UNIVERSITÄT PADERBORN
Die Universität der Informationsgesellschaft

Fakultät für Elektrotechnik, Informatik und Mathematik
Arbeitsgruppe Codes und Kryptographie

Evaluation of Pairing Optimization for Embedded Platforms

Master's Thesis

in Partial Fulfillment of the Requirements for the Degree of

Master of Science

by
Martin Sosniak
Dahlienstr. 1
59063 Hamm

submitted to
Prof. Dr. Johannes Blömer
and
Jun.-Prof. Dr. Christian Plessl

Paderborn, 23. March 2015

Declaration
(Translation from German)

I hereby declare that I prepared this thesis entirely on my own and have not used outside sources without declaration in the text. Any concepts or quotations applicable to these sources are clearly attributed to them. This thesis has not been submitted in the same or substantially similar version, not even in part, to any other authority for grading and has not been published elsewhere.

Original Declaration Text in German:

Erklärung

Ich versichere, dass ich die Arbeit ohne fremde Hilfe und ohne Benutzung anderer als der angegebenen Quellen angefertigt habe und dass die Arbeit in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegen hat und von dieser als Teil einer Prüfungsleistung angenommen worden ist. Alle Ausführungen, die wörtlich oder sinngemäß übernommen worden sind, sind als solche gekennzeichnet.

City, Date	Signature

Contents

List of Figures

List of Algorithms

1 Introduction

A bilinear pairing takes two arguments and maps them to one value. More formal, let $\mathbb{G}_1$, $\mathbb{G}_2$ be additive groups of points of an elliptic curve and $\mathbb{G}_T$ be a subgroup of a multiplicative group of a finite extension field. Additionally, let $\mathbb{G}_1$, $\mathbb{G}_T$ be groups of prime order r and let $\mathbb{G}_2$ be a group where each element has order dividing r. A bilinear pairing e is an efficiently computable function

$$e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$$

such that:

1. Non-degeneracy:

$$e(P, Q) = 1_{\mathbb{G}_T} \text{ for all } Q \in \mathbb{G}_2 \text{ if and only if } P = \mathcal{O}$$

   and similarly

$$e(P, Q) = 1_{\mathbb{G}_T} \text{ for all } P \in \mathbb{G}_1 \text{ if and only if } Q = \mathcal{O}.$$

2. Bilinearity: for all $P_1, P_2 \in \mathbb{G}_1, Q_1, Q_2 \in \mathbb{G}_2$:

$$e(P_1 + P_2, Q_1) = e(P_1, Q_1)e(P_2, Q_1)$$

$$e(P_1, Q_1 + Q_2) = e(P_1, Q_1)e(P_1, Q_2)$$

From bilinearity we can follow:

$$e(aU, bV) = e(U, V)^{ab} = e(bU, aV) \ \forall U \in \mathbb{G}_1, V \in \mathbb{G}_2, a, b \in \mathbb{Z}.$$

In the recent years bilinear pairings were used to design new cryptographic protocols, such as Identity-Based encryption [9], Attribute-Based encryption [14], short group signatures [8] and communication-efficient multi-party key agreement [17]. The core property of bilinear pairings that allows the construction of the above stated schemes is their linearity in both arguments, the bilinearity.

Concrete realizations of bilinear pairings are, for example, the Weil-pairing [22], the (reduced) Tate-pairing [11], [10] and the optimal Ate-pairing [6], [10], [24]. All these pairings are computed with the Miller algorithm [22] or one of its variants.

As with any cryptographic protocol, to be applicable in practice it is required to be computable in few tenths of a second. Unfortunately it turned out that the computation of bilinear pairings consumes a couple million of processor cycles

and is therefore very time intensive. As a result of that, the computation of pairings build a major bottleneck of the above stated protocols. Therefore it is interesting to find ways to optimize the run time of these computations. Indeed there is a lot of ongoing research in optimizing the algorithms respectively formulas and exploiting the hardware/processor properties to decrease the number of cycles respectively the run time. Hence, researchers managed to compute bilinear pairings in few tenths of a second on desktop class CPUs [2], [1], [6], [24], [10]. Computing pairings on resource constraint systems such as micro controllers, e.g. CPUs with word size at most 32-bit, very small or even no cache, no floating point unit, few registers, is still very time intensive.

Our work

In this master's thesis we present detailed complexity analyses of the arithmetic needed to implement bilinear pairings. On the base of the analyses we implement and evaluate the reduced Tate pairing for one specific case, i.e. embedding degree $k = 12$ and a 254-bit prime. All algorithms that are implemented on top of the base field arithmetic are presented in this thesis. For the implementation we choose among all algorithms presented in this thesis only the most efficient.

We consider the base field arithmetic as a given building block and focus on the building blocks extension field arithmetic (Section 3.1), elliptic curve arithmetic and line evaluation arithmetic (Section 3.2) and finally on the arithmetic of the reduced Tate-pairing. For every of those building blocks we present different optimizations or implementation strategies. For example, we compare Karatsuba multiplication and Schoolbook multiplication in extension fields, provide algorithms for elliptic curve operations merged with the evaluation of line equations and we consider twists on elliptic curves that reduces the complexity of bilinear pairings.

All analyses are done on two different abstractions. At first, every analysis is done for a general case. In a second step we use the general analyses to compute the complexity for a specific case, e.g. for an extension field with specific extension degree, an elliptic curve with explicit parameters, or a concrete bilinear pairing for a given security level.

This approach has two advantages. The analyses are applicable for many different cases, especially for those we do not consider in this thesis. For example, our analyses for the multiplication in extension fields are applicable for any extension degree that can be represented by $2^i 3^j$. The complexity for point addition and line evaluation is done for any curve that is specified by $E : y^2 = x^3 + ax + b$. For point doubling and tangent evaluation we restricted to the case where $a = -3$ or $a = 0$, which fit to many elliptic curve families. The complexity analysis of the reduced Tate-pairing is applicable to any embedding degree. The second advantage relies on the fact, that we consider the hardware platform and the base field arithmetic as black boxes. All presented algorithms make no assumptions on the hardware and they do not rely on a special base field

arithmetic. This means, one can use our analyses to estimate run times on its hardware platform under consideration. All that one needs, are the timings for multiplication, addition and inversion of his particular base field implementation - independent of whether the base field arithmetic is implemented in software or hardware.

In Chapter 4 we validate and justify the practical correctness of our analyses. For that we compare the theoretical estimates with the practical measurements. To be able to justify the practical correctness of our analyses we implemented on top of ESCRYPT's[1] library for base field arithmetic an extension field arithmetic, elliptic curve and line evaluation arithmetic and the reduced Tate pairing for embedding degree $k = 12$ and prime $|p| = 254$. We use this implementation to measure the timings of

- the base field implementation of ESCRYPT, for fields over primes of size 160-bit and 254-bit (Section 4.2.1),
- the extension field arithmetic for embedding degree $k = 12$ and prime $|p| = 254$ (Section 4.2.2),
- the reduced Tate pairing for embedding degree $k = 12$ and prime p with $|p| = 254$ (Section 4.2.3),

on the ARM Cortex M3 processor. To justify the correctness of our theoretical estimates we feed our analyses with the base field timings and compare the estimated timings with the practical measurements. Our theoretical estimates differ only by at most -4% which shows that our theoretical analyses are applicable for calculating estimates. Thus, we use our theoretical analyses to estimate the run time of the reduced Tate pairing for embedding degree $k = 6$ and prime p with $|p| = 160$.

Furthermore, we show that computing pairings on affine coordinates on the ARM Cortex M3 with ESCRYPT's base field library is not as efficient as computing pairings on Jacobian coordinates. We show that computation on affine coordinates are as efficient as on Jacobian coordinates if one inversion in the base field under consideration costs at most as 11.5 multiplications in the same base field. In ESCRYPT's library a base field inversion is as efficient as 20 respectively 17 multiplications for primes of size 254-bit and 160-bit, see Section 4.2.1.

We have that a computation a Miller operation costs $7.3666s$ for $k = 12$ and a prime of size 254-bit, whereas computing the Miller operation for the case $k = 6$ and a prime of size 160-bit costs $1.462s$, see Section 4.2.3. These timings are completely not applicable for practical purposes. In Section 4.3 we discuss what are the problems with our implementation and what can be done to improve the runtime. In Section 4.3.1 we show where our implementation lacks efficiency and how big are the run time improvements by resolving these efficiency lacks. Section 4.3.2 discusses the structure of the costs. There we give hints which parts

[1] This thesis was done in cooperation with company ESCRYPT GmbH - Embedded Security, Lise-Meitner-Allee 4, 44801 Bochum, Germany

of the implementation promise the biggest improvement impact if they will be optimized. Section 4.3.2 also gives a very raw lower bound for the computation of the Miller loop on the ARM Cortex M3 processor.

2 Preliminaries

This chapter introduces all necessary objects needed to define and compute pairings. We begin with fixing notation in Section 2.1. In Section 2.2 we introduce finite fields and finite extension fields. Section 2.3 defines elliptic curves and shows that the points of an elliptic curve form an abelian group. The chapter ends with Section 2.4 where we introduce bilinear pairings and define the Tate and the reduced Tate pairing.

2.1 Notation

Here we introduce notation that we use through the entire thesis.

Sets The set of natural numbers $\{0, 1, 2, 3, \ldots\}$ is denoted by $\mathbb{N}_0$. The set of natural numbers without 0 is defined as $\mathbb{N} = \mathbb{N}_0 \setminus \{0\}$. The set of integers $\{\ldots, -3, -2, -1, 0, 1, 2, 3, \ldots\}$ is denoted by $\mathbb{Z}$.

Numbers We denote by p a prime number and by q a prime power p^n for $n \in \mathbb{N}$.

Length Let $x \in \mathbb{N}$ then

$$|x| := \lfloor \log_2(x) \rfloor + 1,$$

is the binary description length of the number x.

Hamming weight We define the Hamming weight as the number of $1'$s in the binary representation of a number $r \in \mathbb{N}$. We denote the Hamming weight of $r \in \mathbb{N}$ as $h(r)$.

2.2 Finite Fields and Field Extensions

In this thesis we will analyze the complexity of extension field arithmetic. Therefore we give a short introduction to these fields. The following statements are taken from [19] Chapter 1 and 2. We begin with the notion of finite fields.

Theorem 2.1. *Let $\mathbb{Z}_n := \mathbb{Z}/n\mathbb{Z}$ with $n \in \mathbb{N}$ and let p be a prime number. Then $\mathbb{F}_p = \mathbb{Z}_p$ is a finite field with p elements.*

Note that p is called the characteristic of the finite field $\mathbb{F}_p$.

For the notion of finite extension fields we need to consider polynomials with coefficients of a specific domain.

Definition 2.2. *Let F be a finite field. Then $F[x]$ denotes the set of all polynomials with coefficients in F.*

The following theorem states the existence of a so called finite extension field. The elements of such fields are polynomials.

Theorem 2.3. *[19]Let F be a finite field. For $f \in F[x]$, the residue class ring $F[x]/(f)$ is a field if and only if f is irreducible over F.*

For this we give a short example. Let f be an irreducible $n-$degree polynomial over $\mathbb{F}_p$. Then $\mathbb{F}_p[x]/(f) = \{\sum_{i=0}^{n-1} a_i x^i | a_i \in \mathbb{F}_p\}$. We call such fields extension fields of degree n and denote them by $\mathbb{F}_{p^n}$.

The next theorem states that we can find irreducible polynomials of arbitrary degrees. With this we are able to define finite extension fields of arbitrary degrees.

Theorem 2.4. *Let F be a finite field, then for every $n \in \mathbb{N}$ there exists an irreducible $n-$degree polynomial in $F[x]$.*

The following theorem specifies the number of elements of a finite extension field with a certain degree and characteristic.

Theorem 2.5. *Let F be a finite field with p elements and $f \in F[x]$ an irreducible polynomial of degree $n \in \mathbb{N}$. Then $\mathbb{F} = F[x]/(f)$ is a finite field with p^n elements.*

Theorem 2.6. *If two finite fields F and K have the same number of elements then F and K are isomorphic.*

The following last theorem gives us an important inside into the structure of finite extension fields. This statement allows us to build extension fields by so called *field towers*.

Theorem 2.7 (Subfield Criterion)**.** *[19] Let $\mathbb{F}_q$ be the finite field with $q = p^n$ elements, $n \in \mathbb{N}$. Then every subfield of $\mathbb{F}_q$ has order p^m, where m is a positive divisor of n. Conversely, if m is a positive divisor of n, then there is exactly one subfield of $\mathbb{F}_q$ with p^m elements.*

For the definition of bilinear pairings in Section 2.4 we will need the notion of an *algebraic closure*. For that we give the following three definitions.

Definition 2.8. *[19] Let K be a finite field, let $f \in K[x]$ be of positive degree and F an extension field of K. Then f is said to split in F if f can be written as a product of linear factors in $F[x]$. That is, if there exists elements $\alpha_1, \alpha_2, \ldots, \alpha_n \in F$ such that*

$$f(x) = a(x - \alpha_1)(x - \alpha_2) \cdots (x - \alpha_n),$$

where a is the leading coefficient of f.

Thus, f *splits in* F if all roots of f belongs to the extension field F.

Definition 2.9. *A field F is said to be algebraically closed if every polynomial with coefficients in F has a root in F.*

Definition 2.10. *The field $\overline{F}$ is called an algebraic closure of F if $\overline{F}$ is algebraic over F and if every polynomial $f(x) \in F[x]$ splits completely over $\overline{F}$, so that $\overline{F}$ can be said to contain all the elements that are algebraic over F.*

In the following we consider the arithmetic of finite extension fields. The arithmetic operations addition and multiplication in $\mathbb{F}_{q^k}$ are done as known from ordinary polynomial arithmetic. Except at the end, the result of an operation is reduced modulo an irreducible polynomial. This means, one does polynomial division and considers the remainder as the final result. For that we consider a simple example [26]:

Example 2.11. We consider the field $\mathbb{F}_{2^4}$ and the irreducible polynomial $P(x) = x^4+x+1$. Let $C'(x) = x^5+x^3+x^2+x$ be a product of two polynomials. Consider only those monomials of $C'(x)$ with degree greater equal to the degree of the polynomial $P(x)$. According to that we consider here just the monomial x^5. Reduce this monomial as follows:

$$\begin{aligned} x^4 &= 1 \cdot P(x) + (x+1) \\ x^4 &\equiv x+1 \mod P(x) \\ x^5 &\equiv x^2 + x \mod P(x). \end{aligned}$$

We have now the reduced expression for $x^5$and can insert it into $C'(x)$ to compute the final result:

$$\begin{aligned} C(x) &\equiv C'(x) \mod P(x) \\ C(x) &\equiv (x^2+x) + (x^3+x^2+x) = x^3. \end{aligned}$$

Specific extension fields In this thesis we will consider extension fields of degree 6 and 12. Due to Theorem 2.3 and Theorem 2.4 we can define such fields as $\mathbb{F}_p[x]/(f)$ and $\mathbb{F}_p[x]/(g)$ where f and g are irreducible polynomials over $\mathbb{F}_p$ of degree 6 respectively 12. For our purposes we will use a different approach. We will define the desired extension fields by field towers. Field towers are fields that are defined over more than one extension. For that we consider a finite field $\mathbb{F}_p$ and a factorization of the desired extension degree $k = k_0 \cdot k_1 \cdot k_2 \cdot \ldots \cdot k_m$, with $k_0 = 1$. Now, for every $i \in \{1, \ldots, m\}$ one defines an extension field as follows: $\mathbb{F}_{p^{k_1 \cdot \ldots \cdot k_i}} = \mathbb{F}_{p^{k_0 \cdot \ldots \cdot k_{i-1}}}[x]/(f_i)$ where f_i is an irreducible polynomial in $\mathbb{F}_{p^{k_0 \cdot \ldots \cdot k_{i-1}}}[x]$ of degree k_i. We say that we build the extension field of degree k by the tower $\mathbb{F}_p \to \mathbb{F}_{p^{k_1}} \to \mathbb{F}_{p^{k_1 \cdot k_2}} \to \ldots \to \mathbb{F}_{p^{k_1 \cdot k_2 \cdot \ldots \cdot k_m}}$.

Since we can factor 6 by $2 \cdot 3$ and 12 by $2 \cdot 3 \cdot 2$ we can build the desired extension fields as follows: We start with degree $k = 6$ and extend the resulting

field tower again to get an extension of degree $k = 12$. For the following tower we need to have $p \equiv 1 \mod 4$ to ensure that -1 is a quadratic non-residue. According to [2], we build the first extension field by the tower $\mathbb{F}_p \to \mathbb{F}_{p^2} \to \mathbb{F}_{p^6}$:

$$\mathbb{F}_{p^2} := \mathbb{F}_p[x]/(P(x)), \text{ where } P(x) = x^2 - u \text{ with } u = -1 \text{ and } T := x \mod P(x). \tag{2.1}$$

$$\mathbb{F}_{p^6} := \mathbb{F}_{p^2}[y]/(Q(y)), \text{ where } Q(y) = y^3 - v \text{ with } v = T + 1 \text{ and } S := y \mod Q(y). \tag{2.2}$$

This is, an element from $\mathbb{F}_{p^2}$ can be represented with a polynomial of degree 1 with coefficients from $\mathbb{F}_p$ and an element from $\mathbb{F}_{p^6}$ can be represented with a polynomial of degree 2 with coefficients from $\mathbb{F}_{p^2}$.

More explicit, let $A \in \mathbb{F}_{p^2}$ then

$$A = a_0 + a_1 T,$$

where $a_0, a_1 \in \mathbb{F}_p$.

And, let $B \in \mathbb{F}_{p^6}$ then

$$\begin{aligned} B =& b_0 + b_1 S + b_2 S^2 \\ =& (b_{00} + b_{01}T) + (b_{10} + b_{11}T)S + (b_{20} + b_{21}T)S^2, \end{aligned}$$

where $b_i \in \mathbb{F}_{p^2}$ and $b_{i,j} \in \mathbb{F}_p$. We have also the following representation of an $\mathbb{F}_{p^6}$ element:

$$B = b_{00} + b_{10}S + b_{20}S^2 + (b_{01} + b_{11}S + b_{21}S^2)T.$$

This is, by changing the order of the coefficients we can also interpret B as a quadratic extension of a cubic extension. We will use this representation in Chapter 3 in the context of elliptic curve twists.

To come up with an extension field $\mathbb{F}_{p^{12}}$ we apply the tower $\mathbb{F}_p \to \mathbb{F}_{p^2} \to \mathbb{F}_{p^6} \to \mathbb{F}_{p^{12}}$. This is, we extend the above tower and we get [2]:

$$\mathbb{F}_{p^{12}} := \mathbb{F}_{p^6}[z]/(R(z)), \text{ where } R(z) = z^2 - S \text{ and } U := z \mod R(z) \tag{2.3}$$

Hence, an element $C \in \mathbb{F}_{p^{12}}$ can be represented as polynomials of degree 1 with coefficients in $\mathbb{F}_{p^6}$ and we can also represent it by a polynomial of degree 6 with coefficients in $\mathbb{F}_{p^2}$:

$$\begin{aligned} C =& c_0 + c_1 U, \text{ where } c_0, c_1 \in \mathbb{F}_{p^6} \\ =& (c_{0,0} + c_{0,1}S + c_{0,2}S^2) + (c_{1,0} + c_{1,1}S + c_{1,2}S^2)U, \text{ where } c_{i,j} \in \mathbb{F}_{p^2} \\ =& c_{0,0} + c_{1,0}U + c_{0,1}U^2 + c_{1,1}U^3 + c_{0,2}U^4 + c_{1,2}U^5. \end{aligned}$$

We will also use the latter representation of C in the context of elliptic curve twists in Chapter 3.

Note that the irreducible polynomials P, Q and R are binomials. The analyses in Chapter 3 assume that the irreducible polynomials of the extension fields are binomials and the analyses are not applicable for tower fields whose irreducible polynomials are not binomials.

2.3 Elliptic Curves

Definition 2.12 ([15, Ch. 3])**.** *An elliptic curve E over a field K is defined by an equation*

$$E : y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6 \tag{2.4}$$

where $a_1, a_2, a_3, a_4, a_5, a_6 \in K$ and $\Delta \neq 0$, where Δ is the discriminant of E and is defined as follows:

$$\begin{aligned}
\Delta &= -d_2^2d_8 - 8d_4^3 - 27d_6^2 + 9d_2d_4d_6 \\
d_2 &= a_1^2 + 4a_2 \\
d_4 &= 2a_4 + a_1a_3 \\
d_6 &= a_3^2 + 4a_6 \\
d_8 &= a_1^2a_6 + 4a_2a_6 - a_1a_3a_4 + a_2a_3^2 - a_4^2.
\end{aligned}$$

If L is an extension field of K, then the set of L-rational points on E is

$$E(L) = \{(x, y) \in L \times L : y^2 + a_1xy + a_3y - x - a_2x^2 - a_4x - a_6 = 0\} \cup \infty$$

where ∞ is the point at infinity. The set $E(L)$ contains the points that satisfy E. Note that if E is defined over K, then E is also defined over any extension field of K.

(2.4) *is called Weierstrass equation. A Weierstrass equation defined over a field K with characteristic different from 2 and 3 can be simplified considerably by applying admissible changes of variables [15, Ch. 3]. We will not go deeper into this, but state that E simplifies over fields with characteristic different from 2 and 3 to*

$$E : y^2 = x^3 + ax + b, \tag{2.5}$$

where the discriminant is $\Delta = -16(4a^3 + 27b^2)$. We call (2.5) the *short Weierstrass form* and we will use this form throughout the remainder of this thesis. More details on this can be found in [15, Section 3.1.1].

