UNIVERSITA' DEGLI STUDI ROMA TRE
Facoltà di Scienze Matematiche, Fisiche e Naturali
Dipartimento di Matematica

Trace Zero Varieties in Pairing-based Cryptography

Relatori:
Prof. Antonio LIOY
Prof. Roberto M. AVANZI

Coordinatore:
Prof. Renato SPIGLER

Tesi di Dottorato di:
Emanuele CESENA
XXI ciclo

Anno Accademico 2009–2010

Introduction

The term *cryptography*, by etymology or simply by association of ideas, suggests its connection with secret messages and this is made clear from the definition that we find in Wikipedia: “The practice and study of hiding information”.

In light of the results of the fundamental article of Diffie and Hellman *New Directions in Cryptography* [DH76] this simple definition of cryptography seems to require a supplement: nowadays states, public organizations and private individuals can not only exchange information in secrecy, but also sign electronic documents so that the digital signature is easily verifiable, but not falsifiable. To make this possible, Diffie and Hellman introduce the concept of *public-key cryptosystem*: the use of public keys allows us to exchange secret keys without having to meet in person (for instance, using channels in clear on the network) or generate/verify digital signatures.

The first example of a public key cryptosystem is proposed in 1978 by Rivest, Shamir and Adleman [RSA78]. Diffie and Hellman also advance the idea of establishing a cryptographic system on the *discrete logarithm problem* (DLP) in a finite field, an idea which they attribute to Prof. Gill of Stanford University, and carried out for the first time in 1985 by ElGamal [ElG85].

In the same year Miller and Koblitz [Mil86a, Kob87] propose to use the group of rational points of an elliptic curve over a finite field. Compared to those already mentioned, elliptic curves allow greater flexibility in building the group and the use of smaller keys at the same level of security. Four years later Koblitz [Kob89] indicates the Jacobian variety of hyperelliptic curves as another possible candidate for the construction of cryptosystems.

In 1998 Frey [Fre98, Fre01] suggests to use *Trace Zero Varieties* (TZV) for cryptosystems based on the DLP. The starting point of Frey’s construction is, in the simplest case, an elliptic curve $\mathcal{E}$ defined over a finite field $\mathbb{F}_q$. Let $\mathbb{F}_{q^r}/\mathbb{F}_q$ be a finite extension. The group $\mathcal{E}(\mathbb{F}_{q^r})$ contains $\mathcal{E}(\mathbb{F}_q)$ and the Frobenius automorphism $\mathbb{F}_{q^r}/\mathbb{F}_q$ extends to $\mathcal{E}(\mathbb{F}_{q^r})$ in a natural way. The TZV is a subgroup of $\mathcal{E}(\mathbb{F}_{q^r})$ (more precisely, a subvariety of the Weil restriction of scalars of $\mathcal{E}(\mathbb{F}_{q^r})$ from $\mathbb{F}_{q^r}$ to $\mathbb{F}_q$) which is globally invariant under the action of the Frobenius and isomorphic to the quotient $\mathcal{E}(\mathbb{F}_{q^r})/\mathcal{E}(\mathbb{F}_q)$. It is exactly such action of the Frobenius to make the computation of scalar multiplication on TZV particularly efficient.

Several authors address the study of TZV: Naumann [Nau99] and Blady [Bla02] consider TZV of elliptic curves over extension fields of degree 3 ($r = 3$); Weimerskirch [Wei01] analyzes the case for extension fields of degree 5; finally Lange [Lan01, Lan04] builds TZV from the Jacobian variety of hyperelliptic curves of genus two, over extension fields of degree 3. Avanzi and Lange [AL04] compare the performance of these three kinds of TZV over fields of odd characteristic. Avanzi and Cesena [Ces04, AC08b] compare the same three types of TZV defined over binary fields, underlining similarities and main differences between TZV defined over fields of even and odd characteristic.

The performance of an elliptic curve cryptosystem depends mainly on two aspects that one needs to consider when implementing the scalar multiplication: the first is the choice of the coordinate system used to represent points, such as classical affine coordinates where a point is represented by the pair (x, y), and the second is the use or not of precomputation.

In this work we extend the results of [AC08b] over binary fields, by taking into account different types of coordinate systems and evaluating the effect of using or not precomputation (the latter has already been considered in literature).

For elliptic curves several coordinate systems are available. The basic idea is to avoid inversions in the field (typically expensive) and to speed up the operation of doubling a point on the curve, which is the one with the major impact on scalar multiplication, especially when using precomputation. Among the proposed coordinate systems for binary elliptic curves, we mention projective coordinates in which a point is represented by the tuple (X, Y, Z) that corresponds to $(x, y) = (X/Z, Y/Z)$ and López-Dahab coordinates where (X, Y, Z) corresponds to $(x, y) = (X/Z, Y/Z^2)$ (in Section 4.2 we extend the discussion to other coordinate systems).

The main result of our analysis – reported in Section 4.2.4, Tables 4.3, C.4, C.5 and C.6 – is that TZV of elliptic curves over extensions of degree 5 are the most efficient groups suitable to build cryptographic systems based on the DLP. On our Intel platform (32-bit), at 80-bit security they are about 10% faster (20% using precomputation) and at 96-bit security they are about 22% faster (30% using precomputation) than elliptic curves with López-Dahab coordinates (extended, to be precise). For TZV, the affine coordinates appear to be the most efficient. This is because we work in extension fields, where the bad impact of inversions is reduced (an inversion in a extension only requires a single inversion in the ground field).

Nowadays it is easy to be blinded by the incredible amount of memory and computational power which is available in laptops and personal computers. However, it is important to stress that there are countless applications – where cryptography is important and often overlooked – that are or need to be deployed on devices with limited resources, like mobile phones or wireless sensors.

In such cases using affine coordinates and avoiding precomputation can be the only way to cope with the constrains imposed by the scenario and TZV turn out to be an excellent solution to improve performance. Indeed, if we limit to consider affine coordinates, we confirm the results in [AC08b] that TZV of elliptic curves are always much more efficient than elliptic curves themselves (by factors about 1.5 for extension of degree 3 and more than 2 for degree 5).

To further assess the validity of our results, we perform experiments also on a PowerPC machine, still a relatively powerful server, but the idea is to have a comparison also with an architecture more similar to what can be found in many embedded devices. Here the advantage of TZV is even more accentuated.

Finally, following an idea of [HKA06], in Section 4.2.2 we develop for TZV a new coordinate system, the *compressed López-Dahab coordinates*, in which a $\mathbb{F}_{q^r}$-rational point is represented by the tuple $(X, Y, z) \in \mathbb{F}_{q^r} \times \mathbb{F}_{q^r} \times \mathbb{F}_q$ that corresponds to $(x, y) = (X/z, Y/z^2)$. The difference with López-Dahab coordinates is therefore that the coordinate z is in $\mathbb{F}_q$, thus smaller. This is made possible by a particular operation available in extensions, called *pseudo-inversion* (see Section 4.1.2), that does not involve inversions in the ground field.

This new coordinate system turns out to be on average $8 \div 10\%$ faster than López-Dahab coordinates, and generally presents similar performance to affine coordinates. We want to remark that, since they do not require inversions in the ground field, compressed coordinates become so much more effective as worst the inversion is, thus they are attractive for devices with constrained resources.

Returning to the history of public-key cryptography, a big step forward has been made with the introduction of pairing-based cryptography. Pairing, from the mathematical point of view, is a non-degenerate, bilinear map and to use it in practical applications, we additionally require that it is efficiently computable. Algebraic geometry gives us two examples of pairing that meet the above definition: the Weil pairing and the Lichtenbaum-Tate pairing, that we shall simply call Tate pairing. The latter is particularly interesting for cryptography because it has better qualities from the computational point of view, at least for moderate security levels.

The first use of pairing in cryptography dates back to 1993, when it is exploited by Menezes, Okamoto and Vanstone [MOV93] to attack cryptosystems by reducing the DLP in the group of rational points of an elliptic curve to the DLP in a finite field. This attack is called the MOV attack.

We have to wait until 2000 to see authors rediscover pairing and use it “for good”, starting to develop cryptographic protocols and schemes based on pairing: Sakai, Ohgishi and Kasahara introduce the first pairing-based key-agreement and signature schemes, and Joux [Jou00] extends the Diffie-Hellman key agreement protocol to a three-party, one-round protocol.

Another fundamental construction is the first Identiy-Based Encryption (IBE) scheme realized in 2001 by Boneh and Franklin [BF01]. In IBE, user’s public key is derived from some known aspects of her identity, such as her name or e-mail address and this eliminates the key distribution or certification problems. The construction of a workable and provably secure scheme was an open problem posed by Shamir and dating back to 1984.

These key contributions have been the trigger for an actual explosion of interest in pairing-based cryptography, which led in recent years the definition of many protocols and schemes and motivated the research for ever more efficient implementations.

Pairing meets TZV in 2002, when Rubin and Silverberg [RS02] propose to use supersingular abelian varieties of dimension greater than one to improve the security of pairing-based cryptosystems. Besides Jacobian varieties of hyperelliptic curves, the other significant example is the class of TZV (called primitive subgroups in that paper), which can be constructed from elliptic curves.

The original work of Rubin and Silverberg and their more recent results presented in [RS09] constitute the motivation of our research. Notably, supersingular TZV of elliptic curves allow to achieve higher “security per bit” than supersingular elliptic curves themselves: in characteristic 3 ($r = 5$) TZV represent the first example of supersingular abelian varieties with security parameter greater than 6 (in fact 7.5); in characteristic 2 ($r = 3$) TZV present an alternative to supersingular *elliptic curves over* $\mathbb{F}_{3^m}$ which is more efficient, simpler to implement and with equivalent security properties.

The computation of pairing over TZV is already taken into account by Barreto et al. [BK$^+$02, BG$^+$07], who define the η and η_T pairings on supersingular abelian varieties. Other pairings, such as the (twisted) Ate pairing [HSV06] and its extended versions [MK$^+$07, LLP09, Ver08] can be naturally defined on TZV too. However no work in literature considered using the Frobenius available in TZV to speed-up the computation of pairing.

The focus of the present work is exactly that: develop a new algorithm to compute the Tate pairing on TZV exploiting the action of the Frobenius. Our result applies to supersingular TZV in characteristic 2, 3 and $p > 3$, that are described respectively in Lemmas 3.9, 3.10 and 3.11.

In Theorem 3.14 we derive a new formula for the Tate pairing $\hat{t}(P, Q)$:

$$\hat{t}(P, Q) = \left(\prod_{i=0}^{r-1} f_{q,P} \left(Q^{\sigma_i} \right)^{q^{i(r+1)}} \right)^{M \frac{a}{r} q^{a-1}} ,$$

where $f_{q,P}$ is the Miller function, σ_i is proper power of the Frobenius endomorphism, a and M are constants depending on the curve.

The previous formula yields a new method to compute the Tate pairing over supersingular TZV. We evaluate $f_{q,P}$ at the r points Q^{σ_i} (raising each evaluation to the proper power $q^{i(r+1)}$). At the end we compute the final exponentiation to $M \frac{a}{r} q^{a-1}$. This is summarized in Algorithm 3.2.

The algorithm is suitable for a parallel implementation, requiring r processors and achieving a Miller loop of “length” q. Moreover, both in a parallel and in a sequential model, an implementation with precomputation of the multiples of P requires the storage of only $\log_2 q$ points. Implementation details are given in Section 4.3, where we also propose a variant of the point compression algorithm of Rubin and Silverber [RS02] in characteristic 2 which is more efficient and requires no inversion in the field.

Experimental results presented in Section 4.3.5 and Tables 4.6, C.10, C.11 and C.12 show that the parallel version of our new algorithm is on average $25 \div 30\%$ faster than any previous pairing algorithm – notably the η_T described in [BG$^+$07].

Besides the computation of pairing, in Section 4.2.6 we also analyze the performance of scalar multiplication on supersingular elliptic curves and TZV. Experimental results are reported in Tables 4.5, C.7, C.8 and C.9. Supersingular TZV are much faster than the corresponding elliptic curves and, as we already mentioned, they also allow to achieve higher security per bit. For instance, if we look data on Intel platform (32-bit), scalar multiplication on the supersingular TZV over $\mathbb{F}_{2^{103}}$ $(r = 3)$ is 4 times faster than on the corresponding elliptic curve defined over $\mathbb{F}_{2^{307}}$, pairing can be up to 30% faster exploiting the parallel version of the new algorithm and finally points can be compressed to 208 bits, allowing a reduction of storage or bandwidth by a factor about 3/2.

In conclusion we have seen that TZV, ordinary and supersingular, have many interesting features that make them attractive for building cryptosystems based on DLP as well as pairing-based cryptosystems. They are also well suited for implementations on devices with constrained resources at moderate security levels.

Foundation

The following publications and pre-prints form the foundation of this thesis:

- R. M. Avanzi and E. Cesena. **Pairing on Supersingular Trace Zero Varieties.** Cryptology ePrint Archive, Report 2008/404, 2008.
  `http://eprint.iacr.org/2008/404`
  A preliminary version of this work was presented at Eurocrypt 2009 poster session (cf. Appendix A).

- R. M. Avanzi and E. Cesena. **Trace Zero Varieties over Fields of Characteristic 2 for Cryptographic Applications.** In J. Hirschfeld, J. Chaumine, and R. Rolland, editors, *Algebraic geometry and its applications*, volume 5 of *Number Theory and Its Applications*, pp. 188–215. World Scientific, 2008. Proceedings of the first SAGA conference, Papeete, 7-11 May 2007.

- E. Cesena, H. Löhr, G. Ramunno, A.-R. Sadeghi, and D. Vernizzi. **Anonymous Authentication with TLS and DAA.** Submitted to TRUST 2010.

- E. Cesena, G. Ramunno, and D. Vernizzi. **Towards Trusted Broadcast Encryption.** TrustCom 2008: The 2008 International Symposium on Trusted Computing. Zhang Jia Jie (Hunan, China), 18-20 November 2008, pp. 2125–2130.

Organization

The remainder of this dissertation is organized as follows.

Chapter 1 introduces pairing-based cryptography with a few relevant cryptographic schemes taken from the literature; this serves as a practical motivation and we will return to the schemes during the course of the dissertation.

In Chapter 2 we arrange the mathematical background with particular focus on the Lichtenbaum-Tate pairing. Most of this chapter is taken from two interventions of Prof. Frey at the first Symposium on Algebraic Geometry and its Applications (SAGA 2007) and at the GTEM Workshop on Pairings (Essen, 2009). We adapt the theory to better fit the case of TZV.

Chapter 3 is the core of this dissertation. We take a more computational oriented approach: we discuss the arithmetic in the ideal class group of a TZV, we review techniques for pairing computation taken from the literature and we develop a new algorithm for the computation of the Tate pairing over supersingular TZV which exploits the action of the q^{th} power Frobenius endomorphism. In the end of the chapter we discuss the security of TZV and we provide explicit examples of curves that are used in the experiments.

