` and `opB` are selected by their respective select signals and the `selCache` signal is set to use the two operands. In this scenario, the two modules *Shift+Mask* remain the two signals unchanged. Following this, the two values are multiplied and the result is stored in the destination register $\mathtt{r}_{\mathtt{dest}}$. The signal `doMultiply` triggers the multiplication and is used for operand isolation to reduce power consumption if the unit is not in use.

A bit in the control register, represented by `doMAC`, is set to activate the multiply-accumulate unit. Its operation is also triggered by the `doMultiply` signal. In its first cycle, the two operands are selected according to the registers defined in the `mul` instruction. In addition, these are stored in the respective operand caches, which is controlled by the signal `storeOp`. This is necessary, since the multiplication is performed as a background task over four cycles and another instruction that is performed might otherwise change one of the

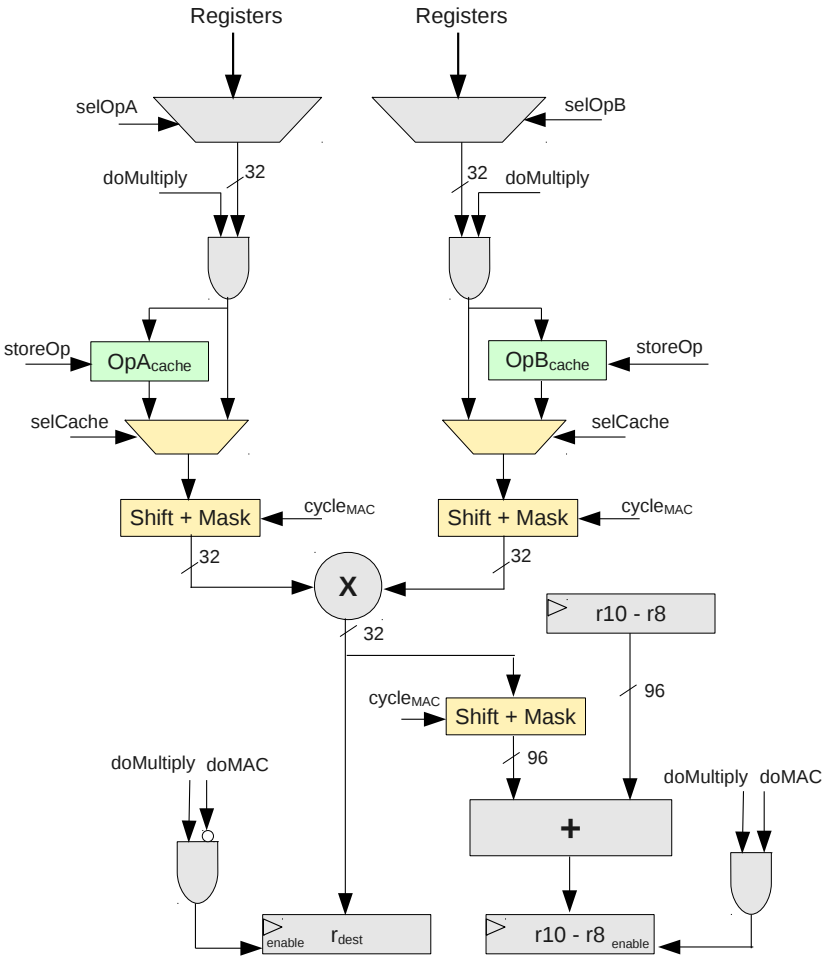

Figure 6.4.: Hardware design of the multiply-accumulate extension *MAC-2*.

operands in cycles 2-4. The unit *Shift+Mask* extracts 16-bit parts from the two operands and aligns them correctly. This is basically the same as what is done using standard instructions in Lines 5-8 of Algorithm 8. Consequently, the 32 bit $\times$ 32 bit $\rightarrow$ 32 bit multiplier is used as a 16 bit $\times$ 16 bit $\rightarrow$ 32 bit multiplier in this case. The resulting product is then aligned correctly by the successive *Shift+Mask* unit and added to the accumulator, which is, here too, represented by the registers `r8-r10`. The corresponding sum is stored again in registers `r8-r10`. In the remaining three cycles, basically the same is done, but other instructions might be performed in parallel. For example, the operands for the next multiplication could be loaded from the memory. To assure constant input during all four cycles, the operands have to be selected from the two operand caches in cycles 2-4. In these three cycles, the units *Shift+Mask*, which are controlled by the signal `cycleMAC`, successively shift and align the operands and the result such that all necessary multiplications are done and accumulated.

Since loading the operands can be done parallel to the multiply-accumulate instruction, one would expect the same performance for *MAC-2* as for *MAC-1*. However, whenever the accumulator has to be shifted or to be stored, `nop` instructions must be inserted manually

to wait for the result of the multiply-accumulate to be ready. Besides this, Algorithm 13 can be used. Regarding the (de-)activation of the multiply-accumulate instruction, the same applies for this extension as for the extension *MAC-1*.

6.6. Conclusion

In this section, both the hardware and the software for the identity-based encryption scheme in embedded environments was investigated. In the beginning, clear goals were defined that should be fulfilled, namely performance, side-channel security and keeping the demand for resources low. Following, the microprocessor-based hardware platform and the Cortex-M0+ processor were introduced. After giving an overview on the software's architecture, the focus was turned to the concrete implementation. From basic computations in a prime field, successively the implementation aspects of extension fields, elliptic curve groups, and pairings were illustrated. Next, the approach to testing was outlined. Besides assembler optimizations of prime field arithmetic, the following section covered several optimization tricks at different layers. Finally, two variants of a multiply-accumulate instruction-set extension were presented that help to speed up the prime field multiplication, and consequently elliptic curve operations, pairings, and identity-based encryption as well.

7. Side-Channel Analysis

The identity-based encryption scheme has the requirement to withstand several of the attacks presented in Chapter 5, specifically timing, simple power analysis, and differential power analysis attacks. Partially, the measures to fulfill these requirements have already been discussed in the previous chapter. Nevertheless, a detailed investigation of side-channel security is done in this chapter. In Section 7.1, the security of the encapsulation routine is analyzed, while Section 7.2 points out the vulnerability of the decapsulation routine against differential power analysis attacks. As a result, a simple countermeasure is presented.

7.1. Encapsluate

The encapsulation routine of the implemented identity-based encryption scheme BB_1-KEM from Section 4.3 is reformulated in Algorithm 14. As pointed out in the previous section, all algorithms were designed to consume constant runtime, that is, the hash functions, the modular multiplication on Line 3, the elliptic curve operations on Lines 4-7, and the exponentiation on Line 8 have runtime independent of the data being processed. The security parameter is denoted κ.

Algorithm 14 BB_1 IBE KEM encapsulation routine.

Input: $id \in \{0,1\}^*, \kappa$
Output: $K \in \{0,1\}^\ell, C \in E(\mathbb{F}_p) \times E(\mathbb{F}_p)$
1: $s \leftarrow \texttt{rand}(\kappa)$
2: $h_{id} \leftarrow H'(id)$
3: $t \leftarrow h_{id} \cdot s \bmod n$
4: $C_1 \leftarrow s \cdot G_3$
5: $C_0 \leftarrow t \cdot G_1$
6: $C_1 \leftarrow C_0 + C_1$
7: $C_0 \leftarrow s \cdot G$
8: $K \leftarrow H'(v_0^s)$
9: **return** $(K, (C_0, C1))$

With respect to the various types of attacks mentioned in Chapter 5, security of the encapsulation routine is given as follows:

Timing attacks are not possible since any operations performed have constant runtime, that is, the operations in the finite field have constant runtime. In addition, exponentiation in the extension field and elliptic curve point multiplication were implemented using a Montgomery ladder that yields constant runtime if the most significant bit of the exponent or the factor respectively is set to one.

Simple power analysis cannot be applied to the point multiplications and the exponentiation, because the combination of constant runtime execution of finite field arithmetic

and Montgomery ladders results in regular sequences in the power trace that are indistinguishable by visual inspection. However, the computation of the parameter t may pose a vulnerability.

Differential power analysis, refined power analysis do not work as the number s is chosen randomly at each invocation. For the same reason, the address-bit DPA cannot be performed as well.

Comparative power analysis may be possible as two point multiplications with the same value s are done. Nevertheless, the doubling attack is not feasible, because a new random secret s is used at each invocation.

Safe-error attacks cannot be successfully applied, because the random secret s changes at each invocation.

Electromagnetic attacks depend on the attackers capability to determine which memory locations are accessed at certain points of time.

Template attacks may possibly be used to extract the random secret s as indicated by Herbst and Medwed [HM09].

For all these types of attacks, the interesting target is the random number s, which leaks the session key. On the other hand, a session key alone may not be as interesting as someones private key, which can be used to reveal any of the ciphertexts intended for the respective party. This type of attack may be done during decapsulation.

7.2. Decapsulate

Algorithm 15 states the routine for decapsulation of the ciphertext C of the BB_1 identity-based encryption scheme from Section 4.3. This routine is more critical than the encapsulation routine since it does not involve random numbers by default and relies on the identity's secret private key D_{id} that must not be compromised.

Algorithm 15 BB_1 IBE KEM decapsulation routine.

Input: $C \in E(\mathbb{F}_p) \times E(\mathbb{F}_p), D_{id} \in E(\mathbb{F}_{p^2}) \times E(\mathbb{F}_{p^2})$
Output: $K \in \{0,1\}^\ell$,
1: $T \leftarrow -C_1$
2: $f \leftarrow \texttt{optate\_mul}(C_0, D_{id,0}, T, D_{id,1})$ ▷ $a_{opt}(C_0, D_{id,0}) \cdot a_{opt}(T, D_{id,1})$
3: $K \leftarrow H'(f)$
4: **return** K

The routine $\texttt{optate\_mul}(a, b, c, d)$ simply performs two optimal Ate pairings $a_{opt}(a, b)$ and $a_{opt}(c, d)$ and multiplies their result. Note that this implementation is optimized such that the squarings in $\mathbb{F}_{p^{12}}$ during the Miller loop are shared between both pairing computations. Further, there is only one single final exponentiation to be performed for both pairings. From Algorithm 6 and the line evaluation formulas in Appendix A, one can see that the computation of a bilinear pairing has constant runtime since the underlying finite field and elliptic curve operations have constant runtime. Further, the branches in Algorithm 6 are admissible, because the parameter $s = |6u + 2|$ that controls the loop is constant for a specific implementation. Consequently, we argue that the implementation of the decapsulation routine is secure against timing and attacks.

7.2.1. Differential Power Analysis Attack

A different picture unveils when investigating the security of the decapsulation routine against differential power analysis attacks. In more detail, a DPA attack could be performed in practice for the decapsulation routine in Algorithm 15. At first, the attack is described in general. Following, the concrete setup as well as the results of a practical attack are explained. Finally, a countermeasure to this type of DPA attack is proposed.

Attack Description

Decapsulation of a ciphertext C in the BB_1 IBE KEM involves the computation of the two pairings $a_{opt}(C_0, D_{id,0})$ and $a_{opt}(-C_1, D_{id,1})$. The attack is capable of extracting D_{id} from these two computations. However, the two points $D_{id,0}$ and $D_{id,1}$ have to be extracted separately. Therefore, the attack is first described for the point $D_{id,0}$ only.

When computing the optimal Ate pairing $a_{opt}(C_0, D_{id,0})$ according to Algorithm 6, the homogeneous projective point T is initialized with $(D_{id,0,x}, D_{id,0,y}, 1)$ in Line 1, where $D_{id,0,x}$ and $D_{id,0,y}$ denote the x- and y-coordinate of the point $D_{id,0}$. In Line 3 of the Miller loop, the tangent line in T is evaluated in the point P, which equals C_0 in this scenario. Therefore, the formulas from Appendix A are used, which involve the computation of $L_{1,0}P_x$. In the first iteration of the Miller loop, this evaluates to $3 \cdot D_{id,0,x}^2 \cdot C_{0,x}$, which seems perfectly suitable to mount a DPA attack.

At first, the value $L_{1,0} = 3 \cdot D_{id,0,x}^2 \in \mathbb{F}_{p^2}$ is computed on the cryptographic device based on the private key's x-coordinate. The value $L_{1,0}$ is then multiplied with the x-coordinate of the input ciphertext, namely $C_{0,x} \in \mathbb{F}_p$. In a typical scenario, the attacker can freely choose the input ciphertext that is used for the decapsulation routine. Consequently, power traces of the finite field multiplication $L_{1,0} \cdot C_{0,x}$ can be collected for an arbitrary number of different inputs. Following, statistical methods are applied to extract the unknown intermediate $L_{1,0}$. Note that the finite field multiplication $L_{1,0} \cdot C_{0,x}$ is actually two prime field multiplications in $\mathbb{F}_p$, which have to be attacked separately.

Once the value $L_{1,0}$ is revealed, one can simply recover $D_{id,0,x}$ by computing $\sqrt{L_{1,0}/3} \in \mathbb{F}_{p^2}$. If one of the $\mathbb{F}_p$-coefficients of the value $L_{1,0} \in \mathbb{F}_{p^2}$ is not immediately divisible by three, one simply adds the modulus p once or twice such that the coefficient becomes divisible by three. The square root in $\mathbb{F}_{p^2}$ is computed using a specialized version of the Tonelli-Shanks algorithm as shown by Adj and Rodríguez-Henríquez [AR12, Algorithm 9]. The respective y-coordinate of the private key $D_{id,0}$ can be calculated using the curve equation $y^2 = x^3 + b$.

In a similar manner, the value $D_{id,1}$ is revealed from the pairing computation $a_{opt}(C_0, D_{id,1})$. In particular, the two prime field multiplications in $L_{1,0} \cdot (-C_1)_x$, where $L_{1,0} = 3 \cdot D_{id,1,x}^2$, recover the second point of the private key. Afterwards, the full private key of the attacked identity is known.

Practical Attack

The attack was performed using a Sasebo G board [UA13] that comes with two Xilinx Virtex-II pro FPGAs. Therefore, the hardware platform described in Section 6.1.1 was deployed to the control FPGA—the Xilinx Virtex-II pro xc2vp30—and connected to the PC using the RS232 interface. A MATLAB Side-Channel Analysis (SCA) toolbox was used to communicate with the cryptographic device and to retrieve the power traces from the LeCroy LC584 oscilloscope. In order to trigger the oscilloscope correctly, a trigger

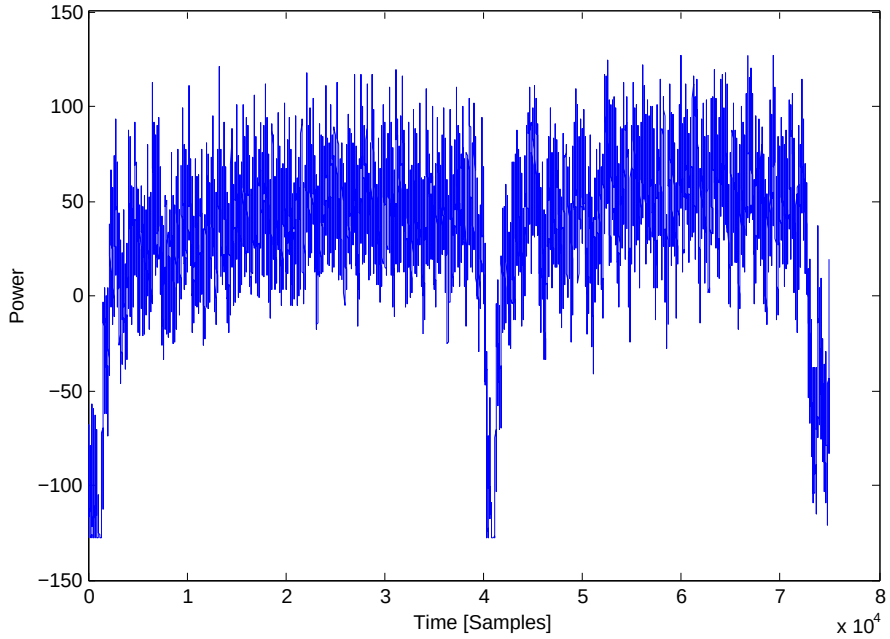

Figure 7.1.: Power trace of a multiplication in $\mathbb{F}_p$ at 250 MS/s sampling rate and 25 Mhz clock frequency.

signal was sent from the FPGA to the oscilloscope just before the attacked prime field multiplication takes place. The power consumption was measured on an 1 Ω resistor on the line from the FPGA core to VCC using a differential probe. The FPGA was operated at a frequency of 25 Mhz such that the sampling rate of the oscilloscope is a multiple of the clock frequency.

A power trace of a prime field multiplication is shown in Figure 7.1. There are two periods of higher power consumption visible, which correspond to the multiplication of the two operands and to the reduction step. Both the multiplication and the reduction step are implemented using product scanning. Two such prime field multiplications have to be analyzed in order to successfully reveal the x-coordinate of the point $D_{id,0} \in E(\mathbb{F}_{p^2})$ of the private key $D_{id} \in E(\mathbb{F}_{p^2}) \times E(\mathbb{F}_{p^2})$, In particular, the multiplications $L_{1,0,0} \cdot C_{0,x} \in \mathbb{F}_p$ and $L_{1,0,1} \cdot C_{0,x} \in \mathbb{F}_p$ are attacked, where $L_{1,0} \in \mathbb{F}_{p^2}$ denotes the intermediate value that is derived from the private key's x-coordinate $D_{id,0,x}$, and $C_{0,x}$ denotes the x-coordinate of the first point of the ciphertext. Attacking a prime field multiplication was described by Hutter et al. [Hut+09]. Basically, the partial products that appear in the multi-precision integer multiplication of the secret intermediate $L_{0,1}$ and the known ciphertext $C_{0,x}$ are attacked to reveal the single words of the secret value.

Let $a[i]$ denote the i-th word of a multi-precision integer a. To attack the i-th word of $L_{1,0,0}$, the product of $C_{0,x}[0]$ and $L_{1,0,0}[i]$ is computed for n different inputs of C and for every possible value of $L_{1,0,0}[i]$. Note that the key space for a single word of $L_{1,0,0}$ consists of 2^{32} possibilities. Since we use 16-bit multiplications on the Cortex-M0+, the attack is simplified by attacking only the 16-bit half-words of $L_{1,0,0}$. This reduces the key-space to 2^{16} choices, but doubles the number of attacks to be performed to reveal the whole secret intermediate $L_{1,0,0}$. This leads to the following reformulation of the attack.