For our proposes, we distinguish two kinds of curves. We say a curve is *smooth* if its discriminant $\Delta \neq 0$. This means the curve has no point which has two or

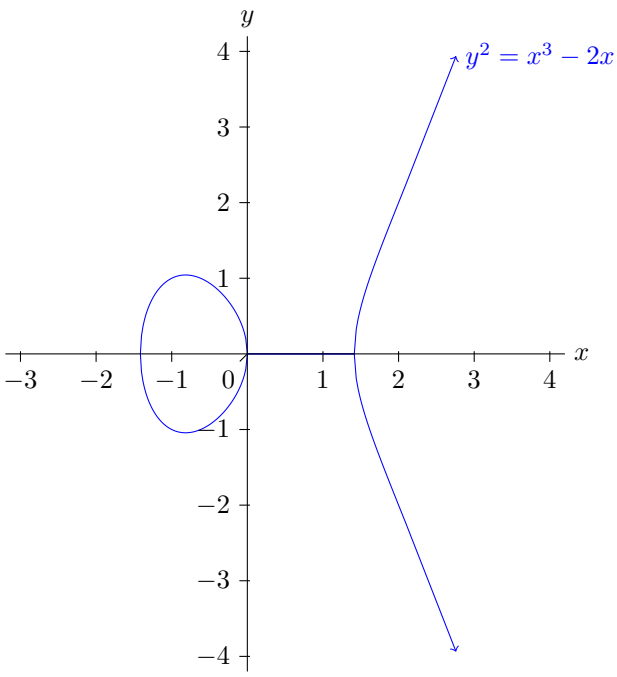

Figure 2.1: E_1 over $\mathbb{R}$

more distinct tangents lines. Otherwise, we say a curve is *singular*. Therefore singular curves have the property that they intersect itself at one point.

To get a better concept of elliptic curves consider

$$E_1 : y^2 = x^3 - 2x,$$
$$E_2 : y^2 = x^3 + \frac{1}{4}x + \frac{5}{4}$$
$$\text{and}$$
$$E_3 : y^2 = x^3 - 3x + 2$$

plotted in Figure 2.1, Figure 2.2 and Figure 2.3 over the reals. In the plot one can see that E_1 and E_2 are smooth curves whereas E_3 is a singular curve, since it has two tangents at the point $(1, 0)$. Note that plots of elliptic curves over finite fields of cryptographic interest do not look like the plots in Figure 2.1, Figure 2.2 and Figure 2.3. Plotting an elliptic curve that is defined over a finite field results in scattered points.

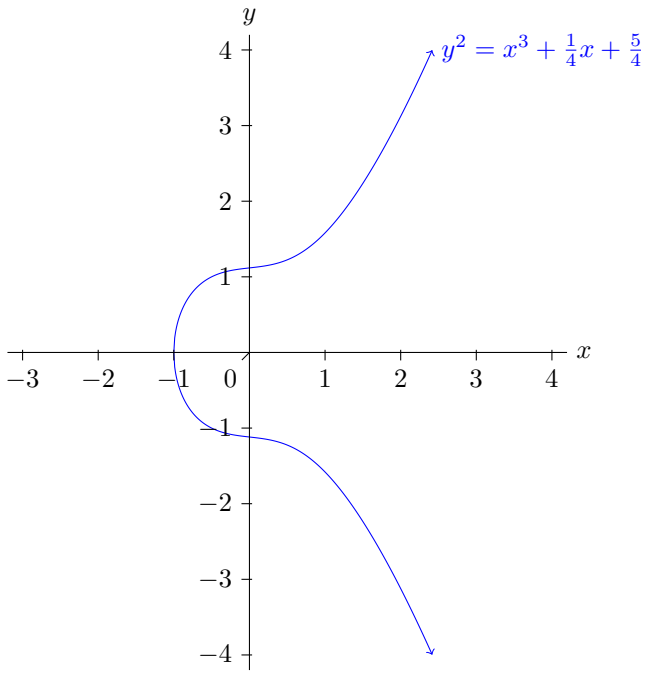

Figure 2.2: E_2 over $\mathbb{R}$

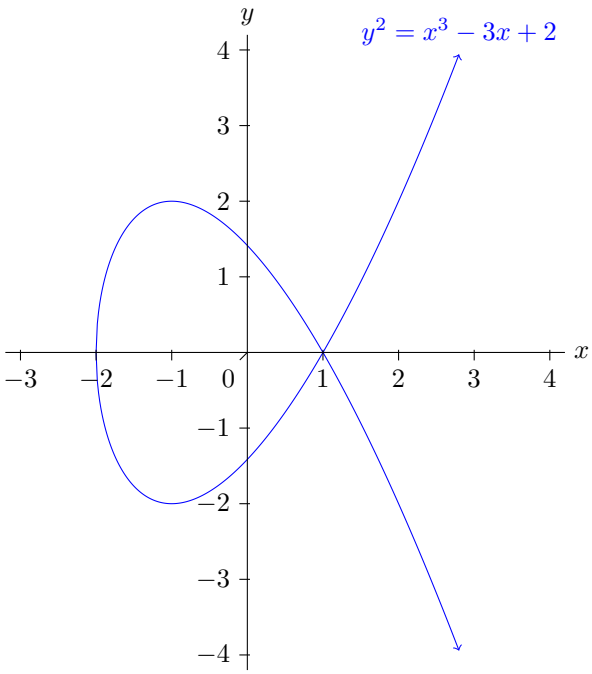

Figure 2.3: E_2 over $\mathbb{R}$

2.3.1 Group law

Let E be an elliptic curve defined over the field K. Then the set $E(K)$ forms an abelian group. Due to the chord-and-tangent rule we can define an adding operation for two points of $E(K)$ such that their result is again an element of $E(K)$. The identity element is ∞. We explain the addition rule geometrically, later we will introduce explicit formulas. Let $P = (x_1, y_1)$ and $Q = (x_2, y_2)$ be two distinct points of the elliptic curve E. Then the sum of P and Q is defined as follows: Draw a line through the points P and Q; since the curve is cubic, this line intersects the curve at another point. Reflect this point along the x-axis and obtain the point R, which is defined as $P + Q$.

In the case where $P = Q$ we compute the sum $P + P = [2]P$. This operation is also called the *double* of P and is geometrically defined as follows: Draw a tangent line to the elliptic curve E at point P. This line intersects the curve E at another point. Reflect this point along the x-axis to obtain the point R, which is defined as $P + P$.

We define $[n]P = P + \ldots + P$ as the n-fold addition of the point P.

Group law for $E : y^2 = x^3 + ax + b$ defined over K with characteristic $\neq 2, 3$ [15, Ch. 3]

1. *Identity.* $P + \infty = \infty + P = P$ for all $P \in E(K)$.

2. *Negatives.* If $P = (x, y) \in E(K)$, then $(x, y) + (x, -y) = \infty$. The point $(x, -y)$ is denoted by $-P$ and is called the negative of P. Also, $-\infty = \infty$.

3. *Point Addition.* Let $P = (x_1, y_1) \in E(K)$ and $Q = (x_2, y_2) \in E(K)$, where $P \neq \pm Q$. Then $P + Q = (x_3, y_3)$,where

$$x_3 = \left(\frac{y_2 - y_1}{x_2 - x_1}\right)^2 - x_1 - x_2 \text{ and } y_3 = \left(\frac{y_2 - y_1}{x_2 - x_1}\right)(x_1 - x_3) - y_1. \quad (2.6)$$

4. *Point doubling.* Let $P = (x_1, y_1) \in E(K)$, where $y_1 \neq 0$. Then $[2]P = (x_3, y_3)$, where

$$x_3 = \left(\frac{3x_1^2 + a}{2y_1}\right)^2 - 2x_1 \text{ and } y_3 = \left(\frac{3x_1^2 + a}{2y_1}\right)(x_1 - x_3) - y_1. \quad (2.7)$$

2.3.2 Properties of Elliptic Curve Groups

Let E be an elliptic curve over $\mathbb{F}_q$. The number of points in $E(\mathbb{F}_q)$ is called the *order* of E over $\mathbb{F}_q$, denoted $\#E(\mathbb{F}_q)$. We know that $1 \leq \#E(\mathbb{F}_q) \leq 2q + 1$, since the Weierstrass equation has at most two solutions for each $x \in \mathbb{F}_q$. The Hasse-Weil Theorem provides tighter bounds.

Theorem 2.13 (Hasse-Weil). *[18, Ch. 2] Let E be an elliptic curve defined over $\mathbb{F}_q$, then the order of the group $E(\mathbb{F}_{q^k})$ with $k \in \mathbb{N}_0$ satisfies*

$$\#E(\mathbb{F}_{q^k}) = q^k + 1 - t_k, \text{with } |t_k| \leq 2\sqrt{q^k},$$

where t_k is called the trace of E over $\mathbb{F}_q$.

Definition 2.14 (Torsion Points). *Let K be a finite field, let E be an elliptic curve defined over K and $r \in \mathbb{Z}$. We call every point $P \in E(K)$ with $[r]P = \mathcal{O}$ an r-torsion point. The set of all r-torsion points in $E(K)$ is denoted by $E(K)[r]$.*

We also say that the points in $E(K)[r]$ have order r.

Definition 2.15 (Embedding Degree). *[18, Ch. 2] Let E be an elliptic curve defined over $\mathbb{F}_q$ and $r|\#E(\mathbb{F}_q)$ with $gcd(r, q) = 1$, then the embedding degree is the smallest positive integer k such that $r|(q^k - 1)$.*

This means, k is the smallest integer such that the field $\mathbb{F}_{q^k}$ contains the r-th roots of unity [20].

Definition 2.16 (Super-singular Curves). *[15, Ch. 3] Let p be the characteristic of $\mathbb{F}_q$. An elliptic curve E defined over $\mathbb{F}_q$ is super-singular if p divides t, where t is the trace. If p does not divide t, then E is non-super-singular.*

We call non-super-singular curves also ordinary curves.

2.3.3 Restrictions

We will only consider non-super-singular respectively ordinary curves, since we will deal with characteristic $\neq 2, 3$ fields and embedding degrees[1] ≥ 6, [20, Section 4.6].

2.4 Bilinear Pairings

This section gives an abstract definition of bilinear pairings. After that we introduce *divisors* which we need to define concrete pairings. We finish this section with defining concrete pairings such as the *Tate*, the *reduced Tate* and the *Ate* pairing.

Definition 2.17 (Bilinear Pairing). *[7, Ch. 9] Let n be a positive integer. Let $\mathbb{G}_1$ and $\mathbb{G}_2$ be abelian groups written in additive notation with identity element $\mathcal{O}$. Suppose that $\mathbb{G}_1$ and $\mathbb{G}_2$ have exponent n (i.e. $[n]P = \mathcal{O}$ for all $P \in \mathbb{G}_1, \mathbb{G}_2$). Suppose $\mathbb{G}_3$ is a cyclic group of order n written in multiplicative notation with identity element 1. A pairing is a function*

$$e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_3.$$

[1] see Definition Definition 2.15

All pairings we consider will satisfy the following additional properties:
Bilinearity: *For all $P, P' \in \mathbb{G}_1$ and all $Q, Q' \in \mathbb{G}_2$ we have*

$$e(P + P', Q) = e(P, Q)e(P', Q)$$
$$\textit{and}$$
$$e(P, Q + Q') = e(P, Q)e(P, Q').$$

Non-degeneracy:

- *For all $P \in \mathbb{G}_1$, with $P \neq 0$, there is some $Q \in \mathbb{G}_2$ such that $e(P, Q) \neq 1$.*
- *For all $Q \in \mathbb{G}_2$, with $Q \neq 0$, there is some $P \in \mathbb{G}_2$ such that $e(P, Q) \neq 1$.*

2.4.1 Divisors

For computing pairings one needs to evaluate rational functions. We will specify these functions up to a constant multiple. A standard method to specify functions is to define their zeros and poles. Therefore we define *divisors.* With divisors we have a notion that encapsulates points and their multiplicities in one formal sum. For working with divisors we need the following two definitions.

Definition 2.18. *Let K be a field. Then the set of polynomials on the elliptic curve $E : y^2 = x^3 + ax + b$ is defined as*

$$K[E] = K[x, y]/(y^2 - x^3 - ax - b).$$

Definition 2.19. *Let K be a field and $E : y^2 = x^3 + ax + b$ an elliptic curve. Then the function field of E over K is defined as*

$$K(E) = \left\{ \frac{f(x, y)}{g(x, y)} | f, g \in K(E), g \neq 0 \right\}.$$

Definition 2.20 (Divisor)**.** *[18, Ch. 2] Let E be an elliptic curve over $\mathbb{F}_q$, then a divisor D is a formal sum of points over the algebraic closure of $\mathbb{F}_q$*

$$D = \sum_{P \in E} c_P(P)$$

with only finitely many non-zero coefficients $c_p \in \mathbb{Z}$.

Definition 2.21 (Degree of Divisor)**.** *[18, Ch. 2] The degree of a divisor D is*

$$deg(D) = \sum_{P \in E} c_P.$$

The support of a divisor D is the set of points with non-zero coefficients in the sum, denoted $supp(D)$. Furthermore we define $ord_P(D) = c_P$.

Definition 2.22 (Vanishing order)**.** *[18, Ch. 2] The order of vanishing of a non-constant function $f \in \overline{\mathbb{F}}_q(E)^*$ at the point P is*

$$ord_P(f) = c_P.$$

Now to define a function f up to a constant multiple we can provide a formal sum. Every zero and every pole of f appears in the sum with its non-zero multiplicity. The zeros and the poles are the points of the sum and the multiplicities are the points' coefficients. All other points have coefficient 0. The multiplicity of a zero is always > 0, whereas the multiplicity of a pole is always < 0. Hence, for each non-constant rational function $f \in \overline{\mathbb{F}}_q(E)^*$ we can associate the divisor $div(f) = \sum_{P \in E} ord_P(f)(P)$.

Note that, for two functions f, g we have that $div(f \cdot g) = div(f) + div(g)$ and $div(f/g) = div(f) - div(g)$.

Next, we introduce the notion of a *Miller function* that plays a central role in the definition and computation of all pairings.

Definition 2.23 (Miller function)**.** *[18, Ch. 2] Let E be an elliptic curve defined over $\mathbb{F}_q$, then any function $f \in \overline{\mathbb{F}}_q(E)$ with divisor*

$$div(f) = n(P) - ([n]P) - (n-1)(\mathcal{O})$$

for some $n \in \mathbb{Z}$ and $P \in E$ is called a Miller function (also called Weil function) and will be denoted as $f_{n,P}$.

Definition 2.24. *[18, Ch. 2] Let E be an elliptic curve defined over $\mathbb{F}_q$, $f \in \overline{\mathbb{F}}_q(E)$ and D a divisor on E with $supp(D) \cap supp(div(f)) = \emptyset$, then by definition we set*

$$f(D) = \prod_{P \in E} f(P)^{ord_P(D)}.$$

Note, multiplicative constants of functions have no influence on the functions divisors. Therefore, if divisor D has degree zero, then the multiplicative constants of f in the above product cancel out. In this case $f(D)$ just depends on $div(f)$.

If D is a divisor of a function $g \in \overline{\mathbb{F}}_q(E)$ we have the following theorem.

Theorem 2.25 (Weil Reciprocity)**.** *[18, Ch. 2] Let E be an elliptic curve and f, g non-zero functions on E with $supp(div(f)) \cap supp(div(g)) = \emptyset$, then $f(div(g)) = g(div(f))$.*

This Theorem states that, if f and g have disjoint zeros and poles then evaluating f at the divisor of g has the same value as evaluating g at the divisor of f.

Definition 2.26 (Equivalence of Divisors)**.** *[18, Ch. 2] Two divisors D and D' are called linearly equivalent, denoted $D \sim D'$ if $D' = D + div(f)$ for some function f.*

2.4.2 Tate Pairing

In the following we define the Tate pairing and the reduced Tate pairing. We will see that the reduced Tate pairing is very similar to the Tate pairing. The reduced Tate pairing is used in practice to standardize the result of the bilinear pairing [16].

Definition 2.27 (Tate Pairing)**.** *[18, Ch. 2] Let E be an elliptic curve over $\mathbb{F}_q$ and $r|\#E(\mathbb{F}_q)$ with r and q coprime, and let k be the embedding degree, then the Tate pairing $\boldsymbol{t}_r$ is the map*

$$\boldsymbol{t}_r : \begin{cases} E(\mathbb{F}_{q^k})[r] \times E(\mathbb{F}_{q^k})/rE(\mathbb{F}_{q^k}) & \to \mathbb{F}_{q^k}^*/(\mathbb{F}_{q^k}^*)^r \\ (P,Q) & \mapsto \boldsymbol{t}_r(P,Q) = f_{r,P}(D_Q), \end{cases}$$

where $div(f_r) = r(P) - r(\mathcal{O})$ and $D_Q \sim (Q) - (\mathcal{O})$ coprime with $div(f_r)$.

Computing the Tate Pairing To compute the Tate pairing we have to evaluate the Miller function $f_{r,P}$ at D_Q (see Definition 2.23). We can do this by using the Miller algorithm, given in Algorithm 1 and [22, 18, Ch. 2]. The Miller algorithm does this by constructing the function Miller $f_{r,P}$. For constructing this function it is necessary to compute the point $[r]P$. This is done by the well known *Double-and-Add* algorithm for elliptic curve point multiplication. The Miller algorithm extends the *Double-and-Add* algorithm by the lines 1, 4 and 7. By these lines the Miller function is constructed and evaluated.

In the following we explain how the Miller algorithm constructs the function $f_{r,P}$. Denote by $l_{[i]P,[j]P}$ the equation of the line that goes through the points $[i]P$ and $[j]P$. Note that, if $[i]P = [j]P$ then $l_{[i]P,[j]P} = l_{[i]P,[i]P}$ is the equation of the tangent line at $[i]P$. And denote by $v_{[i+j]P}$ the equation of the vertical line through $[i+j]P$. The divisor of $l_{[i]P,[j]P}$ is

$$div(l_{[i]P,[j]P}) = ([i]P) + ([j]P) + (-([i+j]P)) - 3(\mathcal{O})$$

and the divisor of $v_{[i+j]P}$ is

$$div(v_{[i+j]P}) = ([i+j]P) + (-([i+j]P)) - 2(\mathcal{O}).$$

To compute $f_{r,P}$ we would like to have a formula that gives us $f_{i+j,P}$, since then we can compute $f_{r,P}$ by using the *Double-and-Add* approach. Exploiting the *Double-and-Add* approach is important, since it provides us a possibility to compute $[r]P$ and $f_{r,P}$ in linear many *double* and *add* operations in the description size of r! Fortunately, we have that

$$f_{i+j,P} = f_{i,P} \cdot f_{j,P} \cdot \frac{l_{[i]P,[j]P}}{v_{[i+j]P}}. \tag{2.8}$$

To compute a *double* operation one just calculates $f_{i+i,P} = f_{2i,P} = f_{i,P}^2 \cdot \frac{l_{[i]P,[i]P}}{v_{[i+i]P}}$.

We show the correctness of (2.8) by analyzing the divisor of the right hand side of it. Then we will see that it fulfills the definition of $f_{i+j,P}$ (Definition 2.23).

$$\begin{aligned}
div(f_{i,P} \cdot f_{j,P} \cdot \frac{l_{[i]P,[j]P}}{v_{[i+j]P}}) &= div(f_{i,P}) + div(f_{j,P}) + div(l_{[i]P,[j]P}) - div(v_{[i+j]P}) \\
&= \big(i(P) - ([i]P) - (i-1)(\mathcal{O})\big) \\
&+ \big(j(P) - ([j]P) - (j-1)(\mathcal{O})\big) \\
&+ \big(([i]P) + ([j]P) + (-([i+j]P)) - 3(\mathcal{O})\big) \\
&- \big(([i+j]P) + (-([i+j]P)) - 2(\mathcal{O})\big) \\
&= \big((i+j)(P) - ([i+j]P) - (i+j-1-1+3-2)(\mathcal{O})\big) \\
&= div(f_{i+j,P})
\end{aligned}$$

All these results in the following algorithm [18, Ch. 2]. Assume that $r \in \mathbb{N}$ has binary representation $r_t \dots r_0$. Note that r's binary representation has $t+1$ bits.

Algorithm 1 Miller algorithm for Tate Pairing

Require: $r \in \mathbb{N}, P \in E[r]$ and divisor D with $supp(D) \cap \{P, \mathcal{O}\} = \emptyset$.
Ensure: $f_{n,P}(D)$.

```
f ← 1
T ← P
for i = t − 1, ..., 0 do
    f ← f^2 · l_{T,T}(D)/v_{[2]T}(D)
    T ← [2]T
    if r_i = 1 then
        f ← f · l_{T,P}(D)/v_{T+P}(D)
        T ← T + P
    end if
end for
return f
```

At the end of the algorithm we have that $f = f_{r,P}(D)$ and $T = rP = \mathcal{O}$.

2.4.3 Reduced Tate Pairing

The Tate pairing is only defined up to a co-set of $(\mathbb{F}_{q^k}^*)^r$. But for protocols it is important to compute unique elements. Therefore we define the reduced Tate pairing to obtain a unique element from $\mathbb{F}_{q^k}^*$. In fact, the reduced pairing maps to the r-th roots of unity in $\mathbb{F}_{q^k}^*$, which we will denote by μ_r.