In Chapter 4 we deal with the implementation details and we provide experimental results. For emotional reasons most of this chapter is devoted to the implementation in characteristic two. Notably, in this chapter we define new compressed López-Dahab coordinates for ordinary TZV of elliptic curves, we analyze the performance of scalar multiplication both on ordinary and supersingular TZV, we detail the parallel and the sequential versions of the new algorithm for the Tate pairing, including experimental results.

Finally, Chapter 5 is devoted to real-world applications and notably to the Trusted Computing technology: due to my personal and somehow atypical research experience at Politecnico di Torino, I have known a more applied way to do research and I wish to include in the dissertation some results obtained in this area. Pairing here is used as a black-box toolkit and actually these topics originated my interest in pairing computation. Unfortunately, due to technical details that will be clarified at proper time, the algorithm for pairing computation developed in this dissertation is probably not the best candidate for such applications, but the hope is to continue my research and let these two routes converge.

Reading Threads

Assuming that someone else other than reviewers could be interested in reading this work, I think is useful – because of the wide range of topics treated – to provide some reading threads, at least for the two main figures that I guess may never be interested in my thesis, that for simplicity I shall call mathematician and engineer.

Chapter 1 is thought for everyone, even non-technical people with a “small” background (to know what is cryptography for, what is a group, maybe an elliptic curve).

Then the mathematician hopefully will find interesting Chapter 2 (mostly thanks to Prof. Frey) and is encouraged to read Chapter 3. Finally, in Chapter 4 she may find some curiosities about technical details, like compressed López-Dahab coordinates or the new algorithm for the Tate pairing, or even some numbers coming from experimental results. Chapter 5 can be ignored at all.

On his side, the engineer must absolutely skip Chapter 2 and perhaps also Chapter 3. I hope he will find useful Chapter 4 for reference material and for the explicit algorithms for compressed López-Dahab coordinates and for Tate pairing in several flavours, as well as experimental data supporting the strength of TZV. Finally he is encouraged to read Chapter 5 about Trusted Computing technology, even if he doesn’t trust it.

Whatever will be your reading thread (and your maybe-forced motivation in reading this work) let me wish you a good reading... and of course a Merry Christmas and a Happy New Year!

21st of December 2009
Emanuele Cesena

Notation

In the following we present the notation and conventions adopted:

$\mathbb{Z},\ \mathbb{Q},\ \mathbb{R},\ \mathbb{C}$	Ring of integers; rational, real, complex fields
q	Power of prime, usually 2^m or 3^m with m prime, or p prime
$\mathbb{F}_q$	Galois field of order q
$r,\ \mathbb{F}_{q^r}$	Degree of the extension to build TZV, extension field
$k,\ \mathbb{F}_{q^k}$	Embedding degree, extension field to compute pairing
K	In Chapter 2: perfect field – In Chapter 3: either $\mathbb{F}_q$ or $\mathbb{F}_{q^r}$
$K^*,\ \overline{K}$	Multiplicative group, algebraic closure of K
$\mathrm{G}_K,\ \mathrm{G}(L/K)$	Absolute Galois group of K, Galois group of the extension L/K
$\mathcal{C}$	Complete, non-singular and absolutely irriducible curve
$\mathcal{E}$	Elliptic curve
$\mathcal{C}_O,\ O$	Non-singular, affine curve and its ring of holomorphic functions
$P = (x_P, y_P)$	Affine point of a curve, with coordinates (x_P, y_P)
$\mathcal{C}(K)$	Set of K-rationals points of the curve $\mathcal{C}$
$F = K(\mathcal{C})$	Field of functions of the curve $\mathcal{C}$
$\mathrm{Pic}^0(\mathcal{C})$	Picard group of the curve $\mathcal{C}$
$\mathcal{J}_\mathcal{C}$	Jacobian variety of the curve $\mathcal{C}$
$D,\ \overline{D},\ \langle u(x), v(x)\rangle$	A divisor D, the class of D, and the class represented by polynomials $u(x), v(x)$ (cf. Mumford representation)
σ	q^{th} power Frobenius automorphism of the field and q^{th} power Frobenius endomorphism of the curve
π	$(q^r)^{\mathrm{th}}$ power Frobenius automorphism of the field and $(q^r)^{\mathrm{th}}$ power Frobenius endomorphism of the curve
$\mathrm{Id},\ \mathrm{Tr}$	Identity map, trace map (of the field or of the curve)
$\mathbb{G},\ \mathbb{G}_1,\ \mathbb{G}_2,\ \mathbb{G}_T$	Groups (usually additive notation, except $\mathbb{G}_T$)
ℓ	Prime number, usually the size of the group(s)
$e\colon \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$	Pairing, i.e. non-degenerate, bilinear map
$W(\cdot,\cdot),\ T(\cdot,\cdot),\ \hat{t}(\cdot,\cdot)$	Weil, Tate and reduced Tate pairings
$f_{n,P}(\cdot)$	Miller function

Contents

List of Tables, Figures and Algorithms

List of Tables

List of Figures

List of Algorithms

Chapter 1

Pairing-based Cryptography

Cryptographers love tradition.
If we were to use "Andy" and "Barbara" as the principals,
no one would believe anything in this chapter.

Until the '70s, cryptographic algorithms were what today we define *symmetric-key* based and the purpose of cryptography was essentially to allow two people, which for us will be Alice and Bob, of exchanging private messages without a third person, the evil Eve, could decipher them once intercepted.

The term symmetric-key means that the information necessary to encrypt a message – the key – is also sufficient to decrypt a ciphertext and thus it must be kept secret. Furthermore, if Alice and Bob want to communicate in private, they have to exchange in a secure way such a key.

The state of affairs has been radically transformed by Diffie and Hellman, with the invention of *public-key cryptography.*

The key point is that if Bob wants to send a message to Alice, the latter shall publicly describe an algorithm for encryption, including her public key. So, even Eve is aware of how to encrypt a message, but this information is insufficient to decipher intercepted ciphertexts.

We must however point out that symmetric-key cryptography is not exceeded by public-key, rather it is still widely used because it is fairly more efficient.

A major step forward in public-key cryptography is the introduction of *pairing-based cryptography.* In 2000, several authors show that the Weil pairing can be used for "good" and start building cryptographic schemes based on pairing: Sakai, Ohgishi and Kasahara introduce the first pairing-based key-agreement and signature schemes, and Joux extends the Diffie-Hellman key agreement protocol to a three-party, one-round protocol.

Again, pairing is not a final solution by itself, but requires to operate with other cryptographic primitives, mainly due to efficiency reasons. Roughly speaking, if public-key cryptography is one order of magnitude slower than symmetric-key cryptography, then pairing looses another order of magnitude in performance with respect to public-key cryptography.

It is therefore important to restrict the use of pairing to the early and most critical stages of a protocol deployed in a real-world application and, at the same time, this motivates the research for ever more efficient implementations.

1.1 Applications of Pairing-based Cryptography

Public-key cryptography extended the application of cryptography, which originally was the sole protection of messages confidentiality through *encryption*, by introducing two new fundamental primitives: *key agreement*, to share symmetric keys, and *digital signature*, to protect the integrity, authenticate and provide non-repudiation of messages.

Pairing-based cryptography enhances the possibilities of public-key cryptography, by allowing efficient construction of several protocols for identity-based encryption, homomorphic encryption, group signatures, key sharing, signcryption, etc.

Inspired by Dutta et al. [DBS04], we give the following classification of these protocols. Another source of reference is The Pairing-Based Crypto Lounge [Bar].

Encryption.

– *Identity-based encryption (IBE):* In IBE, the public key distribution problem is eliminated by making each user's public key derivable from some known aspect of his identity. When Bob wants to send a message to Alice, he encrypts the message using Alice's public key which is derived from her identity (e.g. her name "Alice" or her e-mail address `alice@paesedellemeraviglie.it`). Alice obtains her private key from a third party called a Private Key generator (PKG) and, on receiving the encrypted message, she can decrypt it. Alice's private key that PKG generates is a function of a master key and Alice's identity. The PKG has the capability of key escrow, in the sense that it can generate any private key, and hence decrypt any ciphertext. Hierarchical IBE was proposed to limit the scope of key escrow in applications where it is not desirable.

– *Homomorphic encryption:* This is a form of encryption where one can perform a specific algebraic operation on the plaintext by performing a (possibly different) operation on the ciphertext. Homomorphic encryption schemes are malleable by design and are thus unsuited for secure data transmission. On the other hand, the homomorphic property of various cryptosystems can be used to create secure storage in database with (limited) search capabilities, secure voting systems, collision-resistant hash functions and private information retrieval schemes.

– *(Public-key) Broadcast encryption (BE):* It provides an efficient solution to broadcast encrypted information to many users. In a BE scheme, a center set-up the system, distribute keys to users and broadcast encrypted messages to a subset of authorized users. A public-key scheme allows anyone knowing the center and users' public keys to send messages to users, thus the center is no longer the sole source of broadcast messages.

Signature.

– *Short Signature:* Pairing allows the construction of signature schemes that produce shorter signatures than usual, around half of the size for equivalent security strength. These are required in environments with space and bandwidth constraints or when a human is asked to manually key in the signature.

– *Aggregate Signature:* Consider n users, each one having a public-private key pair and signing a (possibly different) message. A public aggregation algorithm outputs a compressed short signature on input all the users' signatures. This aggregation of

n signatures can be done by anyone. Additionally, there is an aggregate verification algorithm all the public keys, all the messages and the aggregate signature as input and decides whether the aggregate signature is valid. Aggregate signature schemes are used in secure border gateway protocols for compressing the list of signatures on distinct messages issued by distinct parties.

– *Group Signature:* Group signature permits any member of a group to sign on behalf of the group. Anyone can verify the signature with a group public key while no one can know the identity of the signer except a special entity called group manager. Group signature provides anonymity of users with the property that the group manager (or revocation authority) can identify the signer. In general, it is computationally hard to decide whether two different signatures were issued by the same member. This property, referred as partial anonymity or pseudonimity, is a feature of specific group signature schemes.

– *Proxy Signature:* A proxy signature allows an entity, called the delegator, to delegate its signing rights to another entity, called a proxy signer. The proxy signer signs messages on behalf of the delegator, in case of temporal absence, lack of time or computational power. Proxy signatures have found numerous practical applications where delegation of rights is quite common, particularly in distributed systems, grid computing, mobile agent applications, distributed shared object systems and mobile communications.

Key agreement.

Key agreement is required in situations where two or more parties want to communicate securely among themselves. The situation where three or more parties share a common key is often called conference keying. In this situation, the parties can securely send and receive messages among each other based on the agreed key.

Threshold.

Threshold cryptography approach is useful to remove single point failure or to avoid the centralization of the power. In a (t, n)-threshold scheme, $t \leq n$, a secret information is distributed among n users such that any subset of t or more users is allowed to reconstruct it. The computation is performed preserving security even in the presence of an active adversary that can corrupt up to $t - 1$ users.

Miscellaneous.

– *Chameleon Hash:* This is a hash function associated with a pair of public-private keys. Anyone who knows the public key can compute the associated hash function and, without the knowledge of associated private key, the chameleon hash function is collision resistant. However, the private key allows the holder to easily find collisions for every given input. Chameleon hashing is basically a non-interactive commitment scheme and find applications in constructing chameleon signatures, where the recipient can verify that the signature of a certain message M is valid, but can not prove that the signer actually signed M and not another message.

– *Signcryption:* It combines the functionality of signature and encryption in a more efficient manner than a straightforward composition of an encryption scheme with a signature scheme: the idea of signcryption scheme is to perform encryption and signature in a single logical step in order to obtain confidentiality, integrity, authentication and non-repudiation more efficiently than the sign-then-encrypt approach.

1.2 Diffie–Hellman and Tripartite Diffie–Hellman

In 1976, Diffie and Hellman describe a protocol that allows Alice and Bob to share a common secret key [DH76]. This protocol is extended in 2000 by Joux to a three-parties, one-round key agreement protocol [Jou00].

Let $(\mathbb{G}, +)$ be a group of order ℓ and fix a generator $P \in \mathbb{G}$.

(*setup*) Publicly describe $\mathbb{G}$ and $P \in \mathbb{G}$.

1. A and B select random integers a and $b \pmod{\ell}$.
2. They exchange respectively aP and bP.
3. They each obtain $K = abP$, by computing respectively $a(bP)$ and $b(aP)$.

Alice and Bob now share the secret element $K \in \mathbb{G}$.

Let us go back to Eve, so far left behind the scene. Her task is to reconstruct abP given P, aP and bP. This problem is known as the *Diffie-Hellman problem* (DHP). Note that if Eve could (efficiently) compute a given P and aP, she would solve DHP. This second problem is called the *discrete logarithm problem* (DLP).

In fact, significant progress has been made towards showing that over many groups the DHP is almost as hard as the DLP, but there is no proof to date that either the DHP (or the DLP) is a hard problem, except in generic groups.

A possible variation of DHP is the *computational Diffie-Hellman problem* (CDHP), i.e. given P, aP, bP and cP decide whether $cP = abP$ or not. Clearly solving DHP allows to solve CDHP. In the end of this section, we are going to show that these two problems are not equivalent in general.

Without going into further details, for a secure and efficient implementation, we require:

- For efficiency, the group operation in $\mathbb{G}$ must be computable in polynomial time.
- For security, the discrete logarithm in $\mathbb{G}$ should have higher complexity, possibly exponential.

Hence, we want a group $\mathbb{G}$ where the *scalar multiplication* is a so-called *one-way function*, that is to say a function easy to compute, but whose inverse is not.

It is worth noting that if $\mathbb{G}$ has order ℓ, given the factorization of ℓ and thanks to the Pohlig-Hellman algorithm [PH78], the DLP in $\mathbb{G}$ can be reduced in polynomial time to the DLP is the subgroups of prime order of $\mathbb{G}$. Hence we can assume ℓ is prime.

We now let pairing come into the game and describe Joux's three-parties, one-round key agreement protocol. Let $(\mathbb{G}_T, \cdot)$ be a group of order ℓ and suppose there exists an efficiently computable map:

$$e \colon \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T \ ,$$

with the following properties:

1. (Non-degeneracy) there exist $P,\ Q \in \mathbb{G}$ such that $e(P,Q) \neq 1_{\mathbb{G}_T}$.
2. (Bilinearity) for every $P,\ Q \in \mathbb{G}$: $e(aP, bQ) = e(P,Q)^{ab}$ for all $a, b \in \mathbb{Z}$.

Alice, Bob and Charlie want to set up a common key. Without pairing, they can use a conference keying protocol (e.g. Burmester and Desmedt's protocol [BD95]) which requires two rounds of interaction. With pairing a single round is sufficient:

(*setup*) Publicly describe $\mathbb{G}$, $P \in \mathbb{G}$, $\mathbb{G}_T$ and $e(\cdot,\cdot)$.

1. A, B and C select random integers a, b and $c \pmod{\ell}$.
2. They respectively broadcast aP, bP and cP.
3. They each obtain $k = e(P,P)^{abc}$, by computing respectively $e(bP,cP)^a$, $e(aP,cP)^b$ and $e(aP,bP)^c$.