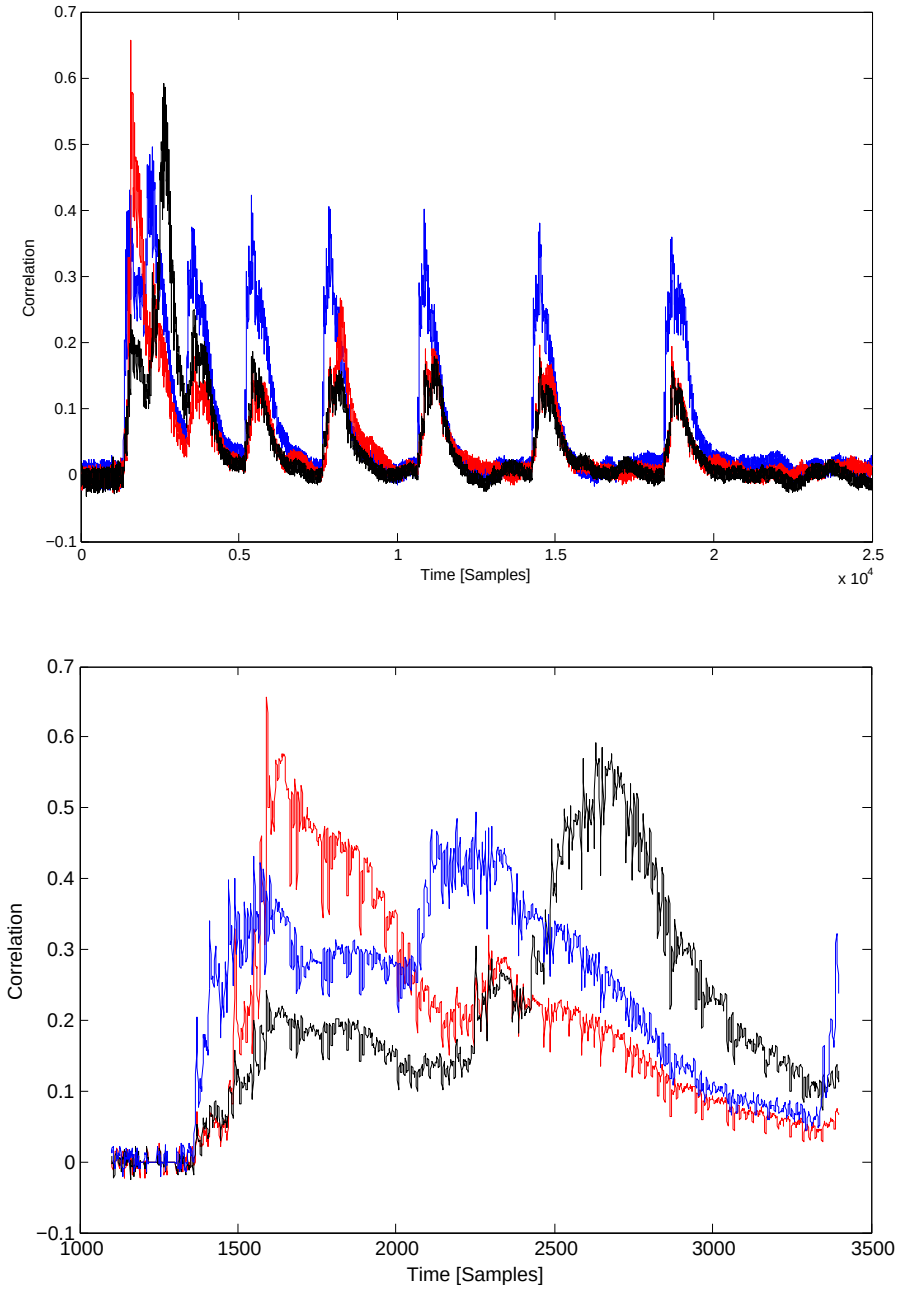

Figure 7.2.: (Zoomed) Correlation of three different key candidates over time: red and black mark the correct hypothesis for half-words 0 and 3, respectively, and blue depicts a wrong hypothesis.

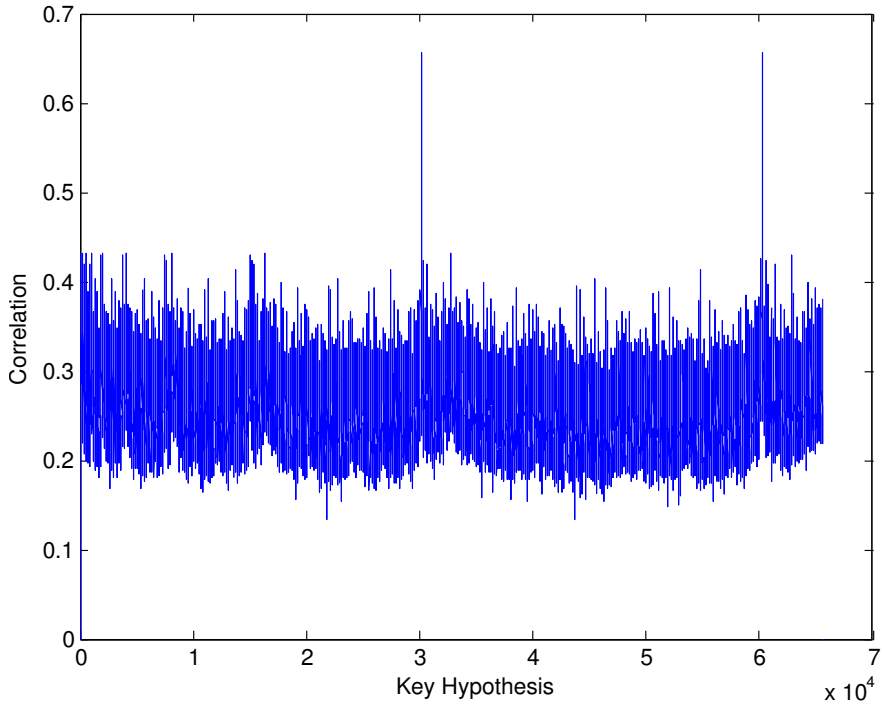

Figure 7.3.: Correlation as a function of the hypothesis for half-word zero.

Let now denote $a[i]$ the i-th **half-word** of the multi-precision integer a. In order to attack the i-th half-word of $L_{1,0,0}$, the product of $C_{0,x}[0]$ and $L_{1,0,0}[i]$ is computed for n different inputs of C and for every possible value of $L_{1,0,0}[i]$. The hamming weights of these products are stored in a matrix of size $n \times 2^{16}$, which constitutes the hypothesis matrix. For each of the n different inputs of C, a power trace is measured using the aforementioned setup. This results in a trace matrix of size $n \times t$, where t denotes the number of samples captured in the power trace. Computing a Pearson correlation of those two matrices gives a correlation matrix of size $2^{16} \times t$. This matrix indicates the correlation for each key hypothesis at every point of time. Further, the hypothesis with the highest correlation peak can be extracted and is assumed to be the correct one.

Figure 7.2 illustrates the correlation of three different key hypothesis over time when attacking partial products of $16 \times 16 \rightarrow 32$ bit multiplications. The hypothesis marked by the blue line suggests that there is some relevant data processed, but does not have a peak high enough to conclude that the hypothesis is correct. Using this correlation trace, one can simply find the positions in the trace, where the multiplications of interest take place. The remaining two lines have peaks that are significantly higher. In particular, one can see that the red line has its highest peak when the multiplication of the least-significant half-word takes place. It marks the correct key hypothesis for half-word zero. Similarly, the black line has its highest peak when the multiplication of half-word three is done. For this and the following side-channel analysis, 10,000 power traces were used.

The correlation traces over time have been processed by extracting their maximum peak for each of the partial products. For example, from Figure 7.2 it is known that the partial product of half-word zero of the input and half-word zero of the unknown intermediate

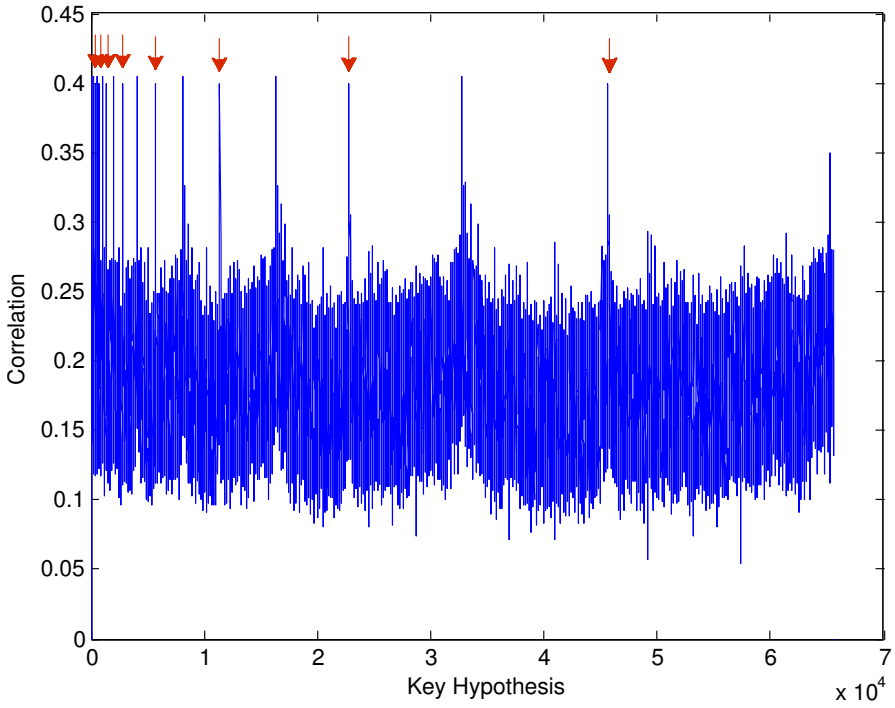

Figure 7.4.: Correlation as a function of the hypothesis for half-word eight.

is computed somewhere between samples 1300 and 1700. Hence, the maximum peak for each hypothesis is extracted from this range in order to find the correct hypothesis for half-word zero. The result of this evaluation leads to the interpretation of correlation as a function of the hypothesis as it is shown in Figure 7.3. This figure clearly shows two peaks that mark two key candidates for half-word zero. The same is done for the remaining half-words as well.

Figure 7.4 additionally shows the result for half-word eight. This plot obviously has many more peaks. The correct ones are marked by red arrows. The remaining peaks are false positives caused by powers of two that clearly have some correlation. However, there are still eight key candidates. These indicate shifts of the correct hypothesis. Since multiplication is a linear operation, the result of a multiplication with a shifted operand results in the same hamming weight as multiplication with the original operand if only zeros are shifted out. Therefore, these shifted versions of the correct hypothesis also correlate. The two peaks in Figure 7.3 are also shifted variants of the same value. This becomes even more obvious when the scale of the x-axis is considered. The numeric value of the hypothesis causing the left peak is half of the numeric value of the hypothesis causing the right peak. This is equivalent to a shift by one bit. Similar observations can be made in Figure 7.4. In the worst case, one has 16 valid key candidates that are shifted equivalents for each half-word. For a 256-bit prime, this results in $16^{16} = 2^{64}$ possible keys, which is not satisfying. Therefore, the partial sums are considered as well.

Using the partial products, 1-16 key candidates can be filtered out from 2^{16} for each half-word. These can then be used to attack two partial sums occurring in the multi-precision integer multiplication. Let a and b denote two multi-precision integers consisting

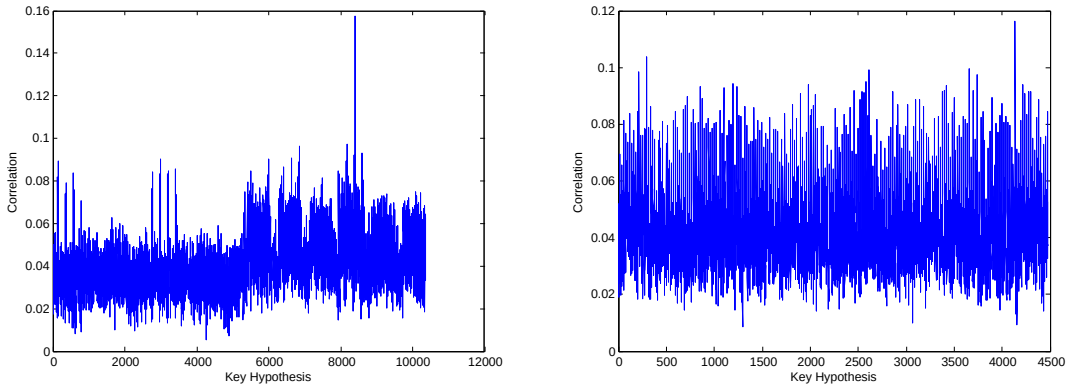

Figure 7.5.: Correlation as a function of the hypothesis when attacking the sums.

of each eight 32-bit words and let $a[i]\ \forall\ 0 \leq i < 8$ denote the i-th word of a. In a product-scanning multiplication, the two sums $a[0]b[3] + a[1]b[2] + a[2]b[1] + a[3]b[0]$ and $a[4]b[7] + a[5]b[6] + a[6]b[5] + a[7]b[4]$ appear as intermediate results in the accumulator. Using the appropriate key candidates learned from attacking the partial products and the n different ciphertexts, two more hypothesis matrices can be built based on the hamming weights of these two sums. These matrices are correlated with the power traces already captured and thus a DPA is performed that reveals the two 128-bit halves of the full 256-bit value in $\mathbb{F}_p$.

The correlations resulting from attacking the two sums are shown in Figure 7.5. For the lower four words of the 256-bit value under attack, a clear peak is given. The upper four words also give a peak, but less prominent. Note that the two results involve a different number of possible keys on the x-axis as the hypothesis is built from the key candidates found during the attack of the single half-words. Further, the numbers do not give a hint about the numerical value of the concrete key, but refer to the indices of a matrix containing the full 128-bit hypothesis. The correlation was computed from 10,000 power traces. It was also tried reducing the number of used power traces, but the upper four words could not be extracted with 9,000 or less power traces.

Attacking the partial products to reveal the single half-words of the unknown intermediate is possible using a smaller number of power traces though. Figure 7.6 shows the correlation in dependence of the key hypothesis when attacking half-word zero using 10,000, 5,000, 1,000, 500, 300 and 100 power traces. For more than 1,000 traces, one cannot recognize any difference. When using 500 power traces, the peaks are already slightly smaller. This effect is even more significant for 300 traces. Performing the DPA with merely 100 traces gives peaks indeed, but these are hardly recognized as the right ones. A different picture may unveil when doing the same experiment for a different half-word, for example, half-word eight as previously shown in Figure 7.4. There, it may become increasingly hard to recognize the right peaks even when using more than 500 power traces. In addition, more false positives may appear in this analysis.

Countermeasure

In order to make this kind of DPA attack impossible, a simple countermeasure has been created. Earlier, it was mentioned that the point $T \in E(\mathbb{F}_{p^2})$ in the formulas in Ap-

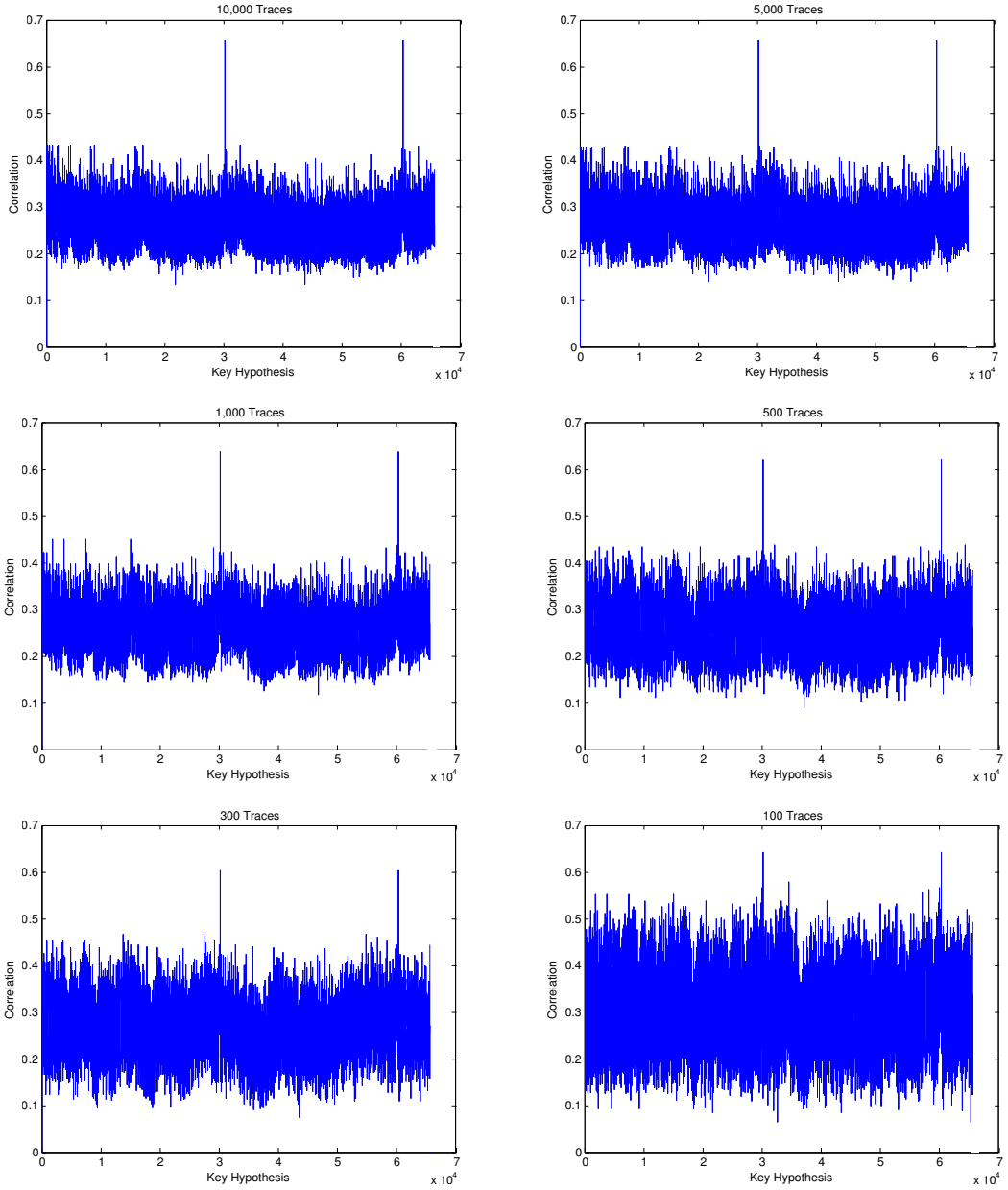

Figure 7.6.: Correlation as a function of the hypothesis when attacking the multiplication with half-word zero of the unknown intermediate using different numbers of power traces.

pendix A equals parts of the identity's private key when computing the first loop of the optimal ate pairings of the decapsulation routine. This makes possible the DPA attack and is due to the initialization of the point T with the parameter Q in Line 1 of Algorithm 6.

Instead of initializing this homogeneous projective point as $T = (Q_x, Q_y, 1)$, one simply generates a random number λ, which is used as the z-coordinate in the representation of homogeneous projective coordinates. Consequently, the initialization is done as $T = (Q_x \cdot \lambda, Q_y \cdot \lambda, \lambda)$, which costs merely two multiplications in $\mathbb{F}_{p^2}$. The point Q, which is set to the identity's private key, is hence randomized and invulnerable to this attack. Note that the point P is always used in affine coordinates and is not randomized at all. In the scenario of this thesis, the point P is always set to the publicly known ciphertext, which is why not randomizing P does not constitute a problem. If the point P was set to a private secret, it would also have to be randomized. However, this adaption is more difficult since the formulas in Appendix A work with P in affine coordinates.