Definition 2.28 (Reduced Tate Pairing)**.** *[18, Ch. 2] Let E be an elliptic curve over $\mathbb{F}_q$ with $r|\#E(\mathbb{F}_q)$ and $gcd(r,q)$=1, and let k be the embedding degree, then the reduced Tate pairing t_r is the map*

$$t_r : \begin{cases} E(\mathbb{F}_{q^k})[r] \times E(\mathbb{F}_{q^k})/rE(\mathbb{F}_{q^k}) & \to \mu_r \subset \mathbb{F}_{q^k}^* \\ (P,Q) & \mapsto t_r(P,Q) = \boldsymbol{t}_r(P,Q)^{(q^k-1)/r}. \end{cases}$$

Computing the Reduced Tate Pairing Note that $\boldsymbol{t}_r(P,Q)^{(q^k-1)/r} = f_{r,P}(D_Q)^{(q^k-1)/r}$. This means, to compute the reduced Tate pairing we can apply Algorithm 1 and append a calculation for the exponentiation by $(q^k-1)/r$, see Algorithm 2.

Algorithm 2 Miller algorithm for Reduced Tate Pairing

Require: $r \in \mathbb{N}, P \in E[r]$ and divisor D with $supp(D) \cap \{P, \mathcal{O}\} = \emptyset$.
Ensure: $f_{r,P}(D)^{(q^k-1)/r}$.

```
f ← f_{r,P}(D)                    ▷ computed by Algorithm 1
f ← f^{(q^k-1)/r}
return f
```

A useful property of the reduced Tate pairing is the following: let $r|N|(q^k-1)$, then for $P \in E(F_{q^k})[r]$ and $Q \in E(F_{q^k})/rE(F_{q^k})$ we have the equality $t_r(P,Q) = t_N(P,Q)$ [18, Ch. 2].

3 Optimization Techniques

This chapter analyses optimization techniques for the computation of bilinear pairings. The computation of bilinear pairings bases on four different components. In a bottom to top view we have at first the component for the base field arithmetic. On top of this lies the component for the extension field arithmetic. The next component is the elliptic curve arithmetic. At the very top there is the bilinear pairing calculation itself. For every component several optimization techniques do exist. We will structure this chapter according to this bottom to top view and present selected optimization for every component.

For this chapter, we assume that we run a processor with word size n. The subscript q^i specifies the field $\mathbb{F}_{q^i}$, where $i \in \mathbb{N}$. To denote costs, we will use the abbreviations M_{q^i} for multiplication, A_{q^i} for addition, R_{q^i} for operation that are due to reductions, S_{q^i} for the space requirement and W for words. For example, $M_{q^6} = 6M_{q^2} + 9A_{q^2} + R_{q^6}$ and $R_{q^6} = 4M_q + 8A_q$ denotes that a multiplication of two $\mathbb{F}_{q^6}$ elements costs 6 multiplications over the field $\mathbb{F}_{q^2}$, 9 additions over the field $\mathbb{F}_{q^2}$, 4 multiplication and 8 additions over $\mathbb{F}_q$ due to reduction in $\mathbb{F}_{q^6}$. We set $S_{q^i} = i\lceil |q|/n \rceil W$ since representing an element from $\mathbb{F}_{q^i}$ requires i coefficients of $|q|$ bits that has to be portioned over words of size n.

In all analyses we will treat subtractions over a field K as additions over the field K. We analyze the complexity of algorithms in terms of the number of different base field operations and the number of words for storing temporary results. The input and output variables are not considered as memory usage of an algorithm.

We will often consider additions over extension fields. Since we give the complexity in terms of base field operations we state that $A_{q^k} = kA_q$. This is, an addition of two extension field elements from $\mathbb{F}_{q^k}$ can be computed by k additions of k base field elements from $\mathbb{F}_q$.

When designing algorithms that computes mathematical operations it is a common practice in the cryptographic community/industry to provide interfaces of the following form: Let *op* be a function and we have that $c = op(a, b)$ is the result of the operation *op* executed on the parameters a and b. In the programming language C one would implement this interface by the following signature `void op(type *c, const type *a, const type *b)`. A call of this function could look like `op(c, a, b);` where `a,b,c` are variables of the type `type *`. We will design each function such that calls of the form `op(c, c, b);` and `op(c, a, c);` are also possible. This means, the functions do not write into the result variable as long as the input is needed. Such functions have the disadvantage that they may need more space but on the other side their interfaces are less prone to error.

3.1 Extension Field Arithmetic

In this section we will analyze the costs of the Schoolbook and Karatsuba multiplication methods. Here we will also use the abbreviation S_{Mul,q^k} which denotes the number of words needed for computing a multiplication over $\mathbb{F}_{q^k}$ by using the multiplication method Mul. For Mul we will use the abbreviations SB and KS for Schoolbook and Karatsuba multiplication.

3.1.1 Multiplication in Field Towers

Here we will analyze the costs of multiplication in field towers. We will do this for two different approaches. One approach uses the straightforward multiplication method learned in school and the other approach applies the multiplication method of Karatsuba. At the end of this Section we compare both approaches.

3.1.1.1 Schoolbook Multiplication in Field Towers

This section is structured into four parts. The first part analyses schoolbook multiplication in quadratic extensions. The second part analyses schoolbook multiplication in cubic extensions. In the both last parts we apply the former analyses to calculate the costs for multiplication in the extension fields $\mathbb{F}_{q^6}$ and $\mathbb{F}_{q^{12}}$ (see 2.2 and 2.3).

Part 1: Schoolbook Multiplication in Quadratic Extensions The following algorithm shows how to multiply in quadratic extensions of the form $\mathbb{F}_{q^2} = \mathbb{F}_q[x]/(P(x))$ where $P(x)$ is an irreducible binomial over $\mathbb{F}_q[x]$ of degree 2, e.g. $P(x) = x^2 - u$. Each addition and multiplication is computed over the sub-field $\mathbb{F}_q$.

Algorithm 3 Schoolbook Multiplication in Quadratic Extensions

Require: $a = a_0 + a_1x, b = b_0 + b_1x \in \mathbb{F}_{q^2} = \mathbb{F}_q[x]/(x^2 - u)$.
Ensure: $c = a \cdot b = (c_0 + c_1x) \in \mathbb{F}_{q^2}$.

1: $T_0 \leftarrow a_0 \cdot b_1$
2: $T_1 \leftarrow a_1 \cdot b_0$
3: $T_1 \leftarrow T_1 + T_0$
4: $T_0 \leftarrow a_1 \cdot b_1$
5: $T_0 \leftarrow T_0 \cdot u$ // reduce: $x^2 \equiv u$
6: $c_0 \leftarrow a_0 \cdot b_0$
7: $c_0 \leftarrow c_0 + T_0$
8: $c_1 \leftarrow T_1$
9: **return** $c_0 + c_1x$

Theorem 3.1. *Let $\mathbb{F}_{q^2} := \mathbb{F}_q[x]/(P(x))$ where $P(x) = x^2 - u$ is an irreducible binomial over $\mathbb{F}_q[x]$. Let $a, b \in \mathbb{F}_{q^2}$, then Algorithm 3 computes $c = a \cdot b \in \mathbb{F}_{q^2}$*

with 4 *multiplications,* 1 *addition,* 1 *additional multiplication and addition in* $\mathbb{F}_q$ *due to reduction and memory for two* $\mathbb{F}_q$ *elements, i.e.* $M_{q^2} = 4M_q + 1A_q + R_{q^2}$, $R_{q^2} = 1M_q + 1A_q$ *and* $S_{SB,q^2} = 2 \cdot S_q$.

Proof. We first show the correctness of the algorithm and prove then its complexity.

Correctness The algorithm has to compute the following formula:

$$\begin{aligned} a \cdot b =& (a_0 + a_1 x) \cdot (b_0 + b_1 x) \\ =& a_0 b_0 + a_1 b_0 x + b_1 a_0 x + a_1 b_1 x^2 \\ =& [a_0 b_0 + (a_1 b_1) u] + [(a_1 b_0 + b_1 a_0) x]. \end{aligned}$$

By substituting the result $c_0 + c_1 T$ of Algorithm 3 successively with the algorithms steps, we get that the algorithm actually computes the correct formula.

Complexity By simply counting the operations we get 5 multiplications and 2 additions in the field $\mathbb{F}_q$. Note that the multiplication in step 5 and the addition in step 7 is due to the reduction, i.e. $R_{q^2} = 1M_q + 1A_q$. Thus the plain multiplication of a and b requires 4 multiplications and 1 addition as claimed for the schoolbook method. The algorithm needs the temporary variable T_0 and T_1 of the type $\mathbb{F}_q$. Hence, the memory usage is $2 \cdot S_q$ words. □

Part 2: Schoolbook Multiplication in Cubic Extensions Algorithm 4 shows how to multiply in cubic extensions of the form $\mathbb{F}_{q^3} = \mathbb{F}_q[y]/(Q(y))$ where $Q(y)$ is an irreducible binomial over $\mathbb{F}_q$ of degree 2, e.g. $Q(y) = y^3 - v$. Each addition and multiplication is computed over the field $\mathbb{F}_q$.

Theorem 3.2. *Let* $\mathbb{F}_{q^3} = \mathbb{F}_q[y]/(Q(y))$ *where* $Q(y) = (y^3 - v)$ *is an irreducible binomial over* $\mathbb{F}_q$. *Let* $a, b \in \mathbb{F}_{q^3}$, *then Algorithm 4 computes* $c = a \cdot b \in \mathbb{F}_{q^3}$ *with* 9 *multiplications,* 4 *additions,*2 *additional multiplications and additions in* $\mathbb{F}_q$ *due to reduction and memory for three field elements* $\mathbb{F}_q$, *i.e.* $M_{q^3} = 9M_q + 4A_q + R_{q^3}$, $R_{q^3} = 2M_q + 2A_q$ *and* $S_{SB,q^3} = 3 \cdot S_q$.

Proof. We start with the correctness and prove then the complexity.

Correctness The algorithm has to compute the following formula:

$$\begin{aligned} a \cdot b =& (a_0 + a_1 y + a_2 y^2) \cdot (b_0 + b_1 y + b_2 y^2) \\ =& a_0 b_0 + a_0 b_1 y + a_0 b_2 y^2 + a_1 b_0 y + a_1 b_1 y^2 + a_1 b_2 y^3 + a_2 b_0 y^2 + a_2 b_1 y^3 + a_2 b_2 y^4 \\ =& [a_0 b_0 + (a_1 b_2 + a_2 b_1) v] + [a_0 b_1 + a_1 b_0 + a_2 b_2 v] \cdot y + [a_0 b_2 + a_1 b_1 + a_2 b_0] \cdot y^2. \end{aligned}$$

The correctness can be shown by substituting the result $c_0 + c_1 y + c_2 y^2$ successively with the algorithm steps.

Algorithm 4 Schoolbook Multiplication in Cubic Extensions

Require: $a = a_0 + a_1 y + a_2 y^2, b = b_0 + b_1 y + b_2 y^2 \in \mathbb{F}_{q^3} = \mathbb{F}_q[y]/(y^3 - v)$.
Ensure: $c = a \cdot b = (c_0 + c_1 y + c_2 y^2) \in \mathbb{F}_{q^3}$.

1: Step 2-6: $T_2 = (a_2 b_0 + a_1 b_1 + a_0 b_2) S^2$
2: $T_2 \leftarrow a_2 \cdot b_0$
3: $T_0 \leftarrow a_1 \cdot b_1$
4: $T_2 \leftarrow T_2 + T_0$
5: $T_0 \leftarrow a_0 \cdot b_2$
6: $T_2 \leftarrow T_2 + T_0$
7: Step 8-13: $T_1 = (a_2 b_2 \cdot v + a_1 b_0 + a_0 b_1) S$
8: $T_1 \leftarrow a_2 \cdot b_2$
9: $T_0 \leftarrow T_1 \cdot v$ // reduce: $y^4 \equiv v \cdot S$
10: $T_1 \leftarrow a_1 \cdot b_0$
11: $T_1 \leftarrow T_1 + T_0$
12: $T_0 \leftarrow a_0 \cdot b_1$
13: $T_1 \leftarrow T_1 + T_0$
14: Step 15-22: $T_0 = a_2 b_1 \cdot v + a_1 b_2 \cdot v + a_0 b_0$
15: $T_0 \leftarrow a_2 \cdot b_1$
16: $c_1 \leftarrow a_1 \cdot b_2$
17: $T_0 \leftarrow T_0 + c_1$
18: $c_0 \leftarrow a_0 \cdot b_0$
19: $T_0 \leftarrow T_0 \cdot v$ // reduce: $y^3 \equiv v$
20: $c_0 \leftarrow c_0 + T_0$
21: $c_1 \leftarrow T_1$
22: $c_2 \leftarrow T_2$
23: **return** $c_0 + c_1 y + c_2 y^2$

Complexity The algorithm needs 11 multiplications and 6 additions in the field $\mathbb{F}_q$. The multiplication in step 9 and 19 as well as the additions in step 11 and 20 are due to reduction, i.e. $R_{q^3} = 2M_q + 2A_q$. The algorithm needs memory for the variables T_0, T_1 and T_2 and this results in a memory requirement of $3 \cdot S_q$ words. □

Part 3: Schoolbook multiplication in $\mathbb{F}_{p^6}$ Here we analyze the costs of schoolbook multiplication for the extension field $\mathbb{F}_{p^6}$ as defined in 2.2. We defined this extension field over the tower $\mathbb{F}_p \to \mathbb{F}_{p^2} \to \mathbb{F}_{p^6}$. This means, this is a *cubic extension of a quadratic extension of the field* $\mathbb{F}_p$. Thus, to compute multiplications in $\mathbb{F}_{p^6}$ via the schoolbook method one has to apply Algorithm 4 that has to work on the sub-field $\mathbb{F}_{p^2}$ and Algorithm 3 that has to work on the sub-field $\mathbb{F}_p$. This also tells us how to calculate the resulting costs of multiplication in terms of base field operations.

Lemma 3.3. *Let $\mathbb{F}_{p^2}$ and $\mathbb{F}_{p^6}$ be defined as in 2.1 and 2.2. Let Algorithm 4 run on inputs from $\mathbb{F}_{p^6}$ and Algorithm 3 run on inputs from $\mathbb{F}_{p^2}$* then multiplication in $\mathbb{F}_{p^6}$ costs $M_{p^6} = 36M_p + 45A_p$ and $S_{SB,p^6} = 8 \cdot S_p$.

Proof. Since we defined $\mathbb{F}_{p^2}$ as $\mathbb{F}_p[x]/(P(x))$, where $P(x) = x^2 - u$ with $u = -1$, we can exchange the multiplication $T_0 \cdot u$ in step 5 of Algorithm 3 by $p - T_0$, since $T_0 \cdot (-1) \equiv p - T_0$. Thus, the choice of the irreducible polynomial saves one multiplication. Due to this and Theorem 3.1 we know that a multiplication in $\mathbb{F}_{p^2}$ costs $M_{p^2} = 4M_p + 1A_p + R_{p^2}$ with $R_{p^2} = 2A_p$ and $S_{p^2} = 2 \cdot S_p$.

We defined $\mathbb{F}_{p^6}$ as $\mathbb{F}_{p^2}[y]/(Q(y))$, where $Q(y) = y^3 - v$ with $v = u+1$ and $u = x \mod P(x)$. Therefore, we can exchange two multiplications in Algorithm 4. At first we exchange the multiplication $T_1 \cdot v = c_1(x+1)$ in step 9. We have $T_1 \in \mathbb{F}_{p^2}$, thus T_1 is of the form $a_0 + a_1 x$ and $a_0, a_1 \in \mathbb{F}_p$. We calculate now the product and analyze then which operations we need to implement step 9 for $v = 1 + x$:

$$\begin{aligned} T_1 \cdot (x+1)y =& (a_0 + a_1 x) \cdot (x+1)y \\ =& (a_0 + a_1 x + a_0 x + a_1 x^2)y \\ =& (a_0 + q - a_1 + (a_0 + a_1)x)y. \end{aligned}$$

Hence, we need 2 additions and 1 subtraction over the field $\mathbb{F}_p$ to compute $T_1 \cdot (x+1)y$.

Second, we exchange the multiplication in step 19: $T_0 \cdot v = T_0 \cdot (x+1)$. Since $T_0 \in \mathbb{F}_{p^2}$ this results in the same as before. Thus we need here also 2 additions and 1 subtraction over the field $\mathbb{F}_p$. Hence, these multiplications cost 6 additions over the field $\mathbb{F}_p$ and we have $R_{p^6} = 6A_p + 2A_{p^2} = 10A_p$. All in all, we have$M_{p^6} = 9M_{p^2} + 4A_{p^2} + R_{p^6}$.

Thus, the number of field operations for the schoolbook multiplication of two

$\mathbb{F}_{p^6}$ elements is:

$$\begin{aligned}
M_{p^6} =& 9M_{p^2} + 4A_{p^2} + R_{p^6} \\
=& 9 \cdot (4M_p + 1A_p + R_{p^2}) + 4A_{p^2} + 10A_p \\
=& 36M_p + 9A_p + 4 \cdot 2A_p + 9R_{p^2} + R_{p^6} \\
=& 36M_p + 17A_p + 28A_p \\
=& 36M_p + 45A_p.
\end{aligned}$$

Due to Theorem 3.2 Algorithm 4 needs three additional $\mathbb{F}_{p^2}$ elements and due to Theorem 3.1 Algorithm 3 needs two additional $\mathbb{F}_p$ elements. This results in a total space requirement of

$$\begin{aligned}
S_{SB,p^6} &= 3 \cdot 2 \cdot \lceil |p|/n \rceil W + 2 \cdot \lceil |p|/n \rceil W \\
&= 8 \cdot \lceil |p|/n \rceil W \\
&= 8 \cdot S_p
\end{aligned}$$

since an element from $\mathbb{F}_{p^2}$ is described by to elements from $\mathbb{F}_p$ and an element from $\mathbb{F}_p$ needs $|p| = \lfloor \log_2(p) \rfloor + 1$ bits. Note that, although the multiplication algorithms for the cubic and quadratic extension are called several times, it is enough to add up their basic memory requirement just one time, since the algorithms are executed sequentially such that the memory space is reusable. □

Part 4: Schoolbook multiplication in $\mathbb{F}_{p^{12}}$ We will now analyze the costs of multiplication for the extension field $\mathbb{F}_{p^{12}}$ as defined in 2.3 . Since we defined $\mathbb{F}_{p^{12}}$as the tower $\mathbb{F}_p \to \mathbb{F}_{p^2} \to \mathbb{F}_{p^6} \to \mathbb{F}_{p^{12}}$ we can reuse the former analysis for the tower $\mathbb{F}_p \to \mathbb{F}_{p^2} \to \mathbb{F}_{p^6}$ and just analyze the costs of the quadratic extension of $\mathbb{F}_{p^6}$ namely $\mathbb{F}_{p^{12}}$. Thus, we have to analyze the costs of Algorithm 3 running on inputs from $\mathbb{F}_{p^{12}}$.

Lemma 3.4. *Let $\mathbb{F}_{p^{12}}$ be defined as in* (2.3)*. Let Algorithm 3 run on inputs from* $\mathbb{F}_{p^{12}}$ then multiplication in $\mathbb{F}_{p^{12}}$ costs $M_{p^{12}} = 144M_p + 195A_p$ and $S_{SB,p^{12}} = 20 \cdot S_p$ additional space.

Proof. Since $\mathbb{F}_{p^{12}} := \mathbb{F}_{p^6}[z]/(R(z))$, where $R(z) = z^2 - w$ with $w = y \mod Q(y)$, we can exchange the multiplication $T_0 \cdot u$ (at this point $u = y$) in step 5 of Algorithm 3. Let $T_0 = a_0 + a_1 y + a_2 y^2 \in \mathbb{F}_{p^6}$ and let $a_2 = a_{20} + a_{21}x \in \mathbb{F}_{p^2}$, then we have to compute:

$$\begin{aligned}
T_0 \cdot v =& (a_0 + a_1 y + a_2 y^2) \cdot y \\
=& (a_0 y + a_1 y^2 + a_2 y^3) \\
=& (a_2(u+1) + a_0 y + a_1 y^2) \\
=& ([a_{20} + p - a_{21} + a_{21}u + a_{20}u] + a_0 y + a_1 y^2).
\end{aligned}$$

This multiplication just effects the values of a_2. The other coefficients a_0 and

a_1 are just shifted within the polynomial. The shifts are for free, since in step 7 we have to add this result to c_0. There we can split the $\mathbb{F}_{p^6}$ addition in three $\mathbb{F}_{p^2}$ additions - which costs the same - and add up simply the correct parts of the summands in step 7. Thus, the costs for the multiplication $T_0 \cdot y$ are just 2 additions and 1 subtraction over the base field $\mathbb{F}_p$.

All in all, we have $M_{p^{12}} = 4M_{p^6} + 2A_{p^6} + 3A_p$.