The element $k \in \mathbb{G}_T$ is used as common secret.

To break the protocol security, Eve should compute $e(P,P)^{abc}$ given P, aP, bP and cP. This is called the *bilinear Diffie-Hellman problem* (BDHP). It is easy to show that solving DHP allows to solve BDHP: if one can compute abP given P, aP and bP, then she can pair $e(abP, cP) = e(P,P)^{abc}$. Currently the equivalence between BDHP and DHP is unknown.

Since in this work we will also discuss real-world applications, it is important to remark that Joux's protocol is only secure against passive attackers, while it is vulnerable to man-in-the-middle attack conducted by an active adversary because it is not authenticated: the purpose of this introductory description is just to warm up and get familiar with pairing.

Before concluding this section, we show that pairing allows to easily solve the decisional Diffie-Hellman problem: given P, aP, bP and cP, one can decide whether $cP = abP$ by checking if $e(P, cP) = e(aP, bP)$, which holds if and only if $c = ab$. Thus we have a group $\mathbb{G}$ where we can assume that the DHP is hard, while the DDHP is easy to solve. In Section 1.5.1 we will show that such a group allows an efficient signature scheme. But before continuing with applications, it is important to present in more detail the group $\mathbb{G}$, its history and pairing as well.

1.3 The Group $\mathbb{G}$: Elliptic Curves and Pairings

We now turn the attention to the group $\mathbb{G}$. The most simple choice (the same proposed by Diffie and Hellman) is to select the multiplicative group of a finite field $\mathbb{F}_q$.

In 1985, Miller and Koblitz [Mil86a, Kob87] propose to use the group of rational points of an elliptic curve $\mathcal{E}/\mathbb{F}_q$. They identify two advantages: (1) a greater flexibility in the choice of the group (in the sense that, fixed q a power of a prime, there exists a unique multiplicative group $\mathbb{F}_q^*$, but many elliptic curves $\mathcal{E}/\mathbb{F}_q$) and (2) the absence of a sub-exponential algorithm for DLP, provided that $\mathcal{E}$ and q are properly chosen.

Working with elliptic curves allows to reduce the size of the field; to give a comparison the security provided by the multiplicative group of a field of 1248 bits is equivalent to that of an elliptic curve defined over a field of 160 bits (cf. [ECR09] and Section 3.7).

Originally cryptosystems based on elliptic curves were defined on supersingular curves. This was mainly due to the difficulty of constructing ordinary curves with a number of points of the required size, and to the more efficient arithmetic that supersingular curves provide with respect to ordinary ones.

In 1993, Menezes, Okamoto and Vanstone [MOV93] found an attack, called the MOV attack, which reduces the DLP on an elliptic curve to the DLP in a finite field. More precisely, if $\mathcal{E}$ is an elliptic curve over a field $\mathbb{F}_q$, the DLP in $\mathcal{E}(\mathbb{F}_q)$ can be reduced to the DLP in (the multiplicative group of) an appropriate extension of $\mathbb{F}_q$. The degree of such an extension is called *embedding degree* and usually denoted by k.

Let us try to understand with an example the importance of the embedding degree k for the security of a cryptosystem. Consider a field $\mathbb{F}_q$ with $q \approx 2^{160}$ and an elliptic curve $\mathcal{E}/\mathbb{F}_q$. As already mentioned, this provides a security equivalent to that of a finite field $\mathbb{F}_{q'}$, with $q' \approx 2^{1248}$. Exploiting the MOV attack, the DLP in $\mathcal{E}(\mathbb{F}_q)$ can be reduced to the DLP in $\mathbb{F}^*_{q^k}$. So, if $\mathbb{F}_q$ has order 2^{160}, the extension order has $2^{160\times k}$ and to ensure the security of our system we must have $160 \times k > 1248$, i.e. about $k > 6$.

The MOV attack exploits the *Weil pairing* (for the definition see, e.g., [Sil86, see Section III.8]) which is a bilinear, non-degenerate map $W(\cdot,\cdot)$ that can be efficiently computed with Miller's algorithm [Mil86b, Mil04]. Suppose one has to compute a, given P and aP. Fixed a point Q such that $W(P,Q) \neq 1$, she can compute $\alpha = W(P,Q)$ and $\beta = W(aP,Q)$. Hence α, $\beta \in \mathbb{F}^*_{q^k}$ and, by bilinearity, $\beta = \alpha^a$: by solving the DLP in $\mathbb{F}^*_{q^k}$ one can find a, i.e. solve the DLP on the elliptic curve.

In 1994, Frey and Ruck [FR94] show that the *Lichtenbaum-Tate pairing*, which is another bilinear and non-degenerate map, provides an alternative to the Weil pairing particularly interesting from the computational point of view.

In summary, the MOV attack exploits the Weil (or the Tate) pairing to reduce the DLP from $\mathcal{E}(\mathbb{F}_q)$ to $\mathbb{F}^*_{q^k}$, and k must be large enough, so that this attack does not lead to security losses.

Luckily, for a "random" elliptic curve $\mathcal{E}/\mathbb{F}_q$, k is approximately q, hence the MOV attack is not an issue. On the contrary, for supersingular elliptic curves it is known that $k \leq 6$, and for this reason supersingular curves were soon abandoned, because considered less secure. The abandonment was also supported by the research on point counting on elliptic curves: the Schoof's algorithm [Sch95] and subsequent variants have helped to efficiently build ordinary elliptic curves suitable for cryptographic applications.

The birth of pairing-based cryptography represents the revenge of supersingular elliptic curves: again, the research has originated in the supersingular world, which is more efficient and in some sense easier to deal with.

Next, the research developed in several directions: generalizing elliptic curves to Jacobians of hyperelliptic curves or abelian varieties [RS02, BK+02], and extending the definition of pairing to ordinary curves [HSV06]. We can observe a similar development as for the history of cryptography based on DLP.

The definition of pairing on ordinary curves has opened an important chapter of research on the construction of pairing-friendly curves, i.e. curves with a prescribed and relatively small embedding degree [FST06].

1.4 Types of Pairings

In Section 1.2 we gave a first definition of pairing, i.e. a bilinear, non-degenerate and efficiently computable map $e\colon \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$. Such a definition is usually referred as *symmetric pairing*, because pairing takes both arguments in $\mathbb{G}$.

In practice, symmetric pairings can be instantiated by using suitable supersingular curves or abelian varieties. However, in order to allow a wider range of curves to be used, the definition must be enhanced.

According to Galbraith et al. [GPS08], we define pairing as a map:

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

where $\mathbb{G}_1$, $\mathbb{G}_2$ and G_T are three groups of prime order ℓ, and we require:

1. (Non-degeneracy) $e(P,Q) \neq 1_{\mathbb{G}_T}$ for all $Q \in \mathbb{G}_2$ if and only if $P = 0_{\mathbb{G}_1}$, and similarly $e(P,Q) \neq 1_{\mathbb{G}_T}$ for all $P \in \mathbb{G}_1$ if and only if $Q = 0_{\mathbb{G}_2}$.
2. (Bilinearity) for every $P \in \mathbb{G}_1$ and $Q \in \mathbb{G}_2$: $e(aP,bQ) = e(P,Q)^{ab}$ for all $a,b \in \mathbb{Z}$.

Note we do not consider here pairings defined on groups with composite order, e.g. RSA moduli.

Finally, we classify pairings in three types:

Type 1: $\mathbb{G}_1 = \mathbb{G}_2$, i.e. symmetric pairing.

Type 2: $\mathbb{G}_1 \neq \mathbb{G}_2$, but there is an efficiently computable homomorphism $\phi\colon \mathbb{G}_2 \to \mathbb{G}_1$.

Type 3: $\mathbb{G}_1 \neq \mathbb{G}_2$, and there are not efficiently computable homomorphisms between $\mathbb{G}_1$ and $\mathbb{G}_2$.

In Type 1 we include the case where an efficiently computable homomorphism exists between $\mathbb{G}_2$ and $\mathbb{G}_1$ and also its inverse is efficiently computable. This is in fact the case for supersingular elliptic curves. Let $\mathcal{E}/\mathbb{F}_q$ be an elliptic curve and let $\mathbb{G}_1$ be the subgroup of $\mathcal{E}(\mathbb{F}_q)$ of order ℓ. It is known that the ℓ-torsion group $\mathcal{E}[\ell] \simeq \mathbb{Z}/\ell\mathbb{Z} \times \mathbb{Z}/\ell\mathbb{Z}$ and, without loss of generality, we can take the first component to be $\mathbb{G}_1$. Let $\mathbb{G}_2$ be the other component[1]. If $\mathcal{E}$ is supersingular, a *distortion map* $\psi\colon \mathbb{G}_1 \to \mathbb{G}_2$ allows to identify the two components of the ℓ-torsion group, and hence a symmetric pairing can be defined:

$$e(P,Q) = W(P,\psi(Q)) \ ,$$

where W is the Weil pairing. Note that, without the intervention of the distortion map, the Weil pairing restricted to $\mathbb{G}_1 \times \mathbb{G}_1$ would be degenerate. A similar construction can be done with the Tate pairing as well.

Type 2 pairings have been introduced as they allow a greater variety of pairings to be used, notably those constructed on ordinary curves. Additionally, because of the map ϕ, security proofs usually require only minimal changes with respect to symmetric pairings. Unfortunately, it turns out that there is no efficient and known method to hash to an element of $\mathbb{G}_2$, and this may be required in some practical applications.

Finally, Type 3 pairings permit hashing to any element of any group including $\mathbb{G}_2$. Additionally, their more flexible nature allows a greater number of assumptions which in turn allow more cryptosystems to be built. Furthermore, there exist optimal constructions of pairing-friendly elliptic curves, for instance Barreto-Naehrig curves [BN05].

In this work we concentrate on symmetric pairing defined on supersingular abelian varieties, but we shall in fact distinguish between $\mathbb{G}_1$ and $\mathbb{G}_2$ – even if a distortion map allows to identify the two groups – to give a better understanding of what the two groups precisely are.

[1]In the sequel, we are going to describe these two components in more detail.

1.5 Cryptographic Schemes Based on Pairing

This section provides a bridge between the mathematical notion of pairing that will be the core of this dissertation and real-world applications presented in Chapter 5.

We present two concrete schemes built on pairing: the Boneh, Lynn and Shacham's short signature scheme [BLS02] and the Boneh, Gentry and Waters' broadcast encryption scheme [BGW05].

The former, together with identity-based encryption [BF01], is probably the most significant advance in cryptography due to the pairing: it allows to halve the size of a digital signature and it is the foundation for several variant of signature schemes. Among all, group signatures can be used as a building block for an anonymous authentication system; in Section 5.3 we present a concrete implementation of an anonymous authentication system based on the Transport Layer Security [DR08] and Direct Anonymous Attestation [BCC04] protocols.

Broadcast encryption provides an efficient solution to broadcast encrypted information to many users and pairing is the toolkit that allowed the migration from symmetric to public-key cryptography: the use of identity-based encryption was initially proposed for such a migration by Dodis and Fazio [DF03], then several scheme built on pairing were defined. Here we describe a quite basic construction, which has been improved in recent years (e.g. [BW06, AKS09]). The cryptographic solution provided by broadcast encryption protects information on the communication channel, but in actual application the users' platforms represent a major issue. In Section 5.2 we propose a solution based on Trusted Computing and we use the scheme presented here as a reference.

1.5.1 Short Signature

We have seen in the end of Section 1.2 that if a cyclic group $\mathbb{G}$ is endowed with a bilinear map, then we have a group where the DHP is thought to be hard, yet the DDHP variant is easy to solve. This group implies a signature scheme, often referred to as the BLS (short) signature scheme [BLS02]:

(*setup*) Choose a group $\mathbb{G}$ of prime order ℓ. Publish a generator $P \in \mathbb{G}$.

(*key gen.*) Choose a random $a \pmod{\ell}$. Output the public key $A = aP$ and the private key a.

(*sign*) Given a message $H \in \mathbb{G}$, output $\sigma = aH$.

(*verify*) Given a message-signature pair (H, σ) and public key A, check that $e(P, \sigma) = e(A, H)$.

The simple abstract definition of pairing allows to proof that this signature scheme is secure against existential forgery under a chosen-message attack in the random oracle model, assuming DHP problem is hard in $\mathbb{G}$.

The major contribution of this scheme is however the size of the signature, that requires a concrete description of the pairing to be understood.

Let $\mathbb{G}$ be the set of rational points of an elliptic curve $\mathcal{E}/\mathbb{F}_q$. Hence the signature σ is a $\mathbb{F}_q$-rational point of $\mathcal{E}$, that can be represented by its x-coordinate (plus 1 bit to distinguish between σ and $-\sigma$). On the contrary, in other signature schemes like ECDSA [ANS05] a signature is a pair of elements of the field, thus double in length.

This signature scheme in fact has other useful properties, including batched verification, and allows the simple construction of aggregate, group and verifiably-encrypted signatures [BG+03]. Among these, group signatures are an interesting tool to build anonymous authentication systems.

1.5.2 Broadcast Encryption

Broadcast encryption allows to efficiently and securely send a plaintext M to many users. The corresponding ciphertext usually looks like a pair $(\mathrm{Hdr}, \{M\}_K)$: the plaintext is encrypted by means of symmetric-key cryptography with a session key K and a header Hdr is put before allowing legitimate users to recover the key K and hence to decrypt the message.

A trivial way to set up a broadcast encryption scheme is to provide each user i with a symmetric key k_i and to form Hdr by encrypting the session key K for each user allowed to receive the message. The header in this case is quite large and, even bad, its size grows linearly with the number of users n. Several works in literature face the problem to limit the size of the header, for instance by increasing the number of keys assigned to each user. We refer, e.g., to [NP00, HS02].

Pairing allows a very efficient construction, where both the header and the private key of each user have constant size [BGW05]:

(*setup*) This algorithm gets in input the number of users n and proceeds as follows.

Choose a group $\mathbb{G}$ of prime order ℓ. Pick a random generator $P \in \mathbb{G}$ and a random integer $\alpha \pmod{\ell}$. Compute $P_i = \alpha^i P$ for $i = 1, 2, \ldots, n, n+2, \ldots, 2n$. Next choose a random $\gamma \pmod{\ell}$ and set $V = \gamma P$. The public key is

$$PK = (P, P_1, \ldots, P_n, P_{n+2}, \ldots, P_{2n}, V) \in \mathbb{G}^{2n+1} \quad .$$

The private key for user $i \in \{1, \ldots, n\}$ is $D_i = \gamma P_i$. Note that $D_i = \alpha^i V$. The algorithm outputs the public key PK and the n private keys $D_1, \ldots, D_n$.

(*encrypt*) Given a subset of users S and a public key PK, pick a random integer $t \pmod{\ell}$ and set $K = e(P_{n+1}, P)^t$. Next set:

$$\mathrm{Hdr} = \left(tP \ \ , \ \ t(V + \sum_{j \in S} P_{n+1-j}) \right) \in \mathbb{G}^2 \quad .$$

Output the pair (Hdr, K): Hdr is the header of the message and K is the (symmetric) key to encypt the plaintext.