7.2.2. Other Attacks

As already mentioned, timing attacks are not possible since everything has constant runtime. DPA attacks are prevented by the just introduced countermeasure. Safe-error and address-bit DPA attacks are not an issue, because there are no dummy operations involved in the pairing computation. Electromagnetic attacks are restricted in the same way as power analysis attacks, because one cannot learn anything from memory accesses alone in the context of pairing computations. However, attackers might recover the secret point if they are able to recognize the words transferred on the bus. Refined power analysis and zero-value attacks are possible, but are expected to take the same effort as brute-force as it is not possible to attack single bits consecutively. Moreover, security against simple power analysis attacks is assumed since power traces should always reveal the same patterns. This, however, was not tested and remains an open question. Further, template attacks may be possible by first building a template that recovers the random value as shown by Herbst and Medwed [HM09].

7.3. Conclusion

This chapter was dedicated to analyzing the security of the presented implementation of the BB_1 IBE KEM. At first, the encapsulation routine was investigated, where merely comparative side-channel attacks and template attacks seem feasible. In the next step, analysis of the decapsulation routine revealed a possible DPA attack that allows extraction of the identity's private key. To counteract this, the parameter $Q \in E(\mathbb{F}_{p^2})$ in the optimal Ate pairing has to be randomized. It is a lucky coincidence that this can be easily done for the parameter Q, but involves adaption of a whole range of evaluation formulas for the point $P \in E(\mathbb{F}_p)$. However, the input to point P is public in this application and the formulas can be kept as they are. Template attacks may again pose a threat to the security of the decapsulation routine. The original goal to have an implementation secure against timing and DPA attacks is fulfilled. However, assessment of the security against SPA attacks and comparative side-channel attacks needs a more detailed analysis, which may be subject to future work.

8. Evaluation

In the previous chapters, the concept of bilinear maps and identity-based encryption was introduced. The respective implementation in a constrained environment had to satisfy several conditions, such as good runtime, low demand for resources and side-channel security. The attainment of the latter has been investigated in Chapter 7. Evaluation of the achieved performance and the implementation's demand for resources is content of this chapter. As pointed out earlier in this work, the finite field arithmetic is crucial for having reasonable performance. Therefore, the results in this respect are looked at first in Section 8.1. Following, Section 8.2 investigates the efficiency of the pairing implementation for both the software implementation and the whole hardware platform. Consecutive to analyzing the performance of the BB_1 IBE KEM's implementation in Section 8.3, Section 8.4 does a comparison with related work to see how the platform presented here ranks.

8.1. Finite Field Arithmetic

An overview of the cycle counts of addition, multiplication and squaring in the finite fields used for the curves BN158, BN254 and BN256 is given in Table 8.1. Although the same number of words are needed to represent finite field elements for the curves BN256 and BN254, multiplication and squaring is faster for the latter since the sparse form of the prime is exploited to speed up the reduction step. Another effect that can be seen is, that the number of cycles needed for addition scales linearly with the degree of the extension field, for example, addition in $\mathbb{F}_{p^2}$ takes about twice as long as addition in $\mathbb{F}_p$. For multiplication and squaring, cycle counts scale quadratically with the degree of the extension field. This leads to rather long runtime for multiplications and squarings in $\mathbb{F}_{p^{12}}$. Similarly, performance of addition scales linearly with the size of the prime given by the curve. Multiplication and squaring, however, scale quadratically in this respect.

Another evaluation of this kind is given by Table 8.2, which shows the performance of inversion in relation to multiplication in the finite fields used for the respective curves. One can see, that, especially in the prime field $\mathbb{F}_p$, inversion is massively more expensive than multiplication. For example, the ratio of cycle counts for inversion and multiplication is 297 for BN254. A large part of this observation is contributed to performing inversion using Fermat's little theorem. However, inversion is not vulnerable to timing and SPA

Table 8.1.: Cycle counts for addition, multiplication and squaring in the finite fields $\mathbb{F}_p$, $\mathbb{F}_{p^2}$, and $\mathbb{F}_{p^{12}}$ using the assembler optimized prime field operations.

Curve	**Addition**			**Multiplication**			**Squaring**		
	$\mathbb{F}_p$	$\mathbb{F}_{p^2}$	$\mathbb{F}_{p^{12}}$	$\mathbb{F}_p$	$\mathbb{F}_{p^2}$	$\mathbb{F}_{p^{12}}$	$\mathbb{F}_p$	$\mathbb{F}_{p^2}$	$\mathbb{F}_{p^{12}}$
BN256	167	350	2,163	3,862	12,354	248,998	3,862	8,217	160,708
BN254	167	350	2,163	3,462	11,154	225,158	3,462	7,417	144,428
BN158	112	240	1,503	1,575	5,239	111,111	1,575	3,485	72,625

Table 8.2.: Cycle counts for multiplication and inversion in the finite fields $\mathbb{F}_p$, $\mathbb{F}_{p^2}$, and $\mathbb{F}_{p^{12}}$ and the respective inversion-to-multiplication ratios.

Curve	**Multiplication**			**Inversion**			**I/M Ratio**		
	$\mathbb{F}_p$	$\mathbb{F}_{p^2}$	$\mathbb{F}_{p^{12}}$	$\mathbb{F}_p$	$\mathbb{F}_{p^2}$	$\mathbb{F}_{p^{12}}$	$\mathbb{F}_p$	$\mathbb{F}_{p^2}$	$\mathbb{F}_{p^{12}}$
BN256	3,862	12,354	248,998	1,111,882	1,127,649	1,615,925	287.9	91.3	6.5
BN254	3,462	11,154	225,158	1,028,298	1,042,465	1,481,541	297.0	93.5	6.6
BN158	1,575	5,239	111,111	290,930	297,460	511,111	184.7	56.8	4.6

Table 8.3.: Cycle counts for addition, multiplication and squaring in the finite fields $\mathbb{F}_p$, $\mathbb{F}_{p^2}$, and $\mathbb{F}_{p^{12}}$ for the implementation variants of the curve BN254. Additionally, the performance relative to the standard implementation is visualized.

Variant	**Addition**			**Multiplication**			**Squaring**		
	$\mathbb{F}_p$	$\mathbb{F}_{p^2}$	$\mathbb{F}_{p^{12}}$	$\mathbb{F}_p$	$\mathbb{F}_{p^2}$	$\mathbb{F}_{p^{12}}$	$\mathbb{F}_p$	$\mathbb{F}_{p^2}$	$\mathbb{F}_{p^{12}}$
Standard[a]	167	350	2,163	3,462	11,154	225,158	3,462	7,417	144,428
LR in $\mathbb{F}_p$[b]	208	438	2,691	3,462	11,434	237,518	3,462	7,569	153,773
LR-$\mathbb{F}_{p^2}$[c]	167	350	2,163	3,782	10,053	205,340	3,782	8,056	132,317
LR-$\mathbb{F}_{p^2}$ *MAC-1*	167	350	2,163	1,199	3,356	84,794	1,199	2,890	58,650
LR-$\mathbb{F}_{p^2}$ *MAC-2*	167	350	2,163	1,316	3,635	89,816	1,316	3,124	61,719
Relative Performance									
LR in $\mathbb{F}_p$	+24.6%	+25%	+24.4%	±0.0%	+2.5%	+5.5%	±0%	+2.0%	+ 6.5%
LR-$\mathbb{F}_{p^2}$	±0.0%	±0.0%	±0.0%	+9.2%	−9.9%	−8.8%	+9.2%	+8.6%	−8.4%
LR-$\mathbb{F}_{p^2}$ *MAC-1*	±0.0%	±0.0%	±0.0%	−65.4%	−69.9%	−62.3%	−65.4%	−61.0%	−59.4%
LR-$\mathbb{F}_{p^2}$ *MAC-2*	±0.0%	±0.0%	±0.0%	−62.0%	−67.4%	−60.1%	−62.0%	−57.9%	−57.3%

[a] Plain assembler-optimized implementation with FIPS Montgomery multiplication in $\mathbb{F}_p$.
[b] Lazy reduction in $\mathbb{F}_p$.
[c] Multiplication in $\mathbb{F}_{p^2}$ uses lazy reduction technique by Sánchez and Rodríguez-Henríquez [SR13].

attacks in this case. Since direct formulas are used to reduce the problem of inversion in extension fields to inversion in their respective subfields, the inversion to multiplication ratio becomes better with higher extension degrees, for example, the ratio is only 6.6 in $\mathbb{F}_{p^{12}}$ for BN254. Consequently, there is a point, where projective coordinates may be more expensive than affine coordinates when computing elliptic curve operations. This, however, is not the case in this work. In Table 8.2 one can also observe that inversion in $\mathbb{F}_p$ scales approximately with the third power of the prime, which splits into a quadratic part from multiplication and a linear part from exponentiation.

In Table 8.3 performance of addition, multiplication and squaring are presented for different finite field implementations for the curve BN254. Performance of the variants is also given relatively to the standard implementation, which was also used in Table 8.1. Addition, and subtraction accordingly, only changes its runtime when the lazy reduction in $\mathbb{F}_p$ is employed. In particular, speed of addition worsens by a quarter throughout the used fields. This effect has already been indicated in Section 6.4.1. It is due to additional integer additions and subtractions of the modulus that need to be done independently of the value since the implementation should resist timing attacks. Even though multiplications and squarings in $\mathbb{F}_p$ do not lose speed in this case, they do in the extension fields $\mathbb{F}_{p^2}$ and $\mathbb{F}_{p^{12}}$ as they also resort to additions and subtractions in the subfield.

Table 8.4.: Cycle counts for multiplication, squaring and inversion in the finite fields $\mathbb{F}_p$, $\mathbb{F}_{p^2}$, and $\mathbb{F}_{p^{12}}$ for the implementation variants of the curve BN158. Additionally, the performance relative to the standard implementation is visualized.

Variant	**Multiplication**			**Squaring**			**Inversion**		
	$\mathbb{F}_p$	$\mathbb{F}_{p^2}$	$\mathbb{F}_{p^{12}}$	$\mathbb{F}_p$	$\mathbb{F}_{p^2}$	$\mathbb{F}_{p^{12}}$	$\mathbb{F}_p$	$\mathbb{F}_{p^2}$	$\mathbb{F}_{p^{12}}$
Standard[a]	1,575	5,239	111,111	1,575	3,485	72,625	290,930	297,460	511,111
LR-$\mathbb{F}_{p^2}$[b]	1,800	4,757	102,435	1,800	3,934	67,323	331,459	338,884	536,563
LR-$\mathbb{F}_{p^2}$ *MAC-1*	646	1,856	50,217	646	1,626	35,412	123,739	126,548	218,076
LR-$\mathbb{F}_{p^2}$ *MAC-2*	718	2,027	53,295	718	1,770	27,293	136,699	139,796	237,597
Relative Performance									
LR-$\mathbb{F}_{p^2}$	+14.3%	−9.2%	−7.8%	+14.3%	+12.9%	−7.3%	+13.9%	+13.9%	+5.0%
LR-$\mathbb{F}_{p^2}$ *MAC-1*	−59.0%	−64.6%	−54.8%	−59.0%	−53.3%	−51,2%	−57.5%	−57.5%	−57.3%
LR-$\mathbb{F}_{p^2}$ *MAC-2*	−54.4%	−61.3%	−52.0%	−54.4%	−49.2%	−48.6%	−53.0%	−53.0%	−53.5%

[a] Plain assembler-optimized implementation with FIPS Montgomery multiplication in $\mathbb{F}_p$.
[b] Multiplication in $\mathbb{F}_{p^2}$ uses lazy reduction technique by Sánchez and Rodríguez-Henríquez [SR13].

When employing the lazy reduction technique in the multiplication in $\mathbb{F}_{p^2}$ as shown by Sánchez and Rodríguez-Henríquez [SR13], multiplication in $\mathbb{F}_{p^2}$ speeds up by roughly 10%. Contrary to that, multiplication slows down by 10% in $\mathbb{F}_p$, because the FIPS variant of the Montgomery multiplication in $\mathbb{F}_p$ is split up into a separate multiplication and reduction step in order to keep the size of the program memory low. If this was not done this way, basically the same unrolled loop would stay in memory twice and make the platform less efficient. The results are similar for squaring. When equipping the platform with either of the two multiply-accumulate instructions *MAC-1* or *MAC-2*, multiplication speeds up between 60% and 70% for each of the fields implemented. The best results are reached for $\mathbb{F}_{p^2}$, which is a consequence of the speed-up coming from the lazy reduction technique in $\mathbb{F}_{p^2}$. The extension *MAC-1* gives results that are a slightly better than the results from extension *MAC-2*, which is due to the additional `nop` instructions needed to wait for the finished results in the latter case.

A similar evaluation has been done using the curve BN158 and is shown in Table 8.4. It does not contain an implementation using lazy reduction in $\mathbb{F}_p$ as it was only implemented for BN254. Addition performs equally well for all the variants shown and can be looked up in Table 8.1. Cycle counts are significantly lower than for BN254, but the observations are mainly the same. Applying the trick by Sánchez and Rodríguez-Henríquez [SR13] yields the same speed-up for multiplications in $\mathbb{F}_{p^2}$, but worsens multiplication even by 14% in $\mathbb{F}_p$. Using one of the instruction-set extensions yields performance increases between 48% and 59%, which is less than for BN254. This can be explained by the fact that complexity of multiplication scales quadratically with the size of the prime and hence becomes less significant in the overall performance if smaller primes are used. The table also shows, that inversion behaves similar to multiplication when comparing the different variants. Nevertheless, since inversion in any of the extension fields relies on inversion (and hence multiplication) in $\mathbb{F}_p$, one can observe that inversion in $\mathbb{F}_{p^{12}}$ is 5% slower than in the standard version after applying the lazy reduction technique in $\mathbb{F}_{p^2}$, which is contrary to the speed-up observed for multiplication and squaring in $\mathbb{F}_{p^{12}}$.

Table 8.5.: Memory requirements for an optimal Ate pairing and cycle counts for pairing computations and point multiplications.

Curve	Memory			Pairing		Point Multiplication	
	Stack [Bytes]	RAM [Bytes]	ROM [Bytes]	Single [kCycles]	Product [kCycles]	$E(\mathbb{F}_p)$ [kCycles]	$E(\mathbb{F}_{p^2})$ [kCycles]
BN256	2,488	2,692	14,024	61,395	84,183	16,323	54,420
BN254	2,488	2,692	13,120	50,825	69,658	14,709	49,271
BN158	1,636	1,768	9,284	18,346	24,897	4,205	14,707

8.2. Pairing

Evaluation of the bilinear map, in particular the optimal Ate pairing, is done with respect to the software itself and the overall hardware platform. In addition, performance of elliptic curve point multiplications is investigated.

8.2.1. Software

The software implementations of both the optimal Ate pairing and the elliptic curve point multiplications are compared with respect to the three different curves BN256, BN254 and BN158 in Table 8.5. A pairing computation can be done in 61 million clock cycles for BN256. Using the implementation for BN254, this is reduced to 50 million clock cycles, providing almost the same level of security though. At the 80-bit security level, an optimal Ate pairing can be computed in 18 million cycles. Computing the product of two pairings is only about 35% slower than computing a single pairing, because some of the computations can be done once for both. Point multiplications in $E(\mathbb{F}_p)$ are clearly faster than in $E(\mathbb{F}_{p^2})$. Point multiplications in $E(\mathbb{F}_{p^2})$ are slower than computing a pairing though, but the difference is only small for the curve BN254. This is interesting since pairings seem computationally extremely more complex due to their operations in $\mathbb{F}_{p^{12}}$. The main reason for this may be the need for constant runtime, which is automatically given for the pairing computation as the loop parameter s is fixed, but needs to be added expensively for point multiplication. The computation of pairings takes about 3-4 times the number of clock cycles of point multiplications in $E(\mathbb{F}_p)$. With respect to memory, pairings at the 128-bit security level need about 2,7 KB of RAM including stack. Due to the smaller elements for BN158, about 930 byte less RAM is needed in this case, which indicates linear behavior. Any additional memory needed is allocated on the stack, which makes possible the utilization of the memory for other applications when the pairing computation is done. With 13,1 KB, the program memory is 900 byte smaller for the curve BN254 than for the curve BN256, which is due to the saved operations in the unrolled reduction step of the Montgomery multiplication. The smaller code in BN158 results from smaller unrolled loops.

The performance of pairing computations and the elliptic curve point multiplications using BN254 is shown in Table 8.6 for the different implementation variants. Performance is additionally evaluated in relation to the standard version. When applying the lazy reduction technique to the multiplication in $\mathbb{F}_{p^2}$, approximately 4.5% more RAM is needed, which is acceptable as runtime of pairing computations is reduced by 6%. Contrary to that, point multiplications in $E(\mathbb{F}_p)$ become slower by 9% due to the splitting of the Montgomery multiplication. Adding randomness to the pairing computation to prevent

Table 8.6.: Memory requirements for an optimal Ate pairing and cycle counts for pairing computations and point multiplications for the implementation variants of the curve BN254.

Variant	**Memory**			**Pairing**		**Point Multiplication**	
	Stack [Bytes]	RAM [Bytes]	ROM [Bytes]	Single [kCycles]	Product [kCycles]	$E(\mathbb{F}_p)$ [kCycles]	$E(\mathbb{F}_{p^2})$ [kCycles]
Standard[a]	2,488	2,692	13,120	50,825	69,658	14,709	49,271
LR-$\mathbb{F}_{p^2}$[b]	2,600	2,804	13,752	47,585	65,612	16,027	47,153
LR_{sc}-$\mathbb{F}_{p^2}$[c]	2,600	2,824	14,740	47,643	65,728	16,027	47,153
LR_{sc}-$\mathbb{F}_{p^2}$ *MAC-1*	2,596	2,820	9,496	20,536	27,690	5,390	16,877
LR_{sc}-$\mathbb{F}_{p^2}$ *MAC-2*	2,596	2,820	9,728	21,680	29,303	5,872	18,174
Relative Performance							
LR-$\mathbb{F}_{p^2}$	+4.5%	+4.2%	+4.8%	−6.4%	−5.8%	+9.0%	−4.3%
LR_{sc}-$\mathbb{F}_{p^2}$	+4.5%	+4.9%	+12.4%	−6.3%	−5.6%	+9.0%	−4.3%
LR_{sc}-$\mathbb{F}_{p^2}$ *MAC-1*	+4.3%	+4.8%	−27.6%	−59.6%	−60.2%	−63.4%	−65.7%
LR_{sc}-$\mathbb{F}_{p^2}$ *MAC-2*	+4.3%	+4.8%	−25.9%	−57.3%	−57.9%	−60.1%	−63.1%

[a] Plain assembler-optimized implementation with FIPS Montgomery multiplication in $\mathbb{F}_p$.
[b] Multiplication in $\mathbb{F}_{p^2}$ uses lazy reduction technique by Sánchez and Rodríguez-Henríquez [SR13].
[c] Additionally incorporates the countermeasure against DPA attacks.

DPA attacks has only negligible impact on runtime. However, the size of the program memory increases by additional 7.6%, because a cryptographically secure pseudorandom number generator has to be included. The two instruction-set extensions *MAC-1* and *MAC-2* in turn reduce program memory size by a quarter as the unrolled loops shrink. Further, a massive speedup of around 60% is observed for pairing computations and point multiplications. Hence, the improvement observed for finite field multiplications directly leads to a speed-up in pairing computations.