Due to Lemma 3.3, Theorem 3.1 and the arguments made above the number of base field operations for the schoolbook multiplication of two $\mathbb{F}_{p^{12}}$ elements is:

$$\begin{aligned} M_{p^{12}} =& 4M_{p^6} + 2A_{p^6} + 3A_p \\ =& 4(36M_p + 45A_p) + 2 \cdot 6A_p + 3A_p \\ =& 144M_p + 195A_p. \end{aligned}$$

Algorithm 3 running on inputs from $\mathbb{F}_{p^{12}}$ needs memory space for the variables T_0 and T_1 of the type $\mathbb{F}_{p^6}$. Due to this and Lemma 3.3 the total memory requirement is:

$$\begin{aligned} S_{SB,p^{12}} =& 2 \cdot 6 \cdot \lceil |p|/n \rceil W + S_{SB,p^6} \\ =& 2 \cdot 6 \cdot \lceil |p|/n \rceil W + 8 \cdot \lceil |p|/n \rceil W \\ =& 20 \cdot S_p. \end{aligned}$$

□

3.1.1.2 Karatsuba Multiplication in Field Towers

As in the section before, this section is structured into four parts. The first part analyses Karatsuba multiplication in quadratic extensions. The second part analyses Karatsuba multiplication in cubic extensions. In the both last parts we apply these analyses to calculate the multiplication costs for extension fields with extension degree 6 and 12.

The algorithms presented in this section are based on the algorithms for multiplying in extension fields from[2]. We modify the algorithms such that their memory usage is minimized and we prove their correctness.

Part 1: Karatsuba Multiplication in Quadratic Extensions Algorithm 5 shows how to multiply with the Karatsuba method [21] in quadratic extensions of the form $\mathbb{F}_{q^2} = \mathbb{F}_q[x]/(P(x))$ where $P(x)$ is an irreducible binomial over $\mathbb{F}_q[x]$ of degree 2, e.g. $P(x) = x^2 - u$. Each multiplication and addition of Algorithm 5 is over the field $\mathbb{F}_q$.

Theorem 3.5. *Let $\mathbb{F}_{q^2} := \mathbb{F}_q[x]/(P(x))$ where $P(x) = x^2 - u$ is an irreducible binomial over $\mathbb{F}_q[x]$. Let $a, b \in \mathbb{F}_{q^2}$, then Algorithm 5 computes $c = a \cdot b \in \mathbb{F}_{q^2}$ with 3 multiplications, 3 additions, 2 subtractions, 1 additional multiplication due to*

Algorithm 5 Karatsuba Multiplication in Quadratic Extensions

Require: $a = a_0 + a_1x, b = b_0 + b_1x \in \mathbb{F}_{q^2} = \mathbb{F}_q[x]/(x^2 - u)$.
Ensure: $c = a \cdot b = (c_0 + c_1x) \in \mathbb{F}_{q^2}$.

1: $T_0 \leftarrow b_0 + b_1$
2: $T_1 \leftarrow a_0 + a_1$
3: $T_1 \leftarrow T_1 \cdot T_0$
4: $T_0 \leftarrow a_1 \cdot b_1$
5: $c_0 \leftarrow a_0 \cdot b_0$
6: $c_1 \leftarrow T_1 - c_0$
7: $c_1 \leftarrow c_1 - T_0$
8: $T_0 \leftarrow T_0 \cdot u$ // reduce: $x^2 \equiv u$
9: $c_0 \leftarrow c_0 + T_0$
10: **return** $c_0 + c_1x$

reduction and memory for two elements from $\mathbb{F}_q$, i.e. $M_{q^2} = 3M_q + 4A_q + R_{q^2}$ and $S_{KS,q^2} = 2 \cdot S_q$.

Proof. We split the proof in the parts correctness and complexity.

Correctness The Karatsuba method requires to compute the product of a and b in the following way:

$$a \cdot b = a_0 \cdot b_0 + a_1 \cdot b_1 \cdot u + [(a_0 + a_1) \cdot (b_0 + b_1) - (a_0 \cdot b_0 + a_1 \cdot b_1)]x. \quad (3.1)$$

This method requires in our case only 3 multiplications to compute $a \cdot b$, namely $a_0 \cdot b_0$, $a_1 \cdot b_1$ and $(a_0 + a_1) \cdot (b_0 + b_1)$. The results of the two products $a_0 \cdot b_0$ and $a_1 \cdot b_1$ are just reused. The term within the parentheses is:

$$\begin{aligned}((a_0 + a_1) \cdot (b_0 + b_1) - (a_0 \cdot b_0 + a_1 \cdot b_1)) =& a_0b_0 + a_1b_0 + a_0b_1 + a_1b_1 - a_0b_0 - a_1b_1 \\ =& a_1b_0 + a_0b_1.\end{aligned}$$

Substituted in (3.1) we get

$$a \cdot b = a_0b_0 + a_1b_1u + [a_1b_0i + a_0b_1]x$$

which is exactly what we want to compute.

One can show that Algorithm 5 computes $a \cdot b$ according to Karatsuba's method by substituting the result $c_0 + c_1x$ successively with the algorithm's steps.

Complexity The algorithm uses 3 multiplications and 4 additions over the field $\mathbb{F}_q$ and 1 additional multiplication and addition due to reduction over the field $\mathbb{F}_q$, hence $R_{q^2} = 1M_q + 1A_q$. Furthermore, it uses the two variables T_0 and T_1 that require $2 \cdot S_q$. □

Part 2: Karatsuba Multiplication in Cubic Extensions Algorithm 6 shows how to multiply with the Karatsuba method in cubic extensions of the form $\mathbb{F}_{q^3} = \mathbb{F}_q[y]/(Q(y))$ where $Q(y)$ is an irreducible binomial over $\mathbb{F}_q$ of degree 2, e.g. $Q(y) = y^3 - v$. Each addition and multiplication is computed over the field $\mathbb{F}_q$.

Algorithm 6 Karatsuba Multiplication in Cubic Extensions

Require: $a = a_0 + a_1 y + a_2 y^2, b = b_0 + b_1 y + b_2 y^2 \in \mathbb{F}_{q^3} = \mathbb{F}_q[y]/(y^3 - v)$.
Ensure: $c = a \cdot b = (c_0 + c_1 y + c_2 y^2) \in \mathbb{F}_{q^3}$.

1: Step 2-8: $t_0 = a_0 b_0 + v \cdot [(a_1 + a_2) \cdot (b_1 + b_2) - a_1 b_1 - a_2 b_2]$
2: $V_0 \leftarrow a_0 \cdot b_0, V_1 \leftarrow a_1 \cdot b_1, V_2 \leftarrow a_2 \cdot b_2$
3: $t_0 \leftarrow a_1 + a_2, t_1 \leftarrow b_1 + b_2$
4: $t_0 \leftarrow t_0 \cdot t_1$
5: $t_1 \leftarrow V_1 + V_2$
6: $t_0 \leftarrow t_0 - t_1$
7: $t_1 \leftarrow t_0 \cdot v$
8: $t_0 \leftarrow t_1 + V_0$
9: Step 10-15: $t_1 = (a_0 + a_1) \cdot (b_0 + b_1) - a_0 b_0 - a_1 b_1 + v a_2 b_2$
10: $t_1 \leftarrow a_0 + a_1, t_2 \leftarrow b_0 + b_1$
11: $t_1 \leftarrow t_1 \cdot t_2$
12: $t_1 \leftarrow t_1 - V_0$
13: $t_1 \leftarrow t_1 - V_1$
14: $t_2 \leftarrow V_2 \cdot v$
15: $t_1 \leftarrow t_1 + t_2$
16: Step 17-21: $c_2 = (a_0 + a_2) \cdot (b_0 + b_2) - a_0 b_0 - a_2 b_2 + a_1 b_1$
17: $t_2 \leftarrow a_0 + a_2, c_2 \leftarrow b_0 + b_2$
18: $c_2 \leftarrow t_2 \cdot c_2$
19: $c_2 \leftarrow c_2 - V_0$
20: $c_2 \leftarrow c_2 - V_2$
21: $c_2 \leftarrow c_2 + V_1$
22: $c_1 \leftarrow t_1$
23: $c_0 \leftarrow t_0$
24: **return** $c_0 + c_1 y + c_2 y^2$

Theorem 3.6. *Let $\mathbb{F}_{q^3} = \mathbb{F}_q[y]/(Q(y))$ where $Q(y) = (y^3 - v)$ is an irreducible binomial over $\mathbb{F}_q$. Let $a, b \in \mathbb{F}_{q^3}$, then Algorithm 6 computes $c = a \cdot b \in \mathbb{F}_{q^3}$ with 6 multiplications, 13 additions, 2 multiplication and 2 additions over $\mathbb{F}_q$ due to reduction and memory for 6 elements from $\mathbb{F}_q$, i.e. $M_{q^3} = 6M_q + 13A_q + R_{q^3}$, $R_{q^3} = 2M_q + 2A_q$ and $S_{KS,q^3} = 6 \cdot S_q$.*

Proof. We first prove the correctness and then the complexity of Algorithm 6.

Correctness The Karatsuba method requires to compute the product of a and b in the following way [2]:

$$\begin{aligned} a \cdot b =& (a_0 + a_1 y + a_2 y^2) \cdot (b_0 + b_1 y + b_2 y^2) \\ =& [((a_1 + a_2) \cdot (b_1 + b_2) - a_1 b_1 - a_2 b_2) \cdot v + a_0 b_0] \\ &+ [(a_0 + a_1) \cdot (b_0 + b_1) - a_0 b_0 - a_1 b_1 + a_2 b_2 \cdot v] y \\ &+ [(a_0 + a_2) \cdot (b_0 + b_2) - a_0 b_0 - a_2 b_2 + a_1 b_1] v^2. \end{aligned} \tag{3.2}$$

This strategy requires only 6 multiplications, namely $(a_1 + a_2) \cdot (b_1 + b_2)$, $(a_0 + a_1) \cdot (b_0 + b_1)$, $(a_0 + a_2) \cdot (b_0 + b_2)$ and $a_0 b_0$, $a_1 b_1$ and $a_2 b_2$.

Substituting the result $c_0 + c_1 y + c_2 y^2$ of Algorithm 6 successively with the Algorithms steps, one can show that it calculates $a \cdot b$ according to (3.2).

Complexity To compute the product $a \cdot b$ Algorithm 6 needs 6 multiplications and 13 additions over the field $\mathbb{F}_q$, 2 additional multiplications and additions over the field $\mathbb{F}_q$ due to reduction, hence $R_{q^3} = 2M_q + 2A_q$, and memory for the 6 variables V_0, V_1, V_2 and t_0, t_1, t_2 of the type $\mathbb{F}_q$ that require $6 \cdot S_q$. □

Part 3: Karatsuba Multiplication in $\mathbb{F}_{p^6}$ Here we analyze the costs of Karatsuba multiplication for the extension field $\mathbb{F}_{p^6}$ as defined in 2.2. We defined the field $\mathbb{F}_{q^6}$ by the tower $\mathbb{F}_p \to \mathbb{F}_{p^2} \to \mathbb{F}_{p^6}$. Since this is a cubic extension of a quadratic extension we can apply Algorithm 6 and Algorithm 5 to compute multiplications over the field $\mathbb{F}_{p^6}$. For that we run Algorithm 6 on inputs from $\mathbb{F}_{p^6}$ and Algorithm 6 runs Algorithm 5 on inputs from $\mathbb{F}_{p^2}$. According to this strategy we analyze the multiplication costs over $\mathbb{F}_{p^6}$ by using Theorem 3.5 and Theorem 3.6.

Lemma 3.7. *Let $\mathbb{F}_{p^2}$ and $\mathbb{F}_{p^6}$ be defined as in 2.1 and 2.2. Let Algorithm 6 run on inputs from $\mathbb{F}_{p^6}$ and Algorithm 5 run on inputs from $\mathbb{F}_{p^2}$* then multiplication in $\mathbb{F}_{p^6}$ costs $M_{p^6} = 18M_p + 68A_p$ and $S_{KS,q^6} = 14 \cdot S_p$ additional space.

Proof. Since we defined $\mathbb{F}_{p^2}$ as $\mathbb{F}_p[x]/(P(x))$, where $P(x) = x^2 - u$ with $u = -1$, we can exchange the multiplication $T_0 \cdot u$ in step 7 of Algorithm 5 by $p - T_0$, since $T_0 \cdot (-1) \equiv p - T_0$. Thus, the choice of the irreducible polynomial saves one multiplication. Due to this and Theorem 3.5 we know that a multiplication in $\mathbb{F}_{p^2}$ costs $M_{p^2} = 3M_p + 4A_p + R_{p^2}$ where $R_{p^2} = 2A_p$.

We defined $\mathbb{F}_{p^6}$ as $\mathbb{F}_{p^2}[y]/(Q(y))$, where $Q(y) = y^3 - v$ with $v = u+1$ and $u = x \mod P(x)$. Therefore we can exchange the multiplications in step 7 and step 14 of Algorithm 6. In step 7 we want to compute $t_1 \leftarrow t_0 \cdot v$ with $t_1, t_0 \in \mathbb{F}_{p^2}$ and $t_0 = d_0 + d_1 x$. For this we consider the multiplication with $v = x + 1$ and

we get:

$$\begin{aligned} t_0 \cdot (x+1) =& t_0 \cdot x + t_0 \\ =& [d_0 + d_1 x^2] + [d_0 + d_1]x \\ =& [d_0 - d_1] - [d_0 + d_1]x. \end{aligned}$$

Hence we can exchange the multiplication $t_0 \cdot v$ by 1 addition and 1 subtraction over the field $\mathbb{F}_p$. The same applies to the multiplication $t_2 \leftarrow V_2 \cdot v$ in step 14. All in all, we get that these multiplications costs 4 base field additions instead of 2 multiplications over the field $\mathbb{F}_{p^2}$. Therefore we get $R_{p^6} = 4A_p + 2A_p$. Due to this and Theorem 3.6 we know that a multiplication in $\mathbb{F}_{p^6}$ costs $M_{p^6} = 6M_{p^2} + 13A_{p^2} + R_{p^6}$.

Thus, the number of base field operations for Karatsuba multiplication of two $\mathbb{F}_{p^6}$ elements is:

$$\begin{aligned} M_{p^6} =& 6M_{p^2} + 13A_{p^2} + R_{p^6} \\ =& 6 \cdot (3M_p + 6A_p) + 26A_p + 6A_p \\ =& 18M_p + 36A_p + 26A_p + 6A_p \\ =& 18M_p + 68A_p. \end{aligned}$$

Due to Theorem 3.5 Algorithm 5 needs memory for $2\lceil |p|/n \rceil$ words and due to 3.6 Algorithm 6 needs memory for $6 \cdot 2 \cdot \lceil |p|/n \rceil$ words. This results in a total memory requirement of

$$\begin{aligned} S_{KS,p^6} =& 6 \cdot 2 \cdot \lceil |p|/n \rceil W + 2\lceil |p|/n \rceil W \\ =& 14 \cdot \lceil |p|/n \rceil W \\ =& 14 \cdot S_p \end{aligned}$$

□

Part 4: Karatsuba Multiplication in $\mathbb{F}_{p^{12}}$ Here we analyze the costs of Karatsuba multiplication over the extension field $\mathbb{F}_{p^{12}}$ as defined in 2.3. Since we defined $\mathbb{F}_{p^{12}}$ as a quadratic extension of the field $\mathbb{F}_{p^6}$ we can run Algorithm 5 on inputs from $\mathbb{F}_{p^{12}}$ to compute multiplication over $\mathbb{F}_{p^{12}}$. Algorithm 5 then applies Algorithm 6 to compute multiplication over $\mathbb{F}_{p^6}$. How to compute multiplications over $\mathbb{F}_{p^6}$ we discussed in the former analysis. Hence, we can also apply the former analysis to analyze the multiplication costs over the field $\mathbb{F}_{p^{12}}$.

Lemma 3.8. *Let $\mathbb{F}_{p^{12}}$ be defined as in* (2.3)*. Let Algorithm 5 run on inputs from* $\mathbb{F}_{p^{12}}$ then multiplication in $\mathbb{F}_{p^{12}}$ costs $M_{p^{12}} = 54M_p + 237A_p$ and $S_{KS,p^{12}} = 26 \cdot S_p$ additional space.

Proof. Since $\mathbb{F}_{p^{12}} = \mathbb{F}_{p^6}[z]/(R(z))$, where $R(z) = z^2 - w$ with $w = y \mod Q(y)$, we can exchange the multiplication $T_0 \cdot u$ where $u = y$ in step 8 of Algorithm 5.

Let $T_0 = a_1 b_1 z^2 = (d_0 + d_1 y + d_2 y^2) z^2 \in \mathbb{F}_{q^6}$ and let $d_2 = d_{20} + d_{21} x \in \mathbb{F}_{q^2}$, then we have to compute:

$$\begin{aligned} T_0 \cdot y =& (d_0 + d_1 y + d_2 y^2) \cdot y \\ =& (d_0 y + d_1 y^2 + d_2 y^3) \\ =& (d_2 (x+1) + d_0 y + d_1 y^2) \\ =& ([d_{20} + p - d_{21} + d_{21} x + d_{20} x] + d_0 y + d_1 y^2). \end{aligned}$$

This multiplication just effects the values of d_2. The other coefficients d_0 and d_1 are just shifted within the polynomial. The shifts are for free, since in step 9 we have to add $T_0 \cdot y$ to c_0. There we are able to split the $\mathbb{F}_{p^6}$ addition in three $\mathbb{F}_{p^2}$ additions - which costs the same - and add up simply the correct parts of the summands in step 4. Thus, the costs for the multiplication $T_0 \cdot y$ are just 2 additions and 1 subtraction over the base field $\mathbb{F}_p$.

All in we have $M_{p^{12}} = 3M_{p^6} + 5A_{p^6} + 3A_p$.

Due to Lemma 3.7, Theorem 3.5 and the arguments made above the number of base field operations for the schoolbook multiplication of two $\mathbb{F}_{p^{12}}$ elements is:

$$\begin{aligned} M_{p^{12}} =& 3M_{p^6} + 5A_{p^6} + 3A_p \\ =& 3(18M_p + 68A_p) + 30A_p + 3A_p \\ =& 54M_p + 204A_p + 33A_p \\ =& 54M_p + 237A_p. \end{aligned}$$

To multiply two $\mathbb{F}_{p^{12}}$ elements we need memory for the variables T_0, T_1 as well as the memory requirement proved in Lemma 3.7:

$$\begin{aligned} S_{KS,p^{12}} =& 2 \cdot (6 \cdot \lceil |p|/n \rceil W) + S_{KS,p^6} \\ =& 12 \cdot \lceil |p|/n \rceil W + 14 \cdot \lceil |p|/n \rceil W \\ =& 26 \cdot \lceil |p|/n \rceil W \\ =& 26 \cdot S_p. \end{aligned}$$

□

3.1.1.3 Schoolbook vs. Karatsuba

In the following table we summarize the results of the former analyses. Each entry shows the number of operations and the required space under a particular method and extension degree k.

k	Schoolbook	Karatsuba
6	$36M_p + 45A_p$, $8S_p$	$18M_p + 68A_p$, $14S_p$
12	$144M_p + 195A_p$, $20S_p$	$54M_p + 237A_p$, $26S_p$

Table 3.1: Schoolbook multiplication vs. Karatsuba multiplication

Of course, the Schoolbook method requires for both cases more multiplications over $\mathbb{F}_p$ than the Karatsuba method. Whereas for saving multiplications the Karatsuba method requires more additions than the Schoolbook method. But usually multiplications have a much greater impact on the run time than additions, such that the Karatsuba method will require less run time than the Schoolbook method.

This is just true if one ignores the space requirements. We see that the Schoolbook method does not need as much space as the Karatsuba method. The difference between the space requirements is in both cases $6S_p$. With this difference the Schoolbook method could be nearly as efficient as the Karatsuba method or even outperform it. For $k = 6$ this is the case, if we would have that one load and store procedure of $1S_p$ is at least as expensive as $3M_p$. For $k = 12$ one load and store procedure of $1S_p$ would have to be at least as expensive as $15M_p$. Otherwise the Karatsuba method will outperform the Schoolbook method.

3.2 Elliptic Curve Arithmetic and Line Equations

This section considers the arithmetic of elliptic curves and optimizations for it. In particular we will consider ordinary elliptic curves of prime order with embedding degree 6 and 12. One optimization is to reuse temporary results of the double and add calculations for computing the line equations, see for example Algorithm 1 at line 4 and 7. Therefore this section also considers the arithmetic of the line equations. In particular we will present algorithms that compute the elliptic curve operations and line equations at once.

We will denote costs of elliptic curve operations as follows: by $\mathcal{A}_{Curve,Coord}$ we denote the costs of an add operation for the curve "$Curve$" on coordinates of the system "$Coord$". We denote the costs of a double operation by $\mathcal{D}_{Curve,Coord}$ where the elliptic curve and the coordinate system is specified as before. We will use the abbreviations MNT and BN for specific elliptic curves, Jac and Aff for Jacobian respectively affine coordinates. The costs for computing tangents and lines will be denoted by $\mathcal{T}$ and $\mathcal{L}$.