(*decrypt*) Given a user $i \in S$ with private key P_i, given the public key PK and a message header $\mathrm{Hdr} = (C_0, C_1)$, output the (symmetric) key:

$$K = e(P_i, C_1) \ / \ e\left(D_i + \sum_{j \in S, j \neq i} P_{n+1-j+i} \ \ , \ \ C_0 \right) \quad .$$

This scheme is correct and fully collusion resistant against chosen-ciphertext attacks (i.e. non-adaptive adversaries), assuming the *bilinear Diffie-Hellman Exponent problem* is hard: given as input a vector of $2n + 1$ elements

$$(Q, P, \alpha P, \alpha^2 P, \ldots, \alpha^n P, \alpha^{n+2} P, \ldots, \alpha^{2n} P) \in \mathbb{G}^{2n+1} \quad ,$$

output $e(P,Q)^{\alpha^{n+1}}$. Note that the input vector lacks the term $\alpha^{n+1}P$, so that the bilinear map seems to be of little help in computing the required value.

We can see that private keys are only 1 group element and the header Hdr has constant size (2 elements in $\mathbb{G}$). The drawback is that the size of the public key is $O(n)$.

We note that, in the encryption algorithm, $e(P_{n+1}, P)$ can be computed as $e(P_n, P_1)$ and stored, so encryption requires no pairing. Furthermore, if the set of legitimate users S is quite large, the computation time is dominated by the large number of operations in $\mathbb{G}$ needed to compute the sum (both in the encryption and decryption). However it is possible to cache some partial sums and speed up the overall computation. For further details we refer to the original paper [BGW05].

As already mentioned, this is the first example of broadcast encryption scheme with short ciphertext and private keys, but the price is to have a very big public key. In recent years several extensions were proposed, for instance [BW06, AKS09].

1.6 Trace Zero Varieties in Cryptography

In this section we introduce in a quite informal way Trace Zero Varieties (TZV) and we present the main reasons motivating the research on this field. Before concluding – the section and the chapter – we will be ready to state the objectives of the present work.

In 1998 Frey [Fre98, Fre01] suggests to use TZV for cryptographic applications because their properties can be exploited to implement fast arithmetic, while the hardness of the DLP can be reduced to the hardness of the DLP in other known groups which are used in practice.

Roughly speaking, if $\mathcal{C}$ is a hyperelliptic curve of genus g defined over a finite field of q elements $\mathbb{F}_q$, the *trace zero (sub)subgroup of $\mathcal{C}$ over a field extension of degree r* is a subgroup $\mathbb{G}$ of the Jacobian variety $\mathcal{J}_{\mathcal{C}}(\mathbb{F}_{q^r})$ of $\mathcal{C}$ over $\mathbb{F}_{q^r}$, which is isomorphic to the quotient group $\mathcal{J}_{\mathcal{C}}(\mathbb{F}_{q^r})/\mathcal{J}_{\mathcal{C}}(\mathbb{F}_q)$. It is possible to prove that it is a codimension g subvariety of the Weil restriction of scalars of $\mathcal{J}_{\mathcal{C}}$ from $\mathbb{F}_{q^r}$ to $\mathbb{F}_q$, whence the name *Trace Zero Variety*.

The group used for cryptographic applications is a subgroup $\mathbb{G}_1$ of $\mathbb{G}$ of large prime order ℓ and of small index in $\mathbb{G}$. We can assume that $\mathbb{G}_1$ is the only subgroup of order ℓ of $\mathbb{G}$, thus the Frobenius endomorphism σ of the Jacobian induced by the relative Frobenius automorphism of the field extension $\mathbb{F}_{q^r}/\mathbb{F}_q$ acts on $\mathbb{G}_1$ as a group automorphism. Since $\mathbb{G}_1$ is cyclic, σ acts as a scalar multiplier, i.e. for each $D \in \mathbb{G}_1$ we have $\sigma(D) = sD$ for some $s \in \mathbb{Z}$ (that depends on $\mathbb{G}_1$). Similarly to the setting of Koblitz curves [Kob91, Sol00], the Frobenius endomorphism can be used to speed up the scalar multiplication in $\mathbb{G}_1$. Another advantage that TZV share with Koblitz curves is that it is quite easy (from a computational point of view) to determine the cardinality of the group of rational points.

Several authors address the study of TZV: Naumann [Nau99] and Blady [Bla02] consider TZV of elliptic curves over extension fields of degree 3 ($r = 3$); Weimerskirch [Wei01] analyzes the case for extension fields of degree 5; finally Lange [Lan01, Lan04] builds TZV from the Jacobian variety of hyperelliptic curves of genus two, over extension fields of degree 3. From a cryptographic point of view, these are the only relevant cases, since for higher extension or higher genus the groups obtained are susceptible to Weil descent attacks [Lan01].

Avanzi and Lange [AL04] compare the performance of these three kinds of TZV over fields of odd characteristic. Avanzi and Cesena [Ces04, AC08b] compare the same three types of TZV defined over binary fields, underlining similarities and main differences between TZV defined over fields of even and odd characteristic.

As early as 2002, Rubin and Silverberg [RS02] propose to use supersingular abelian varieties of dimension greater than one to improve the security of pairing-based cryptosystems. Besides Jacobian varieties of hyperelliptic curves of genus greater than one, the other significant example is the class of TZV (called primitive subgroups in that paper), which can be constructed from elliptic curves.

The original work of Rubin and Silverberg and their more recent results presented in [RS09] constitute the motivation of our research. Notably, supersingular TZV of elliptic curves allow to achieve higher "security per bit" than supersingular elliptic curves themselves: in characteristic 3 ($r = 5$) TZV represent the first example of supersingular abelian varieties with security parameter greater than 6 (in fact 7.5); in characteristic 2 ($r = 3$) TZV present an alternative to supersingular *elliptic curves over* $\mathbb{F}_{3^m}$ which is more efficient, simpler to implement and with equivalent security properties.

The computation of pairing over TZV is already taken into account by Barreto et al. [BK⁺02, BG⁺07], who define the η and η_T pairings on supersingular abelian varieties. Other pairings, such as the (twisted) Ate pairing [HSV06] and its extended versions [MK⁺07, LLP09, Ver08] can be naturally defined on TZV too. All these pairings exploit the action of the $(q^r)^{\text{th}}$ power Frobenius endomorphism π and they can in fact be defined not only on the TZV, but on the whole $\mathcal{J}(\mathbb{F}_{q^r})$.

The main purpose of this work is to derive a new algorithm for computing the Tate pairing over supersingular TZV that exploits the action of the q^{th} power Frobenius endomorphism σ, allowing for a finer "splitting" of the computation in several independent execution threads.

This work not only focus on pairing, but also reviews results on "classical" DLP-based cryptography, notably to achieve two objectives: extend the work to supersingular TZV, showing that they outperform elliptic curves in actual applications also because of a more efficient arithmetic in $\mathbb{G}_1$; update previous results obtained in the settings of ordinary curves, by analyzing better coordinates systems than the sole affine coordinates. In this context we also introduce a new coordinate system: the compressed López-Dahab coordinates.

Beside the theoretical discussion, implementation takes a relevant place: in fact a new system, no matter how clever it is, will not attract the attention of the cryptographic community unless it achieves better performance, for a comparable security level, than already established systems.

Chapter 2

Background

In this chapter we introduce the Lichtenbaum-Tate pairing on Jacobian varieties of hyperelliptic curves.

The core of the chapter is inspired by two interventions by prof. Frey at the first SAGA conference (Papeete, 2007) [Fre08] and at the GTEM Workshop on Pairings (Essen, 2009). The material was supplemented with the Handbook of Elliptic and Hyperelliptic Curve Cryptography [AC+05]. For background on curves, we adopt the notation of Lorenzini [Lor96] and we refer to Lange's Ph.D. dissertation [Lan01] for an extended discussion.

Through this chapter, K will be a perfect field, $\overline{K}$ its algebraic (and separable) closure and $\mathrm{G}_K = \mathrm{Aut}_K(\overline{K})$ the absolute Galois group. We will focus on a complete (often projective), non-singular and absolutely irreducible curve $\mathcal{C}/K$, with at least one K-rational point. We denote with $K(\mathcal{C})$ its function field.

Our destination point will be the definition of the Lichtenbaum-Tate pairing on the Jacobian of a curve $\mathcal{C}_O$ defined over a finite field $\mathbb{F}_q$, but to achieve our purpose we need to consider its lift $\mathcal{C}$ over a local field K with residue field $\mathbb{F}_q$. Since this is our direction, we additionally assume that $\mathcal{C}$ has good reduction, i.e. $\mathcal{C}_O$ is also non-singular.

It is important to make a clarification regarding the Frobenius endomorphism: in this, but especially in the next chapter, we are dealing with two instances of the Frobenius, the q^{th} power Frobenius σ and the $(q^r)^{\text{th}}$ power Frobenius π (where r is a small prime number). The former is specifically used to build trace zero varieties and, if we just consider this chapter, it will be used in Sections 2.1 and 2.2. In the remainder of the chapter, we will make use of π but, with abuse of notation, we shall call it the q^{th} power Frobenius endomorphism too. This allows us to introduce the Lichtenbaum-Tate pairing – without unduly complicating the notation – on the Jacobian of $\mathcal{C}_O/\mathbb{F}_q$. Soon thereafter we shall change such a Jacobian in a trace zero variety, so π will honestly become the $(q^r)^{\text{th}}$ power Frobenius endomorphism. That is the reason to have two distinct names for, at least in this chapter, the same object, i.e. the Frobenius.

2.1 Hyperelliptic Curve in Cryptography

Let K be a perfect field. A curve $\mathcal{C}/K$ is called *hyperelliptic curve* if it is not the projective line and the corresponding function field $K(\mathcal{C})$ contains at least an element x such that $K(\mathcal{C})/K(x)$ is a Galois extension of degree 2.

If L/K is a field extension, we denote with $\mathrm{G}(L/K)$ the Galois group of L/K and with

$\mathcal{C}_L/L$ the *extension of scalars* of $\mathcal{C}/K$ to L/K. In particular we denote $\overline{\mathcal{C}}/\overline{K} := \mathcal{C}_{\overline{K}}/\overline{K}$.

The free abelian group generated by the L-rational points of $\mathcal{C}_L/L$ is called *divisors group* of $\mathcal{C}$ over L/K and denoted by $\mathrm{Div}(\mathcal{C}_L/L)$. It is possible to associate to every function $f \in L(\mathcal{C})$ a divisor denoted (f). Divisors coming from a function are called *principal divisors* and their set $\mathrm{Princ}(\mathcal{C}_L/L)$ is a normal subgroup of $\mathrm{Div}(\mathcal{C}_L/L)$. The *Picard group* or *divisors class group* is the quotient group of the divisors group modulo the subgroup of principal divisors and it is denoted by $\mathrm{Pic}(\mathcal{C}_L/L)$.

To simplify the notation, we shall omit the field of definition when it is clear, so $\mathrm{Pic}(\mathcal{C})$ refers to $\mathrm{Pic}(\mathcal{C}/K)$, $\mathrm{Pic}(\mathcal{C}_L)$ refers to $\mathrm{Pic}(\mathcal{C}_L/L)$, $\mathrm{Pic}(\overline{\mathcal{C}})$ refers to $\mathrm{Pic}(\mathcal{C}_{\overline{K}}/\overline{K})$ and similarly for other groups as Div and Princ.

In cryptography, we are interested in the subgroup $\mathrm{Pic}^0(\mathcal{C})$ of the divisors with degree 0 modulo principal divisors. If K is a finite field, then $\mathrm{Pic}^0(\mathcal{C})$ is a finite group and its order is called *class number of* $\mathcal{C}/K$.

Denote with $\mathcal{O}_P$ the ring of K-rational functions defined in P, with maximal ideal $\mathcal{M}_P$. Let B be a Dedekind domain and assume $K \subset B$. Denote with $\mathrm{Div}(B)$ the group of divisors of B, i.e. the group of divisors of its quotient field over K. Let $\mathrm{Cl}(B)$ be the group of ideal class of B. Define $U = \{P \in \mathcal{C}(K) \mid \mathcal{O}_P \subset B\}$.

We have the following commutative diagram with exact rows:

$$\begin{tikzcd}
(1) \arrow[r] & K^* \arrow[r] \arrow[d] & K(\mathcal{C})^* \arrow[r, "\mathrm{div}"] \arrow[d, equal] & \mathrm{Div}^0(\mathcal{C}) \arrow[r] \arrow[d, "\mathrm{res}"] & \mathrm{Pic}^0(\mathcal{C}) \arrow[r] \arrow[d, "\varphi"] & (0) \\
(1) \arrow[r] & B^* \arrow[r] & K(\mathcal{C})^* \arrow[r, "\mathrm{div}_B"] & \mathrm{Div}(B) \arrow[r] & \mathrm{Cl}(B) \arrow[r] & (0)
\end{tikzcd}$$

Where we denoted with φ the map between $\mathrm{Pic}^0(\mathcal{C})$ and $\mathrm{Cl}(B)$, explicitly given by:

$$\varphi\colon \mathrm{Pic}^0(\mathcal{C}) \to \mathrm{Cl}(B), \quad \text{class of} \sum_{P \in \mathcal{C}(K)} a_P P \mapsto \prod_{P \in U} (\text{class of } \mathcal{M}_P \cap B)^{a_P} \ .$$

If φ is bijective, we can identify the two groups. This is the most interesting case for applications and we simply note that the curves we shall consider belong to this situation. We will use such a correspondence between $\mathrm{Pic}^0(\mathcal{C})$ and $\mathrm{Cl}(B)$ to obtain an efficient arithmetic, since the operation of multiplication of ideals can be made explicit by operations in the ring of polynomials $K[X, Y]$.

From the Riemann-Roch theorem we can define the *genus of* $\mathcal{C}$, denoted with g. A complete, non-singular curve of genus 1 is called an *elliptic curve*. An important property of the genus is that it is invariant with respect to extension of scalars.

Let now $K = \mathbb{F}_q$ be a finite field. The q^{th} power Frobenius automorphism

$$\mathrm{G}_K \ni \sigma\colon \overline{K} \to \overline{K}, \ a \mapsto a^q$$

induces an endomorphism of $\mathcal{C}$ of degree q that we shall also call *Frobenius endomorphism* and denote with σ. It extends to $\overline{\mathcal{C}}$, as well as to the Picard group $\mathrm{Pic}^0(\mathcal{C})$.

The Frobenius endomorphism satisfies its characteristic polynomial $\chi(T) \in \mathbb{Z}[T]$ and the Hasse-Weil theorem relates the complex roots of χ with the class number of $\mathcal{C}$ over any extension $\mathbb{F}_{q^r}/\mathbb{F}_q$. In the following theorem we resume some important properties of χ.

Theorem 2.1. *Let the factorization of* $\chi(T)$ *over* $\mathbb{C}$ *be:* $\chi(T) = \prod_{i=1}^{2g}(T - \tau_i)$. *Then:*

1. *The roots of* χ *satisfy* $|\tau_i| = \sqrt{q}$.