The same evaluation is done for the curve BN158 in Table 8.7. The requirements for RAM behave very similar to the implementations for the curve BN254. In absolute values, program memory size also increases similarly when adding the random number generator, but it shrinks less when making use of one of the two multiply-accumulate extensions. For this curve, only 10% of program memory can be saved if one of these extensions is used. With respect to runtime of pairing computations and the elliptic curve operations, the picture is very similar to the one conveyed by the curve BN254: the lazy reduction technique in $\mathbb{F}_{p^2}$ speeds pairings up by 5%, but slows down point multiplications in $E(\mathbb{F}_p)$ by 13.6%. The instruction-set extensions achieve improvements by up to 59%.

8.2.2. Hardware

The hardware platforms capable of computing point multiplications in $E(\mathbb{F}_p)$ and optimal Ate pairings are compared for the three different curves in Table 8.8. The typical power values, which were determined using power simulations of the respective chips at 10 Mhz, are roughly the same for all three curves. Runtime is equivalent to the respective software platforms investigated in Section 8.2.1 and is given in milliseconds at a clock frequency of 10 Mhz. One can observe that computing a pairing takes 6.1 seconds for the curve BN254 in this realistic scenario. This runtime is reduced to 5 and 1.8 seconds when

Table 8.7.: Memory requirements for an optimal Ate pairing and cycle counts for pairing computations and point multiplications for the implementation variants of the curve BN158.

Variant	**Memory**			**Pairing**		**Point Multiplication**	
	Stack [Bytes]	RAM [Bytes]	ROM [Bytes]	Single [kCycles]	Product [kCycles]	$E(\mathbb{F}_p)$ [kCycles]	$E(\mathbb{F}_{p^2})$ [kCycles]
Standard[a]	1,636	1,768	9,284	18,346	24,897	4,205	14,707
LR-$\mathbb{F}_{p^2}$[b]	1,700	1,832	9,680	17,360	23,707	4,778	14,327
LR_{sc}-$\mathbb{F}_{p^2}$[c]	1,700	1,852	10,640	23,765	27,098	4,778	14,327
LR_{sc}-$\mathbb{F}_{p^2}$ *MAC-1*	1,696	1,848	8,308	8,788	11,745	1,836	6,000
LR_{sc}-$\mathbb{F}_{p^2}$ *MAC-2*	1,696	1,848	8,452	9,300	12,461	2,020	6,500
Relative Performance							
LR-$\mathbb{F}_{p^2}$	+3.9%	+3.6%	+4.3%	−5.4%	−4.8%	+13.6%	−2.6%
LR_{sc}-$\mathbb{F}_{p^2}$	+3.9%	+4.8%	+14.6%	−5.2%	−4.5%	+13.6%	−2.6%
LR_{sc}-$\mathbb{F}_{p^2}$ *MAC-1*	+3.7%	+4.5%	−10.5%	−52.1%	−52.8%	−56.3%	−59.2%
LR_{sc}-$\mathbb{F}_{p^2}$ *MAC-2*	+3.7%	+4.5%	−9.0%	−49.3%	−49.9%	−52.0%	−55.8%

[a] Plain assembler-optimized implementation with FIPS Montgomery multiplication in $\mathbb{F}_p$.
[b] Multiplication in $\mathbb{F}_{p^2}$ uses lazy reduction technique by Sánchez and Rodríguez-Henríquez [SR13].
[c] Additionally incorporates the countermeasure against DPA attacks.

Table 8.8.: Comparison of the hardware platforms capable of computing point multiplications and pairings at 10 Mhz.

Curve	**Area**				**Power**	**Runtime**		**Energy**	
	RAM [kGE]	ROM [kGE]	Core [kGE]	Total [kGE]	Typ. [mW]	Pairing [ms]	Point Mul. [ms]	Pairing [mJ]	Point Mul. [mJ]
BN256	15.3	15.6	16.3	48.0	0.66	6,140	1,632	4.04	1.07
BN254	15.3	15.6	16.3	48.0	0.66	5,083	1,471	3.34	0.97
BN158	10.9	11.9	16.3	39.9	0.67	1,835	421	1.23	0.28

pairings are computed for the curves BN254 and BN158 respectively. Point multiplications are significantly faster and can be performed in a range of 420 and 1,630 milliseconds depending on the curve being used. With respect to energy, the same ratios can be observed when comparing the three curves: for a BN256 curve about 4 mJ of energy are needed to compute an optimal Ate pairing, which drops to 3.3 mJ for BN254 and 1.2 mJ for BN158. Accordingly, point multiplications consume less energy than pairing computations.

Another important characteristic of hardware platforms is their area. The figures in Table 8.8 and all the following are given in gate equivalents for a 130 nm UMC process. Both RAM and ROM were implemented using macros of appropriate size. The core area consists of the processor and a bus arbiter. The total area additionally consists of a RS232 interface for communication. The hardware platforms for BN256 and BN254 are equal in size when the standard implementation is used. Total chip area shrinks by roughly 17% if the smaller curve BN158 is used.

The hardware platforms for the various implementation variants using the curve BN254 are shown in Table 8.9. The first thing that catches the eye is that power consumption

Table 8.9.: Comparison of the various hardware platforms capable of computing point multiplications and pairings using a 10 Mhz clock and the curve BN254.

Variant	**Area**				**Power**	**Runtime**		**Energy**	
	RAM [kGE]	ROM [kGE]	Core [kGE]	Total [kGE]	Typ. [mW]	Pairing [ms]	Mult.[a] [ms]	Pairing [mJ]	Mult. [mJ]
Standard[b]	15.3	15.6	16.3	48.0	0.66	5,083	1,471	3.34	0.97
LR-$\mathbb{F}_{p^2}$[c]	15.8	15.6	16.3	48.5	0.66	4,759	1,603	3.14	1.06
LR_{sc}-$\mathbb{F}_{p^2}$[d]	15.9	17.0	16.3	50.0	0.68	4,764	1,603	3.24	1.09
LR_{sc}-$\mathbb{F}_{p^2}$ *MAC-1*	15.9	11.9	19.8	48.4	0.74	2,054	539	1.51	0.40
LR_{sc}-$\mathbb{F}_{p^2}$ *MAC-2*	15.9	11.9	17.6	46.2	1.09	2,168	587	2.35	0.64
Relative Performance									
LR-$\mathbb{F}_{p^2}$	+3.5%	±0.0%	±0.0%	+1.1%	±0.0%	−6.4%	+9.0%	−6.1%	+9.3%
LR_{sc}-$\mathbb{F}_{p^2}$	+4.0%	+9.1%	±0.0%	+4.2%	+3.4%	−6.3%	+9.0%	−3.1%	+12.7%
LR_{sc}-$\mathbb{F}_{p^2}$ *MAC-1*	+4.0%	−23.7%	+21.5%	+0.9%	+11.9%	−59.6%	−63.4%	−54.8%	−59.0%
LR_{sc}-$\mathbb{F}_{p^2}$ *MAC-2*	+4.0%	−23.7%	+8.0%	−3.7%	+65.0%	−57.3%	−60.1%	−29.6%	−34.1%

[a] Elliptic curve point multiplication in $E(\mathbb{F}_p)$.
[b] Plain assembler-optimized implementation with FIPS Montgomery multiplication in $\mathbb{F}_p$.
[c] Multiplication in $\mathbb{F}_{p^2}$ uses lazy reduction technique by Sánchez and Rodríguez-Henríquez [SR13].
[d] Additionally incorporates the countermeasure against DPA attacks.

is more or less the same for all versions that do not have hardware extensions employed. Adding the extension *MAC-1* raises the typical power consumption by about 12%. This value is even topped by a 65% gain when utilizing *MAC-2*. Using the lazy reduction technique for multiplication in $\mathbb{F}_{p^2}$, overall chip area and area for RAM increases slightly. Pairing computations consume less energy in less time, and point multiplications have increased demand for energy. Making the pairing computation secure against DPA attacks increases chip size by about 4%. Nevertheless, the effect on chip area vanished when employing the *MAC-1* instruction-set extension. For the extension *MAC-2*, even a smaller chip size can be observed since the increased size of the processor core is overcompensated by the reduced need for program memory. Significant differences can be recognized for the amount of energy needed: the extension *MAC-1* reduces energy consumption for both pairing computation and point multiplication by over 50%. Despite similar runtime for the extension *MAC-2*, this version is less energy-efficient than the version incorporating *MAC-1*. Nevertheless, it consumes only 70% of the energy originally needed. The area-efficiency and energy-efficiency of these platforms is also visualized in Figure 8.1 and Figure 8.2 respectively.

This kind of evaluation has also been done for the hardware platforms capable of performing computations using the BN158 curve, which is shown in Table 8.10. More or less the same observations can be made as for the curve BN254 in Table 8.9. One remarkable difference is, that employing one of the multiply-accumulate extensions does not lead to equal or reduced chip size relative to the standard variant for BN158. The reason for this is, that there were no suitable ROM macros available that would reflect the reduced size of the program memory. Energy consumption of pairing computations and elliptic curve point multiplications is reduced significantly by using the extensions. However, the effect is about 10% less than it is for the curve BN254. The same can be observed for the respective runtimes.

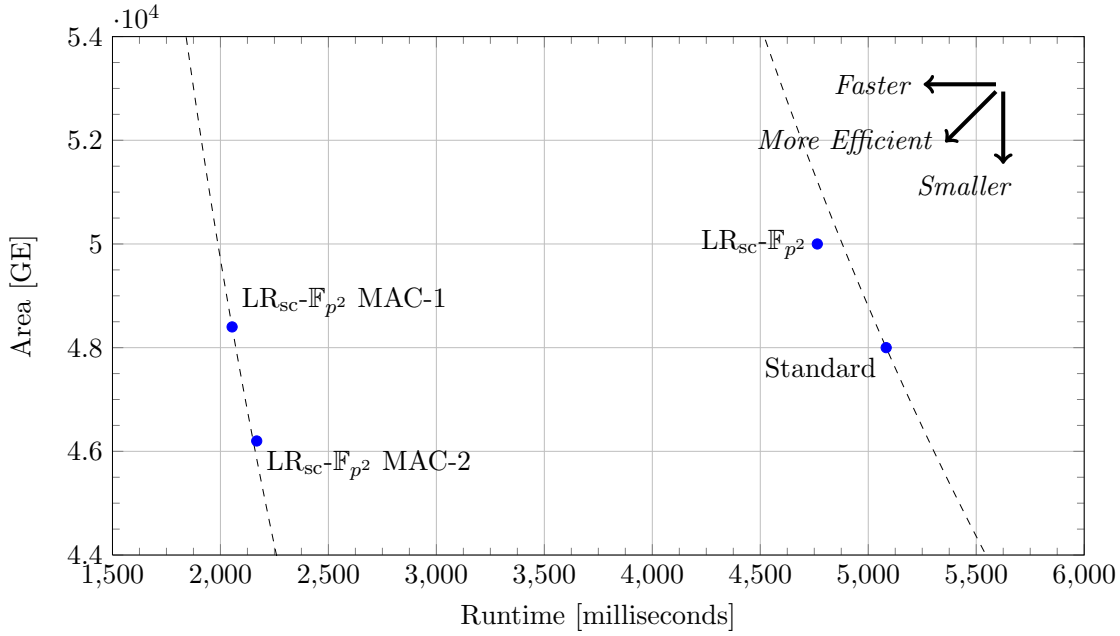

Figure 8.1.: Area-runtime characteristics for pairing computations using BN254 at 10 MHz.

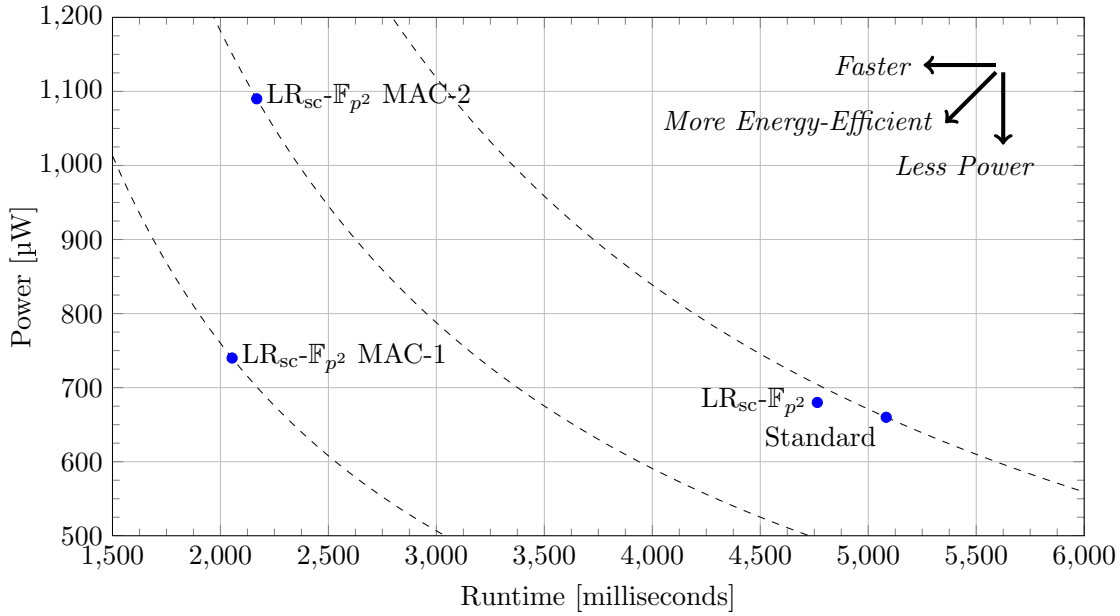

Figure 8.2.: Power-runtime characteristics for pairing computations using BN254 at 10 MHz.

Table 8.10.: Comparison of the various hardware platforms capable of computing point multiplications and pairings using a 10 Mhz clock and the curve BN158.

Variant	**Area**				**Power**	**Runtime**		**Energy**	
	RAM [kGE]	ROM [kGE]	Core [kGE]	Total [kGE]	Typ. [mW]	Pairing [ms]	Mult.[a] [ms]	Pairing [mJ]	Mult. [mJ]
Standard[b]	10.9	11.9	16.3	39.9	0.67	1,835	421	1.23	0.28
LR-$\mathbb{F}_{p^2}$[c]	11.2	11.9	16.3	40.2	0.67	1,736	478	1.17	0.32
LR_{sc}-$\mathbb{F}_{p^2}$[d]	11.3	13.8	16.3	42.2	0.68	1,739	478	1.18	0.32
LR_{sc}-$\mathbb{F}_{p^2}$ *MAC-1*	11.3	11.9	19.8	43.8	0.74	879	184	0.65	0.14
LR_{sc}-$\mathbb{F}_{p^2}$ *MAC-2*	11.3	11.9	17.6	41.6	1.09	930	202	1.01	0.22
Relative Performance									
LR-$\mathbb{F}_{p^2}$	+2.6%	±0.0%	±0.0%	+0.7%	±0.0%	−5.4%	+13.6%	−5.0%	+14.1%
LR_{sc}-$\mathbb{F}_{p^2}$	+3.4%	+15.6%	±0.0%	+5.6%	+5.6%	−5.2%	+13.6%	−4.3%	+14.7%
LR_{sc}-$\mathbb{F}_{p^2}$ *MAC-1*	+3.4%	±0.0%	+21.5%	+9.7%	+10.8%	−52.1%	−56.3%	−46.9%	−51.6%
LR_{sc}-$\mathbb{F}_{p^2}$ *MAC-2*	+3.4%	±0.0%	+8.0%	+4.2%	+61.6%	−49.3%	−52.0%	−18.0%	−22.3%

[a] Elliptic curve point multiplication in $E(\mathbb{F}_p)$.
[b] Plain assembler-optimized implementation with FIPS Montgomery multiplication in $\mathbb{F}_p$.
[c] Multiplication in $\mathbb{F}_{p^2}$ uses lazy reduction technique by Sánchez and Rodríguez-Henríquez [SR13].
[d] Additionally incorporates the countermeasure against DPA attacks.

8.3. Identity-Based Encryption Scheme

The original goal of this work was to equip embedded platforms with identity-based encryption. In the following, it is evaluated how well this goal is achieved in terms of the software imlementation and the respective hardware platform.

8.3.1. Software

The memory requirements and runtime for the BB_1 IBE KEM are listed in Table 8.11. At the 128-bit security level, encapsulation takes 136 and 123 million clock cycles for the curves BN256 and BN254 respectively. This number drops significantly to 37 million when reducing security to 80 bit. Decapsulation is significantly faster than encapsulation: decapsulating a session key takes, depending on the curve, only 55-65% of the runtime needed for encapsulation. The implementations using BN256 and BN254 both need 2,848 bytes of RAM. Their demand for constant memory is 19,668 bytes and 18,872 bytes, respectively. When the curve BN158 is used, the demand for RAM drops to 1,900 bytes. The effect on constant memory is less, but still significant as it shrinks by 23%.

Detailed evaluation of the identity-based encryption implementations for the curve BN254 are shown in Table 8.12. The table also shows performance relative to the standard implementation used in Table 8.11. The lazy reduction technique in the multiplication in $\mathbb{F}_{p^2}$, however, reduces runtime of both routines. This is remarkable since encapsulation involves several point multiplications in $E(\mathbb{F}_p)$, which, in the previous section, have been shown to be less efficient after applying this trick. The downside is a slightly increased memory footprint. Application of the countermeasure to prevent DPA attacks only has little impact on decapsulation performance. Using either of the instruction-set extensions *MAC-1* and *MAC-2* significantly speeds up the operations involved in the identity-based

Table 8.11.: Memory requirements and runtime of the BB_1 IBE KEM using the three curves BN256, BN254, and BN158.

Curve	**Memory**			**Encapsulate**	**Decapsulate**
	Stack [Bytes]	RAM [Bytes]	ROM [Bytes]	[kCycles]	[kCycles]
BN256	2,824	2,848	19,668	136,822	84,638
BN254	2,824	2,848	18,872	123,320	70,107
BN158	1,876	1,900	14,704	37,700	25,149

Table 8.12.: Memory requirements and runtime of the BB_1 IBE KEM for the implementation variants of the curve BN254.

Variant	**Memory**			**Encapsulate**	**Decapsulate**
	Stack [Bytes]	RAM [Bytes]	ROM [Bytes]	[kCycles]	[kCycles]
Standard[a]	2,824	2,848	18,872	123,320	70,107
LR-$\mathbb{F}_{p^2}$[b]	2,936	2,960	19,480	121,194	66,066
LR_{sc}-$\mathbb{F}_{p^2}$[c]	2,936	2,960	19,500	121,194	66,183
LR_{sc}-$\mathbb{F}_{p^2}$ *MAC-1*	2,932	2,956	14,252	47,925	28,103
LR_{sc}-$\mathbb{F}_{p^2}$ *MAC-2*	2,932	2,956	14,488	51,096	29,717
Relative Performance					
LR-$\mathbb{F}_{p^2}$	+4.0%	+3.9%	+3.2%	−1.7%	−5.8%
LR_{sc}-$\mathbb{F}_{p^2}$	+4.0%	+3.9%	+3.3%	−1.7%	−5.6%
LR_{sc}-$\mathbb{F}_{p^2}$ *MAC-1*	+3.8%	+3.8%	−24.5%	−61.1%	−59.9%
LR_{sc}-$\mathbb{F}_{p^2}$ *MAC-2*	+3.8%	+3.8%	−23.2%	−58.6%	−57.6%

[a] Plain assembler-optimized implementation with FIPS Montgomery multiplication in $\mathbb{F}_p$.
[b] Multiplication in $\mathbb{F}_{p^2}$ uses lazy reduction technique by Sánchez and Rodríguez-Henríquez [SR13].
[c] Additionally incorporates the countermeasure against DPA attacks.

encryption scheme: runtime plummets by about 60%. In addition, the demand for constant memory shrinks by almost a quarter since the unrolled loops become smaller. Consequently, encapsulation can be done in about 50 million clock cycles, and decapsulation is done in approximately 29 million cycles when the multiply-accumulate extensions are used. Still, decapsulation is much faster than encapsulation.