3.2.1 Jacobian Coordinates and Group law

For computing an add or double operation for elliptic curve points one may implement the formulas (2.6) and (2.7), respectively. These formulas compute elliptic curve points in affine coordinates. For that one has to compute the inverse of $(x_2 - x_1)$ respectively $2y_1$. But computing the inverse usually is more

expensive than computing a few multiplications, additions and / or subtractions. We can avoid the inversions by representing the points of an elliptic curve by Jacobian coordinates. In this case the inversions are replaced by a few multiplications, additions and subtractions. We can do the calculations in Jacobian coordinates instead of affine coordinates since there exists an one-to-one transformation between these two classes of coordinates [15, Ch. 3]:

Let K be a field, and let c and d be positive integers. One can define an equivalence relation $\sim$ on the set $K^3 \setminus \{(0,0,0)\}$ of nonzero triples over K by

$$(X_1, Y_1, Z_1) \sim (X_2, Y_2, Z_2) \text{ if } X_1 = \lambda^c X_2, Y_1 = \lambda^d Y_2, Z_1 = \lambda Z_2 \text{ for some } \lambda \in K^*.$$

The equivalence class containing $(X, Y, Z) \in K^3 \setminus \{(0,0,0)\}$ is

$$(X : Y : Z) = \{(\lambda^c X, \lambda^d Y, \lambda Z) : \lambda \in K^*\}.$$

We call $(X : Y : Z)$ a projective point, and (X, Y, Z) is called a representative of $(X : Y : Z)$. Notice that if $(X', Y', Z') \in (X : Y : Z)$ then $(X' : Y' : Z') = (X : Y : Z)$; that is, any element of an equivalence class can serve as its representative. In particular, if $Z \neq 0$, then $(X/Z^c, Y/Z^d, 1)$ is a representative of the projective point $(X : Y : Z)$, and in fact is the only representative with Z-coordinate equal to 1. Jacobian coordinates are denoted by $(X : Y : Z)$ where $c = 2$ and $d = 3$. The coordinate $(X : Y : Z)$, $Z \neq 0$, corresponds to the affine point $(X/Z^2, Y/Z^3)$ and the affine point (X, Y) corresponds to the Jacobi coordinate $(X : Y : 1)$. The projective form of the short Weierstrass equation defined over the field K is

$$E : Y^2 = X^3 + aXZ^4 + bZ^6.$$

The point at infinity corresponds to $(1 : 1 : 0)$,while the negative of $(X : Y : Z)$ is $(X : -Y : Z)$. In the following we present the formulas for the double and add operation on Jacobi elliptic curve points. In the remainder of this thesis we will use just the notation $(X : Y : Z)$ to denote a Jacobian coordinate, irrespective if we mean the actual representative or the projective point.

Point addition using mixed Jacobian-affine coordinates. Note that for our purpose it is enough to use formulas for mixed Jacobian-affine coordinates, since we can simply call the Miller Algorithm (see Algorithm 1) with affine coordinates. Within the Miller-Algorithm we just copy the parameter P and transform it to a Jacobian coordinate. After that, all add operations have to be computed on one Jacobian and one affine coordinate. For these we can perfectly apply the following formulas. Let $P = (X_1 : Y_1 : Z_1) \in E$, $Z_1 \neq 0$, and $Q = (X_2, Y_2)$, and suppose that $P \neq \pm Q$. Since $P = (X_1/Z_1^2 : Y_1/Z_2^3 : 1)$, we can use the addition formula for E in affine coordinates ((2.6)) to compute $P + Q = (X_3' : Y_3' : 1)$,

obtaining

$$X_3' = \left(\frac{Y_2 - \frac{Y_1}{Z_1^3}}{X_2 - \frac{X_1}{Z_1^2}}\right)^2 - \frac{X_1}{Z_1^2} - X_2 = \left(\frac{Y_2 Z_1^3 - Y_1}{(X_2 Z_1^2 - X_1)Z_1}\right)^2 - \frac{X_1}{Z_1^2} - X_2$$

and

$$Y_3' = \left(\frac{Y_2 - \frac{Y_1}{Z_1^3}}{X_2 - \frac{X_1}{Z_1^2}}\right)\left(\frac{X_1}{Z_1^2} - X_3'\right) - \frac{Y_1}{Z_1^3} = \left(\frac{Y_2 Z_1^3 - Y_1}{(X_2 Z_1^2 - X_1)Z_1}\right)\left(\frac{X_1}{Z_1^2} - X_3'\right) - \frac{Y_1}{Z_1^3}.$$

To eliminate denominators in the expression for X_3' and Y_3', we set $X_3 = X_3' \cdot Z_3^2$ and $Y_3 = Y_3' \cdot Z_3^3$ where $Z_3 = (X_2 Z_1^2 - X_1)Z_1$, and obtain the following formulas for computing $P + Q = (X_3 : Y_3 : Z_3)$ in Jacobian coordinates:

$$\begin{aligned} X_3 =& (Y_2 Z_1^3 - Y_1)^2 - (X_2 Z_1^2 - X_1)^2 (X_1 + X_2 Z_1^2) \\ Y_3 =& (Y_2 Z_1^3 - Y_1)[X_1 (X_2 Z_1^2 - X_1)^2 - X_3] - Y_1 (X_2 Z_1^2 - X_1)^3 \\ Z_3 =& (X_2 Z_1^2 - X_1) Z_1. \end{aligned} \tag{3.3}$$

Point doubling. Let $P = (X_1 : Y_1 : Z_1) \in E$, and suppose that $P \neq -P$. Since $P = (X_1/Z_1^2 : Y_1/Z_1^3 : 1)$, we can use the doubling formula for E in affine coordinates ((2.7)) to compute $[2]P = (X_3' : Y_3' : 1)$, obtaining

$$X_3' = \left(\frac{3\frac{X_1^2}{Z_1^4} + a}{2\frac{Y_1}{Z_1^3}}\right)^2 - 2\frac{X_1}{Z_1^2} = \frac{(3X_1^2 + aZ_1^4)^2 - 8X_1 Y_1^2}{4Y_1^2 Z_1^2}$$

and

$$Y_3' = \left(\frac{3\frac{X_1^2}{Z_1^4} + a}{2\frac{Y_1}{Z_1^3}}\right)\left(\frac{X_1}{Z_1^2} - X_3'\right) - \frac{Y_1}{Z_1^3} = \frac{3X_1^2 + aZ_1^4}{2Y_1 Z_1}\left(\frac{X_1}{Z_1^2} - X_3'\right) - \frac{Y_1}{Z_1^3}.$$

These formulas are derived by substituting the variables x_1, x_3 and y_1 in (2.7) by $X_1/Z_1^2, X_3'$ and Y_1/Z_1^3. To eliminate the denominators in the expression for X_3' and Y_3', we set $X_3 = X_3' \cdot Z_3^2$ and $Y_3 = Y_3' \cdot Z_3^3$ where $Z_3 = 2Y_1 Z_1$, and obtain the following formulas for computing $[2]P = (X_3 : Y_3 : Z_3)$ in Jacobian coordinates:

$$\begin{aligned} X_3 =& (3X_1^2 + aZ_1^4)^2 - 8X_1 Y_1^2 \\ Y_3 =& (3X_1^2 + aZ_1^4)(4X_1 Y_1^2 - X_3) - 8Y_1^4 \\ Z_3 =& 2Y_1 Z_1. \end{aligned} \tag{3.4}$$

3.2.2 Line Equations

During the Miller loop one has to evaluate the tangent $l_{T,T}$, the line $l_{T,P}$, such as the vertical lines $v_{[2]T}$ and v_{T+P} at a specific point. Due to denominator elimination we can ignore the calculation of the vertical lines [3], thus we only consider formulas for the line equations. In the following we give the line and tangent equations in affine coordinates and show how to transform the line equation into mixed Jacobian-affine coordinates and the tangent equation into Jacobian coordinates.

Line through T and P The equation for a line $l_{T,P}$ where $T = (x_1, y_1)$ and $P = (x_2, y_2)$ are affine coordinates is as follows:

$$l_{T,P}(x, y) = y - y_2 - \frac{y_2 - y_1}{x_2 - x_1} \cdot (x - x_1). \tag{3.5}$$

For mixed Jacobian-affine coordinates $T = (X_1 : Y_1 : Z_1)$ and $P = (X_2, Y_2)$ we get:

$$l_{T,P}(x, y) = y - Y_2 - \frac{Y_2 - \frac{Y_1}{Z_1^3}}{X_2 - \frac{X_1}{Z_1^3}}(x - X_2).$$

To eliminate the denominators we multiply the equation by $(X_2 - \frac{X_1}{Z_1^2})$ and Z_1^3. Note that these multiplications are just multiplications by constants that does not change the zeros of $l_{T,P}(x, y)$, e.g. the divisor of $l_{T,P}(x, y)$ does not change. Thus we get:

$$l_{T,P}(x, y) = (y - Y_2)(X_2 Z_1^2 - X_1)Z_1 - (Y_2 Z_1^3 - Y_1)(x - X_2). \tag{3.6}$$

Tangent at T The equation for a tangent $l_{T,T}$ where $T = (x_1, y_1)$ is an affine coordinates is as follows[1]:

$$l_{T,T}(x, y) = y - y_1 - \left(\frac{3x_1^2 + a}{2y_1}\right)(x - x_1). \tag{3.7}$$

For a Jacobian coordinate $T = (X_1 : Y_1 : Z_1)$ we get:

$$l_{T,T}(x, y) = y - \frac{Y_1}{Z_1^3} - \left(\frac{3\frac{X_1^2}{Y_1^4} + a}{2\frac{Y_1}{Z_1^3}}\right)(x - \frac{X_1}{Z_1^2}).$$

To eliminate denominators we multiply the equation by $2\frac{Y_1}{Z_1^3}$ and Z_1^6. Note, as before, these multiplications are just multiplications by constants that does

[1] The variable a belongs to $E : y^2 = x^3 + ax + b$

not change the divisor of $l_{T,T}(x, y)$. Thus we get:

$$l_{T,T}(x, y) = (yZ_1^3 - Y_1)(2Y_1) - (3X_1^2 + aZ_1^4)(xZ_1^2 - X_1). \tag{3.8}$$

3.2.3 Computing Point Operations and Line Equations

In this section we present algorithms that compute point addition and doubling as well as the line equations on Jacobian coordinates. Some algorithms are taken from *Explicit-Formulas-Database* that is due to Daniel J. Bernstein and Tanja Lange [5]. We will modify these algorithms such that their memory usage is reduced.

At first we will analyze the algorithms costs for arbitrary embedding degrees. In the next step we will consider two special elliptic curve families with embedding degree 6 and 12. For these explicit embedding degrees we will analyze the costs by applying the general analysis.

3.2.3.1 Computing $T + P$ and $l_{T,P}(Q)$

By looking at the formulas for computing $T + P$ and $l_{T,P}(Q)$ we note that this operations are independent of the curves parameters a and b (see (3.3) and (3.6)). Therefore, we can design an algorithm that computes $T+P$ and $l_{T,P}(Q)$ for any curve of the form $E : y^2 = x^3 + ax + b$.

Theorem 3.9. *Let $k \in \mathbb{N}$, $T = (X_1 : Y_1 : Z_1) \in \mathbb{F}_q^3$, $P = (X_2, Y_2) \in \mathbb{F}_q^2$, $Q = (X_4, Y_4) \in \mathbb{F}_{q^k}^2$ then computing $T + P$ and $l_{T,P}(Q)$ costs $\mathcal{A}_{Curve,Jac} = 12M_q + 6A_q$ and $\mathcal{L}_{Curve,Jac} = (2k)M_q + 1A_{q^k} + 2A_q$ with a space requirement of $S_{Curve,Jac,add} = 1S_{q^k} + 7S_q$.*

Proof. Algorithm 7 computes two results. At first it computes the add operation $T+P$ on Jacobian coordinates for elliptic curves of the form $E : y^2 = x^3+ax+b$. Second, it evaluates the line through T and P at the affine point Q. For this, the algorithm implements (3.3) and (3.6).

Algorithm 7 Jacobian-affine Point Adding and Line Evaluation

Require: $T = (X_1 : Y_1 : Z_1) \in \mathbb{F}_q^3, P = (X_2, Y_2) \in \mathbb{F}_q^2, Q = (X_4, Y_4) \in \mathbb{F}_{q^k}^2, E : y^2 = x^3 + ax + b$

Ensure: $T + P = (X_3 : Y_3 : Z_3) \in \mathbb{F}_q^3, l_{T,P}(Q) = L \in \mathbb{F}_{q^k}$

1: /* *** Point Adding *** */
2: $A \leftarrow Z_1^2$ ▷ $A = Z_1^2$
3: $B \leftarrow X_2 \cdot A$ ▷ $B = X_2 Z_1^2$
4: $C \leftarrow X_1 + B$ ▷ $C = X_1 + X_2 Z_1^2$
5: $u_1 \leftarrow B - X_1$ ▷ $u_1 = X_2 Z_1^2 - X_1$
6: $D \leftarrow u_1^2$ ▷ $D = (X_2 Z_1^2 - X_1)^2$
7: $C \leftarrow D \cdot C$ ▷ $C = (X_2 Z_1^2 - X_1)^2 \cdot (X_1 + X_2 Z_1^2)$
8: $t_0 \leftarrow A \cdot Z_1$ ▷ $t_0 = Z_1^3$
9: $t_0 \leftarrow Y_2 \cdot t_0$ ▷ $t_0 = Y_2 Z_1^3$
10: $E \leftarrow t_0 - Y_1$ ▷ $E = Y_2 Z_1^3 - Y_1$
11: $t_0 \leftarrow E^2$ ▷ $t_0 = (Y_2 Z_1^3 - Y_1)^2$
12: $C \leftarrow t_0 - C$ ▷ $C = X_3 = (Y_2 Z_1^3 - Y_1)^2 - (X_2 Z_1^2 - X_1)^2 \cdot (X_1 + X_2 Z_1^2)$
13: $t_0 \leftarrow X_1 \cdot D$ ▷ $t_0 = X_1 (X_2 Z_1^2 - X_1)^2$
14: $t_0 \leftarrow t_0 - C$ ▷ $t_0 = X_1 (X_2 Z_1^2 - X_1)^2 - X_3$
15: $B \leftarrow E \cdot t_0$ ▷ $B = (Y_2 Z_1^3 - Y_1) \cdot [X_1 (X_2 Z_1^2 - X_1)^2 - X_3]$
16: $t_0 \leftarrow D \cdot u_1$ ▷ $t_0 = (X_2 Z_1^2 - X_1)^3$
17: $t_0 \leftarrow Y_1 \cdot t_0$ ▷ $t_0 = Y_1 (X_2 Z_1^2 - X_1)^3$
18: $Y_3 \leftarrow B - t_0$ ▷ $Y_3 = (Y_2 Z_1^3 - Y_1) \cdot [X_1 (X_2 Z_1^2 - X_1)^2 - X_3] - Y_1 (X_2 Z_1^2 - X_1)^3$
19: $Z_3 \leftarrow u_1 \cdot Z_1$ ▷ $Z_3 = (X_2 Z_1^2 - X_1) Z_1$
20: /* *** Line Evaluation *** */
21: $L \leftarrow Y_4 - Y_2$ ▷ $L = Y_4 - Y_2$
22: $L \leftarrow L \cdot Z_3$ ▷ $L = (Y_4 - Y_2) Z_3$
23: $T_0 \leftarrow X_4 - X_2$ ▷ $T_0 = X_4 - X_2$
24: $T_0 \leftarrow T_0 \cdot E$ ▷ $T_0 = (X_4 - X_2)(Y_2 Z_1^3 - Y_1)$
25: $L \leftarrow L - T_0$ ▷ $L = (Y_4 - Y_2) Z_3 - (X_4 - X_2)(Y_2 Z_1^3 - Y_1)$
26: $X_3 \leftarrow C$
27: **return** $(X_3, Y_3, Z_3), L$

Correctness At the end we have

$$\begin{aligned}
X_3 =& (Y_2 Z_1^3 - Y_1)^2 - (X_2 Z_1^2 - X_1)^2 \cdot (X_1 + X_2 Z_1^2), \\
Y_3 =& (Y_2 Z_1^3 - Y_1) \cdot [X_1 (X_2 Z_1^2 - X_1)^2 - X_3] - Y_1 (X_2 Z_1^2 - X_1)^3, \\
Z_3 =& (X_2 Z_1^2 - X_1) Z_1 \text{ and} \\
L =& (Y_4 - Y_2) Z_3 - (X_4 - X_2)(Y_2 Z_1^3 - Y_1) \\
=& (Y_4 - Y_2)(X_2 Z_1^2 - X_1) Z_1 - (X_4 - X_2)(Y_2 Z_1^3 - Y_1),
\end{aligned}$$

which exactly match (3.3) and (3.6).

Complexity For computing the add operation $T + P$ the algorithm needs $\mathcal{A}_{Curve,Jac} = 12M_q + 5A_q$. Evaluating the line through T and P at the point Q costs $\mathcal{L}_{Curve,Jac} = (2k)M_q + 1A_{q^k} + 2A_q$, where the operations in line 22 and 24 are scalar multiplications which cost kM_q each, since multiplying an $\mathbb{F}_q$ and $\mathbb{F}_{q^k}$ element can be implemented by k multiplications over $\mathbb{F}_q$. The operations in line 21 and 23 are simple subtractions over $\mathbb{F}_q$ and the subtraction in line 25 costs $1A_{q^k}$. The algorithm requires space for the variables A, B, C, D, E, t_0, u_1 of type $\mathbb{F}_q$ and T_0 of type $\mathbb{F}_{q^k}$. Thus the total space requirement is $S_{Curve,Jac,add} = 1S_{q^k} + 7S_q$. □

3.2.3.2 Computing $[2]P$ and $l_{P,P}(Q)$

Computing $[2]P$ and $l_{P,P}(Q)$ depends on the curves parameter a (see 3.4 and 3.8). Most families of elliptic curves have $a = -3$ or $a = 0$. In the following we present algorithms for these two cases. We begin with the case $a = -3$.

Theorem 3.10. *Let $P = (X_1 : Y_1 : Z_1) \in \mathbb{F}_q^3$, $Q = (X_2, Y_2) \in \mathbb{F}_{q^k}^2$ then computing $[2]P$ and $l_{P,P}(Q)$ costs $\mathcal{D}_{Curve,Jac} = 8M_q + 16A_q$ and $\mathcal{T}_{Curve,Jac} = (3k)M_q + 1M_q + 1A_{q^k} + 3A_q$ with a space requirement of $S_{Curve,Jac,dbl} = 1S_{q^k} + 6S_q$.*

Proof. Algorithm 8 computes two results. At first it computes the double operation $[2]P$ on Jacobian coordinates for elliptic curves of the form $E : y^2 = x^3 - 3x + b$. Second, it evaluates the tangent at P at the affine point Q. The double part of Algorithm 8 is due to Daniel Bernstein [5] and is denoted by *dbl-2001-b*. This is, the algorithm implements (3.4) and (3.8) for $a = -3$.

Algorithm 8 Jacobian Point Doubling (dbl-2001-b) and Tangent Evaluation

Require: $P = (X_1 : Y_1 : Z_1) \in \mathbb{F}_q^3, Q = (X_2, Y_2) \in \mathbb{F}_{q^k}^2, E : y^2 = x^3 - 3x + b$

Ensure: $[2]P = (X_3 : Y_3 : Z_3) \in \mathbb{F}_q^3, l_{P,P}(Q) = L \in \mathbb{F}_{q^k}$

1: /* *** Point Doubling *** */
2: $A \leftarrow Z_1^2$ ▷ $A = Z_1^2$
3: $B \leftarrow Y_1^2$ ▷ $B = Y_1^2$
4: $C \leftarrow X_1 \cdot B$ ▷ $C = X_1 Y_1^2$
5: $t_0 \leftarrow X_1 - A$ ▷ $t_0 = X_1 - Z_1^2$
6: $t_1 \leftarrow X_1 + A$ ▷ $t_1 = X_1 + Z_1^2$
7: $t_0 \leftarrow t_0 \cdot t_1$ ▷ $t_0 = (X_1 - Z_1^2)(X_1 + Z_1^2)$
8: $D \leftarrow t_0 + t_0, D \leftarrow D + t_0$ ▷ $D = 3(X_1 - Z_1^2)(X_1 + Z_1^2)$
9: $t_0 \leftarrow D^2$ ▷ $t_0 = [3(X_1 - Z_1^2)(X_1 + Z_1^2)]^2$
10: $t_1 \leftarrow C + C, t_2 \leftarrow t_1 + t_1, t_1 \leftarrow t_2 + t_2$ ▷ $t_1 = 8X_1Y_1^2, t_2 = 4X_1Y_1^2$
11: $C \leftarrow t_0 - t_1$ ▷ $C = X_3 = [3(X_1 - Z_1^2)(X_1 + Z_1^2)]^2 - 8X_1Y_1^2$
12: $t_0 \leftarrow Y_1 + Z_1$ ▷ $t_0 = Y_1 + Z_1$
13: $t_0 \leftarrow t_0^2$ ▷ $t_0 = (Y_1 + Z_1)^2$
14: $t_0 \leftarrow t_0 - B$ ▷ $t_0 = (Y_1 + Z_1)^2 - Y_1^2$
15: $Z_3 \leftarrow t_0 - A$ ▷ $Z_3 = (Y_1 + Z_1)^2 - Y_1^2 - Z_1^2$
16: $t_0 \leftarrow t_2 - C$ ▷ $t_0 = 4X_1Y_1^2 - X_3$
17: $t_1 \leftarrow B^2$ ▷ $t_1 = Y_1^4$
18: $t_1 \leftarrow t_1 + t_1, t_1 \leftarrow t_1 + t_1, t_1 \leftarrow t_1 + t_1$ ▷ $t_1 = 8Y_1^4$
19: $t_2 \leftarrow D \cdot t_0$ ▷ $t_2 = 3(X_1 - Z_1^2)(X_1 + Z_1^2)(4X_1Y_1^2 - X_3)$
20: $Y_3 \leftarrow t_2 - t_1$ ▷ $Y_3 = 3(X_1 - Z_1^2)(X_1 + Z_1^2)(4X_1Y_1^2 - X_3) - 8Y_1^4$
21: /* *** Tangent Evaluation *** */
22: $t_2 \leftarrow Z_3 \cdot A$ ▷ $t_2 = Z_3Z_1^2$
23: $t_2 \leftarrow t_2 \cdot Y_2$ ▷ $t_2 = Z_3Z_1^2Y_2$
24: $L \leftarrow t_2 - B$ ▷ $L = Z_3Z_1^2Y_2 - Y_1^2$
25: $L \leftarrow L - B$ ▷ $L = Z_3Z_1^2Y_2 - 2Y_1^2$
26: $t_2 \leftarrow X_2 \cdot A$ ▷ $t_2 = X_2Z_1^2$
27: $t_2 \leftarrow t_2 - X_1$ ▷ $t_2 = X_2Z_1^2 - X_1$
28: $t_2 \leftarrow D \cdot t_2$ ▷ $t_2 = 3(X_1 - Z_1^2)(X_1 + Z_1^2)(X_2Z_1^2 - X_1)$
29: $L \leftarrow L - t_2$ ▷ $L = Z_3Z_1^2Y_2 - 2Y_1^2 - 3(X_1 - Z_1^2)(X_1 + Z_1^2)(X_2Z_1^2 - X_1)$
30: $X_3 \leftarrow C$
31: **return** $(X_3 : Y_3 : Z_3), L$

Correctness At the end we have

$$\begin{aligned}
X_3 =& [3(X_1 - Z_1^2)(X_1 + Z_1^2)]^2 - 8X_1Y_1^2 \\
=& (3X_1^2 - 3Z_1^4)^2 - 8X_1Y_1^2, \\
Y_3 =& 3(X_1 - Z_1^2)(X_1 + Z_1^2)(4X_1Y_1^2 - X_3) - 8Y_1^4 \\
=& (3X_1^2 - 3Z_1^4)(4X_1Y_1^2 - X_3) - 8Y_1^4, \\
Z_3 =& (Y_1 + Z_1)^2 - Y_1^2 - Z_1^2 \\
=& 2Y_1Z_1 \text{ and} \\
L =& Z_3Z_1^2Y_2 - 2Y_1^2 - 3(X_1 - Z_1^2)(X_1 + Z_1^2)(X_2Z_1^2 - X_1) \\
=& 2Y_1Z_1Z_1^2Y_2 - 2Y_1^2 - (3X_1^2 - 3Z_1^4)(X_2Z_1^2 - X_1) \\
=& (Z_1^3Y_2 - Y_1)(2Y_1) - (3X_1^2 - 3Z_1^4)(X_2Z_1^2 - X_1)
\end{aligned}$$

which exactly match (3.4) and (3.8).