2. *For every integer r, denoted M_r the number of $\mathbb{F}_{q^r}$-rational points of $\mathcal{C}$, we have:*

$$M_r = q^r + 1 - \sum_{i=1}^{2g} \tau_i^r \quad \text{and} \quad |M_r - (q^r + 1)| \leq g\lfloor 2q^{r/2} \rfloor \ .$$

3. *$\chi(T)$ has the following form:*

$$T^{2g} + a_1 T^{2g-1} + a_2 T^{2g-2} + \cdots + a_g T^g + q a_{g-1} T^{g-1} + \cdots + q^{g-1} a_1 T + q^g \ .$$

Setting $a_0 = 1$,

$$a_j = \frac{\sum_{i=1}^{j} \big(M_i - (q^i + 1)\big) a_{j-i}}{j} \ , \qquad \text{for } 1 \leq j \leq g \ .$$

4. *For every integer r, we have:*

$$N_r := \big|\mathrm{Pic}^0(\mathcal{C}_{\mathbb{F}_{q^r}}/\mathbb{F}_{q^r})\big| = \prod_{i=1}^{2g}(1 - \tau_i^r) \quad \text{and} \quad (q^{r/2} - 1)^{2g} \leq N_r \leq (q^{r/2} + 1)^{2g} \ .$$

Hence, knowing the first g numbers M_j we can obtain the full polynomial $\chi(T)$ and thus the class number N_r over any extension, in particular $N_1 = \chi(1)$.

2.2 Jacobian and Trace Zero Variety

Let K be a perfect field and let $\mathcal{C}/K$ be a complete, non-singular curve of genus $g \geq 1$.

If $\mathcal{C}(K)$ is not empty, the choice of a point $P_0 \in \mathcal{C}(K)$ allows to define an injective map $\mathcal{C}(\overline{K}) \to \mathrm{Pic}^0(\overline{\mathcal{C}})$, sending P into the ideal class of $P - P_0$.

Such a map is a bijection if $g = 1$, hence we can identify the group $\mathrm{Pic}^0(\overline{\mathcal{C}})$ with the points of an algebraic variety, the curve $\mathcal{C}$ itself. In general, for every curve $\mathcal{C}$, the group $\mathrm{Pic}^0(\overline{\mathcal{C}})$ can be identified in a functorial way with the set of K-rational points of an abelian variety.

More precisely, it is possible to associate to a curve $\mathcal{C}/K$, in a functorial way, an abelian variety $\mathcal{J}_{\mathcal{C}}/K$ (or $\mathcal{J}/K$ for short) of dimension g, defined over K and called *Jacobian Variety of $\mathcal{C}/K$*, or simply *Jacobian of $\mathcal{C}/K$*, such that:

1. If $K \subseteq L \subseteq \overline{K}$, the set of L-rational points of the Jacobian $\mathcal{J}(L)$ is in bijection with $\mathrm{Pic}^0(\overline{\mathcal{C}})^{\mathrm{G}_L}$.

2. Given $P_0 \in \mathcal{C}(K)$, there exists a morphism of algebraic varieties $\mathcal{C} \to \mathcal{J}$, defined over K, sending P_0 into the neutral element of $\mathcal{J}$. This morphism induces the map $P \mapsto$ class of $(P - P_0)$, over the set of $\overline{K}$-rational points.

In the sequel, with a slightly abuse of notation, we shall indistinctly refer to the group $\mathrm{Pic}^0(\mathcal{C}_L)$ or the Jacobian $\mathcal{J}(L)$.

We now define an endomorphism which will be fundamental for our purposes. Let us fix an extension $L = \mathbb{F}_{q^r}$ of degree r over K and denote with σ the (restriction of the) q^{th} power Frobenius endomorphism in $\mathcal{C}(L)$. In this case $\sigma^r = \mathrm{Id}$.

Definition 2.2. *The* trace $\mathrm{Tr} = \mathrm{Tr}_L \colon \mathcal{C}(L) \to \mathcal{C}(L)$ *is the map given by*

$$\mathrm{Tr} := \sum_{i=0}^{r-1} \sigma^i = \mathrm{Id} + \sigma + \cdots + \sigma^{r-1} \quad .$$

As direct consequence of the definition, the trace extends to the Picard group $\mathrm{Pic}^0(\mathcal{C}_L)$. The kernel of the trace map $\mathbb{G} := \mathrm{Ker}\,\mathrm{Tr}$ is a subgroup of the Picard group, the *trace zero subgroup*.

Actually, from the implementation perspective, this definition would be enough because the arithmetic in the trace zero subgroup is performed with the arithmetic in the full Picard group. It is an open problem to find explicit formulae for the arithmetic in the trace zero subgroup, which are more efficient than the ones available for the full Picard group. For completeness, in the sequel we are going to show that $\mathbb{G}$ is also (isomorphic to) a variety, whence the name "Trace Zero Variety".

Let $\mathcal{J}/L$ a variety on L. Clearly, we are particularly interested in the case where $\mathcal{J}$ is the Jacobian of a curve $\mathcal{C}/L$. We restrict ourselves to the case of affine varieties, but all the discussion can be easily extended to projective varieties.

Definition 2.3. *The* restriction of scalars*, or* Weil descent*,* $\mathrm{Res}_{L/K}(\mathcal{J})$ *is a variety defined over K with the following properties:*

(W1) For every field $K' \subset \overline{K}$ such that $[L \cdot K' \colon K'] = r$ (i.e. K' is linearly disjoint to L), we have the natural identification: $\big(\mathrm{Res}_{L/K}(\mathcal{J})\big)(K') \longrightarrow \mathcal{J}(L \cdot K')$.

(W2) The variety $\big(\mathrm{Res}_{L/K}(\mathcal{J})\big)_L/L$ obtained by $\mathrm{Res}_{L/K}(\mathcal{J})$ through extension of scalars is isomorphic to $\mathcal{J}^r$, the cartesian product of $\mathcal{J}$ with itself r times.

We consider the Galois group $\mathrm{G}(L/K)$ and recall it is isomorphic to $\mathrm{G}\big(L(\mathcal{J})/K(\mathcal{J})\big)$. For $\phi \in \mathrm{G}(L/K)$, and for $f \in L(\mathcal{J})$, let consider the function $\phi \circ f \in L(\mathcal{J})^{\phi}$. We define the image of the variety $\mathcal{J}$ with respect to ϕ as the variety $\mathcal{J}^{\phi}$ corresponding to the field of functions $L(\mathcal{J})^{\phi}/L$.

We now construct the following variety:

$$\mathcal{W} := \prod_{\phi \in \mathrm{G}(L/K)} \mathcal{J}^{\phi} \quad .$$

We shall denote a point of $\mathcal{W}_{\overline{K}}/\overline{K}$ with $P := (\dots, P_\phi, \dots)$, where $P_\phi \in \mathcal{J}^{\phi}(\overline{K})$. Moreover, let $\overline{\tau} \in \mathrm{G}_K$ and denote with τ its restriction to L. Then:

$$\overline{\tau}(P) := (\dots, Q_\phi, \dots) \ , \qquad \text{with} \qquad Q_\phi = \overline{\tau}(P_{\tau^{-1}\circ\phi}) \ .$$

Theorem 2.4. *The variety $\mathcal{W}$ is the restriction of scalars $\mathrm{Res}_{L/K}(\mathcal{J})$ of $\mathcal{J}$.*

We now explicitly construct the restriction of scalars of an affine variety $\mathcal{J}/L$ embedded in the affine space $\mathbb{A}^n_L$ and given as the set of zeros of a system of m equations:

$$F_i(x_1, \dots, x_n) = 0 \qquad i = 1, \dots, m \ ,$$

with $F_i(x) \in L[x_1, \dots, x_n]$.

Let $\{u_1, \dots, u_r\}$ be a base of L, seen as vector space over K. Define the nr variables y_{ij} as

$$x_j = u_1 y_{1j} + \cdots + u_r y_{rj} \qquad j = 1, \dots, n \ .$$

Now substitute the variables x_j in the relationships that define $\mathcal{J}$, obtaining:

$$F_i(y) = g_{i1}(y)u_1 + \cdots + g_{ir}(y)u_r \qquad i = 1, \ldots, m \ ,$$

where $g_{ij} \in L[y_{11}, \ldots, y_{nr}]$. Since the u_i are linearly independent, we define $\mathcal{W}$ through the mr equations:

$$g_{ij}(y) = 0 \ .$$

One can proof that $\mathcal{W}$ build in this way is the restriction of scalars of the variety $\mathcal{J}$.

Example 2.5. *Let $K = \mathbb{F}_2$, $L = K(\alpha)$, with $\alpha^2 + \alpha + 1 = 0$. Let us consider the elliptic curve $\mathcal{E}/L$*

$$\mathcal{E}: \ y^2 + xy + x^3 + 1 = 0 \ .$$

Substitute $x = x_0 + x_1\alpha + x_2\alpha^2$ and $y = y_0 + y_1\alpha + y_2\alpha^2$:

$$(y_0 + y_1\alpha + y_2\alpha^2)^2 + (x_0 + x_1\alpha + x_2\alpha^2)(y_0 + y_1\alpha + y_2\alpha^2) + (x_0 + x_1\alpha + x_2\alpha^2)^3 + 1 \ .$$

Expanding, we obtain the following system of equations:

$$\begin{aligned} y_0^2 + x_0y_0 + x_1y_2 + x_2y_1 + x_0^3 + x_1^3 + x_2^3 + 1 &= 0 \\ y_2^2 + x_0y_1 + x_1y_0 + x_2y_2 + x_0x_2^2 + x_1x_0^2 + x_2x_1^2 &= 0 \\ y_1^2 + x_0y_2 + x_1y_1 + x_2y_0 + x_0x_1^2 + x_1x_2^2 + x_2x_0^2 &= 0 \end{aligned}$$

which defines the restriction of scalars $\mathrm{Res}_{L/K}(\mathcal{E})$ of $\mathcal{E}$.

The restriction of scalars is a variety of dimension greater than $\mathcal{J}$. For instance, if $\mathcal{C}/L$ is a complete, non-singular curve of genus g, its Jacobian $\mathcal{J}$ is a variety of dimension g and if L/K is an extension of degree r, then $\mathrm{Res}_{L/K}(\mathcal{J})$ has dimension rg. It is easy to guess that we can define subvarieties in $\mathrm{Res}_{L/K}(\mathcal{J})$ that we could not define in $\mathcal{J}$.

Example 2.6. *Recall the elliptic curve $\mathcal{E}$ from Example 2.5. We can construct subvarieties of $\mathrm{Res}_{L/K}(\mathcal{E})$ by intersecting with the hyperplanes $x_i = 0$ or $y_i = 0$ ($i = 1, 2, 3$). These subvarieties, called* sections*, have dimension 5 over K. Clearly these are not subvarieties of $\mathcal{E}$ since it is an elliptic curve, hence it has dimension 1 (over L).*

Let now, for simplicity, $K = \mathbb{F}_q$ and $L = \mathbb{F}_{q^r}$. Let $\mathcal{J}$ be a variety defined over K, for instance the Jacobian of a complete, non-singular curve $\mathcal{C}/K$ of genus g. Let $\sigma \in \mathrm{G}(L/K)$ be the Frobenius automorphism.

Since $\mathcal{J}$ is defined over K, we have $\mathcal{J}^\sigma = \mathcal{J}$. Note, however, that $\mathrm{Res}_{L/K}(\mathcal{J})$ is not $\mathbb{F}_q$-isomorphic to $\mathcal{J}^r$, because of the action of the Galois group.

We can however embed $\mathcal{J}$ into $\mathrm{Res}_{L/K}(\mathcal{J})$ "diagonally":

$$\delta \colon \mathcal{J}(\overline{K}) \to \big(\mathrm{Res}_{L/K}(\mathcal{J})\big)(\overline{K}), \ P \mapsto (\ldots, \sigma^i(P), \ldots) \in \prod_{i=0}^{r-1} \mathcal{J}.$$

The map δ allows to identify $\mathcal{J}$ with a subvariety of $\mathrm{Res}_{L/K}(\mathcal{J})$.

Let now suppose that $\mathcal{J}$ is an abelian variety. Then we find an abelian subvariety complementary to $\mathcal{J}$, in $\mathrm{Res}_{L/K}(\mathcal{J})$.

We use the existence of an automorphism γ of order r in $\mathrm{Res}_{L/K}(\mathcal{J})$, defined by

$$P = (\ldots, P_i, \ldots) \mapsto \gamma(P) := (\ldots, P_{i-1 \bmod r}, \ldots).$$

This map is an automorphism and it is defined over $\mathbb{F}_q$ since it permutes with σ.

Denote with $\mathcal{J}_r$ the kernel of the endomorphism $\sum_{i=0}^{r-1} \gamma^i$. This is an abelian subvariety of $\mathrm{Res}_{L/K}(\mathcal{J})$, called the *trace zero subvariety.*

We point out some of its properties in the following proposition.

Proposition 2.7. *Let $\mathcal{J}$ be an abelian variety defined over $\mathbb{F}_q$. We shall use the representation with product of $\mathrm{Res}_{\mathbb{F}_{q^r}/\mathbb{F}_q}(\mathcal{J})$ and define γ as the automorphism induced by the cyclic permutation of the factors. Then we have:*

1. *$\mathcal{J}$ can be embedded (diagonally) in $\mathrm{Res}_{\mathbb{F}_{q^r}/\mathbb{F}_q}(\mathcal{J})$ or, equivalently, $\mathcal{J} = \mathrm{Ker}(\gamma - \mathrm{Id})$.*
2. *The image of $\gamma - \mathrm{Id}$ is the trace zero subvariety $\mathcal{J}_r$.*
3. *The $\mathbb{F}_q$-rational points of $\mathcal{J}_r$ are the points $P \in \mathcal{J}(\mathbb{F}_{q^r})$ with $\mathrm{Tr}(P) = 0$,*
4. *$\mathcal{J}$ and $\mathcal{J}_r$ generate $\mathrm{Res}_{\mathbb{F}_{q^r}/\mathbb{F}_q}(\mathcal{J})$, and intersect in the r-torsion subgroup.*

2.3 Duality in Arithmetic Geometry

Let S be a (non-empty) set and C an abelian group. The set C^S of the functions from S to C becomes in a natural way an abelian group. In many contexts S and C are endowed with a topology and in this case we tacitly assume that the functions are continuous.

The evaluation map:

$$S \times C^S \to C$$

is non-degenerate and $\mathbb{Z}$-linear in the second argument.

Let $C^{(S)}$ be the set of all functions g with the property that $g(s) = 0_C$ for all but finitely many $s \in S$. Obviously $C^{(S)}$ is a subgroup of C^S.

Example 2.8. *Take $C = \mathbb{Z}$. $\mathbb{Z}^{(S)}$ is the free abelian group generated by S. Its elements are functions $g \colon S \to \mathbb{Z}$ which have value different from 0 only for finitely many elements in S.*

We can define the degree $\deg(g) := \sum_{s \in S} g(s)$. The elements of degree 0 form a subgroup denoted ${\mathbb{Z}^{(S)}}^0$.