As in the previous sections, this evaluation was also done for the curve BN158 and is shown in Table 8.13. The results are practically the same as for the curve BN254, but the speed-up of the encapsulation routine when using lazy reduction in $\mathbb{F}_{p^2}$ has almost vanished. On the other hand, the increase of the needed memory's size is less significant, which ranges between 3.4% for RAM and 2.6% for ROM. Using one of the two instruction-set extensions *MAC-1* and *MAC-2* halves the cycle counts for encapsulation and decapsulation, leading to about 18 and 12 million clock cycles respectively. Further, 13.2% of constant memory can be saved when using the instruction-set extensions. Nevertheless, the memory-saving effect is less for BN158 than it is for BN254.

Table 8.13.: Memory requirements and runtime of the BB_1 IBE KEM for the implementation variants of the curve BN158.

Variant	**Memory**			**Encapsulate**	**Decapsulate**
	Stack [Bytes]	RAM [Bytes]	ROM [Bytes]	[kCycles]	[kCycles]
Standard[a]	1,876	1,900	14,704	37,700	25,149
LR-$\mathbb{F}_{p^2}$[b]	1,940	1,964	15,076	37,642	23,962
LR_{sc}-$\mathbb{F}_{p^2}$[c]	1,940	1,964	15,096	37,642	24,020
LR_{sc}-$\mathbb{F}_{p^2}$ *MAC-1*	1,936	1,960	12,764	16,960	11,982
LR_{sc}-$\mathbb{F}_{p^2}$ *MAC-2*	1,936	1,960	12,908	18,136	12,699
Relative Performance					
LR-$\mathbb{F}_{p^2}$	+3.4%	+3.4%	+2.5 %	−0.2%	−4.7%
LR_{sc}-$\mathbb{F}_{p^2}$	+3.4%	+3.4%	+2.6 %	−0.2%	−4.5%
LR_{sc}-$\mathbb{F}_{p^2}$ *MAC-1*	+3.2%	+3.2%	−13.2%	−55.0%	−52.4%
LR_{sc}-$\mathbb{F}_{p^2}$ *MAC-2*	+3.2%	+3.2%	−12.2%	−51.9%	−49.5%

[a]Plain assembler-optimized implementation with FIPS Montgomery multiplication in $\mathbb{F}_p$.
[b]Multiplication in $\mathbb{F}_{p^2}$ uses lazy reduction technique by Sánchez and Rodríguez-Henríquez [SR13].
[c]Additionally incorporates the countermeasure against DPA attacks.

Table 8.14.: Comparison of the hardware platforms for the BB_1 IBE KEM at 10 Mhz.

Curve	**Area**				**Power**	**Runtime**		**Energy**	
	RAM [kGE]	ROM [kGE]	Core [kGE]	Total [kGE]	Typ. [mW]	Enc. [ms]	Dec. [ms]	Enc. [mJ]	Dec. [mJ]
BN256	16.0	20.1	16.3	53.2	0.71	13,682	8,464	9.68	5.99
BN254	16.0	20.1	16.3	53.2	0.71	12,332	7,011	8.73	4.96
BN158	11.6	17.0	16.3	45.7	0.70	3,770	2,515	2.64	1.76

8.3.2. Hardware

Finally, the hardware platforms capable of performing identity-based encryption are compared for the three curves BN256, BN254 and BN158 in Table 8.14. Runtime and power measurements are given for a clock frequency of 10 Mhz. Among the three platforms, BN158 is of course the fastest, but also provides the least security. Identity-based encryption at the 128-bit security level comes at the cost of merely 53,300 gate equivalents using an 130 nm UMC process. This number drops to 45,700 when reducing security to 80 bits. Consequently, chip area shrinks less than the size of the prime in use. Encapsulation is done in roughly 12 seconds at the 128-bit security level using a 10 Mhz clock, which is significantly slower than decapsulation, which is performed in 7 seconds. Therefore, 10.7 mJ and 6.1 mJ of energy are needed respectively. At the 80-bit security level, encapsulation and decapsulation are done in 3.7 and 2.5 seconds, which also reduces the demand for energy to 2.6 mJ and 1.7 mJ.

The different variants of the hardware platforms capable of performing the operations of the BB_1 IBE KEM are compared for the BN254 curve in Table 8.15. The lazy reduction technique in $\mathbb{F}_{p^2}$ reduces runtime of encapsulation and decapsulation by 1.7% and 5.8%, respectively. On the other hand, chip area is increased slightly by the higher memory

Table 8.15.: Comparison of the various hardware platforms for the BB_1 IBE KEM using a 10 Mhz clock and the curve BN254.

Variant	**Area**				**Power**	**Runtime**		**Energy**	
	RAM [kGE]	ROM [kGE]	Core [kGE]	Total [kGE]	Typ. [mW]	Enc. [ms]	Dec. [ms]	Enc. [mJ]	Dec. [mJ]
Standard[a]	16.0	20.1	16.3	53.2	0.71	12,332	7,011	8.73	4.96
LR-$\mathbb{F}_{p^2}$[b]	16.6	20.1	16.3	53.8	0.71	12,119	6,607	8.56	4.67
LR_{sc}-$\mathbb{F}_{p^2}$[c]	16.6	20.1	16.3	53.8	0.71	12,119	6,618	8.56	4.67
LR_{sc}-$\mathbb{F}_{p^2}$ *MAC-1*	16.5	15.6	19.8	52.7	0.74	4,793	2,810	3.56	2.09
LR_{sc}-$\mathbb{F}_{p^2}$ *MAC-2*	16.5	17.0	17.6	51.9	1.10	5,110	2,972	5.60	3.26
Relative Performance									
LR-$\mathbb{F}_{p^2}$	+3.3%	±0.0%	±0.0%	+1.0%	±0.0%	−1.7%	−5.8%	−1.9%	−5.9%
LR_{sc}-$\mathbb{F}_{p^2}$	+3.3%	±0.0%	±0.0%	+1.0%	±0.0%	−1,7%	−5.6%	−1.9%	−5.8%
LR_{sc}-$\mathbb{F}_{p^2}$ *MAC-1*	+3.1%	−22.5%	+21.5%	−1.0%	+4.9%	−61.1%	−59.9%	−59.2%	−57.9%
LR_{sc}-$\mathbb{F}_{p^2}$ *MAC-2*	+3.1%	−15.5%	+8.0%	−2.5%	+54.9%	−58.6%	−57.6%	−35.8%	−34.3%

[a] Plain assembler-optimized implementation with FIPS Montgomery multiplication in $\mathbb{F}_p$.
[b] Multiplication in $\mathbb{F}_{p^2}$ uses lazy reduction technique by Sánchez and Rodríguez-Henríquez [SR13].
[c] Additionally incorporates the countermeasure against DPA attacks.

requirements. Inclusion of the countermeasure to DPA attacks reduces performance and energy efficiency slightly, but is negligibly. More interesting is the usage of either of the two multiply-accumulate extensions *MAC-1* and *MAC-2*. These reduce the requirements for constant memory by up to 20%, but reduce chip size only slightly, because the extensions increase the processor core's size. On the other hand, runtime drops massively by up to 60%, also yielding lower energy consumption. Since power consumption for *MAC-2* is increased by one half, the platform using *MAC-1* is more energy-efficient.

Finally, the impact of the several implementation variants on their respective hardware platforms for the curve BN158 are presented in Table 8.16. Similar to BN254, both runtime and demand for energy are reduced when employing the lazy reduction technique in the multiplication in $\mathbb{F}_{p^2}$. This remains valid for the secured implementation. However, chip size increases minimally. The two instruction-set extensions *MAC-1* and *MAC-2* further reduce runtime of encapsulation and decapsulation by 50%. This is also reflected in energy consumption. However, the energy reduction for the extension *MAC-2* is only 20-25% since power consumption rises by 50% in this case. On the other hand, *MAC-1* leads to an overall 5% larger chip, while there is a negligible effect on chip size when using *MAC-2*.

8.4. Related Work

Many implementations of bilinear pairings have been published in recent years. In the earlier years, the focus was on pairings over binary curves, since they are generally faster. However, the results by Joux [Jou13] and Göloğlu et al. [Göl+13] indicate that pairings over fields with small characteristic might be less secure. Therefore, the comparison to related work is restricted to those implementations that are based on finite fields with large prime characteristic.

Table 8.16.: Comparison of the various hardware platforms for the BB_1 IBE KEM using a 10 Mhz clock and the curve BN2158.

Variant	**Area**				**Power**	**Runtime**		**Energy**	
	RAM [kGE]	ROM [kGE]	Core [kGE]	Total [kGE]	Typ. [mW]	Enc. [ms]	Dec. [ms]	Enc. [mJ]	Dec. [mJ]
Standard[a]	11.6	17.0	16.3	45.7	0.70	3,770	2,515	2.64	1.76
LR-$\mathbb{F}_{p^2}$[b]	11.9	17.0	16.3	46.0	0.70	3,764	2,396	2.64	1.68
LR_{sc}-$\mathbb{F}_{p^2}$[c]	11.9	17.0	16.3	46.0	0.70	3,764	2,402	2.64	1.69
LR_{sc}-$\mathbb{F}_{p^2}$ *MAC-1*	11.9	15.6	19.8	48.1	0.74	1,696	1,198	1.26	0.89
LR_{sc}-$\mathbb{F}_{p^2}$ *MAC-2*	11.9	15.6	17.6	45.8	1.10	1,814	1,270	1.99	1.39
Relative Performance									
LR-$\mathbb{F}_{p^2}$	+2.5%	±0.0%	±0.0%	+0.6%	±0.0%	−0.2%	−4.7%	−0.1%	−4.7%
LR_{sc}-$\mathbb{F}_{p^2}$	+2.5%	±0.0%	±0.0%	+0.6%	±0.0%	−0.2%	−4.5%	±0.0%	−4.3%
LR_{sc}-$\mathbb{F}_{p^2}$ *MAC-1*	+2.5%	−8.3%	+21.5%	+5.2%	+5.9%	−55.0%	−52.4%	−52.3%	−49.5%
LR_{sc}-$\mathbb{F}_{p^2}$ *MAC-2*	+2.5%	−8.3%	+8.0%	+0.4%	+56.3%	−51.9%	−49.5%	−24.8%	−21.1%

[a] Plain assembler-optimized implementation with FIPS Montgomery multiplication in $\mathbb{F}_p$.
[b] Multiplication in $\mathbb{F}_{p^2}$ uses lazy reduction technique by Sánchez and Rodríguez-Henríquez [SR13].
[c] Additionally incorporates the countermeasure against DPA attacks.

In Table 8.17, performance of pairing implementations providing about 128 bits of security is shown. The MSP430 platform used by Gouvêa [GL09; GOL12a] is targeted at similar applications as the Cortex-M0+. Consequently, these implementations seem good for comparison. Runtime of the optimal Ate pairing on the MSP430 is about 67% slower than on the Cortex-M0+. If its successor MSP430X is used, the Cortex-M0+ still performs 41% better. However, if an MSP430 with integrated 32-bit multiplication unit is utilized, the MSP430 seems to be equally fast. Contrary to that, the implementation in this work based on the Cortex-M0+ needs 40%-57% less RAM and 60% less ROM. In this respect, the platform presented is clearly more efficient. When further using one of the two instruction-set extensions *MAC-1* or *MAC-2*, performance of the Cortex-M0+-based platform becomes even better compared to the MSP430.

The implementations on the Cortex-A and Intel processors are listed to give a sense of their performance relative to the Cortex-M0+. Note that these are typically clocked at over 1 GHz, while the Cortex-M0+ is clocked between 10 MHz and 50 MHz. Therefore, cycle counts are not as comparative as for the MSP430 in this case. Using the much more powerful Cortex-A9, the implementation by Acar et al. [Aca+13] needs 7% more cycles compared to this work's implementation. However, Grewal et al. [Gre+13] and Sánchez and Rodríguez-Henríquez [SR13] presented implementations that exploit the advantages of the application processors Cortex-A9 and Cortex-A15 and yield about 75% better cycle counts than on the Cortex-M0+. Even better results were shown by Sánchez and Rodríguez-Henríquez [SR13] when the NEON extensions are used. These yield a similar speed-up as the two proposed instruction-set extensions *MAC-1* and *MAC-2*. In terms of cycle counts, the implementation on the Intel i7 by Beuchat et al. [Beu+10] is about 20 times faster than this work, which is even topped by the results on the Intel i5 by Aranha et al. [Ara+11]. However, both the Intel i5 and i7 are massively more powerful. Note that none of these implementations but the ones presented in this thesis provide sufficient security against side-channel attacks.

Table 8.17.: Comparison of pairing implementations providing roughly 128 bits of security.

Variant	Platform	κ [Bit]	RAM [Bytes]	ROM [Bytes]	Pairing [kCycles]	Type
LR_{sc}-$\mathbb{F}_{p^2}$	Cortex-M0+	127	2,824	14,740	47,644	Opt. Ate
LR_{sc}-$\mathbb{F}_{p^2}$ *MAC-1*	Cortex-M0+	127	2,820	9,496	20,536	Opt. Ate
LR_{sc}-$\mathbb{F}_{p^2}$ *MAC-2*	Cortex-M0+	127	2,820	9,728	21,681	Opt. Ate
Gouvêa [GL09]	MSP430	128	4,700	36,200	117,598	Opt. Ate
Gouvêa [GOL12a]	MSP430	127	6,500	36,000	79,440	Opt. Ate
	MSP430X	127	6,500	35,200	67,688	Opt. Ate
	MSP430/MPY32	127	6,500	34,400	47,736	Opt. Ate
Acar [Aca+13]	Cortex-A9	127	-	-	51,010	Opt. Ate
Sánchez [SR13]	Cortex-A15	127	-	-	13,618	Opt. Ate
	Cortex-A15/NEON	127	-	-	5,838	Opt. Ate
Grewal [Gre+13]	Cortex-A9	127	-	-	11,886	Opt. Ate
Beuchat [Beu+10]	Intel i7	128	-	-	2,330	Opt. Ate
Aranha [Ara+11]	Intel i5	127	-	-	1,703	Opt. Ate

Table 8.18.: Comparison of pairing implementations providing less than 128 bits of security.

Variant	Platform	κ [Bit]	RAM [Bytes]	ROM [Bytes]	Pairing [kCycles]	Type
LR_{sc}-$\mathbb{F}_{p^2}$	Cortex-M0+	79	1,852	10,640	17,389	Opt. Ate
LR_{sc}-$\mathbb{F}_{p^2}$ *MAC-1*	Cortex-M0+	79	1,848	8,308	8,789	Opt. Ate
LR_{sc}-$\mathbb{F}_{p^2}$ *MAC-2*	Cortex-M0+	79	1,848	8,452	9,300	Opt. Ate
TinyTate[Oli+07]	ATMega	64	1,831	18,384	223,034	Tate
NanoECC [Szc+08]	ATMega	80	2,500	71,900	132,373	Ate
Szczechowiak[Szc+09]	ATMega	64	3,390	60,910	54,800	Tate
Gouvêa [GL09]	MSP430	70	2,300	28,900	40,869	Tate

A comparison of pairing implementations at a lower level of security is done in Table 8.18. Comparable work uses the slower Tate and Ate pairings. The implementation on the Cortex-M0+ results in massively lower memory requirements and runtime that is 2.3-12 times faster. This becomes even more obvious when considering the provided level of security, which, apart from the implementation by Szczechowiak et al. [Szc+08], is significantly lower for the related work.

The positioning of the hardware platform for computing pairings is shown in Table 8.19. When not using instruction-set extensions, the implementations provided in this thesis are massively slower. Apart from one variant presented by Kammler et al. [Kam+09], the implementations by Kammler et al. and by Fan, Vercauteren, and Verbauwhede [FVV09] are even faster than the presented platform when instruction-set extensions are used. In addition, their platforms can be operated at clock frequencies of 338 Mhz and 204 Mhz respectively, while the Cortex-M0+-based platform is typically clocked at 50 Mhz at most. Contrary to that, the microcontroller-based approach is more general and may hence be used for other applications as well. Addtionally, the platform is over 60% smaller. Figure 8.3 summarizes these results and shows, that the implementation by Fan, Vercauteren,

Table 8.19.: Comparison of hardware platforms for computing pairings using a 130 nm process.

Variant	Platform	κ [Bit]	Area [GE]	Pairing [kCycles]	Type
LR_{sc}-$\mathbb{F}_{p^2}$	Cortex-M0+	127	50,008	47,644	Opt. Ate
LR_{sc}-$\mathbb{F}_{p^2}$ *MAC-1*	Cortex-M0+	127	48,289	20,536	Opt. Ate
LR_{sc}-$\mathbb{F}_{p^2}$ *MAC-2*	Cortex-M0+	127	46,202	21,681	Opt. Ate
LR_{sc}-$\mathbb{F}_{p^2}$	Cortex-M0+	79	42,144	17,389	Opt. Ate
LR_{sc}-$\mathbb{F}_{p^2}$ *MAC-1*	Cortex-M0+	79	43,796	8,789	Opt. Ate
LR_{sc}-$\mathbb{F}_{p^2}$ *MAC-2*	Cortex-M0+	79	41,609	9,300	Opt. Ate
Kammler [Kam+09]	RISC Core	128	164,000	5,340	Opt. Ate
	RISC Core	128	145,000	6,490	Opt. Ate
	RISC Core	128	130,000	10,816	Opt. Ate
Fan [FVV09]	ASIC	128	183,000	593	R-Ate
	ASIC	128	183,000	862	Ate

and Verbauwhede is the most efficient. The implementations by Kammler et al. are approximately as efficient as the two Cortex-M0+ platforms equipped with instruction-set extensions. The least-efficient platform is based on a plain Cortex-M0+ microprocessor.

8.5. Conclusion

In this chapter, performance of the implementations provided in this thesis was evaluated. Since finite field arithmetic is crucial, this was looked at in the first place. It was shown how runtime of addition, multiplication and inversion scales in extension fields and depending on the size of the prime being used. Consequently, the pairing implementation along with the elliptic curve point multiplication was investigated. It turned out, that pairing computations are 3-4 times slower than point multiplications in $E(\mathbb{F}_p)$. Further, lazy reduction in the base field $\mathbb{F}_p$ was shown not to be beneficial. However, the lazy reduction trick in the multiplication in $\mathbb{F}_{p^2}$ improved runtime noticeably. This was remarkably enhanced by using either of the instruction-set extensions. Additionally, the demand for memory is reduced by these extensions. The effect of these optimizations could also be observed in a similar manner for the BB_1 IBE KEM. There, it turned out that encapsulation is significantly more expensive than decapsulation. Moreover, the need for constant memory is clearly higher than for a simple pairing computation.