Complexity For computing $[2]P$ Algorithm 8 needs $8M_q$ and $16A_q$. For evaluating $l_{P,P}(Q)$ the algorithm needs $(3k)M_{q^k} + 1M_q + 1A_{q^k} + 3A_q$ where the operations in line 22, 24-25 and 27 are counted as operations over $\mathbb{F}_q$, the multiplications in line 23, 26 and 28 are scalar multiplications which cost $(3k)M_q$ and the subtraction in line 29 costs $1A_{q^k}$. The algorithm uses the variables A, B, C, D, t_0, t_1 of type $\mathbb{F}_q$ and t_2 of type $\mathbb{F}_{q^k}$. Thus the total space requirement is $S_{Curve,Jac,dbl} = 1S_{q^k} + 6S_q$. □

Now we consider the algorithm for the case $a = 0$.

Theorem 3.11. *Let $P = (X_1 : Y_1 : Z_1) \in \mathbb{F}_q^3$, $Q = (X_2, Y_2) \in \mathbb{F}_{q^k}^2$ then computing $[2]P$ and $l_{P,P}(Q)$ costs $\mathcal{D}_{Curve,Jac} = 7M_q + 14A_q$ and $\mathcal{T}_{Curve,Jac} = (3k)M_q + 2M_q + 3A_{q^k} + 3A_q$ with a space requirement of $S_{Curve,Jac,dbl} = 2S_{q^k} + 8S_q$.*

Proof. Algorithm 9 computes two results. At first it computes the double operation $[2]P$ on Jacobian coordinates for elliptic curves of the form $E : y^2 = x^3 + b$. Second, it evaluates the tangent at P at the affine point Q. This is, the algorithm implements (3.4) and (3.8). The double part of Algorithm 9 is due to Tanja Lange [5] which is denoted as *dbl-2009-l* within the Explicit-Formula Database (EFD) .

Algorithm 9 Jacobian Point Doubling (dbl-2009-l) and Tangent Evaluation

Require: $P = (X_1 : Y_1 : Z_1) \in \mathbb{F}_q^3, Q = (X_2, Y_2) \in \mathbb{F}_{q^k}^2, E : y^2 = x^3 + b$

Ensure: $[2]P = (X_3 : Y_3 : Z_3) \in \mathbb{F}_q^3, l_{P,P}(Q) = L \in \mathbb{F}_{q^{12}}$

/* *** Point Doubling *** */
$A \leftarrow X_1^2$ ▷ $A = X_1^2$
$B \leftarrow Y_1^2$ ▷ $B = Y_1^2$
$C \leftarrow B^2$ ▷ $C = Y_1^4$
$t_0 \leftarrow X_1 + B$ ▷ $t_0 = X_1 + Y_1^2$
$t_0 \leftarrow t_0^2$ ▷ $t_0 = (X_1 + Y_1^2)^2$
$t_0 \leftarrow t_0 - A$ ▷ $t_0 = (X_1 + Y_1^2)^2 - X_1^2$
$t_0 \leftarrow t_0 - C$ ▷ $t_0 = (X_1 + Y_1^2)^2 - X_1^2 - Y_1^4$
$D \leftarrow t_0 + t_0$ ▷ $D = 2[(X_1 + Y_1^2)^2 - X_1^2 - Y_1^4]$
$E \leftarrow A + A, E \leftarrow E + A$ ▷ $E = 3X_1^2$
$F \leftarrow E^2$ ▷ $F = (3X_1^2)^2$
$t_0 \leftarrow D + D$ ▷ $t_0 = 4[(X_1 + Y_1^2)^2 - X_1^2 - Y_1^4]$
$F \leftarrow F - t_0$ ▷ $F = X_3 = (3X_1^2)^2 - 4[(X_1 + Y_1^2)^2 - X_1^2 - Y_1^4]$
$t_0 \leftarrow D - F$ ▷ $t_0 = 2[(X_1 + Y_1^2)^2 - X_1^2 - Y_1^4] - X_3$
$t_1 \leftarrow C + C, t_1 \leftarrow t_1 + t_1, t_1 \leftarrow t_1 + t_1$ ▷ $t_1 = 8Y_1^4$
$t_0 \leftarrow E \cdot t_0$ ▷ $t_0 = 3X_1^2\{2[(X_1 + Y_1^2)^2 - X_1^2 - Y_1^4] - X_3\}$
$Y_3 \leftarrow t_0 - t_1$ ▷ $Y_3 = 3X_1^2\{2[(X_1 + Y_1^2)^2 - X_1^2 - Y_1^4] - X_3\} - 8Y_1^4$
$t_0 \leftarrow Y_1 \cdot Z_1$ ▷ $t_0 = Y_1 Z_1$
$E \leftarrow 2 \cdot t_0$ ▷ $E = Z_3 = 2Y_1 Z_1$
/* *** Tangent Evaluation *** */
$C \leftarrow Z_1^2$ ▷ $C = Z_1^2$
$t_0 \leftarrow Z_3 \cdot C$ ▷ $t_0 = Z_3 Z_1^2$
$L \leftarrow t_0 \cdot Y_2$ ▷ $L = Z_3 Z_1^2 Y_2$
$B \leftarrow B + B$ ▷ $B = 2Y_1^2$
$L \leftarrow L - B$ ▷ $L = Z_3 Z_1^2 Y_2 - 2Y_1^2$
$T_2 \leftarrow C \cdot X_2$ ▷ $T_2 = Z_1^2 X_2$
$T_2 \leftarrow T_2 - X_1$ ▷ $T_2 = Z_1^2 X_2 - X_1$
$T_2 \leftarrow T_2 \cdot A$ ▷ $T_2 = (Z_1^2 X_2 - X_1) X_1^2$
$T_3 \leftarrow T_2 + T_2$ ▷ $T_3 = 2(Z_1^2 X_2 - X_1) X_1^2$
$T_3 \leftarrow T_3 + T_2$ ▷ $T_3 = 3(Z_1^2 X_2 - X_1) X_1^2$
$L \leftarrow L - T_3$ ▷ $L = Z_3 Z_1^2 Y_2 - 2Y_1^2 - 3(Z_1^2 X_2 - X_1) X_1^2$
$X_3 \leftarrow F, Z_3 \leftarrow E$
return $(X_3 : Y_3 : Z_3), L$

Correctness At the end we have

$$\begin{aligned}
X_3 =&(3X_1^2)^2 - 4[(X_1 + Y_1^2)^2 - X_1^2 - Y_1^4] \\
=&(3X_1^2)^2 - 8X_1Y_1^2, \\
Y_3 =&3X_1^2\{2[(X_1 + Y_1^2)^2 - X_1^2 - Y_1^4] - X_3\} - 8Y_1^4 \\
=&3X_1^2(4X_1Y_1^2 - X_3) - 8Y_1^4, \\
Z_3 =&2Y_1Z_1 \text{ and} \\
L =&Z_3Z_1^2Y_2 - 2Y_1^2 - 3(Z_1^2X_2 - X_1)X_1^2 \\
=&(Z_1^3Y_2 - Y_1)(2Y_1) - 3(Z_1^2X_2 - X_1)X_1^2,
\end{aligned}$$

which exactly match (3.4) and (3.8).

Complexity For the double operation the algorithm needs $\mathcal{D}_{Curve,Jac} = 7M_q + 14A_q$. Evaluating the tangent costs $\mathcal{T}_{Curve,Jac} = (3k)M_q + 2M_q + 3A_{q^k} + 3A_q$, where the lines 21-22, 24-25 and 27 are operations over the field $\mathbb{F}_q$, the lines 23, 26 and 28 are scalar multiplications which cost $(3k)(M_q)$. The operations in line 29-31 are operations over $\mathbb{F}_{q^k}$. The algorithm requires space for the variables $A, B, C, D, E, F, t_0, t_1$ and T_2,T_3 of the type $\mathbb{F}_q$ respectively $\mathbb{F}_{q^k}$. This results in a total space requirement of $S_{Curve,Jac,dbl} = 2S_{q^k} + 8S_q$. □

3.2.4 MNT and BN Curves

In this section we introduce two families of curves: The MNT-curves introduced by Miyaji, Nakabayashi and Takano in [23] and the BN-curves introduced by Barreto and Naehrig in [4]. We choose these curves since they are ordinary curves of prime order with embedding degree 6 and 12. We will analyse the costs for these special curves by applying the analyses of the former section.

3.2.4.1 MNT Curve

MNT-curves are ordinary elliptic curves of prime order and have embedding degree $k = 3, 4, 6$. Since we are just interested in the case $k = 6$ the following definition omits the properties for the other embedding degrees.

Definition 3.12 (MNT-Curves)**.** *[18, Ch. 2] Let $E : y^2 = x^3 + ax + b$ be an ordinary elliptic curve with trace $t(l)$, prime order $r(l)$ and $p(l)$ the characteristic of $\mathbb{F}_p$ parametrized by:*

$$\begin{aligned}
t(l) =&1 + 2l \\
r(l) =&4l^2 \pm 2l + 1 \\
p(l) =&4l^2 + 1 > 64
\end{aligned}$$

where $l \in \mathbb{N}$. Such a curve is called an MNT-curve and has embedding degree $k = 6$.

Analysis We can now analyse how expensive it is to compute the point operations and line evaluations on MNT curves. For this we have to set the parameter Q of Algorithm 7 and Algorithm 8 to a point from $E(\mathbb{F}_{p^6})$. This is, $Q = (x, y)$ with $x, y \in \mathbb{F}_{p^6}$ where in the worst case each coefficient of x and y is unequal to 0. We can also choose Q from a special subgroup of $E(\mathbb{F}_{p^6})$ such that not every coefficient of x and y is unequal to 0. Since multiplications by 0 do not have to be computed we will get more efficient algorithms. We can define such subgroups by working with twists on elliptic curves. For a definition of twists see [18, Ch. 2].

An MNT curve admits a twist of degree 2 over $\mathbb{F}_{p^3}$, namely $E'(\mathbb{F}_{p^3}) : T^2y^2 = x^3 + ax + b$ [18, Ch. 2]. Remember that $T = x \mod P(x)$, see (2.1). The isomorphism $E'(\mathbb{F}_{p^3})[r] \to E(\mathbb{F}_{p^6})[r]$ is given by $(x, y) \mapsto (x, yT)$. This is, a point $Q' = (x, y) \in E(\mathbb{F}_{p^3})[r]$ can be mapped to a point $Q = (x, yT) \in E(\mathbb{F}_{p^6})[r]$, but x and yT each has only 3 instead of 6 coefficients unequal to 0. Thus, the isomorphism maps to a subgroup of $E(\mathbb{F}_{p^6})[r]$. Since we do not count multiplications by 0 we can save some multiplications in Algorithm 7 and Algorithm 8 by starting them on a point $Q = (x, yT) \in E(\mathbb{F}_{p^6})[r]$. This we will analyse next.

Lemma 3.13. *Let $T = (X_1 : Y_1 : Z_1) \in \mathbb{F}_p^3$, $P = (X_2, Y_2) \in \mathbb{F}_p^2$, $Q = (X_4, Y_4) = (x, yT) \in \mathbb{F}_{p^6}^2$, $(x, y) \in \mathbb{F}_{p^3}^2$ then computing $T + P$ and $l_{T,P}(Q)$ costs $\mathcal{A}_{MNT,Jac} = 12M_p + 6A_p$ and $\mathcal{L}_{MNT,Jac} = 7M_p + 8A_p$ with a space requirement of $S_{MNT,Jac,add} = 1S_{p^6} + 7S_p$.*

Proof. We run Algorithm 7 with input T, P and Q as defined above. The point addition analysis is the same as in Theorem 3.9. Therefore we have only to consider the line evaluation part. We have that $X_4 = x$ and $Y_4 = yT$ where each variable has only 3 coefficients unequal to 0. Thus, the subtractions in line 21 and 23 cost just 2 additions over $\mathbb{F}_p$. After computing line 21 and 23 the variable L has at most 4 coefficients unequal to 0 and the variable T_0 has at most 3 coefficients unequal to 0. Thus the multiplication in line 22 costs at most $4M_p$ and the multiplication in line 24 costs at most $3M_p$. The subtraction in line 25 costs at most $6A_p$. All in all, we have $\mathcal{L}_{MNT,Jac} = 7M_p + 8A_p$. According to Theorem 3.9 the space requirement is $S_{MNT,Jac,add} = 1S_{p^6} + 7S_p$. □

Lemma 3.14. *Let $P = (X_1 : Y_1 : Z_1) \in \mathbb{F}_p^3$, $Q = (X_2, Y_2) = (x, yT) \in \mathbb{F}_{p^6}^2$, $(x, y) \in \mathbb{F}_{p^3}^2$ then computing $[2]P$ and $l_{P,P}(Q)$ costs $\mathcal{D}_{MNT,Jac} = 8M_p + 16A_p$ and $\mathcal{T}_{MNT,Jac} = 7M_p + 9A_p$ with a space requirement of $S_{MNT,Jac,dbl} = 1S_{p^6} + 6S_p$.*

Proof. We run algorithm Algorithm 8 with input P and Q as defined above. The point doubling analysis is the same as in Theorem 3.10. Therefore we have only to consider the tangent evaluation part. Again, we have $X_4 = x$ and $Y_4 = yT$ where each variable has only 3 coefficients unequal to 0. Thus, the multiplications in line 23 and 26 cost $6M_p$ and the subtractions in line 24, 25 and 27 cost $3A_p$. The multiplication in line 28 costs $1M_p$ and the subtraction in line 29 costs $6A_p$. The multiplication in line 22 costs $1M_p$. All in all we have $\mathcal{T}_{MNT,Jac} = 7M_p + 9A_p$. According to Theorem 3.10 the space requirement is $S_{MNT,Jac,dbl} = 1S_{p^6} + 6S_p$. □

3.2.4.2 BN Curve

BN-curves are ordinary elliptic curves of prime order and have embedding degree $k = 12$.

Definition 3.15 (BN-curves)**.** *[10] Let $E : y^2 = x^3 + b$, with $b \neq 0$ be an elliptic curve with trace $t(l)$, prime order $r(l)$ and $p(l)$ the characteristic of $\mathbb{F}_p$ parametrized by:*

$$\begin{aligned} t(l) &= 6l^2 + 1 \\ r(l) &= 36l^4 - 36l^3 + 18l^2 - 6l + 1 \\ p(l) &= 36l^4 - 36l^3 + 24l^2 - 6l + 1, \end{aligned}$$

where $l \in \mathbb{N}$. Such a curve is called a Barreto-Naehrig curve and has embedding degree $k = 12$.

Note that this curve is completely defined by the parameter l. Barreto and Naehrig also presents an efficient algorithm in [4] how to generate b, t, r and p where the size of the curves order can be chosen.

Analysis For BN curves we can do the same as for MNT curves, instead of choosing Q from the group $E(\mathbb{F}_{p^{12}})$ we will consider a special subgroup.

A BN curve admits a twist of degree 6 over $\mathbb{F}_{p^2}$, namely $E'(\mathbb{F}_{p^2}) : y^2 = x^3 + 3/\xi$. The isomorphism $E'(\mathbb{F}_{p^2})[r] \to E(\mathbb{F}_{p^{12}})[r]$ is given by $(x, y) \mapsto (xU^2, yU^3)$ [18, Ch. 2, Ch. 7]. Remember that $U = z \mod R(z)$, see (2.3). This means, a point $Q' = (x, y) \in E'(\mathbb{F}_{p^2})[r]$ can be transformed to a point $Q = (xU^2, yU^3) \in E(\mathbb{F}_{p^{12}})[r]$, where xU^2 and yU^3 each has only 2 instead of 12 coefficients unequal to 0 . Thus, by running Algorithm 7 and Algorithm 9 on a point $Q = (xU^2, yU^3) \in E(\mathbb{F}_{p^{12}})[r]$ we can save some multiplications. This we will analyse next.

Lemma 3.16. *$T = (X_1 : Y_1 : Z_1) \in \mathbb{F}_p^3$, $P = (X_2, Y_2) \in \mathbb{F}_p^2$, $Q = (X_4, Y_4) = (xU^2, yU^3) \in \mathbb{F}_{p^{12}}^2$, $(x, y) \in \mathbb{F}_{p^2}^2$ then computing $T + P$ and $l_{T,P}(Q)$ costs $\mathcal{A}_{BN,Jac} = 12M_p + 6A_p$ and $\mathcal{L}_{BN,Jac} = 6M_p + 8A_p$ with a space requirement of $S_{BN,Jac,add} = 1S_{p^{12}} + 7S_p$.*

Proof. We run Algorithm 7 with the input T, P and Q as defined above. The analysis for the point addition part is as in Theorem 3.9. Therefore we have only to consider the line evaluation part. We have that $X_4 = xU^2$ and $Y_4 = yU^3$ where each variable has only 2 coefficients unequal to 0. Thus, the subtractions in line 21 and 23 cost just 2 additions over $\mathbb{F}_p$. After computing line 21 and 23, L and T_0 have at most 3 coefficients unequal to 0. Thus, the multiplications in line 22 and 24 cost at most 6 multiplications over $\mathbb{F}_p$ and the subtraction in line 25 costs at most 6 additions over $\mathbb{F}_p$. All in all, we have $\mathcal{L}_{BN,Jac} = 6M_p + 8A_p$. According to Theorem 3.9 the space requirement is $S_{BN,Jac,add} = 1S_{p^{12}} + 7S_p$. □

Lemma 3.17. *Let $P = (X_1 : Y_1 : Z_1) \in \mathbb{F}_p^3$, $Q = (X_2, Y_2) = (xU^2, yU^3) \in \mathbb{F}_{p^{12}}^2$, $(x, y) \in \mathbb{F}_{p^2}^2$ then computing $[2]P$ and $l_{P,P}(Q)$ costs $\mathcal{D}_{BN,Jac} = 7M_p + 14A_p$ and $\mathcal{T}_{BN,Jac} = 9M_p + 15A_p$ with a space requirement of $S_{BN,Jac,dbl} = 2S_{p^{12}} + 8S_p$.*

Proof. We run Algorithm 9 with the input P and Q as defined above. The analysis for the point doubling part is as in Theorem 3.11. Therefore we have only to consider the tangent evaluation part. We have that $X_2 = xU^2$ and $Y_2 = yU^3$ where each variable has only 2 coefficients unequal to 0. Thus, the multiplications in line 23 and 26 cost just 4 multiplications over $\mathbb{F}_p$ and the multiplication in line 28 costs at most 3 multiplications over $\mathbb{F}_p$, since T_2 has at most 3 coefficients unequal to 0. The subtractions in line 25 and 27 cost 2 additions over $\mathbb{F}_p$, the additions in the lines 29-30 cost at most 6 additions over $\mathbb{F}_p$ and the subtraction in line 31 costs at most 6 additions over $\mathbb{F}_p$. The operations in the lines 21, 22 and 24 cost $2M_p + 1A_p$. All in all, we have $\mathcal{T}_{BN,Jac} = 9M_p + 15A_p$. According to Theorem 3.11 we have a total space requirement of $S_{BN,Jac,dbl} = 2S_{p^{12}} + 8S_p$. □

3.3 Bilinear Pairing Computation

This section analyses the costs of the Tate pairing. In particular we concentrate on the Miller operation of the pairing and we do not analyze the costs of the final exponentiation. We denote the costs as follows: $\mathcal{P}_{Tate,r,k}$ is the number of operations of the bilinear pairing t_r (see Definition 2.28) of embedding degree k. The costs of the final exponentiation are denoted by $\mathcal{E}$.