Furthermore, we can embed S in $\mathbb{Z}^{(S)}$ by sending $s \in S$ to the function g_s such that $g_s(s) = 1$ and $g_s(s') = 0$ for $s' \neq s$.

A function f from S to C can be extended linearly, i.e. it becomes an homomorphism from $\mathbb{Z}^{(S)}$ to C denoted again by f:

$$f \colon g \mapsto \sum_{s \in S} g(s) \cdot f(s) \ .$$

In this way C^S is identified with $\mathrm{Hom}\left(\mathbb{Z}^{(S)}, C\right)$. We get a non-degenerate *evaluation pairing*:

$$E \colon \mathbb{Z}^{(S)} \times C^S \to C$$

by

$$(g, f) \mapsto \sum_{s \in S} g(s) \cdot f(s) \ .$$

An important special case is that of $C = \mathbb{Z}$. For a *fixed* $f \in \mathbb{Z}^S$, we define:

$$\deg_f \colon \mathbb{Z}^{(S)} \to \mathbb{Z}$$

by

$$\deg_f(g) := E(g, f) = \sum_{s \in S} g(s) \cdot f(s) \quad .$$

Take $f \equiv 1$. In this case we denote $\deg_f$ by deg and its kernel by ${\mathbb{Z}^{(S)}}^0$, the subgroup of elements of degree 0. This definition is equivalent to the one presented in Example 2.8.

Now assume that S is a group. By restricting the evaluation map from C^S to $\mathrm{Hom}(S, C)$, the group of homomorphisms from S to C, we get:

$$E_0 \colon S \times \mathrm{Hom}(S, C) \to C \quad .$$

E_0 is a pairing, non-degenerate in the second argument: it is linear and non-degenerate in the second argument as evaluation map, and it is a group homomorphism as function of the first argument.

Since C is assumed to be abelian, every homomorphism vanishes on the derivative subgroup S' of S, i.e. the subgroup generated by the commutators. This gives rise to a pairing:

$$E \colon S/S' \times \mathrm{Hom}(S, C) \to C \quad .$$

Example 2.9. *Take $C = \mathbb{R}/\mathbb{Z}$ with the discrete topology and S a topological group. The (topological) group $\mathrm{Hom}(S, \mathbb{R}/\mathbb{Z})$ of continuous homomorphisms is called the Pontryagin dual S^* of S.*

Since $\mathbb{R}/\mathbb{Z}$ is an injective $\mathbb{Z}$-module, the pairing

$$E \colon S/S' \times S^* \to \mathbb{R}/\mathbb{Z}$$

is non-degenerate in both variables.

Let K be a perfect field of characteristic $p \geq 0$. For simplicity we assume that group orders are prime to p. The absolute Galois group G_K is a topological group with profinite topology and hence it is compact.

A Galois module M is a discrete $\mathbb{Z}$-module with continuous G_K-action. In particular this implies that

$$M = \bigcup_U M^U \quad ,$$

where $U < \mathrm{G}_K$ with finite index.

The Galois module M determines a functor

$$\mathcal{M} \colon \{ \text{ fields between } K \text{ and } \overline{K} \text{ } \} \mapsto \{ \text{ Abelian groups } \}$$

sending a field L to M^{G_L}.

Example 2.10. *Take $M = \overline{K}^*$. The corresponding functor is denoted by G_m.*

There is a scheme, also denoted G_m, defined over K such that for commutative algebras R over K we have:

$$G_m(R) = R^* \quad ,$$

the group of invertible elements in $(R, \cdot)$ and so $G_m(L) = \overline{K}^{*\mathrm{G}_L} = L^*$.

It is the spectrum of the coordinate ring $K[X, Y]/(XY - 1)$.

This example is generalized as follows. Assume that $\mathcal{A}$ is a commutative group scheme defined over K. Then $A = \mathcal{A}(\overline{K})$ is a G_K-module.

In general $\mathcal{A}$ is not determined by $\mathcal{A}(\overline{K})$, but this is so if $\mathcal{A}$ is smooth or, in particular, étale. If $\mathcal{A}$ is a finite commutative group scheme with order prime to p, then it is étale over K.

Remark 2.11. *A finite Galois module is always represented by an (affine) étale commutative group scheme, and conversely, the $\overline{K}$-rational points of a finite étale commutative group scheme are a finite Galois module.*

Definition 2.12. *Let A, B, C be Galois modules and let*

$$Q\colon A \times B \to C$$

be a pairing.

Q is a Galois pairing *iff for all $(a, b) \in A \times B$ and $\zeta \in \mathrm{G}_K$ we have:*

$$Q(\zeta \circ a, \zeta \circ b) = \zeta \circ Q(a, b) \ .$$

Let S be a set endowed with the discrete topology with continuous G_K-action, and let C be a Galois module. Then C^S becomes a Galois module by the action $f^\zeta := \zeta \circ f \circ \zeta^{-1}$.

If we apply this definition to $\mathbb{Z}^{(S)}$, where $\mathbb{Z}$ seen as Galois module with trivial action, we can verify that the evaluation pairing

$$E\colon \mathbb{Z}^{(S)} \times C^S \to C$$

is a Galois pairing.

When S is also a G_K-module, restricting to homomorphisms we endow $\mathrm{Hom}(S, C)$ with a natural G_K-module structure and the pairing

$$E\colon S \times \mathrm{Hom}(S, C) \to C$$

is a Galois pairing.

A key example from the arithmetical point of view is to take $C = \overline{K}^*$. For a finite G_K-module A, we call *Cartier dual* the G_K-module $\widehat{A} := \mathrm{Hom}(A, \overline{K}^*)$.

Theorem 2.13. *Let A be a finite G_K-module and $\widehat{A}$ its Cartier dual.*

1. *The evaluation pairing $E\colon A \times \widehat{A} \to \overline{K}^*$ is a non-degenerate Galois pairing.*

2. *If $\mathcal{A}$ is a finite commutative group scheme with order prime to $\mathrm{char}(K)$ then $\widehat{\mathcal{A}} := \mathrm{Hom}(\mathcal{A}, G_m)$ is an étale group scheme (the Cartier dual of $\mathcal{A}$) and $\widehat{\mathcal{A}(\overline{K})} = \widehat{\mathcal{A}}(\overline{K})$. In particular the Cartier dual of $\widehat{A}$ is A.*

We now present two foundamental examples.

Example 2.14. *Take $A = \mu_n$, the group of the roots of unity with order dividing n. As always, we assume n prime to p.*

Then $\mathcal{A} = \mathrm{Ker}(n \cdot \mathrm{Id}_{G_m}) =: G_m[n]$ and we have the Kummer sequence:

$$1 \to G_m[n] \to G_m \to G_m \to 1$$

of group schemes, yielding the exact sequence of Galois modules

$$1 \to \mu_n \to \overline{K}^* \to \overline{K}^* \to 1 \ .$$

The Cartier dual of $G_m[n]$ is the constant group scheme $\mathbb{Z}/n\mathbb{Z}$ (with trivial Galois action) since every endomorphism of μ_n is an exponentiation.

Conversely, if $A = \mathbb{Z}/n\mathbb{Z}$, its Cartier dual is μ_n. Note the Pontryagin dual of $\mathbb{Z}/n\mathbb{Z}$ is $\mathbb{Z}/n\mathbb{Z}$ itself. So as $\mathbb{Z}$-modules we get the same groups, but the Galois action differs as soon as K does not contain all the n^{th} roots of unity.

Example 2.15. *Let $\mathcal{A}$ be an abelian variety defined over K. Take $\mathcal{A}[n] := \mathrm{Ker}(n \cdot \mathrm{Id}_{\mathcal{A}})$.*

Again we have a Kummer sequence

$$0 \to \mathcal{A}[n] \to \mathcal{A} \to \mathcal{A} \to 0$$

yielding the exact sequence of Galois modules

$$0 \to \mathcal{A}(\overline{K})[n] \to \mathcal{A}(\overline{K}) \to \mathcal{A}(\overline{K}) \to 0 \ .$$

There is an abelian variety $\widehat{\mathcal{A}}$ dual to $\mathcal{A}$ such that, in a canonical way, $\widehat{\mathcal{A}[n]}$ is isomorphic to $\widehat{\mathcal{A}}[n]$. In particular we get a non-degenerate Galois pairing between the points of order dividing n of $\mathcal{A}(\overline{K})$ and $\widehat{\mathcal{A}}(\overline{K})$.

An important special case is that $\mathcal{A}$ is principally polarized, e.g. $\mathcal{A}$ is the Jacobian of a projective, absolutely irreducible, non-singular curve, and then it is isomorphic to $\widehat{\mathcal{A}}$. In this case $\mathcal{A}[n]$ is self-dual. This evaluation pairing

$$W_n \colon \mathcal{A}(\overline{K})[n] \times \mathcal{A}(\overline{K})[n] \to \overline{K}^*$$

is called Weil pairing.

Let A, B be two G_K-modules. The tensor product $A \otimes B$ (over $\mathbb{Z}$) becomes in a natural way a G_K-module. We have a natural (and functorial) homomorphism:

$$\cup^{0,0} \colon A^{\mathrm{G}_K} \otimes B^{\mathrm{G}_K} \to (A \otimes B)^{\mathrm{G}_K} \ ,$$

which induces a unique family of homomorphisms:

$$\cup^{p,q} \colon H^p(\mathrm{G}_K, A) \times H^q(\mathrm{G}_K, B) \to H^{p+q}(\mathrm{G}_K, A \otimes B) \ ,$$

with functorial properties with respect to cohomology functors.
$\cup^{p,q}$ is called the *cup product.*

Now assume there is a pairing $Q \colon A \times B \to C$.
Q defines a G_K-homomorphism ϕ_Q from $A \otimes B$ to C, by sending $a \otimes b$ to $Q(a, b)$. We also get induced homomorphisms $\phi_Q^{(n)}$ on the n^{th} cohomology groups.

We can then define a bilinear pairing:

$$Q^{p,q} \colon H^p(\mathrm{G}_K, A) \times H^q(\mathrm{G}_K, B) \to H^{p+q}(\mathrm{G}_K, C)$$

by

$$Q^{p,q} = \phi_Q^{(p+q)} \circ \cup^{p,q} \ .$$

Example 2.16. *Let p, q be non-negative integers with $p + q = 2$. The evaluation pairing induces pairings:*

$$E^{p,q} \colon H^p(\mathrm{G}_K, A) \times H^q(\mathrm{G}_K, \widehat{A}) \to H^2(\mathrm{G}_K, \overline{K}^*) \ .$$

The cohomology group $H^2(\mathrm{G}_K, \overline{K}^*)$ is an important object to study the arithmetic of K. It is called the *Brauer group of K* and denoted by $\mathrm{Br}(K)$.

2.4 The Lichtenbaum-Tate Pairing

Let $\mathcal{J}$ be a principally polarized abelian variety defined over K, e.g. the Jacobian of a projective, non-singular and absolutely irreducible curve. Let, as always, $n \in \mathbb{N}$ be prime to $p = \operatorname{char}(K)$.

The assumption that n is prime to p implies that the Kummer sequence:

$$0 \to \mathcal{J}(\overline{K})[n] \to \mathcal{J}(\overline{K}) \xrightarrow{n} \mathcal{J}(\overline{K}) \to 0$$

is an exact sequence of G_K-modules.

We can therefore apply Galois cohomology and obtain the exact sequence:

$$0 \to \mathcal{J}(K)/n\mathcal{J}(K) \xrightarrow{\delta} H^1\big(\mathrm{G}_K, \mathcal{J}[n](\overline{K})\big) \xrightarrow{\alpha} H^1\big(\mathrm{G}_K, \mathcal{J}(\overline{K})\big)[n] \to 0 \ .$$

This sequence is particular important both in theory and in practice and we describe it in more detail. Recall that for a G_K module M, the group $H^q(\mathrm{G}_K, M)$ is a quotient of the group of q-cocycles modulo the subgroup of q-coboundaries.

We begin by describing the map δ. Let $P \in \mathcal{J}(K)$. Since the multiplication times n is surjective, there exists a point $Q \in \mathcal{J}(\overline{K})$ such that $nQ = P$. Define:

$$\begin{aligned} \delta'(P)\colon\ \mathrm{G}_K &\to \mathcal{J}(\overline{K})[n] \\ \zeta &\mapsto \zeta(Q) - Q \ . \end{aligned}$$

It is easy to check that $\delta'(P)$ is a 1-cocycle with image in $\mathcal{J}(\overline{K})[n]$ and that another choice of Q' with $nQ' = P$ chances this cocycle by a coboundary, and so we get a well defined map from $\mathcal{J}(K)$ to $H^1\big(\mathrm{G}_K, \mathcal{J}[n](\overline{K})\big)$. Another immediate check shows that the kernel of this map is exactly $n\mathcal{J}(K)$.

We now turn the attention to α. Using the injection of $\mathcal{J}(\overline{K})[n]$ into $\mathcal{J}(\overline{K})$ we may interpret cocycles with values in $\mathcal{J}(\overline{K})[n]$ as cocycles with values in $\mathcal{J}(\overline{K})$. The map α is obtained going to the quotient modulo coboundaries. Since the arguments of the induced cocycles are points of order n, it follows that the image of α is contained in the subgroup $H^1\big(\mathrm{G}_K, \mathcal{J}(\overline{K})\big)[n]$.

Finally one can check, either directly or by using properties of cohomology, that α is surjective and that the kernel of α is equal to the image of δ.

As seen in the previous section, we have an evaluation pairing:

$$E^{1,1}\colon H^1\big(\mathrm{G}_K, \mathcal{J}[n](\overline{K})\big) \times H^1\big(\mathrm{G}_K, \mathcal{J}[n](\overline{K})\big) \to \operatorname{Br}(K)[n] \ ,$$

where we used that $\mathcal{J}[n](\overline{K})$ is self-dual (cf. Example 2.15).

Definition 2.17. *With the notation as above, the* Tate pairing

$$T_n\colon \mathcal{J}(K)/n\mathcal{J}(K) \times H^1\big(\mathrm{G}_K, \mathcal{J}(\overline{K})\big)[n] \to \operatorname{Br}(K)[n]$$

is given by:

$$T_n\big(P + n\mathcal{J}(K), \gamma\big) = E^{1,1}\big(\delta(P + n\mathcal{J}(K)), \alpha^{-1}(\gamma)\big) \ .$$

It is easy to check that T_n is well defined and bilinear.

Remark 2.18. *The Tate pairing relates three very interesting groups occurring in Arithmetic Geometry: the Morder-Weil group of $\mathcal{J}$, the first cohomology group of $\mathcal{J}$ and the Brauer group of the ground field K.*

We now want to introduce another pairing, the Lichtenbaum pairing, which is essentially an evaluation pairing defined on a curve. Let $\mathcal{C}$ be a projective, non-singular and absolutely irreducible curve defined over K with function field $F = K(\mathcal{C})$ and, for simplicity, a K-rational point.