The impact of in- and decreases in terms of memory size was also reflected in the area of the respective hardware platforms. Using instruction-set extensions raises power consumption, especially when utilizing *MAC-2*. However, the platforms with multiply-accumulate extensions are significantly more energy-efficient and to be preferred in battery-powered systems.

In the context of related work, the pairing implementation provided is clearly superior to any other implementation in the same processor segment, which is largely contributed to the 32-bit architecture. The memory requirements are extraordinary low for the presented platform. Furthermore, the investigated related work lacks countermeasures to prevent side-channel attacks, which typically reduce performance. The provided hardware platform to compute pairings is clearly slower than the ones presented in related work.

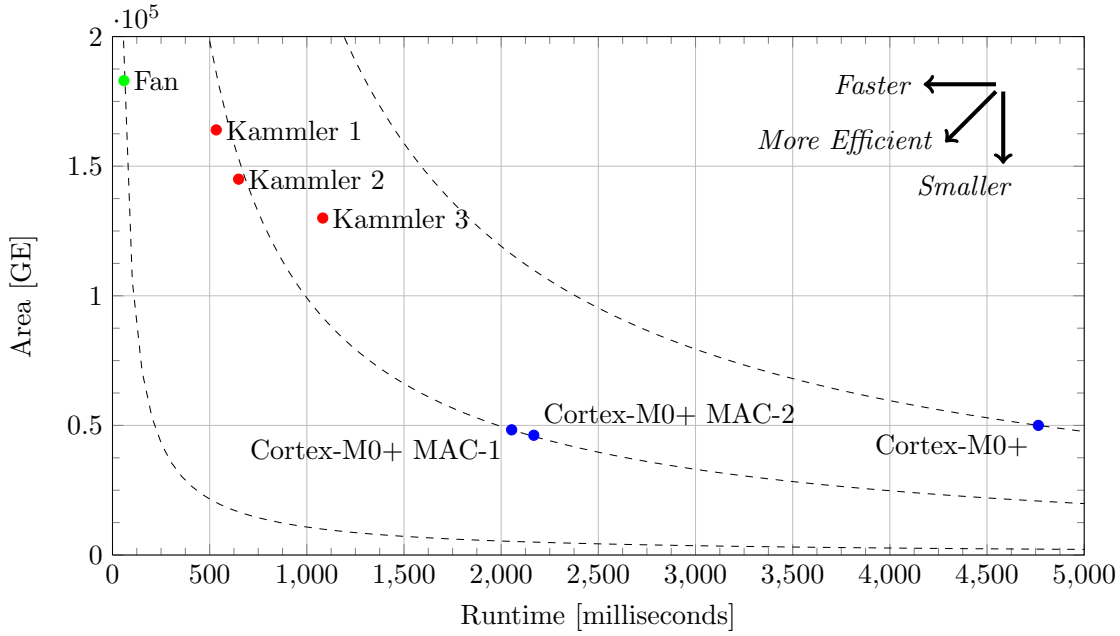

Figure 8.3.: Area-runtime characteristics at 10 MHz.

This drawback becomes even more significant when taking the possible clock frequency into account. Nevertheless, the platform based on the Cortex-M0+ is general-purpose and also significantly smaller.

9. Conclusion

Cryptography nowadays suffers from several problems: symmetric encryption schemes rely on the distribution of a common secret key, while asymmetric schemes have the problem of authenticity of the public keys used for encryption. Consequently, identity-based encryption was presented as a promising alternative, where a publicly known identity-string, for example an e-mail address, is used instead of the public key for data encryption.

Since most published identity-based encryption schemes rely on bilinear pairings, their mathematical background was looked at in more detail. In the most common case, bilinear parings map two elements from groups over elliptic curves to an element in a large extension field, for example, $E(\mathbb{F}_p) \times E(\mathbb{F}_{p^2}) \longrightarrow \mathbb{F}_{p^{12}}$ for BN curves. These maps are efficiently computable and fulfill a bilinearity property, which allows creation of new protocols based on the difficulty of the bilinear Diffie-Hellman problem. However, overall complexity of such systems is high.

Consecutively, various identity-based encryption schemes that followed the scheme by Boneh and Franklin [BF01] were presented. These differ enormously in size of public domain parameters, performance, and the proven notion of security. A special focus was put on the latter, which lead to the conclusion that adaptive-identity security without random oracles is the most secure type of security. It seems, that it provides more security than selective-identity security without random oracles even in its weakest form. Two of the mentioned identity-based encryption schemes, namely the KEM variant of the BB_1 IBE by Boyen [Boy06] and the KEM IBE by Kiltz [Kil06], were investigated in more detail since they seemed suitable for usage in embedded applications. It turned out, that in order to be able to provide sufficient performance on a resource-limited device, the weaker notion of security provided by the BB_1 IBE KEM had to be accepted.

The BB_1 IBE KEM was implemented on a hardware platform based on the Cortex-M0+ microprocessor, which targets the embedded applications. This approach offered clear advantages. First, the processor can be used for other applications as well and is not limited to identity-based encryption. Second, a microprocessor seems most suitable for the irregular data flow that evolves from the executed operations. At last, existing compilers can be used. The layered implementation was then explained on architectural level. Following, the most important implementation aspects and optimization strategies were explained. In this matter, a new method for final exponentiation in the pairing computation could be presented, which significantly reduces the demand for memory. In addition, the parameterization of the prime allows more efficient inversion in the prime field being used. Nevertheless, the main contribution is the optimized finite field arithmetic on the Cortex-M0+ processor, which provides good runtime and needs little memory. This was further improved by equipping the processor with two different multiply-accumulate instruction-set extensions, which make the platform faster and more energy-efficient.

Side-channel attacks pose a serious threat to security applications in embedded environments since an adversary very probably has full access to the device. Therefore, the most important types of side-channel attacks, among them timing attacks, simple and differential power analysis attacks, were discussed. The implementation of the identity-based

encryption scheme was done with respect to preventing many side-channel attacks. This was pointed out partly in Chapter 6, but in more detail in Chapter 7. Moreover, a DPA attack on the pairing computation in the decapsulation routine of the identity-based encryption scheme was shown. It allows extraction of the identity's private key. Fortunately, a simple but effective countermeasure, in particular randomization of the private elliptic curve point in the pairing computation, could be proposed and included in the implementation. If the identity-based encryption scheme was set up differently, that is, the private key was not in $E(\mathbb{F}_{p^2})$, but in $E(\mathbb{F}_p)$, the countermeasure would take considerably more effort.

The various software implementations done for the three curves BN256, BN254 and BN158 were evaluated in detail. It turned out, that encapsulation in the identity-based encryption scheme is roughly by 50-75% more expensive than decapsulation. On a typical platform clocked at 10 MHz, encapsulation is done in 12 and 3.7 seconds at the 128-bit and 80-bit security levels, respectively. The latter seems acceptable, especially if 80 bits of security are enough for the embedded applications. When comparing the proposed architecture to related work, it was found that dedicated hardware platforms consume much more chip area, but are significantly faster. On the other hand, architectures based on processors also targeted at the embedded market have massively longer runtime and memory requirements, which are a number of times higher. Considering side-channel security and the general-purpose architecture, the platform presented in this work seems to fill the current gap between the existing microcontroller-based solutions and dedicated hardware platforms.

Appendix

A. Point Multiplication Formulas

For the routines `AddDblCoZ` and `RecoverFullCoordinatesCoZ` in the elliptic curve point multiplication from Algorithm 5, the variants having both low memory requirements and little computational effort were chosen. Algorithm 16 illustrates the combined double-and-add step using homogeneous projective co-Z coordinates. Correspondingly, Algorithm 17 shows how to recover the full point from the two x-coordinates and the z-coordinate.

In these two Algorithms, X_1, X_2 and Z are variables representing the two points used in the Montgomery ladder. The variables a and $4b$ constitute the parameters of the elliptic curve $y^2 = x^3 + ax + b$. The coordinates x_D and y_D belong to the difference of the two points used in the Montgomery ladder, namely $D = R_2 - R_1 = (x_D, y_D)$. This difference D is invariant during the whole point multiplication and equals the original point P that is to be multiplied.

Algorithm 16 Out-of-place version of `AddDblCoZ` using homogeneous projective co-Z coordinates by Hutter, Joye, and Sierra [HJS11, Alg. 5].

Input: $X_1, X_2, Z, x_D, a, 4b$
Output: X_1, X_2, Z
1: $R_2 \leftarrow Z^2$
2: $R_3 \leftarrow a \cdot R_2$
3: $R_1 \leftarrow Z \cdot R_2$
4: $R_2 \leftarrow 4b \cdot R_1$
5: $R_1 \leftarrow X_2^2$
6: $R_5 \leftarrow R_1 - R_3$
7: $R_4 \leftarrow R_5^2$
8: $R_1 \leftarrow R_1 + R_3$
9: $R_5 \leftarrow X_2 \cdot R_1$
10: $R_5 \leftarrow R_5 + R_5$
11: $R_5 \leftarrow R_5 + R_5$
12: $R_5 \leftarrow R_5 + R_2$
13: $R_1 \leftarrow R_1 + R_3$
14: $R_3 \leftarrow X_1^2$
15: $R_1 \leftarrow R_1 + R_3$
16: $X_1 \leftarrow X_1 - X_2$
17: $X_2 \leftarrow X_2 + X_2$
18: $R_3 \leftarrow X_2 \cdot R_2$
19: $R_4 \leftarrow R_4 - R_3$
20: $R_3 \leftarrow X_1^2$
21: $R_1 \leftarrow R_1 - R_3$
22: $X_1 \leftarrow X_1 + X_2$
23: $X_2 \leftarrow X_1 \cdot R_1$
24: $X_2 \leftarrow X_2 + R_2$
25: $R_2 \leftarrow Z \cdot R_3$
26: $Z \leftarrow x_D \cdot R_2$
27: $X_2 \leftarrow X_2 - Z$
28: $X_1 \leftarrow R_5 \cdot X_2$
29: $X_2 \leftarrow R_3 \cdot R_4$
30: $Z \leftarrow R_2 \cdot R_5$
31: **return** $\{X_1, X_2, Z\}$

Algorithm 17 Out-of-place version of `RecoverFullCoordinatesCoZ` using homogeneous projective co-Z coordinates by Hutter, Joye, and Sierra [HJS11, Alg. 7].

Input: $X_1, X_2, Z, x_D, y_D, a, 4b$
Output: X_1, X_2, z
1: $R_1 \leftarrow x_D \cdot Z$
2: $R_2 \leftarrow X_1 - R_1$
3: $R_3 \leftarrow R_2^2$
4: $R_4 \leftarrow R_3 \cdot X_2$
5: $R_2 \leftarrow R_1 \cdot X_1$
6: $R_1 \leftarrow X_1 + R_1$
7: $X_2 \leftarrow Z^2$
8: $R_3 \leftarrow a \cdot X_2$
9: $R_2 \leftarrow R_2 + R_3$
10: $R_3 \leftarrow R_2 \cdot R_1$
11: $R_3 \leftarrow R_3 - R_4$
12: $R_3 \leftarrow R_3 + R_3$
13: $R_1 \leftarrow y_D + y_D$
14: $R_1 \leftarrow R_1 + R_1$
15: $R_2 \leftarrow R_1 \cdot X_1$
16: $X_1 \leftarrow R_2 \cdot X_2$
17: $R_2 \leftarrow R_2 \cdot Z$
18: $Z \leftarrow R_2 \cdot R_1$
19: $R_4 \leftarrow 4b \cdot R_2$
20: $X_2 \leftarrow R_4 + R_3$
21: **return** $\{X_1, X_2, Z\}$

B. Pairing Evaluation Formulas

Costello, Lange, and Naehrig [CLN10, Section 5] presented fast formulas for interleaved line evaluation and point addition for computing pairings. Accordingly, fast formulas for interleaved line evaluation and and point doubling were shown. In particular, they presented formulas for curves of the form $E : y^2 = x^3 + b$, which have even embedding degree and a sextic twist E'. They use homogeneous projective coordinates, which transform the curve equation to $Y^2Z = X^3 + bZ^3$. Consequently, these formulas are perfectly suitable for BN curves that are used in this thesis.

The double of the point $T = (T_x, T_y, T_z) \in E'(\mathbb{F}_{p^2})$ in projective coordinates is denoted $T' = (T'_x, T'_y, T'_z) = [2](T_x, T_y, T_z) \in E'(\mathbb{F}_{p^2})$ and computed as

$$T'_x = 2T_xT_y(T_y^2 - 9bT_z^2),\ T'_y = T_y^4 + 18bT_y^2T_z^2 - 27b^2T_z^4,\ T'_z = 8T_y^3T_z.$$

Let z denote the root of the irreducible polynomial $z^6 - \xi$ over $\mathbb{F}_{p^2}$, which is used to define $\mathbb{F}_{p^{12}}$. The tangent line in T is represented in homogeneous projective coordinates as

$$\ell_{T,T}(P) = (3bT_z^2 - T_y^2) \cdot z^3 + 3T_x^2 \cdot P_x \cdot z - 2T_yT_z \cdot P_y \in \mathbb{F}_{p^{12}},$$

where $P = (P_x, P_y) \in E(\mathbb{F}_p)$ denotes the point at which the tangent line has to be evaluated. Let $\ell_{T,T}(P) = L_{0,1} \cdot z^3 + L_{1,0} \cdot P_x \cdot z + L_{0,0} \cdot P_y$ and $T' = (T'_x, T'_y, T'_z) = [2](T_x, T_y, T_z)$. The interleaved point doubling and the tangent line evaluation can be done as follows:

$$\begin{aligned}
&A = T_x^2,\ B = T_y^2,\ C = T_z^2,\ D = 3bC,\ E = (T_x + T_y)^2 - A - B,\\
&F = (T_y + T_z)^2 - B - C,\ G = 3D,\ T'_x = E \cdot (B - G),\ T'_y = (B + G)^2 - 12D^2,\\
&T'_z = 4B \cdot F,\ L_{1,0} = 3A,\ L_{0,0} = -F,\ L_{0,1} = D - B.
\end{aligned}$$

This sequence performs in only two multiplications, 7 squarings and one multiplication by the constant b in $\mathbb{F}_{p^2}$. The full evaluation of $\ell_{T,T}(P)$ takes only two more multiplications by $P_x, P_y \in \mathbb{F}_p$ respectively.

According to Costello et al. [Cos+09], the sum of two points $T = (T_x, T_y, T_z)$ and $Q = (Q_x, Q_y, 1) \in E'(\mathbb{F}_{p^2})$ using homogeneous projective coordinates, denoted $T' = (T'_x, T'_y, T'_z) = (T_x, T_y, T_z) + (Q_x, Q_y, 1) \in E'(\mathbb{F}_{p^2})$, is computed as

$$\begin{aligned}
T'_x &= (T_x - T_zQ_x)(T_z(T_y - T_zQ_y)^2 - (T_x + T_zQ_x)(T_x - T_zQ_x)^2),\\
T'_y &= (T_y - T_zQ_y)((2T_x + T_zQ_x)(T_x - T_zQ_x)^2 - T_z(T_y - T_zQ_y)^2) - T_y(T_x - T_zQ_x)^3,\\
T'_z &= T_z(T_x - T_zQ_x)^3.
\end{aligned}$$

The line between the two points T and Q is represented as

$$\begin{aligned}
\ell_{T,Q}(P) = &((T_y - T_zQ_y) \cdot Q_x - (T_x - T_zQ_x) \cdot Q_y) \cdot z^3 -\\
&(T_y - T_zQ_y) \cdot P_x \cdot z + (T_x - T_zQ_x) \cdot P_y,
\end{aligned}$$

where $P = (P_x, P_y) \in E(\mathbb{F}_p)$ denotes the point at which the line is evaluated. Writing $\ell_{T,Q}(P) = L_{0,1} \cdot z^3 + L_{1,0} \cdot P_x \cdot z + L_{0,0} \cdot P_y$ and let $T' = (T'_x, T'_y, T'_z) = (T_x, T_y, T_z) + (Q_x, Q_y, 1)$, the interleaved point addition and the line evaluation can be done as follows:

$$\begin{aligned} A = T_y - T_z Q_y,\ B = T_x - T_z Q_x,\ C = B^2,\ D = B \cdot C, \\ E = C \cdot T_x,\ F = A^2 T_z + D - 2E,\ G = (E - F) \cdot A, \\ T'_x = B \cdot F,\ T'_y = G - D \cdot T_y,\ T'_z = D \cdot T_z, \\ L_{0,0} = B,\ L_{1,0} = -A,\ L_{0,1} = A \cdot Q_x - B \cdot Q_y. \end{aligned}$$

The chain presented takes 11 multiplications and two squarings in $\mathbb{F}_{p^2}$. As for point doubling, the full evaluation of $\ell_{P'_1,P'_2}(S)$ comprises only two more multiplications by $P_x, P_y \in \mathbb{F}_p$ respectively.

Bibliography

[Aca+13] Tolga Acar et al. "Affine pairings on ARM". In: *Proceedings of the 5th international conference on Pairing-Based Cryptography*. Pairing'12. Cologne, Germany: Springer-Verlag, 2013. URL: http://dx.doi.org/10.1007/978-3-642-36334-4_13 (cit. on pp. 97, 98).

[AM93] A. O. L. Atkin and F. Morain. "Elliptic Curves And Primality Proving". In: *Math. Comp* 61 (1993), pp. 29–68 (cit. on p. 27).

[AR12] Gora Adj and Francisco Rodríguez-Henríquez. *Square root computation over even extension fields*. Cryptology ePrint Archive, Report 2012/685. http://eprint.iacr.org/. 2012 (cit. on p. 77).

[Ara+11] DiegoF. Aranha et al. "Faster Explicit Formulas for Computing Pairings over Ordinary Curves". In: *Advances in Cryptology – EUROCRYPT 2011*. Ed. by KennethG. Paterson. Vol. 6632. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2011, pp. 48–68. ISBN: 978-3-642-20464-7. DOI: 10.1007/978-3-642-20465-4_5. URL: http://dx.doi.org/10.1007/978-3-642-20465-4_5 (cit. on pp. 67, 97, 98).

[AT03] Toru Akishita and Tsuyoshi Takagi. "Zero-Value Point Attacks on Elliptic Curve Cryptosystem". In: *Information Security*. Ed. by Colin Boyd and Wenbo Mao. Vol. 2851. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2003, pp. 218–233. ISBN: 978-3-540-20176-2. DOI: 10.1007/10958513_17. URL: http://dx.doi.org/10.1007/10958513_17 (cit. on p. 46).

[Att+05] Nuttapong Attrapadung et al. *Efficient Identity-Based Encryption with Tight Security Reduction*. 2005 (cit. on p. 34).

[Bak+04] S. Baktir et al. "Optimal tower fields for hyperelliptic curve cryptosystems". In: *Signals, Systems and Computers, 2004. Conference Record of the Thirty-Eighth Asilomar Conference on*. Vol. 1. 2004, 522–526 Vol.1. DOI: 10.1109/ACSSC.2004.1399187 (cit. on p. 57).