3.3.1 Tate Pairing

In the following we will use the notations $|r|$ and $h(r)$ which denote the description length of r's binary representation and the Hamming weight of r, see also Section 2.1.

Theorem 3.18. *Let $P = (X_1, Y_1) \in \mathbb{F}_q^2$, $Q = (X_2, Y_2) \in \mathbb{F}_{q^k}^2$ then computing $t_r(P, Q)$ costs*

$$\begin{aligned}\mathcal{P}_{Tate,r,k} =& (|r| - 1) \cdot (\mathcal{D}_{Curve,Coords} + \mathcal{T}_{Curve,Coords} + 2M_{q^k}) + \\ &+ (h(r) - 2) \cdot (\mathcal{A}_{Curve,Coords} + \mathcal{L}_{Curve,Coords} + 1M_{q^k}) + \mathcal{E}\end{aligned}$$

and requires $S_{Tate,k} = 2S_{q^k} + 2S_q + \max(S_{Curve,Jac,add}, S_{Curve,Jac,dbl}, S_{Mul,q^k})$ additional space, where r is the curves prime order and k specifies the embedding degree.

Proof. The following algorithm is taken from [18, Ch. 7]. We omitted the detailed strategy how to implement the final exponentiation, since this is not a part of this thesis.

Algorithm 10 Reduced Tate Pairing

Require: $P = (X_1, Y_1) \in \mathbb{F}_q^2, Q = (X_2, Y_2) \in \mathbb{F}_{q^k}^2$, curve prime order $r \in \mathbb{N}$ with $r|(q^k - 1)$

Ensure: $f = t_r(P, Q) \in \mathbb{F}_{q^k}^*$

1: Write r in binary: $r = \sum_{i=0}^{L-1} r_i 2^i$.
2: $T \leftarrow P; f \leftarrow 1$
3: /* *** Miller Operation *** */
4: **for** i from $L-2$ downto 0 **do**
5: $\quad L \leftarrow l_{T,T}(Q)$
6: $\quad T \leftarrow 2T$
7: $\quad f \leftarrow f^2$
8: $\quad f \leftarrow f \cdot L$
9: $\quad$ **if** $r_i = 1$ and $i \neq 0$ **then**
10: $\quad\quad L \leftarrow l_{T,P}(Q)$
11: $\quad\quad T \leftarrow T + P$
12: $\quad\quad f \leftarrow f \cdot L$
13: /* *** Final Exponentiation *** */
14: $f \leftarrow f^{(p^k-1)/r}$
15: **return** f

The algorithm consists of the parts *Miller Operation* and *Final Exponentiation*. We denote the latter parts complexity by $\mathcal{E}$ and will not analyze it further.

Within the Miller Operation the lines 5-8 are executed $|r| - 1$ times, since the leading 1 of the binary representation of r is skipped. The number of executions of the `if`-block depends on the Hamming weight of r. Since (1) r is an odd prime, the least significant bit of r is always 1 which makes the `if`-condition `FALSE` and (2) the leading bit of r is always skipped, the `if`-block is executed $h(r) - 2$ times.

The lines 5-8 consist of a tangent evaluation, a point doubling and two multiplications over the field $\mathbb{F}_{q^k}$ and cost $\mathcal{T}_{Curve,Coords} + \mathcal{D}_{Curve,Coords} + 2M_{q^k}$. The lines 10-12 consist of a line evaluation, a point addition and one multiplication over the field $\mathbb{F}_{q^k}$ and cost $\mathcal{A}_{Curve,Coord} + \mathcal{L}_{Curve,Coords} + 1M_{q^k}$. This results in the overall costs stated in the theorem.

The algorithm uses the variables $f, L \in \mathbb{F}_{q^k}$ and $T \in \mathbb{F}_q^2$. Furthermore we have to consider the space requirement of the sub routines executed in line 5-8 and 10-12. Since the sub routines are executed sequentially and memory is reusable, we have only to add up the space requirement of the sub routine with the highest space requirement. This is, we consider $\max(S_{Curve,Jac,add}, S_{Curve,Jac,dbl}, S_{Mul,q^k})$. This results in a total space requirement of

$$S_{Tate,k} = 2S_{q^k} + 2S_q + \max(S_{Curve,Jac,add}; S_{Curve,Jac,dbl}; S_{Mul,q^k}).$$

□

4 Practical Evaluation

This chapter uses the former analyses to calculate the costs for real world implementations. We calculate costs for different sets of parameters. The parameters are the particular prime p and the corresponding curve order r, the embedding degree k and the type of curve, and last but not least the specific pairing. We structure this chapter according to the parameter sets. We will not analyze the final exponentiation of the pairings. More information on the final exponentiation for pairings over ordinary elliptic curves can be found in [28] and [18, Ch. 7].

4.1 Tate Pairing

In the following we analyze the costs of the reduced Tate Pairing for the embedding degrees $k = 6, 12$.

4.1.1 Embedding degree $k = 6$, Security level 80 bit

We consider the elliptic curve

$$\begin{aligned} E_1 : \quad y^2 = &\quad x^3 + ax + b \text{ with} \\ a = &\quad -3 \text{ and} \\ b = &\quad \texttt{0x299CE219B7B01348FC2B5007B6AB1EE1005676F7}, \end{aligned}$$

which is together with $l = \texttt{0x5ECF0CB24C7A405A7495}$ an MNT-curve with embedding degree $k = 6$. This particular l is from [27] and it generates the 160-bit prime

$$p = p(l) \quad = \quad \texttt{0x8C72D321E48AA1419B22F914CB43C112B76D7AE5}.$$

The curve $E_1(\mathbb{F}_p)$ has prime order

$$r \quad = \quad \texttt{0x8C72D321E48AA1419B23B6B2E4A85A073822640F}$$

with $h(r) = 67$. Note, the curve E_1 admits a quadratic twist over $\mathbb{F}_{p^3}$, namely

$$E' : \quad T^2 y^2 = \quad x^3 + ax + b.$$

Remember that the isomorphism $E'(\mathbb{F}_{p^3})[r] \to E_1(\mathbb{F}_{p^6})[r]$ is given by $(x, y) \mapsto (x, yT)$, see also Section 3.2.4.1.

Lemma 4.1. *Let $P \in E_1(\mathbb{F}_p)[r] \subset \mathbb{F}_p^2$ and $Q \in E_1(\mathbb{F}_{p^6})[r] \subset \mathbb{F}_{p^6}^2$, where $Q = (x, yT)$ and $(x, y) \in E'(\mathbb{F}_{p^3})[r]$. Then computing the reduced Tate pairing $t_r(P, Q)$ costs $\mathcal{P}_{Tate,r,k} = 12514M_p + 30929A_p$ and $S_{Tate,k} = 28S_p$.*

Proof. We run the Tate Pairing (Algorithm 10) with input $P \in E_1(\mathbb{F}_p)[r] \subset \mathbb{F}_p^2$ and $Q \in E_1(\mathbb{F}_{p^6})[r] \subset \mathbb{F}_{p^6}^2$, where $Q = (x, yT)$ and $(x, y) \in E'(\mathbb{F}_{p^3})$. Then we get the following costs:

- *tangent evaluation* - line 5 (Algorithm 8): According to Lemma 3.14 we have $\mathcal{T}_{MNT,Jac} = 7M_p + 9A_p$.
- *point doubling* - line 6 (Algorithm 8): According to Lemma 3.14 we have $\mathcal{D}_{MNT,Jac} = 8M_p + 16A_p$.
- *squaring* - line 7 (Algorithm 6): according to Lemma 3.7 we have $M_{p^6} = 18M_p + 68A_p$.
- *multiplication* - line 8 and 12 (Algorithm 6): The line respectively tangent evaluation yields an element over $\mathbb{F}_{p^6}$ with 6 coefficients unequal to 0. This means we have to consider this multiplication as a full $\mathbb{F}_{p^6}$ multiplication. Thus, according to Lemma 3.7 we have $M_{p^6} = 18M_p + 68A_p$.
- *line evaluation* - line 10 (Algorithm 7): According to Lemma 3.13 we have $\mathcal{L}_{MNT,Jac} = 7M_p + 8A_p$.
- *point addition* (Algorithm 7): According to Lemma 3.13 we have $\mathcal{A}_{MNT,Jac} = 12M_p + 6A_p$.

According to Theorem 3.18 we get the following overall costs:

$$\begin{aligned}
\mathcal{P}_{Tate,r,k} &= (|r| - 1) \cdot [\mathcal{D}_{MNT,Jac} + \mathcal{T}_{MNT,Jac} + 2M_{p^6}] \\
&+ (h(r) - 2) \cdot [\mathcal{A}_{MNT,Jac-Aff} + \mathcal{L}_{MNT,Jac} + 1M_{p^6}] + \mathcal{E} \\
&= 159 \cdot [8M_p + 16A_p + 7M_p + 9A_p + 36M_p + 136A_p] \\
&+ 65 \cdot [12M_p + 6A_p + 7M_p + 8A_p + 18M_p + 68A_p] + \mathcal{E} \\
&= 159 \cdot [51M_p + 161A_p] + 65 \cdot [37M_p + 82A_p] + \mathcal{E} \\
&= [8109M_p + 25599A_p] + [2405M_p + 5330A_p] + \mathcal{E} \\
&= 12514M_p + 30929A_p + \mathcal{E}.
\end{aligned}$$

The space requirement is as stated in Theorem 3.18. For this we have to evaluate $\max(S_{MNT,Jac,add}, S_{MNT,Jac,dbl}, S_{KS,p^6})$, which is

$$\begin{aligned}
max(1S_{p^6} + 7S_p, 1S_{p^6} + 6S_p, 14S_p) &= max(13S_p, 12S_p, 14S_p) \\
&= 14S_p,
\end{aligned}$$

see Lemma 3.13, Lemma 3.14 andLemma 3.7. Thus the total space requirement is $S_{Tate,k} = 2S_{p^6} + 2S_p + 14S_p = 28S_p$. □

4.1.2 Embedding degree $k = 12$, Security level 128 bit

We consider the elliptic curve $E_2 : y^2 = x^3 - 2$ over $\mathbb{F}_p$ which is together with $l = -(2^{62} + 2^{55} + 1) = -(0x4080000000000001)$ a BN-curve with embedding degree $k = 12$. This particular l is from [25] and it generates the 254-bit prime $p(l)$ and the prime order $r(l)$:

$$p(l) = \texttt{0x2523648240000001BA344D8000000008612100000000013A700000000000013}$$
$$r(l) = \texttt{0x2523648240000001BA344D8000000007FF9F800000000010A10000000000000D}$$

In the following we set $p = p(l)$ and $r = r(l)$. We choose this specific l since it generates an order r with a Hamming weight of just 49. The curve E has a degree-6 twist over $\mathbb{F}_{p^2}$, namely $E' : y^2 = x^3 - 2/\xi$. The isomorphism $E'(\mathbb{F}_{p^2})[r] \to E_2(\mathbb{F}_{p^{12}})[r]$ is given by $(x, y) \mapsto (xU^2, yU^3)$, see also Section 3.2.4.2.

Lemma 4.2. *Let $P \in E_2(\mathbb{F}_p)[r] \subset \mathbb{F}_p^2$, $Q \in E_2(\mathbb{F}_{p^{12}})[r] \subset \mathbb{F}_{p^{12}}^2$, where $Q = (xU^2, yU^3)$ and $(x, y) \in E'(\mathbb{F}_{p^2})[r]$, and r, p be defined as above. Then computing the reduced Tate pairing $t_{r,}(P, Q)$ costs $\mathcal{P}_{Tate,r,k} = 29056M_p + 139056A_p + \mathcal{E}$ and $S_{Tate,k} = 58S_p$.*

Proof. We run the Tate pairing (Algorithm 10) with input $P \in E(\mathbb{F}_p)[r] \subset \mathbb{F}_p^2$ and $Q \in E(\mathbb{F}_{p^{12}})[r] \subset \mathbb{F}_{p^{12}}^2$, where $Q = (xW^2, yW^3)$. Then we get the following costs:

- *tangent evaluation* - line 5 (Algorithm 9): According to Lemma 3.17 we have $\mathcal{T}_{BN,Jac} = 9M_p + 15A_p$.
- *point doubling* - line 6 (Algorithm 9): According to Lemma 3.17 we have $\mathcal{D}_{BN,Jac} = 7M_p + 14A_p$.
- *squaring* - line 7 (Algorithm 5): According to Lemma 3.8 we have $M_{p^{12}} = 54M_p + 237A_p$.
- *multiplication* - line 8 and 12 (Algorithm 5): $l(Q)$ has in both cases (line 8 and 12) the form $a + bU^2 + cU^3$ with $a \in \mathbb{F}_p$ and $b, c \in \mathbb{F}_{p^2}$. Computing this multiplication with the Karatsuba method while treating f and $l(Q)$ as elements from $\mathbb{F}_{p^{12}}$ costs according to Lemma 3.8 $M_{p^{12}} = 54M_p + 237A_p$. Since $l(Q)$ has only 5 coefficients unequal 0, many multiplications have no effect. Hence, for this analysis we do not count these multiplications. Hence computing $f \cdot l(Q)$ costs $35M_p + 237A_p$[1][2].
- *line evaluation* - line 10 (Algorithm 7): According to Lemma 3.16 we have $\mathcal{L}_{BN,Jac} = 6M_p + 8A_p$.

[1] The number of additions is also smaller, but we ignore this here.

[2] One can check that the multiplication $f \cdot l(Q)$ costs only $35M_p$ by simply simulating Algorithm 5 on input f and $l(Q)$, and counting the number of these base field multiplications where every factor is unequal to 0.

- *point addition* - line 11 (Algorithm 7): According to Lemma 3.16 we have $\mathcal{A}_{BN,Jac} = 12M_p + 6A_p$.

According to Theorem 3.18 we get the following overall costs:

$$\begin{aligned}
\mathcal{P}_{Tate,r,k} &= (|r|-1)\cdot[\mathcal{D}_{BN,Jac} + \mathcal{T}_{BN,Jac} + 1M_{p^{12}} + (35M_p + 237A_p)] \\
&+ (h(r)-2)\cdot[\mathcal{A}_{BN,Jac-Aff} + \mathcal{L}_{BN,Jac} + (35M_p + 237A_p)] + \mathcal{E} \\
&= 253\cdot[7M_p + 14A_p + 9M_p + 15A_p + 54M_p + 237A_p + 35M_p + 237A_p)] \\
&+ 47\cdot[12M_p + 6A_p + 6M_p + 8A_p + 35M_p + 237A_p] + \mathcal{E} \\
&= 253\cdot[105M_p + 503A_p] + 47\cdot[53M_p + 251A_p] + \mathcal{E} \\
&= 26565M_p + 127259A_p + 2491M_p + 11797A_p + \mathcal{E} \\
&= 29056M_p + 139056A_p + \mathcal{E}.
\end{aligned}$$

The space requirement is as stated in Theorem 3.18. For this we have to evaluate $\max(S_{BN,Jac,add}, S_{BN,Jac,dbl}, S_{KS,p^{12}})$, which is

$$\begin{aligned}
\max(S_{BN,Jac,add}, S_{BN,Jac,dbl}, S_{KS,p^{12}}) &= \max(19S_p, 32S_p, 26S_p) \\
&= 32S_p,
\end{aligned}$$

see Lemma 3.16, Lemma 3.17 and Lemma 3.8. Thus the total space requirement is $S_{Tate,k} = 2S_{p^{12}} + 2S_p + 32S_p = 58S_p$. □

4.2 Measurements and Estimates

One part of this thesis is an implementation of the reduced Tate pairing. We have implemented the Tate pairing for the case $k = 12$ and $|p| = 254$. For this specific implementation we measured timings which we present in the following. In particular we present timings for operations over $\mathbb{F}_p$ for the field sizes 254 and 160, operations over the extension field $\mathbb{F}_{p^{12}}$ for $|p| = 254$ and the Miller loop of the Tate pairing. We will use these timings to justify the practical correctness of our analyses made before.

All measurements were done on a ARM Cortex M3 at the highest possible clock rate of 166Mhz and we used the flag `-O3` for compiler optimizations. The ARM Cortex M3 has a word length $n = 32$ and 13 general purpose registers. Table 4.1 summarizes the cycles count of the most important instructions for computing bilinear pairings.

Instruction	Signature	Cycles count
`mul32`	$\{0,1\}^{32} \times \{0,1\}^{32} = \{0,1\}^{32}$	1 or 2
`mul64`	$\{0,1\}^{32} \times \{0,1\}^{32} = \{0,1\}^{64}$	$3-5$
`mulaccu64`	$\{0,1\}^{32} \times \{0,1\}^{32} + \{0,1\}^{64} = \{0,1\}^{64}$	$3-6$
`add32`	$\{0,1\}^{32} + \{0,1\}^{32} = \{0,1\}^{32}$	1
`load`	/	first access 2, then 1
`store`	/	at least 2

Table 4.1: Extract of instructions and cycle counts of the ARM Cortex M3

4.2.1 Base Field Arithmetic

In Table 4.2 we present timings for multiplication, addition and inversion over base fields that are defined by primes p_1 and p_2 where $|p_1| = 254$ and $|p_2| = 160$. These timings show us that computing pairings on affine coordinates is not as efficient as on Jacobian coordinates.

Computations on affine coordinates for $|p_1| = 254$ and $k = 12$ We check that at first for the case $|p_1| = 254$ and embedding degree $k = 12$. We see that one inversion is approximately as expensive as 20 multiplications[3]. The point addition $T + P$ on affine coordinates $T = (x_1, y_1), P = (x_2, y_2) \in \mathbb{F}_{p_1}$ (see (2.6)) is computable with 1 inversion, $2M_{p_1}$ and $6A_{p_1}$. The line evaluation $l_{T,P}(Q)$ (see (3.5)) on the affine coordinates T, P as defined above and $Q = (xU^2, yU^3) \subset \mathbb{F}_{p_1^{12}}$ as defined in Lemma 4.2 has the following costs: the needed inversion can be reused from the computation of the point addition and the remaining subtractions and multiplications cost at most $8A_{p_1}$ and $3M_{p_1}$. Having that one inversion costs $20M_{p_1}$ we can estimate the costs for point addition and line evaluation on affine coordinates with $25M_{p_1} + 14A_{p_1}$. A point addition and a line evaluation on Jacobian coordinates cost $12M_{p_1} + 6A_{p_1} + 6M_{p_1} + 8A_{p_1} = 18M_{p_1} + 14A_{p_1}$, see Lemma 3.16.

The point doubling $[2]T$ on the affine coordinate T (see (2.7)) is computable with 1 inversion, $3M_{p_1}$ and at least $7A_{p_1}$, where we do not count the addition with a and we consider the scalar multiplications by 2 and 3 as additions. The tangent evaluation $l_{T,T}(Q)$ on the affine coordinates T and Q as defined above can be estimated with the costs: we can completely reuse the computation of term $(3x_1^2 + a)/2y_1$ and the remaining subtractions and multiplications cost at most $8A_{p_1}$ and $3M_{p_1}$. Then we can estimate the costs by $26M_{p_1} + 15A_{p_1}$. A point doubling and a tangent evaluation on Jacobian coordinates cost $7M_{p_1} + 14A_{p_1} + 9M_{p_1} + 15A_{p_1} = 16M_{p_1} + 29A_{p_1}$, see Lemma 3.17. Thus, as long as 1 inversion in $\mathbb{F}_{p_1}$ is at least as expensive as $11.5M_{p_1}$, point operations and line evaluations on affine coordinates will not outperform those operations on Jacobian coordinates significantly.

[3]The inversion is computed with the Extended Euclidean Algorithm.

Computations on affine coordinates for $|p_2| = 160$ and $k = 6$ Now we check that computing pairings on affine coordinates for the case $|p_2| = 160$ and embedding degree $k = 6$ is also not as efficient as on Jacobian coordinates. In this case we have that one inversion is approximately as expensive as 17 multiplications. The point addition and point doubling on the affine coordinates is as expensive as stated calculated above. Thus, we have only to consider the costs of the line evaluation on affine coordinates. The line evaluation $l_{T,P}(Q)$ (see (3.5)) on the affine coordinates T, P as defined above and $Q = (x, yT) \subset \mathbb{F}_{p_2^6}$ as defined in Lemma 4.1 has the following costs: the inversion can be reused again and the remaining subtractions and multiplications cost at most $8A_{p_2}$ and $3M_{p_2}$. Having that one inversion costs $17M_{p_2}$ we can estimate the costs for point addition and line evaluation on affine coordinates with $22M_{p_2} + 14A_{p_1}$. A point addition and line evaluation on on Jacobian coordinates cost $12M_{p_2} + 6A_{p_2} + 7M_{p_2} + 8A_{p_2} = 19M_{p_2} + 14A_{p_2}$, see Lemma 3.13.