We denote by $\overline{\mathcal{C}}$ the curve obtained from $\mathcal{C}$ by extending the scalars to $\overline{K}$. The function field of $\overline{\mathcal{C}}$ is $\overline{F} = F \cdot \overline{K}$ and G_K acts in a natural way on $\overline{F}$ and $\overline{\mathcal{C}}(\overline{K})$, fixing respectively the sets F and $\mathcal{C}(K)$.

We consider subsets $T \subset \overline{\mathcal{C}}(\overline{K})$ which are G_K-invariant. We denote $\overline{F}_T \subset \overline{F}$ the group of functions on $\overline{\mathcal{C}}$ *without* zeros and poles in T. We can also interpret $\overline{F}_T$ as Galois invariant subset of $(\overline{K}^*)^T$, the set of all the maps from T to $\overline{K}^*$.

The evaluation pairing

$$E_T \colon \mathbb{Z}^{(T)} \times \overline{F}_T \to \overline{K}$$

is a Galois pairing and induces (for $p+q=2$) a pairing

$$E_T^{p,q} \colon H^p\big(\mathrm{G}_K, \mathbb{Z}^{(T)}\big) \times H^q\big(\mathrm{G}_K, \overline{F}_T\big) \to \mathrm{Br}(K) \ .$$

Since $\mathcal{C}$ is assumed to be regular (actually non-singular) we can identify $\mathbb{Z}^{(T)}$ with the group $\mathrm{Div}_T(\overline{\mathcal{C}})$ of divisors on $\overline{\mathcal{C}}$ with support in T.

It is possible to see that $H^1\big(\mathrm{G}_K, \mathrm{Div}_T(\overline{\mathcal{C}})\big) = 0$, hence the case $p = q = 1$ is not interesting. In the following we will concentrate on

$$E_T := E_T^{0,2} \colon \mathcal{D}_T \times H^2\big(\mathrm{G}_K, \overline{F}_T\big) \to \mathrm{Br}(K) \ ,$$

where $\mathcal{D}_T = H^0\big(\mathrm{G}_K, \mathbb{Z}^{(T)}\big)$ can be identified with $\mathrm{Div}_T(\mathcal{C})$, the group of K-rational divisors on $\mathcal{C}$ with support in T.

We would like to extend the pairing E_T to a pairing with $\overline{F}^*$ as domain for the second argument.

Begin with $\gamma \in H^2\big(\mathrm{G}_K, \overline{F}^*\big)$. We represent γ by a cocycle c determined by finitely many functions f_{σ_1,σ_2} as values. This is possible since, because of continuity, there is a finite Galois extension L/K such that γ is the inflation of an element $\gamma_L \in H^2\big(\mathrm{G}(L/K), F \cdot L\big)$.

Given a K-rational divisor D, we wish to evaluate each function f_{σ_1,σ_2} on D, but we have to pay attention to the zeros and poles of f_{σ_1,σ_2}. Let S be the finite set of points on $\overline{\mathcal{C}}$ which occur as zeros or poles of the functions f_{σ_1,σ_2}, and take $T = \overline{\mathcal{C}} \setminus S$.

Next, choose a function $h \in F$ such that $D_h := D + (h)$ is prime to S. So $D_h \in \mathcal{D}_T$ and $E_T(D_h, \gamma)$ is an element in $\mathrm{Br}(K)$. Unfortunately this elements may depend on the choice of h.

Let $h' \in F$ such that $D_{h'} := D+(h')$ is prime to S. We want to evaluate the difference $E_T(D_h - D_{h'}, \gamma)$, which means to evaluate f_{σ_1,σ_2} on the divisor $(\tilde{h})$, where $F \ni \tilde{h} = h - h'$. If the evaluation is trivial, we can conclude that the element $E_T(D_h, \gamma) \in \mathrm{Br}(K)$ does not depend upon the choice of h. Of course we can not expect that the evaluation is always trivial, but we aim to find conditions on γ that make this happen.

By Weil's reciprocity we have:

$$f_{\sigma_1,\sigma_2}\big((\tilde{h})\big) = \tilde{h}\big((f_{\sigma_1,\sigma_2})\big) \ ,$$

and, since $\tilde{h}$ is G_K-invariant, the above expression is trivial if the class of the following cocycle $c'_L \in H^2(\mathrm{G}_K, \mathrm{Div}(\overline{\mathcal{C}}))$ is trivial:

$$\begin{aligned} c'_L \colon \ & \mathrm{G}(L/K) \times \mathrm{G}(L/K) \to \mathrm{Div}(\overline{\mathcal{C}}) \\ & \qquad\qquad \sigma_1, \sigma_2 \mapsto (f_{\sigma_1,\sigma_2}) \ . \end{aligned}$$

This motivate the following:

Definition 2.19. *The* Brauer group $\mathrm{Br}(\mathcal{C})$ of $\mathcal{C}$ *is the kernel of the map*

$$H^2(\mathrm{G}_K, \overline{F}^*) \to H^2(\mathrm{G}_K, \mathrm{Div}(\overline{\mathcal{C}}))$$

induced by sending a function f on $\mathcal{C}$ to its principal divisor (f).

By the discussion above we see that we can define a pairing from $\mathrm{Div}(\mathcal{C}) \times \mathrm{Br}(\mathcal{C})$ in $\mathrm{Br}(K)$ by using appropriate pairing E_T and changing elements in $\mathrm{Div}(\mathcal{C})$ by principal divisors. By definition, the resulting pairing only depends on the divisor class of the K-rational divisors on $\mathcal{C}$.

Proposition 2.20. *Let $\mathcal{C}$ be a projective, non-singular and absolutely irreducible curve defined over K with divisor class group* $\mathrm{Pic}(\mathcal{C})$*.*

Then the evaluation pairing induces a pairing

$$E\colon\ \mathrm{Pic}(\mathcal{C}) \times \mathrm{Br}(\mathcal{C}) \to \mathrm{Br}(K)\ \ .$$

In many cases one is interested to work in $\mathrm{Pic}^0(\mathcal{C})$. As this group consists of classes of divisors on $\mathcal{C}$ of degree 0 modulo the group of principal divisors, we observe that the evaluation of a function f at a divisor of degree 0 depends only on (f), and so E induces a pairing, also denoted by E, from $\mathrm{Pic}^0(\mathcal{C}) \times \overline{\mathrm{Br}}(\mathcal{C})$, where $\overline{\mathrm{Br}}(\mathcal{C})$ is the image of $\mathrm{Br}(\mathcal{C})$ in $H^2\big(\mathrm{G}_K, \mathrm{Princ}(\overline{\mathcal{C}})\big)$ induced by the map $f \mapsto (f)$.

Corollary 2.21. *The evaluation pairing induces a pairing*

$$E\colon\ \mathrm{Pic}^0(\mathcal{C}) \times \overline{\mathrm{Br}}(\mathcal{C}) \to \mathrm{Br}(K)\ \ .$$

It remains to describe the elements of $\overline{\mathrm{Br}}(\mathcal{C})$. We use the exact sequence:

$$0 \to \mathrm{Princ}(\overline{\mathcal{C}}) \to \mathrm{Div}^0(\overline{\mathcal{C}}) \to \mathrm{Pic}^0(\overline{\mathcal{C}}) \to 0\ \ ,$$

and get:

$$0 = H^1\big(\mathrm{G}_K, \mathrm{Div}^0(\overline{\mathcal{C}})\big) \to H^1\big(\mathrm{G}_K, \mathrm{Pic}^0(\overline{\mathcal{C}})\big) \xrightarrow{\delta} H^2\big(\mathrm{G}_K, \mathrm{Princ}(\overline{\mathcal{C}})\big) \to H^2\big(\mathrm{G}_K, \mathrm{Div}^0(\overline{\mathcal{C}})\big)\ \ ,$$

where δ is the connecting homomorphism resulting from cohomology, which associates to $\gamma \in H^1\big(\mathrm{G}_K, \mathrm{Pic}^0(\overline{\mathcal{C}})\big)$ a 2-cocycle from G_K^2 to $\mathrm{Princ}(\overline{\mathcal{C}})$. In other words, given γ we find for each pair $(\sigma_1, \sigma_2) \in \mathrm{G}_K^2$ a function $f_{\sigma_1,\sigma_2} \in \overline{F}$ such that $\delta(\gamma)$ is equal to the class of the 2-cocycle sending (σ_1, σ_2) to the principal divisor (f_{σ_1,σ_2}). Moreover since $H^1\big(\mathrm{G}_K, \mathrm{Div}^0(\overline{\mathcal{C}})\big) = 0$, δ is injective and $\overline{\mathrm{Br}}(\mathcal{C}) = \delta\Big(H^1\big(\mathrm{G}_K, \mathrm{Pic}^0(\overline{\mathcal{C}})\big)\Big)$.

Definition 2.22. *Let $\overline{D} \in Pic^0(\mathcal{C})$ be a K-rational class of degree 0 with divisor $D \in \overline{D}$.*

The Lichtenbaum pairing

$$T_L\colon\ \mathrm{Pic}^0(\mathcal{C}) \times H^1\big(\mathrm{G}_K, \mathrm{Pic}^0(\overline{\mathcal{C}})\big) \to \mathrm{Br}(K)$$

maps $(\overline{D}, \gamma)$ to the class of the 2-cocycles from G_K^2 to $\overline{K}^*$ given by:

$$(\sigma_1, \sigma_2) \mapsto f_{\sigma_1,\sigma_2}(D)\ \ .$$

(Here D has to be chosen such that it is prime to the set of zeros and poles of f_{σ_1,σ_2}, which is always possible.)

We now give an explicit description of the Lichtenbaum pairing.
Take $\gamma \in H^1\big(\mathrm{G}_K, \mathrm{Pic}^0(\overline{\mathcal{C}})\big)$ represented by a cocycle

$$\begin{aligned} c\colon \mathrm{G}_K &\to \mathrm{Pic}^0(\overline{\mathcal{C}}) \\ \zeta &\mapsto \overline{D}_\zeta \end{aligned}$$

and choose $D_\zeta \in \overline{D}_\zeta$.

We have already seen that given γ we find for each pair $(\sigma_1, \sigma_2) \in \mathrm{G}_K^2$ a function $f_{\sigma_1,\sigma_2} \in \overline{F}$ such that $\delta(\gamma)$ is equal to the class of the 2-cocycle sending (σ_1, σ_2) to the principal divisor (f_{σ_1,σ_2}). Explicitly, the principal divisor is given by:

$$(f_{\sigma_1,\sigma_2}) := \sigma_1(D_{\sigma_2}) + D_{\sigma_1} - D_{\sigma_1\sigma_2} \ .$$

Now, choose a divisor $D_0 := \sum_{P\in\overline{\mathcal{C}}(\overline{K})} z_P P \in \overline{D}_0 \in \mathrm{Pic}^0(C)$, such that D_0 is prime to (f_{σ_1,σ_2}). Then $T_L(\overline{D}_0, \gamma)$ is the cohomology class of the cocycle:

$$(\sigma_1, \sigma_2) \mapsto \sum_{P\in\overline{\mathcal{C}}(\overline{K})} f_{\sigma_1,\sigma_2}(P)^{z_P} \in \mathrm{Br}(K) \ .$$

Example 2.23. *Let L/K be a cyclic extension of degree n and let $\mathrm{G}(L/K) = \langle\tau\rangle$.*

Because of Hilbert's Theorem 90, the inflation map from $H^2\big(\mathrm{G}(L/K), F\cdot L\big)$ to $H^2(\mathrm{G}_K, \overline{F})$ is injective. We have the following explicit description of $H^2\big(\mathrm{G}(L/K), F\cdot L\big)$: every cohomology class γ contains a cocycle given by:

$$c_b(\tau^i, \tau^j) = \begin{cases} b & \text{if } i+j \geq n \\ 1 & \text{if } i+j < n \end{cases}$$

with $b \in F$. The cocycles c_b, $c_{b'}$ lie in the same class if and only if $bb'^{-1} \in N_{F\cdot L/F}(F\cdot L)$.

We now turn the attention to $H^1\big(\mathrm{G}(L/K), \mathrm{Pic}^0(\mathcal{C}_L)\big)$. Let γ be the class of a cocycle $c\colon \mathrm{G}(L/K) \to \mathrm{Pic}^0(\mathcal{C}_L)$. Then c is completely determined by the value $\overline{D}_\tau := c(\tau)$. In fact, the cocycle condition implies that $c(\tau^j) = \sum_{i=0}^{j-1} \tau^i(\overline{D}_\tau)$. In particular we get $\mathrm{Tr}_{L/K}(\overline{D}_\tau) = \sum_{i=0}^{n-1} \tau^i(\overline{D}_\tau) = c(\tau^n) = 0$.

Choose a divisor $D \in \overline{D}_\tau$. Then:

$$\mathrm{Tr}_{L/K}(D) = (f_D) \ , \qquad \text{with } f_D \in F \ .$$

Hence $\delta(\gamma)$ is represented by the 2-cocycle:

$$(\tau^i, \tau^j) \mapsto \begin{cases} (f_D) & \text{if } i+j \geq n \\ 1 & \text{if } i+j < n \end{cases}$$

Next choose a divisor $D_0 := \sum_{P\in\overline{\mathcal{C}}(\overline{K})} z_P P \in \overline{D}_0 \in \mathrm{Pic}^0(\mathcal{C})$, such that D_0 is prime to the set of zeros and poles of f_D. Then $T_L(\overline{D}_0, \gamma) \in H^2\big(\mathrm{G}(L/K), L^*\big)$ is represented by the 2-cocycle:

$$(\tau^i, \tau^j) \mapsto \begin{cases} \prod_{P\in\overline{\mathcal{C}}(\overline{K})} f_D(P)^{z_P} & \text{if } i+j \geq n \\ 1 & \text{if } i+j < n \end{cases}$$

This is a cocycle defining a cyclic algebra with center K and splitting field L.

So far we have defined two pairings attached to the Jacobian variety of an hyperelliptic curve, namely the Tate pairing T_n which crucially uses the Weil pairing on torsion points of the Jacobian of order n, and the Lichtenbaum pairing T_L which uses evaluation of functions on the curve. A priori the Lichtenbaum pairing is a universal object, but we can look at it modulo n and get for all natural number prime to $\operatorname{char}(K)$

$$T_{L,n}\colon\ \operatorname{Pic}^0(\mathcal{C})/n\operatorname{Pic}^0(\mathcal{C}) \times H^1\big(\mathrm{G}_K, \operatorname{Pic}^0(\overline{\mathcal{C}})\big)[n] \to \operatorname{Br}(K)[n]\ .$$

Lichtenbaum proves:

Theorem 2.24. *Up to a sign, the pairing $T_{L,n}$ is equal to T_n.*

We shall call T_n the Lichtenbaum-Tate pairing. For most purpose its interpretation as evaluation of functions on $\mathcal{C}$ is useful: we have a description of the Tate pairing on the Jacobian that only uses objects directly defined by the underlying curve. In particular the Weil pairing has completely disappeared.