[Bar+02] Paulo S. L. M. Barreto et al. "Efficient Algorithms for Pairing-Based Cryptosystems". In: *Proceedings of the 22nd Annual International Cryptology Conference on Advances in Cryptology*. CRYPTO '02. London, UK, UK: Springer-Verlag, 2002, pp. 354–368. ISBN: 3-540-44050-X. URL: http://dl.acm.org/citation.cfm?id=646767.704315 (cit. on p. 61).

[Bar+04] Paulo S. L. M. Barreto et al. "Efficient Pairing Computation on Supersingular Abelian Varieties". In: *Designs, Codes and Cryptography*. 2004, pp. 239–271 (cit. on p. 27).

[Bar87] Paul Barrett. "Implementing the Rivest Shamir and Adleman Public Key Encryption Algorithm on a Standard Digital Signal Processor". English. In: *Advances in Cryptology — CRYPTO' 86*. Ed. by AndrewM. Odlyzko. Vol. 263. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 1987, pp. 311–323. ISBN: 978-3-540-18047-0. DOI: 10.1007/3-540-47721-7_24. URL: http://dx.doi.org/10.1007/3-540-47721-7_24 (cit. on p. 53).

[BB04a] Dan Boneh and Xavier Boyen. "Efficient Selective-ID Secure Identity Based Encryption Without Random Oracles". In: *Advances in Cryptology—EUROCRYPT 2004*. Vol. 3027. Lecture Notes in Computer Science. Available at http://www.cs.stanford.edu/~xb/eurocrypt04b/. Berlin: Springer-Verlag, 2004, pp. 223–238 (cit. on pp. 34, 35, 39).

[BB04b] Dan Boneh and Xavier Boyen. "Secure Identity Based Encryption Without Random Oracles". In: *Advances in Cryptology – CRYPTO 2004*. Ed. by Matt Franklin. Vol. 3152. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2004, pp. 443–459. ISBN: 978-3-540-22668-0. DOI: 10.1007/978-3-540-28628-8_27. URL: http://dx.doi.org/10.1007/978-3-540-28628-8_27 (cit. on pp. 21, 35).

[Ben+08] K. Bentahar et al. "Generic Constructions of Identity-Based and Certificateless KEMs". English. In: *Journal of Cryptology* 21.2 (2008), pp. 178–199. ISSN: 0933-2790. DOI: 10.1007/s00145-007-9000-z. URL: http://dx.doi.org/10.1007/s00145-007-9000-z (cit. on p. 40).

[Beu+10] Jean-Luc Beuchat et al. "High-Speed Software Implementation of the Optimal Ate Pairing over Barreto–Naehrig Curves". In: *Pairing-Based Cryptography - Pairing 2010*. Ed. by Marc Joye, Atsuko Miyaji, and Akira Otsuka. Vol. 6487. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2010, pp. 21–39. ISBN: 978-3-642-17454-4. DOI: 10.1007/978-3-642-17455-1_2. URL: http://dx.doi.org/10.1007/978-3-642-17455-1_2 (cit. on pp. 1, 97, 98).

[BF01] Dan Boneh and Matthew Franklin. "Identity-Based Encryption from the Weil Pairing". In: Springer-Verlag, 2001, pp. 213–229 (cit. on pp. 8, 21, 31, 34, 101).

[BG81] Charles H. Bennett and John Gill. "Relative to a Random Oracle A, PA != NPA != co-NPA with Probability 1." In: *SIAM J. Comput.* 10.1 (1981), pp. 96–113. URL: http://dblp.uni-trier.de/db/journals/siamcomp/siamcomp10.html#BennettG81 (cit. on p. 31).

[BK04] Dan Boneh and Jonathan Katz. "Improved Efficiency for CCA-Secure Cryptosystems Built Using Identity-Based Encryption". In: Springer-Verlag, 2004, pp. 87–103 (cit. on p. 34).

[BL13] D. J. Bernstein and T. Lange. *Explicit-Formulas Database*. Aug. 2013 (cit. on p. 18).

[BLS04] PauloS.L.M. Barreto, Ben Lynn, and Michael Scott. "On the Selection of Pairing-Friendly Groups". In: *Selected Areas in Cryptography*. Ed. by Mitsuru Matsui and RobertJ. Zuccherato. Vol. 3006. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2004, pp. 17–25. ISBN: 978-3-540-21370-3. DOI: 10.1007/978-3-540-24654-1_2. URL: http://dx.doi.org/10.1007/978-3-540-24654-1_2 (cit. on p. 61).

[BMW05] Xavier Boyen, Qixiang Mei, and Brent Waters. "Direct Chosen Ciphertext Security from Identity-Based Techniques". In: *In ACM Conference on Computer and Communications Security*. ACM Press, 2005, pp. 320–329 (cit. on pp. 34, 38).

[BN06] PauloS.L.M. Barreto and Michael Naehrig. "Pairing-Friendly Elliptic Curves of Prime Order". In: *Selected Areas in Cryptography*. Ed. by Bart Preneel and Stafford Tavares. Vol. 3897. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2006, pp. 319–331. ISBN: 978-3-540-33108-7. DOI: 10.1007/11693383_22. URL: http://dx.doi.org/10.1007/11693383_22 (cit. on pp. 27–29, 55).

[Bon98] Dan Boneh. "The Decision Diffie-Hellman problem". In: *Algorithmic Number Theory*. Ed. by JoeP. Buhler. Vol. 1423. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 1998, pp. 48–63. ISBN: 978-3-540-64657-0. DOI: 10.1007/BFb0054851. URL: http://dx.doi.org/10.1007/BFb0054851 (cit. on p. 20).

[Boy06] Xavier Boyen. *The BB1 identity-based cryptosystem: a standard for encryption and key encapsulation*. Aug. 2006. URL: http://grouper.ieee.org/groups/1363/IBC/submissions/Boyen-bb1ieee.pdf (cit. on pp. 2, 35–37, 40–42, 52, 101).

[CC05] Liqun Chen and Zhaohui Cheng. "Security proof of Sakai-Kasahara's identity-based encryption scheme". In: *In Proceedings of Cryptography and Coding 2005, LNCS 3706*. Springer-Verlag, 2005, pp. 442–459 (cit. on p. 34).

[CH07] Jaewook Chung and M.A. Hasan. "Asymmetric Squaring Formulae". In: *Computer Arithmetic, 2007. ARITH '07. 18th IEEE Symposium on*. 2007, pp. 113–122. DOI: 10.1109/ARITH.2007.11 (cit. on pp. 56, 68).

[Che+06] L. Chen et al. "An Efficient ID-KEM Based On The Sakai-Kasahara Key Construction". In: *IEE Proceedings of Information Security*. 2006 (cit. on p. 34).

[Che06] JungHee Cheon. "Security Analysis of the Strong Diffie-Hellman Problem". In: *Advances in Cryptology - EUROCRYPT 2006*. Ed. by Serge Vaudenay. Vol. 4004. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2006, pp. 1–11. ISBN: 978-3-540-34546-6. DOI: 10.1007/11761679_1. URL: http://dx.doi.org/10.1007/11761679_1 (cit. on pp. 35, 37, 39, 41).

[CHK03] Ran Canetti, Shai Halevi, and Jonathan Katz. "A Forward-Secure Public-Key Encryption Scheme". English. In: *Advances in Cryptology — EUROCRYPT 2003*. Ed. by Eli Biham. Vol. 2656. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2003, pp. 255–271. ISBN: 978-3-540-14039-9. DOI: 10.1007/3-540-39200-9_16. URL: http://dx.doi.org/10.1007/3-540-39200-9_16 (cit. on p. 34).

[CLN10] Craig Costello, Tanja Lange, and Michael Naehrig. "Faster Pairing Computations on Curves with High-Degree Twists". In: *Public Key Cryptography – PKC 2010*. Ed. by PhongQ. Nguyen and David Pointcheval. Vol. 6056. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2010, pp. 224–242. ISBN: 978-3-642-13012-0. DOI: 10.1007/978-3-642-13013-7_14. URL: http://dx.doi.org/10.1007/978-3-642-13013-7_14 (cit. on pp. 61, 107).

[Coc01] Clifford Cocks. "An identity based encryption scheme based on quadratic residues". In: *IN IMA INT. CONF*. Springer-Verlag, 2001, pp. 360–363 (cit. on p. 8).

[Coh+10] H. Cohen et al. *Handbook of Elliptic and Hyperelliptic Curve Cryptography*. Discrete Mathematics and Its Applications. Taylor & Francis, 2010. ISBN: 9781420034981 (cit. on p. 19).

[Cor99] Jean-Sébastien Coron. "Resistance Against Differential Power Analysis For Elliptic Curve Cryptosystems". English. In: *Cryptographic Hardware and Embedded Systems*. Ed. by ÇetinK. Koç and Christof Paar. Vol. 1717. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 1999, pp. 292–302. ISBN: 978-3-540-66646-2. DOI: 10.1007/3-540-48059-5_25. URL: http://dx.doi.org/10.1007/3-540-48059-5_25 (cit. on p. 46).

[Cos+09] Craig Costello et al. "Faster Pairings on Special Weierstrass Curves". In: *Pairing-Based Cryptography – Pairing 2009*. Ed. by Hovav Shacham and Brent Waters. Vol. 5671. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2009, pp. 89–101. ISBN: 978-3-642-03297-4. DOI: 10.1007/978-3-642-03298-1_7. URL: http://dx.doi.org/10.1007/978-3-642-03298-1_7 (cit. on p. 107).

[Cos13] Craig Costello. *Pairings for Beginners*. 2013. URL: http://www.craigcostello.com.au/pairing/ (cit. on pp. 17, 22–26, 28).

[CS06] Sanjit Chatterjee and Palash Sarkar. "Trading time for space: towards an efficient IBE scheme with short(er) public parameters in the standard model". In: *Proceedings of the 8th international conference on Information Security and Cryptology*. ICISC'05. Seoul, Korea: Springer-Verlag, 2006, pp. 424–440. ISBN: 3-540-33354-1, 978-3-540-33354-8. DOI: 10.1007/11734727_33. URL: http://dx.doi.org/10.1007/11734727_33 (cit. on p. 35).

[CS11] S. Chatterjee and P. Sarkar. *Identity-Based Encryption*. SpringerLink : Bücher. Springer, 2011. ISBN: 9781441993830. URL: http://books.google.fr/books?id=a5mkUnEPMooC (cit. on pp. 31–34, 41).

[Dav72] George I. Davida. "Inverse of elements of a Galois field". In: *Electronics Letters* 8.21 (1972), pp. 518–520. ISSN: 0013-5194. DOI: 10.1049/el:19720378 (cit. on p. 57).

[Dev+06] Augusto Jun Devegili et al. *Multiplication and Squaring on Pairing-Friendly Fields*. Cryptology ePrint Archive, Report 2006/471. http://eprint.iacr.org/. 2006 (cit. on pp. 55, 56).

[DH76] W. Diffie and M.E. Hellman. "New directions in cryptography". In: *Information Theory, IEEE Transactions on* 22.6 (1976), pp. 644–654. ISSN: 0018-9448. DOI: 10.1109/TIT.1976.1055638 (cit. on pp. 3, 20).

[DSD07] AugustoJun Devegili, Michael Scott, and Ricardo Dahab. "Implementing Cryptographic Pairings over Barreto-Naehrig Curves". In: *Pairing-Based Cryptography – Pairing 2007*. Ed. by Tsuyoshi Takagi et al. Vol. 4575. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2007, pp. 197–207. ISBN: 978-3-540-73488-8. DOI: 10.1007/978-3-540-73489-5_10. URL: http://dx.doi.org/10.1007/978-3-540-73489-5_10 (cit. on p. 61).

[Edw07] Harold M. Edwards. "A normal form for elliptic curves". In: *Bulletin of the American Mathematical Society*. 2007, pp. 393–422 (cit. on p. 17).

[ES00] Carl Ellison and Bruce Schneier. "Ten Risks of PKI: What You're Not Being Told About Public Key Infrastructure". In: *Computer Security Journal* 16.1 (2000), pp. 1–7. URL: http://www.schneier.com/paper-pki.pdf (cit. on p. 5).

[FKR12] Laura Fuentes-Castañeda, Edward Knapp, and Francisco Rodríguez-Henríquez. "Faster hashing to G2". In: *Proceedings of the 18th international conference on Selected Areas in Cryptography*. SAC'11. Toronto, ON, Canada: Springer-Verlag, 2012, pp. 412–430. ISBN: 978-3-642-28495-3. DOI: 10.1007/978-3-642-28496-0_25. URL: http://dx.doi.org/10.1007/978-3-642-28496-0_25 (cit. on pp. 61, 69).

[FR94] Gerhard Frey and Hans-Georg Rück. "A remark concerning m-divisibility and the discrete logarithm in the divisor class group of curves". In: *Math. Comput.* 62.206 (Apr. 1994), pp. 865–874. ISSN: 0025-5718. DOI: 10.2307/2153546. URL: http://dx.doi.org/10.2307/2153546 (cit. on p. 20).

[FV03] Pierre-Alain Fouque and Frederic Valette. "The Doubling Attack – Why Upwards Is Better than Downwards". In: *Cryptographic Hardware and Embedded Systems - CHES 2003*. Ed. by ColinD. Walter, ÇetinK. Koç, and Christof Paar. Vol. 2779. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2003, pp. 269–280. ISBN: 978-3-540-40833-8. DOI: 10.1007/978-3-540-45238-6_22. URL: http://dx.doi.org/10.1007/978-3-540-45238-6_22 (cit. on p. 46).

[FV12] Junfeng Fan and Ingrid Verbauwhede. "An Updated Survey on Secure ECC Implementations: Attacks, Countermeasures and Cost". In: *Cryptography and Security: From Theory to Applications*. Ed. by David Naccache. Vol. 6805. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2012, pp. 265–282. ISBN: 978-3-642-28367-3. DOI: 10.1007/978-3-642-28368-0_18. URL: http://dx.doi.org/10.1007/978-3-642-28368-0_18 (cit. on p. 44).

[FVV09] Junfeng Fan, Frederik Vercauteren, and Ingrid Verbauwhede. "Faster $\mathbb{F}_p$-Arithmetic for Cryptographic Pairings on Barreto-Naehrig Curves". In: *Cryptographic Hardware and Embedded Systems - CHES 2009*. Ed. by Christophe Clavier and Kris Gaj. Vol. 5747. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2009, pp. 240–253. ISBN: 978-3-642-04137-2. DOI: 10.1007/978-3-642-04138-9_18. URL: http://dx.doi.org/10.1007/978-3-642-04138-9_18 (cit. on pp. 2, 98, 99).

[Gal06] Steven Galbraith. "Pairings". In: *LONDON MATHEMATICAL SOCIETY LECTURE NOTE SERIES* 317 (2006), p. 183 (cit. on pp. 22, 23, 25, 27).

[Gen06] Craig Gentry. "Practical identity-based encryption without random oracles". In: *Proceedings of the 24th annual international conference on The Theory and Applications of Cryptographic Techniques*. EUROCRYPT'06. St. Petersburg, Russia: Springer-Verlag, 2006, pp. 445–464. ISBN: 3-540-34546-9, 978-3-540-34546-6. DOI: 10.1007/11761679_27. URL: http://dx.doi.org/10.1007/11761679_27 (cit. on p. 35).

[GL09] Conrado Porto Gouvêa and Julio López. "Software Implementation of Pairing-Based Cryptography on Sensor Networks Using the MSP430 Microcontroller". In: *Proceedings of the 10th International Conference on Cryptology in India: Progress in Cryptology*. INDOCRYPT '09. New Delhi, India: Springer-Verlag, 2009. URL: http://dx.doi.org/10.1007/978-3-642-10628-6_17 (cit. on pp. 1, 97, 98).

[GOL12a] ConradoP.L. Gouvêa, LeonardoB. Oliveira, and Julio López. "Efficient software implementation of public-key cryptography on sensor networks using the MSP430X microcontroller". English. In: *Journal of Cryptographic Engineering* 2.1 (2012), pp. 19–29. ISSN: 2190-8508. DOI: 10.1007/s13389-012-0029-z. URL: http://dx.doi.org/10.1007/s13389-012-0029-z (cit. on pp. 1, 97, 98).

[GOL12b] ConradoP.L. Gouvêa, LeonardoB. Oliveira, and Julio López. "Efficient software implementation of public-key cryptography on sensor networks using the MSP430X microcontroller". English. In: *Journal of Cryptographic Engineering* 2.1 (2012), pp. 19–29. ISSN: 2190-8508. DOI: 10.1007/s13389-012-0029-z. URL: http://dx.doi.org/10.1007/s13389-012-0029-z (cit. on p. 64).

[Göl+13] Faruk Göloğlu et al. "On the Function Field Sieve and the Impact of Higher Splitting Probabilities". In: *Advances in Cryptology – CRYPTO 2013*. Ed. by Ran Canetti and JuanA. Garay. Vol. 8043. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2013, pp. 109–128. ISBN: 978-3-642-40083-4. DOI: 10.1007/978-3-642-40084-1_7. URL: http://dx.doi.org/10.1007/978-3-642-40084-1_7 (cit. on p. 96).

[Gou02] Louis Goubin. "A Refined Power-Analysis Attack on Elliptic Curve Cryptosystems". English. In: *Public Key Cryptography — PKC 2003*. Ed. by YvoG. Desmedt. Vol. 2567. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2002, pp. 199–211. ISBN: 978-3-540-00324-3. DOI: 10.1007/3-540-36288-6_15. URL: http://dx.doi.org/10.1007/3-540-36288-6_15 (cit. on p. 46).

[Gou13] Conrado P. L. Gouvêa. *The Frobenius endomorphism with finite fields*. Sept. 2013. URL: http://alicebob.cryptoland.net/the-frobenius-endomorphism-with-finite-fields/ (cit. on p. 57).

[GPS08] Steven D. Galbraith, Kenneth G. Paterson, and Nigel P. Smart. "Pairings for cryptographers". In: *Discrete Appl. Math.* 156.16 (Sept. 2008), pp. 3113–3121. ISSN: 0166-218X. DOI: 10.1016/j.dam.2007.12.010. URL: http://dx.doi.org/10.1016/j.dam.2007.12.010 (cit. on p. 25).

[GR11] Santosh Ghosh and Dipanwita Roychowdhury. "Security of Prime Field Pairing Cryptoprocessor against Differential Power Attack". In: *Security Aspects in Information Technology*. Ed. by Marc Joye, Debdeep Mukhopadhyay, and Michael Tunstall. Vol. 7011. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2011, pp. 16–29. ISBN: 978-3-642-24585-5. DOI: 10.1007/978-3-642-24586-2_4. URL: http://dx.doi.org/10.1007/978-3-642-24586-2_4 (cit. on p. 2).

[Gre+13] Gurleen Grewal et al. “Efficient Implementation of Bilinear Pairings on ARM Processors”. In: *Selected Areas in Cryptography*. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2013. URL: http://dx.doi.org/10.1007/978-3-642-35999-6_11 (cit. on pp. 1, 97, 98).