The tangent evaluation $l_{T,T}(Q)$ (see (3.7)) on the affine coordinates T, P as defined above and $Q = (x, yT) \subset \mathbb{F}_{p_2^6}$ as defined in Lemma 4.1 has the following costs: again we can completely reuse the computation of the term $(3x_1^2 + a)/2y_1$ and the remaining subtractions and multiplications cost at most $8A_{p_2}$ and $3M_{p_2}$. Then we can estimate the costs for computing a point doubling and tangent evaluation on affine coordinates with $23M_{p_2} + 15A_{p_2}$. Here we have that a point doubling and a tangent evaluation on Jacobian coordinates cost at most $8M_{p_2} + 16A_{p_2} + 7M_{p_2} + 9A_{p_2} = 15M_{p_2} + 25A_{p_2}$, see Lemma 3.14. We have the same break-even value as before: as long as 1 inversion in $\mathbb{F}_{p_2}$ is at least as expensive $11.5M_{p_2}$, point operations and line evaluation on affine coordinates will not outperform those operations on Jacobian coordinates significantly.

Computing constant multiplications We can also see that one multiplication is approximately as expensive as 30 additions. Thus, it makes sense to compute multiplications by constants c with $|c| + h(c) - 1 \ll 30$ by a double-and-add approach.

	$10k$ Mul	$10k$ Add	$10k$ Inv
$\mathbb{F}_{p_1}$	$2.1s$	$0.07s$	$42s$
$\mathbb{F}_{p_2}$	$1.05s$	$0.048s$	$18s$

Table 4.2: costs of base field operations in seconds, with $|p_1| = 254$ and $|p_2| = 160$

4.2.2 Extension Field Arithmetic

In Table 4.3 we present timings for Schoolbook and Karatsuba multiplication as well as for additions, each for the fields $\mathbb{F}_{p^{12}}$ and $\mathbb{F}_{p^6}$ with $|p| = 254$. We see that the Karatsuba method clearly outperforms the Schoolbook method. As shown in Table 3.1, the multiplication timings behaves as stated by the theoretical results.

We show this by applying the theorems statements on the timings from table Table 4.2. Then we compare the theoretical estimates with the timings given in Table 4.3 by providing a relative error measurement. We compute the relative error f by the formula $(x_t/x_p - 1) \cdot 100\%$, where x_t is the theoretical estimate and x_p is the timing measured in practice.

- $\mathbb{F}_{p^{12}}$ Karatsuba, applying Lemma 3.8:

$$54 \cdot 2.1s + 237 \cdot 0.07s = 129.99s \quad \approx 132s, \quad f = -1.523\%.$$

- $\mathbb{F}_{p^{12}}$ Schoolbook, applying Lemma 3.4:

$$144 \cdot 2.1s + 195 \cdot 0.07s = 316.05s \quad \approx 325s, \quad f = -2.754\%.$$

- $\mathbb{F}_{p^6}$ Karatsuba, applying Lemma 3.7:

$$18 \cdot 2.1s + 68 \cdot 0.07s = 42.56s \quad \approx 43s, \quad f = -1.023\%.$$

- $\mathbb{F}_{p^6}$ Schoolbook, applying Lemma 3.3:

$$36 \cdot 2.1s + 45 \cdot 0.07s = 78.75s \quad \approx 81s, \quad f = -2.777\%.$$

At first we notice that the theoretical estimates are always beneath the practical timings. We can say that these small errors result from overhead such as functions calls, loops, if-else statements and assignments that are not considered by the theoretical analyses. Thus, all in all, the theoretical analyses fit the practical timings very well, such that the theoretical analyses are useful for further estimates.

	$10k$ Mul Schoolbook	$10k$ Mul Karatsuba	$10k$ Additions
$\mathbb{F}_{p^6}$	$81s$	$43s$	$0.4s$
$\mathbb{F}_{p^{12}}$	$325s$	$132s$	$0.9s$

Table 4.3: costs of extension field operations in seconds, $|p| = 254$

4.2.3 Bilinear Pairings

At first we will validate the theoretical estimate for the reduced Tate pairing for the case $k = 12$ (see Section 4.1.2). Second, we will estimate the run time of the reduced Tate pairing for the case $k = 6$ (see Section 4.1.1).

Validation of Estimate for Reduced Tate Pairing with $k = 12$ and $|p| = 254$
In Table 4.4 we present timings for the computation of the reduced Tate pairing as specified in Section 4.1.2. We compare these timings with the theoretical estimate made in Section 4.1.2. We state that one computation of the reduced

Tate pairing costs $29056M_p + 139056A_p + \mathcal{E}$, see Lemma 4.2. According to table Table 4.2, for the Miller Operation we have a theoretical estimate of $29056 \cdot 0.00021s + 139056 \cdot 0.000007s = 7.075152s$. Compared to the measurement we have a relative error of $f = -3.96\%$.

Miller Operation	Final Exponentiation	Total
7.3666	54.9333	62.3

Table 4.4: costs of reduced Tate pairing as specified in Section 4.1.2

The final exponentiation needs $54.9333s$. This value is that high since we implemented the final exponentiation by a trivial square and multiply approach, that needs $4213M_{p^{12}}$. According to Lemma 3.8 we can express this as $227502M_p + 998481A_p$ which gives a theoretical estimate of $227502 \cdot 0.00021s + 998481 \cdot 0.000007s = 47.77542s + 6.989367s = 54.764787s$. Compared to the measurement this is a relative error of $f = -0.3\%$.

One can drastically reduce the complexity of the final exponentiation by using more advanced exponentiation strategies as explained by Hankerson et al. in [18, Ch. 7]. The analysis made there is comparable with our analysis. Using an advanced exponentiation strategy, they estimate the costs with $7246M_p$ and one inversion in the base field. Thus, using a similar exponentiation strategy, this would lower the costs approximately by factor 30. Hence, considering our timings, the final exponentiation can be done approximately in $2s$, which is a conservative time estimate.

Thus, it is possible to compute the reduced Tate pairing for $k = 12$ and $|p| = 254$ in at most $7.3666s + 2s \approx 9.5s$.

Run Time Estimate for Reduced Tate Pairing with $k = 6$ and $|p| = 160$ In Section 4.1.1 we state that the computation of the reduced Tate pairing costs $12514M_p + 30929A_p + \mathcal{E}$, see Lemma 4.1. According to Table 4.2, we estimate that one computation costs $12514 \cdot 0.000105s + 30929 \cdot 0.0000048s = 1.462s$. As before, the final exponentiation can be computed in at most $2s$, which is a very raw and conservative estimate. Hence, it is possible to compute the reduced Tate pairing for $k = 6$ and $|p| = 160$ in at most $1.462s + 2s \approx 3.5s$.

4.3 Discussion

We have that the reduced Tate pairing for $k = 6$ is $9.5s/3.5s \approx 2.7$ times faster than the reduced Tate pairing for $k = 12$. The better efficiency of the reduced Tate pairing for $k = 6$ comes along with a lower security level of just 80 bit, whereas the reduced Tate pairing for $k = 12$ provides a security level of 128 bit. Thus, we can state that: the lower the security level the faster the computation of the reduced Tate pairing. Note that the factor 2.7 is just a raw estimate. By applying and analyzing an advanced exponentiation strategy we would get

a sharper and greater factor. More precisely, we think that the factor is greater than 2.7.

All in all, the performance of the reduced Tate pairing is not very high on our platform. The timings $9.5s$ and $3.5s$ are far too slow for practical purposes. Many cryptographic protocols that rely on bilinear pairings require several pairing computations within one protocol run. This would result in a protocol run time of several tens of seconds or even minutes which is completely not applicable. For practical purposes we require a bilinear pairing computation in the range of $1ms$. This means, we have to improve the performance of the reduced Tate pairing for $k = 6$ by a factor 3500, and for $k = 12$ by a factor 9500. We think that run times in this range are not achievable on an ARM Cortex M3 @ 166Mhz together with ESCRYPT´s base field library. Nonetheless, we want to show where our implementation lacks efficiency and where it is worse to improve the implementation. Then we will analyse how fast we can get.

4.3.1 Further Optimization

Our implementation lacks efficiency at the following computations.

1. Our implementation does not take advantage of special squaring strategies in extension fields. Hankerson et al. apply in [18, Ch. 7] a special squaring method for the extension field $\mathbb{F}_{p^6}$. This technique only costs $12M_p$ to square in $\mathbb{F}_{p^6}$ and $36M_p$ to square in $\mathbb{F}_{p^6}$. Applying this method for the reduced Tate pairing for $k = 12$ would decrease the number of multiplications by $253 \cdot (54M_p - 36M_p) = 4554M_p$, which saves approximately $4554 \cdot 0.00021s = 0.956s$. For $k = 6$ this method decreases the number of multiplications by $159 \cdot (18M_p - 12M_p) = 954M_p$, which saves approximately $954 \cdot 0.000105s = 0.1s$.

2. Algorithm 7 needs 12 base field multiplications for computing a point addition. The point addition is also computable with just 11 multiplications, see[5, madd-2007-bl]. Applying this algorithm would save 47 multiplications for the case $k = 12$ and 65 multiplications for the case $k = 6$. This results just in a small improvement of $47 \cdot 0.00021s = 0.00987s$ and $65 \cdot 0.000105s = 0.006825s$, respectively.

3. Our implementation does not take advantage of the lazy reduction approach. Aranha et al. state in [2] an improvement of about 20% on several Intel and AMD 64-bit architectures.

Hence, applying these improvements could reduce the run time of $9.5s$ for $k = 12$ to $9.5s - 0.956s - 0.00987s - 0.2 \cdot (9.5s - 0.956s - 0.00987s) \approx 6.8s$ and the run time of $3.5s$ for $k = 6$ to $3.5s - 0.1s - 0.006825s - 0.2(3.5s - 0.1s - 0.006825s) \approx 2.7s$. We still need to improve the performance by a factor 6800 respectively 2700.

From now on the argumentation follows on the base of the improvements stated in 1. and 2.: For computing the Miller loop for $k = 6$ we need $12514M_p -$

$954M_p - 65M_p + 30929A_p = 11495M_p + 30929A_p$ and for $k = 12$ we need $29056M_p - 4554M_p - 47M_p + 139056A_p = 24455M_p + 139056A_p$.

4.3.2 Structure of costs

Here we want to give hints where it is worse to improve the implementation. For $k = 6$ we have that 89% of the run time are due to multiplications and 11% are for additions. For $k = 12$ the proportion is slightly different: 84% of the run time are due to multiplications and 16% are due to additions. Thus, to improve the run time we concentrate on the multiplications. But one should not completely ignore the performance of the additions, since 16% and 11% correspond to $0.97s$ and $0.15s$ which is also far away from our goal of $1ms$.

For reducing the run time we have generally two possibilities: (1) improve the base field arithmetic and (2) reduce the number of base field operations.

Improving the base field arithmetic Here we show which run times we require for the base field arithmetic to reach the goal of $1ms$. From now on we just concentrate on the Miller loop and omit the final exponentiation. This means, we soften our goal to compute the Miller loop within $1ms$ and not the whole pairing. Note, managing to compute the Miller loop within $1ms$ then computing the whole pairing within $2ms$ is also possible.

At first we consider this for the case $k = 6$ and $|p| = 160$. We orientate oneself on the proportion above e.g. we want that 90% of the run time are due to multiplications and 10% are due to the additions. Thus, the multiplications have to be computed within $890\mu s$ and the additions within $110\mu s$. To reach that, we need that $1M_p$ costs at most $890\mu s/11495 \approx 0.000000077s = 77ns$ and $1A_p$ costs at most $110\mu s/30929 \approx 0.0000000036s = 3.6ns$. At 166Mhz one clock cycle of the ARM Cortex M3 costs approximately $T_{Clock} = 6ns$. To compute the Miller loop within $1ms$ we would have to compute $1M_p$ within $77ns/6ns \approx 13$ cycles and $1A_p$ within $3.6ns/6ns \approx 0.5$ cycles. By taking a look at Table 4.1 one immediately sees that this is not achievable, since just one `mul64` instruction costs in the mean $\overline{C}_{mul64} = 4$ cycles. To see what is possible on the ARM Cortex M3 @ 166Mhz, we want to give a very raw lower bound for the run time of the multiplications. For this we estimate the costs of one $1M_p$ multiplication in the number of cycles. We choose the Schoolbook method for multiplying to 160-bit elements. For this we need 25 `mul64` instructions. Applying the Karatsuba method for 160-bit elements we would need at least 27 `mul64` instructions or 81 `mul32` instructions. Thus, the multiplication costs $25 \cdot \overline{C}_{mul64} \cdot T_{Clock} = 600ns$. Note, these are just the costs for the pure multiplication, the costs for the reduction are omitted. Having that $1M_p$ costs at least $600ns$ we can state the lower bound $12514 \cdot 600ns \approx 7.5ms$. Thus, we can state that our software implementation of the reduced Tate pairing for $k = 6$ and $|p| = 160$ on the ARM Cortex M3 processor cannot be faster than $7.5ms$.

Now we consider the case $k = 12$ and $|p| = 254$. Again, we orientate oneself on the proportion above. Thus, the multiplications have to be computed within

$840\mu s$ and the additions within $160\mu s$. For this, we need that $1M_p$ costs at most $840\mu s/24455 \approx 0.000000034s = 34ns$ and $1A_p$ costs at most $160\mu s/139056 \approx 0.0000000012 = 1.2ns$. Again we see that this is not achievable, since the run times are even tougher as in the case for $k = 6$. Here we want also give a very raw lower bound for the run time of the multiplications. This time we choose the Karatsuba method for multiplying 254-bit elements. For this we need 27 `mul64` instructions. Thus, the pure multiplication costs $27 \cdot \overline{C}_{mul64} \cdot T_{Clock} = 648ns$. This results in a total run time of at least $24455 \cdot 648ns \approx 15.8ms$. Thus, we can state that our software implementation of the reduced Tate pairing for $k = 12$ and $|p| = 254$ on the ARM Cortex M3 processor cannot be faster than $15.8ms$.

Reducing the number of base field operations Another possibility to reduce the run time is to save base field operations. Here we want to show which computations of the Miller loop have the biggest impact on the number of base field operations. One can use that to orientate oneself when searching for the computations with the biggest optimization potential.

Generally we can say that the `if`-block of reduced Tate pairing (Algorithm 10) has a much smaller impact than the rest of `for`-loop, since `if`-block is executed only $h(r) - 2$ times whereas the rest is executed $|r - 1|$ times. Thus, one should always concentrate at first on the lines $5 - 8$. Furthermore, one can regulate the impact of the `if`-block by choosing r such that $h(r)$ is as small as possible.

We can say that the point operations and line evaluations have the smallest impact. For $k = 6$ these operations cost $3555M_p + 4885A_p$ and need $0.4s$ which is only 29% of the total run time of the Miller loop. For $k = 12$ these operations cost $4847M_p + 7995A_p$ and need $1.07s$ which is only 18% of the total run time of the Miller loop. The used algorithms for computing the point operations and line evaluations are the best known for short Weierstrass curves with $a = -3$ or $a = 0$ on Jacobian coordinates. Thus, for improving these operations one should concentrate on other types of curves.

From this we can conclude that the squaring in line 7 and the multiplications in line 8 and 12 of Algorithm 10 make use of 71% and 82% of the total run time for the case $k = 6$ respectively $k = 12$. Thus, one should concentrate on optimizing these multiplications.

4.3.3 Outlook

Yet another possibility to reduce the runtime might be to implement parts of the computation in hardware. The best part to start with is the base field multiplication and the appropriate reduction.

One should also consider to implement the Ate pairing. Hankerson et al. show in [18, Ch. 7] that this is possible with only $15722M_p$ for the Miller operation and $7246M_p$ and one inversion in the base field for the case $k = 12$ and $|p| = 254$. This shows that the Ate pairing is more efficient than the reduced Tate pairing. This should be also true for the case $k = 6$ and $|p| = 160$.

Bibliography

[1] Tolga Acar, Kristin Lauter, Michael Naehrig, and Daniel Shumow. Affine pairings on arm. In *Pairing-Based Cryptography–Pairing 2012*, pages 203–209. Springer, 2013.

[2] Diego F Aranha, Koray Karabina, Patrick Longa, Catherine H Gebotys, and Julio López. Faster explicit formulas for computing pairings over ordinary curves. In *Advances in Cryptology–EUROCRYPT 2011*, pages 48–68. Springer, 2011.

[3] Paulo SLM Barreto, Hae Y Kim, Ben Lynn, and Michael Scott. Efficient algorithms for pairing-based cryptosystems. In *Advances in cryptology–CRYPTO 2002*, pages 354–369. Springer, 2002.

[4] Paulo SLM Barreto and Michael Naehrig. Pairing-friendly elliptic curves of prime order. In *Selected areas in cryptography*, pages 319–331. Springer, 2006.

[5] Daniel J. Bernstein and Tanja Lange. Explicit-formulas database. www.hyperelliptic.org, 2007.

[6] Jean-Luc Beuchat, Jorge E González-Díaz, Shigeo Mitsunari, Eiji Okamoto, Francisco Rodríguez-Henríquez, and Tadanori Teruya. High-speed software implementation of the optimal ate pairing over barreto–naehrig curves. In *Pairing-Based Cryptography-Pairing 2010*, pages 21–39. Springer, 2010.

[7] I. Blake, G. Seroussi, N. Smart, and J. W. S. Cassels. *Advances in Elliptic Curve Cryptography (London Mathematical Society Lecture Note Series)*. Cambridge University Press, New York, NY, USA, 2005.

[8] Dan Boneh, Xavier Boyen, and Hovav Shacham. Short group signatures. In *Advances in Cryptology–CRYPTO 2004*, pages 41–55. Springer, 2004.

[9] Dan Boneh and Matt Franklin. Identity-based encryption from the weil pairing. In *Advances in Cryptology–CRYPTO 2001*, pages 213–229. Springer, 2001.

[10] Augusto Jun Devegili, Michael Scott, and Ricardo Dahab. Implementing cryptographic pairings over barreto-naehrig curves. In *Pairing-Based Cryptography–Pairing 2007*, pages 197–207. Springer, 2007.

[11] Steven D Galbraith, Keith Harrison, and David Soldera. Implementing the tate pairing. In *Algorithmic number theory*, pages 324–337. Springer, 2002.

[12] Steven D Galbraith, Kenneth G Paterson, and Nigel P Smart. Pairings for cryptographers. *Discrete Applied Mathematics*, 156(16):3113–3121, 2008.

[13] Conrado PL Gouvêa, Leonardo B Oliveira, and Julio López. Efficient software implementation of public-key cryptography on sensor networks using the msp430x microcontroller. *Journal of Cryptographic Engineering*, 2(1):19–29, 2012.

[14] Vipul Goyal, Omkant Pandey, Amit Sahai, and Brent Waters. Attribute-based encryption for fine-grained access control of encrypted data. In *Proceedings of the 13th ACM conference on Computer and communications security*, pages 89–98. ACM, 2006.

[15] Darrel Hankerson, Scott Vanstone, and Alfred J Menezes. *Guide to elliptic curve cryptography*. Springer, 2004.

[16] Florian Hess, Nigel P Smart, and Frederik Vercauteren. The eta pairing revisited. *Information Theory, IEEE Transactions on*, 52(10):4595–4602, 2006.

[17] Antoine Joux. A one round protocol for tripartite diffie–hellman. In *Algorithmic number theory*, pages 385–393. Springer, 2000.

[18] Marc Joye and Gregory Neven. *Identity-based cryptography*, volume 2. IOS press, 2009.

[19] Rudolf Lidl and Harald Niederreiter. *Finite Fields*. Cambridge University Press, second edition, 1996. Cambridge Books Online.

[20] Ben Lynn. *On the implementation of pairing-based cryptosystems*. PhD thesis, Stanford University, 2007.

[21] Alfred J Menezes, Paul C Van Oorschot, and Scott A Vanstone. *Handbook of applied cryptography*. CRC press, 2010.

[22] Victor S Miller. The weil pairing, and its efficient calculation. *Journal of Cryptology*, 17(4):235–261, 2004.

[23] Atsuko Miyaji, Masaki Nakabayashi, and Shunzou Takano. New explicit conditions of elliptic curve traces for fr-reduction. *IEICE transactions on fundamentals of electronics, communications and computer sciences*, 84(5):1234–1243, 2001.

[24] Michael Naehrig, Ruben Niederhagen, and Peter Schwabe. New software speed records for cryptographic pairings. In *Progress in Cryptology–LATINCRYPT 2010*, pages 109–123. Springer, 2010.

[25] Yasuyuki Nogami, Masataka Akane, Yumi Sakemi, Hidehiro Katou, and Yoshitaka Morikawa. *Integer Variable chi-Based Ate Pairing*. 2008.

[26] Christof Paar and Jan Pelzl. *Understanding Cryptography - A Textbook for Students and Practitioners*. Springer, 2010.

[27] Dan Page, Nigel P Smart, and Frederik Vercauteren. A comparison of mnt curves and supersingular curves. *Applicable Algebra in Engineering, Communication and Computing*, 17(5):379–392, 2006.

[28] Michael Scott, Naomi Benger, Manuel Charlemagne, Luis J Dominguez Perez, and Ezekiel J Kachisa. On the final exponentiation for calculating pairings on ordinary elliptic curves. In *Pairing-Based Cryptography–Pairing 2009*, pages 78–88. Springer, 2009.

[29] Thomas Unterluggauer and Erich Wenger. Efficient pairings and ECC for embedded systems. In *Cryptographic Hardware and Embedded Systems - CHES 2014 - 16th International Workshop, Busan, South Korea, September 23-26, 2014. Proceedings*, pages 298–315, 2014.