[GS02] Craig Gentry and Alice Silverberg. “Hierarchical ID-Based Cryptography”. In: *Proceedings of the 8th International Conference on the Theory and Application of Cryptology and Information Security: Advances in Cryptology*. ASIACRYPT ’02. London, UK, UK: Springer-Verlag, 2002, pp. 548–566. ISBN: 3-540-00171-9. URL: http://dl.acm.org/citation.cfm?id=647098.717144 (cit. on pp. 8, 34).

[GS10] Robert Granger and Michael Scott. “Faster Squaring in the Cyclotomic Subgroup of Sixth Degree Extensions”. In: *Public Key Cryptography – PKC 2010*. Ed. by PhongQ. Nguyen and David Pointcheval. Vol. 6056. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2010, pp. 209–223. ISBN: 978-3-642-13012-0. DOI: 10.1007/978-3-642-13013-7_13. URL: http://dx.doi.org/10.1007/978-3-642-13013-7_13 (cit. on pp. 67, 68).

[HJS11] Michael Hutter, Marc Joye, and Yannick Sierra. “Memory-Constrained Implementations of Elliptic Curve Cryptography in Co-Z Coordinate Representation”. In: *Progress in Cryptology – AFRICACRYPT 2011*. Ed. by Abderrahmane Nitaj and David Pointcheval. Vol. 6737. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2011, pp. 170–187. ISBN: 978-3-642-21968-9. DOI: 10.1007/978-3-642-21969-6_11. URL: http://dx.doi.org/10.1007/978-3-642-21969-6_11 (cit. on pp. 59, 105, 106).

[HM09] Christoph Herbst and Marcel Medwed. “Using Templates to Attack Masked Montgomery Ladder Implementations of Modular Exponentiation”. In: *Information Security Applications*. Ed. by Kyo-Il Chung, Kiwook Sohn, and Moti Yung. Vol. 5379. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2009, pp. 1–13. ISBN: 978-3-642-00305-9. DOI: 10.1007/978-3-642-00306-6_1. URL: http://dx.doi.org/10.1007/978-3-642-00306-6_1 (cit. on pp. 45, 76, 84).

[HR03] Shai Halevi and Phillip Rogaway. “A Tweakable Enciphering Mode”. In: *Advances in Cryptology - CRYPTO 2003*. Ed. by Dan Boneh. Vol. 2729. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2003, pp. 482–499. ISBN: 978-3-540-40674-7. DOI: 10.1007/978-3-540-45146-4_28. URL: http://dx.doi.org/10.1007/978-3-540-45146-4_28 (cit. on p. 40).

[HR04] Shai Halevi and Phillip Rogaway. “A Parallelizable Enciphering Mode”. In: *Proc. RSA Conference 2004 – Cryptographer’s Track*. Springer-Verlag, 2004, pp. 292–304 (cit. on p. 40).

[HSV06] F. Hess, N.P. Smart, and F. Vercauteren. “The Eta Pairing Revisited”. In: *Information Theory, IEEE Transactions on* 52.10 (2006), pp. 4595–4602. ISSN: 0018-9448. DOI: 10.1109/TIT.2006.881709 (cit. on p. 26).

[Hut+09] Michael Hutter et al. “Attacking ECDSA-Enabled RFID Devices”. In: *Applied Cryptography and Network Security*. Ed. by Michel Abdalla et al. Vol. 5536. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2009, pp. 519–534. ISBN: 978-3-642-01956-2. DOI: 10.1007/978-3-642-01957

-9_32. URL: http://dx.doi.org/10.1007/978-3-642-01957-9_32 (cit. on p. 78).

[IEE08] IEEE. *P1363.3TM/D1 Draft Standard for Identity-based Public-key Cryptography Using Pairings.* 2008 (cit. on p. 35).

[IIT03] Kouichi Itoh, Tetsuya Izu, and Masahiko Takenaka. "Address-Bit Differential Power Analysis of Cryptographic Schemes OK-ECDH and OK-ECDSA". English. In: *Cryptographic Hardware and Embedded Systems - CHES 2002*. Ed. by BurtonS. Kaliski, çetinK. Koç, and Christof Paar. Vol. 2523. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2003, pp. 129–143. ISBN: 978-3-540-00409-7. DOI: 10.1007/3-540-36400-5_11. URL: http://dx.doi.org/10.1007/3-540-36400-5_11 (cit. on p. 46).

[Jou13] Antoine Joux. *A new index calculus algorithm with complexity $L(1/4 + o(1))$ in very small characteristic*. Cryptology ePrint Archive, Report 2013/095. http://eprint.iacr.org/. 2013 (cit. on p. 96).

[JY03] Marc Joye and Sung-Ming Yen. "The Montgomery Powering Ladder". English. In: *Cryptographic Hardware and Embedded Systems - CHES 2002*. Ed. by BurtonS. Kaliski, çetinK. Koç, and Christof Paar. Vol. 2523. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2003, pp. 291–302. ISBN: 978-3-540-00409-7. DOI: 10.1007/3-540-36400-5_22. URL: http://dx.doi.org/10.1007/3-540-36400-5_22 (cit. on p. 47).

[KAK96] C.K. Koç, Tolga Acar, and Jr. Kaliski B.S. "Analyzing and comparing Montgomery multiplication algorithms". In: *Micro, IEEE* 16.3 (1996), pp. 26–33. ISSN: 0272-1732. DOI: 10.1109/40.502403 (cit. on pp. 53, 63).

[Kam+09] David Kammler et al. "Designing an ASIP for Cryptographic Pairings over Barreto-Naehrig Curves". In: *Proceedings of the 11th International Workshop on Cryptographic Hardware and Embedded Systems*. CHES '09. Lausanne, Switzerland: Springer-Verlag, 2009, pp. 254–271. ISBN: 978-3-642-04137-2. DOI: 10.1007/978-3-642-04138-9_19. URL: http://dx.doi.org/10.1007/978-3-642-04138-9_19 (cit. on pp. 2, 98, 99).

[Kar10] Koray Karabina. *Squaring in cyclotomic subgroups*. Cryptology ePrint Archive, Report 2010/542. http://eprint.iacr.org/. 2010 (cit. on pp. 67, 68).

[KG06] Eike Kiltz and David Galindo. "Direct chosen-ciphertext secure identity-based key encapsulation without random oracles". In: *In ACISP 2006*. Springer-Verlag, 2006 (cit. on pp. 35, 37).

[Kil06] Eike Kiltz. "Chosen-Ciphertext Secure Identity-Based Encryption in the Standard Model with Short Ciphertexts". In: *In ACISP 2006, volume 4058 of LNCS*. Springer-Verlag, 2006, pp. 336–347 (cit. on pp. 35, 37, 39–42, 101).

[KM05] Neal Koblitz and Alfred Menezes. "Pairing-based Cryptography at High Security Levels". In: *Proceedings of Cryptography and Coding 2005, volume 3796 of LNCS*. Springer-Verlag, 2005, pp. 13–36 (cit. on p. 55).

[Knu97] Donald E. Knuth. *The art of computer programming, volume 2 (3rd ed.): seminumerical algorithms*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1997. ISBN: 0-201-89684-2 (cit. on p. 56).

[KO63] A. Karatsuba and Yu Ofman. *Multiplication of Many-Digital Numbers by Automatic Computers*. 1963 (cit. on p. 55).

[Koc96] Paul C. Kocher. "Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems". In: *Proceedings of the 16th Annual International Cryptology Conference on Advances in Cryptology*. CRYPTO '96. London, UK, UK: Springer-Verlag, 1996, pp. 104–113. ISBN: 3-540-61512-1. URL: http://dl.acm.org/citation.cfm?id=646761.706156 (cit. on p. 44).

[KV08] Eike Kiltz and Yevgeniy Vahlis. "CCA2 secure IBE: standard model efficiency through authenticated symmetric encryption". In: *Proceedings of the 2008 The Cryptopgraphers' Track at the RSA conference on Topics in cryptology*. CT-RSA'08. San Francisco, CA, USA: Springer-Verlag, 2008, pp. 221–238. ISBN: 3-540-79262-7, 978-3-540-79262-8. URL: http://dl.acm.org/citation.cfm?id=1791688.1791708 (cit. on p. 35).

[KW03] Jonathan Katz and Nan Wang. "Efficiency improvements for signature schemes with tight security reductions". In: *Proceedings of the 10th ACM conference on Computer and communications security*. CCS '03. Washington D.C., USA: ACM, 2003, pp. 155–164. ISBN: 1-58113-738-9. DOI: 10.1145/948109.948132. URL: http://doi.acm.org/10.1145/948109.948132 (cit. on p. 34).

[LN86] Rudolf Lidl and Harald Niederreiter. *Introduction to finite fields and their applications*. New York, NY, USA: Cambridge University Press, 1986. ISBN: 0-521-30706-6 (cit. on p. 11).

[Ltd13] ARM Ltd. *Cortex-M0+ Processor*. Sept. 2013. URL: http://www.arm.com/products/processors/cortex-m/cortex-m0plus.php (cit. on p. 50).

[Lyn07] Ben Lynn. "On the implementation of pairing-based cryptosystems". PhD thesis. Stanford University, 2007 (cit. on p. 20).

[Men05] Alfred Menezes. *An introduction to pairing-based cryptography. Notes from lectures given in*. 2005 (cit. on p. 27).

[Mil04] Victor S. Miller. "The Weil Pairing, and Its Efficient Calculation". English. In: *Journal of Cryptology* 17.4 (2004), pp. 235–261. ISSN: 0933-2790. DOI: 10.1007/s00145-004-0315-8. URL: http://dx.doi.org/10.1007/s00145-004-0315-8 (cit. on pp. 26, 29).

[MNT01] Atsuko Miyaji, Masaki Nakabayashi, and Shunzou Takano. *New explicit conditions of elliptic curve traces for FR-reduction*. 2001 (cit. on p. 27).

[MO09] Marcel Medwed and Elisabeth Oswald. "Template Attacks on ECDSA". In: *Information Security Applications*. Ed. by Kyo-Il Chung, Kiwook Sohn, and Moti Yung. Vol. 5379. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2009, pp. 14–27. ISBN: 978-3-642-00305-9. DOI: 10.1007/978-3-642-00306-6_2. URL: http://dx.doi.org/10.1007/978-3-642-00306-6_2 (cit. on p. 45).

[Mon85] Peter L. Montgomery. "Modular Multiplication without Trial Division". In: *Mathematics of Computation* 44.170 (1985), pp. 519–521 (cit. on p. 53).

[Mon87] P. L. Montgomery. *Speeding up the Pollard and elliptic curve methods of factorization*. 1987. DOI: 48(177):243264 (cit. on p. 47).

[MOP07] Stefan Mangard, Elisabeth Oswald, and Thomas Popp. *Power Analysis Attacks: Revealing the Secrets of Smart Cards*. Springer, 2007 (cit. on p. 43).

[Mor87] François Morain. "Building Cyclic Elliptic Curves Modulo Large Primes". In: *Advances in Cryptology - EUROCRYPT '91, Lecture Notes in Computer Science*. 1987, pp. 328–336 (cit. on p. 27).

[MOV93] A.J. Menezes, T. Okamoto, and S.A. Vanstone. "Reducing elliptic curve logarithms to logarithms in a finite field". In: *Information Theory, IEEE Transactions on* 39.5 (1993), pp. 1639–1646. ISSN: 0018-9448. DOI: 10.1109/18.259647 (cit. on p. 20).

[Nac05] David Naccache. "Secure and *Practical* Identity-Based Encryption." In: *IACR Cryptology ePrint Archive* 2005 (2005), p. 369. URL: http://dblp.uni-trier.de/db/journals/iacr/iacr2005.html#Naccache05 (cit. on pp. 35, 40, 41).

[NIS01] NIST. *Digital Signature Standard (DSS)*. 2001. URL: http://csrc.nist.gov/publications/fips/archive/fips186-2/fips186-2.pdf (cit. on p. 51).

[NIS12] NIST. *Secure Hash Standard*. Mar. 2012. URL: http://csrc.nist.gov/publications/fips/fips180-4/fips-180-4.pdf (cit. on p. 51).

[Oli+07] L.B. Oliveira et al. "TinyTate: Computing the Tate Pairing in Resource-Constrained Sensor Nodes". In: *Network Computing and Applications, 2007. NCA 2007. Sixth IEEE International Symposium on*. 2007, pp. 318–323. DOI: 10.1109/NCA.2007.48 (cit. on p. 98).

[Per+11] Geovandro C. C. F. Pereira et al. "A family of implementation-friendly BN elliptic curves". In: *J. Syst. Softw.* 84.8 (Aug. 2011), pp. 1319–1326. ISSN: 0164-1212. DOI: 10.1016/j.jss.2011.03.083. URL: http://dx.doi.org/10.1016/j.jss.2011.03.083 (cit. on pp. 52, 62).

[RCK03] Shai Halevi Ran Canetti and Jonathan Katz. *Chosen-Ciphertext Security from Identity-Based Encryption*. Cryptology ePrint Archive, Report 2003/182. http://eprint.iacr.org/. 2003 (cit. on p. 34).

[Sch85] René Schoof. "Elliptic curves over finite fields and the computation of square roots mod p". In: *Mathematics of computation* 44.170 (1985), pp. 483–494 (cit. on p. 19).

[Sco05] Michael Scott. "Computing the Tate Pairing". In: *Topics in Cryptology – CT-RSA 2005*. Ed. by Alfred Menezes. Vol. 3376. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2005, pp. 293–304. ISBN: 978-3-540-24399-1. DOI: 10.1007/978-3-540-30574-3_20. URL: http://dx.doi.org/10.1007/978-3-540-30574-3_20 (cit. on p. 61).

[Sco+09] Michael Scott et al. "On the Final Exponentiation for Calculating Pairings on Ordinary Elliptic Curves". In: *Pairing-Based Cryptography – Pairing 2009*. Ed. by Hovav Shacham and Brent Waters. Vol. 5671. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2009, pp. 78–88. ISBN: 978-3-642-03297-4. DOI: 10.1007/978-3-642-03298-1_6. URL: http://dx.doi.org/10.1007/978-3-642-03298-1_6 (cit. on p. 61).

[Sha84] Adi Shamir. "Identity-Based Cryptosystems and Signature Schemes". In: *Advances in Cryptology, Proceedings of CRYPTO '84, Santa Barbara, California, USA, August 19-22, 1984, Proceedings*. Vol. 196. Lecture Notes in Computer Science. Springer, 1984, pp. 47–53. DOI: 10.1007/3-540-39568-7_5 (cit. on pp. 1, 5, 31, 34).

[Sil09] J.H. Silverman. *The Arithmetic of Elliptic Curves*. Graduate texts in mathematics. Springer, 2009. ISBN: 9780387094946. URL: http://books.google.at/books?id=Z90CA_EUCCkC (cit. on pp. 17–19).

[SK03] Ryuichi Sakai and Masao Kasahara. *ID based Cryptosystems with Pairing on Elliptic Curve*. Cryptology ePrint Archive, Report 2003/054. http://eprint.iacr.org/. 2003 (cit. on p. 34).

[Sma01] N.P. Smart. "The Hessian Form of an Elliptic Curve". English. In: *Cryptographic Hardware and Embedded Systems — CHES 2001*. Ed. by ÇetinK. Koç, David Naccache, and Christof Paar. Vol. 2162. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2001, pp. 118–125. ISBN: 978-3-540-42521-2. DOI: 10.1007/3-540-44709-1_11. URL: http://dx.doi.org/10.1007/3-540-44709-1_11 (cit. on p. 17).

[SR13] AnaHelena Sánchez and Francisco Rodríguez-Henríquez. "NEON Implementation of an Attribute-Based Encryption Scheme". In: *Applied Cryptography and Network Security*. Ed. by Michael Jacobson et al. Vol. 7954. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2013, pp. 322–338. ISBN: 978-3-642-38979-5. DOI: 10.1007/978-3-642-38980-1_20. URL: http://dx.doi.org/10.1007/978-3-642-38980-1_20 (cit. on pp. 1, 66, 86, 87, 89–91, 93–98).

[Szc+08] Piotr Szczechowiak et al. "NanoECC: Testing the Limits of Elliptic Curve Cryptography in Sensor Networks". In: *Wireless Sensor Networks*. Ed. by Roberto Verdone. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2008. URL: http://dx.doi.org/10.1007/978-3-540-77690-1_19 (cit. on pp. 1, 98).

[Szc+09] Piotr Szczechowiak et al. "On the application of pairing based cryptography to wireless sensor networks". In: *Proceedings of the second ACM conference on Wireless network security*. WiSec '09. Zurich, Switzerland: ACM, 2009. URL: http://doi.acm.org/10.1145/1514274.1514276 (cit. on p. 98).

[UA13] Tohoku University and AIST. *Evaluation Environment for Side-Channel Attacks*. Sept. 2013. URL: http://www.risec.aist.go.jp/project/sasebo/ (cit. on p. 77).

[Unt13] Thomas Unterluggauer. "Xetroc-M0+. An implementation of ARMs Cortex-M0+". Master Project. Graz University of Technology, 2013 (cit. on pp. 2, 50, 62).

[Ver10] F. Vercauteren. "Optimal Pairings". In: *Information Theory, IEEE Transactions on* 56.1 (2010), pp. 455–461. ISSN: 0018-9448. DOI: 10.1109/TIT.2009.2034881 (cit. on pp. 27–29).

[Vol13] Inc. Voltage Security. *Identity-Based Encryption: Information Encryption for Email, Files, Documents, and Databases.* Aug. 2013. URL: http://www.voltage.com/technology/identity-based-encryption/ (cit. on p. 42).

[Wat05] Brent Waters. "Efficient identity-based encryption without random oracles". In: Springer-Verlag, 2005, pp. 114–127 (cit. on pp. 35, 38).

[Wat09] Brent Waters. "Dual System Encryption: Realizing Fully Secure IBE and HIBE under Simple Assumptions". In: *Proceedings of the 29th Annual International Cryptology Conference on Advances in Cryptology*. CRYPTO '09. Santa Barbara, CA: Springer-Verlag, 2009, pp. 619–636. ISBN: 978-3-642-03355-1. DOI: 10.1007/978-3-642-03356-8_36. URL: http://dx.doi.org/10.1007/978-3-642-03356-8_36 (cit. on p. 35).

[WH12] Johannes Winter and Daniel Hein. *Cortex-M0 Simulator*. 2012 (cit. on p. 62).

[WS06] Claire Whelan and Mike Scott. "Side Channel Analysis of Practical Pairing Implementations: Which Path Is More Secure?" In: *Progress in Cryptology - VIETCRYPT 2006*. Ed. by PhongQ. Nguyen. Vol. 4341. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2006, pp. 99–114. ISBN: 978-3-540-68799-3. DOI: 10.1007/11958239_7. URL: http://dx.doi.org/10.1007/11958239_7 (cit. on p. 2).

[WUW13] Erich Wenger, Thomas Unterluggauer, and Mario Werner. "8/16/32 Shades of Elliptic Curve Cryptography on Embedded Processors". In: *Progress in Cryptology - INDOCRYPT 2013*. 2013 (cit. on p. 63).

[YJ00] Sung-Ming Yen and M. Joye. "Checking before output may not be enough against fault-based cryptanalysis". In: *Computers, IEEE Transactions on* 49.9 (2000), pp. 967–970. ISSN: 0018-9340. DOI: 10.1109/12.869328 (cit. on p. 47).