2.5 The Local Lichtenbaum-Tate Pairing

A fascinating object in number theory is class field theory, whose essentials are formulated in a very elegant fundamental duality theorem involving étale cohomology.

Let O be a ring of integers of a number field and $X = \operatorname{Spec}(O)$, let F be a constructible abelian sheaf, i.e. there are finitely many points $\{x_1, \ldots, x_n\}$ such that the pullbacks of F to $X \setminus \{x_1, \ldots, x_n\}$ and to $\{x_1, \ldots, x_n\}$ are locally constant. Denote G_m the group scheme attached to the multiplicative group.

Theorem 2.25. *For $0 \leq i \leq 3$ we have a perfect pairing*

$$H^i_{et}(X, F) \times \operatorname{Ext}^{3-i}_X(F, G_m) \to H^3_{et}(X, G_m) \cong \mathbb{Q}/\mathbb{Z}$$

of finite groups.

From this pairing we get duality theorems both for local fields, e.g. finite algebraic extensions of p-adic fields, and for global fields, which however we shall omit from this dissertation.

We now apply Theorem 2.25 to a finite Galois module A over a local field K with residue field $\mathbb{F}_q$. We assume that the order of A is prime to q.

We interpret O_K, the ring of integers of K, as localization of the ring of integers of a number field, and look at abelian sheaves F trivial outside $\operatorname{Spec}(O_K)$.

We get the *Duality Theorem of Tate.*

Theorem 2.26.

1. *$H^3_{et}(X, G_m)$ is isomorphic (in a natural way) to $\operatorname{Br}(K) = H^2(\mathrm{G}_K, \overline{K}^*)$ and hence this group is isomorphic to $\mathbb{Q}/\mathbb{Z}$.*

2. *Let A be a finite G_K-module with Cartier dual $\widehat{A}$.*

   Then for $0 \leq i \leq 2$ the cohomology groups $H^i(\mathrm{G}_K, A)$ are finite and the evaluation pairing induces non-degenerate pairings:

$$H^i(\mathrm{G}_K, A) \times H^{2-i}(\mathrm{G}_K, \widehat{A}) \to \operatorname{Br}(K)\ .$$

As a consequence of this duality theorem, Tate proves:

Theorem 2.27. *Let $\mathcal{J}$ be an abelian variety (for simplicity principally polarized). The Tate pairing*

$$T_n \colon \mathcal{J}(K)/n\mathcal{J}(K) \times H^1\big(\mathrm{G}_K, \mathcal{J}(\overline{K})\big)[n] \to \mathrm{Br}(K)[n]$$

is a non-degenerate pairing.

Motivated by Theorems 2.24 and 2.27 we aim to obtain an explicit description of a non-degenerate, bilinear pairing defined over an affine, non-singular curve. Further, as already mentioned in Section 2.1, we would like to exploit the correspondence between the Picard group of the curve and an ideal class group (that will be introduced in a while) to obtain a description of the pairing which is also efficiently computable.

Let K be a local field with residue field $\mathbb{F}_q$. Let $\mathcal{C}_O$ be an affine, non-singular curve over K with function field F and let O be the ring of holomorphic functions on $\mathcal{C}_O$. Since $\mathcal{C}_O$ is non-singular, O is a Dedekind domain (non-singularity of $\mathcal{C}_O$ is related to the integral closure of O). Following [Fre08], we denote with $\mathrm{Pic}(O)$ the ideal class group $\mathrm{Cl}(O)$.

Now, there exists a unique projective, irreducible and regular curve $\mathcal{C}$ with function field F and containing $\mathcal{C}_O$ as affine part. Let $\mathcal{J}$ be the Jacobian variety of $\mathcal{C}$.

We would like to relate $\mathrm{Pic}(O)$ and the Jacobian of $\mathcal{C}$. For this, we enlarge the ground field to $\overline{K}$. The integral closure of O in $\overline{F}$ is denoted by $\overline{O}$ and it is the ring of holomorphic functions of the curve $\overline{\mathcal{C}_O}$, the extension of scalars of $\mathcal{C}_O$ to $\overline{K}$. It is easy to proof that $\mathcal{J}(\overline{K})$ is isomorphic to $\mathrm{Pic}(\overline{O})$ and we can thus mimic the construction made in the previous section.

It is worthwhile to state again the fundamental result following from the local duality theorem.

Corollary 2.28 (Lichtenbaum-Tate)**.** *Let K be a local field with residue field $\mathbb{F}_q$, $\mathcal{C}_O$ an affine, non-singular curve over K with ring of holomorphic functions O.*

For every natural number n prime to q, the Lichtenbaum-Tate pairing

$$T_{L,n} \colon \; \mathrm{Pic}(O)/n\,\mathrm{Pic}(O) \times H^1\big(\mathrm{G}_K, \mathrm{Pic}(\overline{O})\big)[n] \to \mathrm{Br}(K)[n]$$

is a non-degenerate pairing.

We now discuss the groups occurring in this pairing, both from the theoretical and algorithmic point of view.

As already noted, we can replace $\mathrm{Pic}(\overline{O})$ by $\mathcal{J}(\overline{K})$. Furthermore we suppose $\mathcal{J}$ has good reduction, i.e. we find equations for $\mathcal{J}$ with coefficients in the ring of integers of K whose reduction modulo the valuation ideal of K defines an abelian variety over $\mathbb{F}_q$.

We remark that all these assumptions perfectly match with the applications that we have in mind. In fact we shall begin with an abelian variety $\mathcal{J}$ over $\mathbb{F}_q$ and lift it to an abelian variety over K.

Let us consider the first group in Tate duality. Using Hensel's lemma we get:

$$\mathcal{J}(K)/n\mathcal{J}(K) \cong \mathcal{J}(\mathbb{F}_q)/n\mathcal{J}(\mathbb{F}_q) \;\;,$$

where with a slight abuse of notation we denoted by $\mathcal{J}$ both the abelian variety over K and its reduction over $\mathbb{F}_q$.

Remark 2.29. *Assume that $n = \ell$ is a prime and that $\mathcal{J}(\mathbb{F}_q)$ has no points of order ℓ^2. Then $\mathcal{J}(\mathbb{F}_q)/\ell\mathcal{J}(\mathbb{F}_q)$ is isomorphic to $\mathcal{J}(\mathbb{F}_q)[\ell]$ in a natural way.*

We now come to the discussion of $H^1\big(\mathrm{G}_K, \mathrm{Pic}(\overline{O})\big)[n]$. Since unramified extensions of K do not split elements in this group we can use an inflation-restriction sequence to change our base field from K to the maximal unramified extension K_{nr} of K and compute the cohomology group over this larger field. We then look for elements which are invariant under the Galois group of K_{nr}/K, which is (topologically) generated by a canonical lift of the Frobenius automorphism π of $\mathbb{F}_q$.

We assume that n is prime to the number of connected components of the special fiber of $\mathcal{J}$. It follows that

$$H^1\big(G(K_{nr}/K), \mathrm{Pic}(\overline{O})\big) = 0 \ .$$

Via restriction we embed $H^1\big(\mathrm{G}_K, \mathrm{Pic}(\overline{O})\big)$ into $H^1\big(G_{K_{nr}}, \mathrm{Pic}(\overline{O})\big)$. The image is equal to the subgroup of elements which are π-invariant as π acts by conjugation on $G_{K_{nr}}$.

Let L_n the unique extension of K_{nr} of degree n which is totally ramified.

Since $\mathcal{J}$ by assumption has good reduction, $\mathcal{J}(K_{nr})[n] = \mathcal{J}(L_n)[n] = \mathcal{J}(\overline{K})[n]$ and hence:

$$H^1\big(\mathrm{G}_K, \mathrm{Pic}(\overline{O})\big)[n] = \mathrm{Hom}\big(G(L_n/K_{ns}), \mathcal{J}[n]\big) \ .$$

By fixing a generator τ of $\mathrm{G}(L_n/K_{nr})$, we can identify $\mathrm{Hom}\big(\mathrm{G}(L_n/K_{nr}), \mathcal{J}[n]\big)$ with $\mathcal{J}[n]$ via the map:

$$\varphi \mapsto \varphi(\tau^{-1}) =: Q_\tau \ .$$

In general this identification is not compatible with Galois action. Let ξ be an uniformizing element of K and $K_{nr} = K(\xi^{1/n})$. Since $\tau(\xi^{1/n}) = \zeta_n \xi^{1/n}$ for an n^{th} root of unity ζ_n and π maps ζ_n to ζ_n^q, we can deduce that π operates on $\langle\tau\rangle$ by conjugation, sending τ to τ^{-q}.

Definition 2.30. *Let $\mathcal{J}(\overline{K})[n]^{(q)}$ be the subgroup of $\mathcal{J}(\overline{K})[n]$ consisting of elements Q with $\pi(Q) = qQ$.*

Proposition 2.31. *Let n be prime to q. Then $H^1\big(\mathrm{G}_K, \mathrm{Pic}(\overline{O})\big)[n]$ is isomorphic to $\mathrm{Hom}\big(\mathrm{G}(L_n/K_{nr}), \mathcal{J}[n]\big)$ and hence, non-canonically since it depends on the choice of τ, to $\mathcal{J}(\overline{K})[n]^{(q)}$.*

Corollary 2.32.

1. *If $\zeta_n \in K$ then $H^1\big(\mathrm{G}_K, \mathrm{Pic}(\overline{O})\big)[n]$ is isomorphic to $\mathrm{Pic}(O)[n]$.*
2. *Let L any extension of K totally ramified of degree n. The $H^1\big(\mathrm{G}_K, \mathrm{Pic}(\overline{O})\big)[n]$ is equal to the kernel of the restriction map from G_K to G_L.*

Finally, we sketch some details about the Brauer group. By definition, $\mathrm{Br}(K)$ consists of the classes of 2-cocycles of G_K with values in $\overline{K}$ modulo 2-coboundaries.

One can interpret these classes as classes of simple K-algebras with center K. The addition in the cohomology group corresponds to the tensor product of algebras and the unit element in $\mathrm{Br}(K)$ corresponds to the class of full matrix algebras. Moreover, if L/K is a separable extension and A is an algebra representing an element $a \in \mathrm{Br}(K)$, then $A \otimes_K L$ represents a class $a_L \in \mathrm{Br}(L)$ and $a_L = \mathrm{res}_{K/L}(a)$.

We remark that as a consequence of Hilbert's Theorem 90, for a Galois extension L/K the inflation map from $H^2(\mathrm{G}(L/K), L^*)$ to $\mathrm{Br}(K)$ is injective, and the kernel of the restriction map $\mathrm{res}_{K/L}$ is $H^2(\mathrm{G}(L/K), L^*)$. So we can study the Brauer group of K by looking at classes of simple K-algebras A with center K and with $A \otimes_K L$ being a full matrix algebra for Galois extensions L/K.

It is especially interesting to look at the case that L/K is a cyclic extension of degree n (this is the case we have encountered in Example 2.23). So, from now on, we assume L/K is cyclic of degree n. Algebras corresponding to elements in $H^2(\mathrm{G}(L/K), L^*)$ are called *cyclic algebras*. It is a general fact that Tate cohomology groups of cyclic groups are periodic with period 2 and so $H^2(\mathrm{G}(L/K), L^*) \cong K^*/N_{L/K}L^*$. Such an isomorphism is made explicit in the following way: let τ be the generator of $\mathrm{G}(L/K)$ and take $b \in K^*$. The map $f_{\tau,b}\colon \mathrm{G}(L/K) \times \mathrm{G}(L/K) \to L^*$ given by

$$f_{\tau,b}(\tau^i, \tau^j) = \begin{cases} b & \text{if } i+j \geq n \\ 1 & \text{if } i+j < n \end{cases}$$

is a 2-cocycle and its class determines an element in $\mathrm{Br}(K)[n]$. For two elements b, b' the cocycles $f_{\tau,b}$ and $f_{\tau,b'}$ are in the same cohomology class if and only if $bb'^{-1} \in N_{L/K}L^*$. We denote the corresponding class of cyclic algebras by $(L, \tau, b \cdot N_{L/K}L^*)$.

The most important case for applications is that of L totally ramified of degree n. We assume L/K is Galois. It follows that L/K is cyclic and $K \supset \mu_n$, the set of n^{th} roots of unity.

Let, as always, τ be a generator of $\mathrm{G}(L/K)$. We have:

$$K^*/N_{L/K}L^* \cong \mathbb{F}_q^*/\left(\mathbb{F}_q^*\right)^n \ .$$

Thus an element $f_{\tau,b} \in H^2(\mathrm{G}(L/K), L^*)$ is determined by the triple:

$$\left(L, \tau, b \cdot \left(\mathbb{F}_q^*\right)^n\right) \ .$$

Similarly as in Remark 2.29, if $n = \ell$ is prime, the map:

$$b \cdot \left(\mathbb{F}_q^*\right)^\ell \mapsto b^{\frac{q-1}{\ell}}$$

produces an isomorphism between $\mathbb{F}_q^*/\left(\mathbb{F}_q^*\right)^\ell$ and $\mu_\ell \subset \mathbb{F}_q^*$. It is desirable for practical applications to exploit this isomorphism because it allows a unique representation of the elements in $\mathrm{Br}(K)[\ell]$.

So far we have assumed that $\mu_n \subset K$, but we remark that often, in practice, this is not the case. A description useful for algorithmic purposes is, at the moment, only available if one extends the field K so that it contains the roots of unity and then uses the results presented above. Unfortunately this requires to work in a field which is, in general, much larger than K. It is a challenge to do better.

2.6 Algorithmic Description of the Tate Pairing

As before, we assume that K is a local field with residue field F_q.

We recall the notation introduced in previous section. Let $\mathcal{C}_O$ be an affine, non-singular curve over K and let $\mathcal{C}$ be the unique projective, irreducible and regular curve with the same function field and containing $\mathcal{C}_O$ as affine part. Let $\mathcal{J}$ be the Jacobian variety of $\mathcal{C}$. We simplify the situation and we assume that $\mathcal{C}$ has good reduction, hence it is the lift of a non-singular curve defined over $\mathbb{F}_q$. Moreover, we suppose that only one point at infinity is missing on $\mathcal{C}_O$.

So we have a non-degenerate pairing:

$$T_{L,n}\colon \mathcal{J}(K)/n\mathcal{J}(K) \times H^1\big(\mathrm{G}_K, \mathcal{J}(\overline{K})\big)[n] \to \mathrm{Br}(K)[n] \enspace .$$

Definition 2.33. *The* embedding degree k *is the smallest number with* $q^k \equiv 1 \bmod n$. *Equivalently,* k *is the degree of the smallest extension of* K *containing the* n^{th} *roots of unity.*

Let ζ_n be a n^{th} root of unity and define $K_n := K(\zeta_n)$. It is a local field with residue field $\mathbb{F}_{q^k}$. As above, we choose an uniformizing element $\xi \in K$, define $L := K_n(\xi^{1/n})$ and take a generator τ of $\mathrm{G}(L/K_n)$.

As seen in Section 2.5, we can identify $H^1\big(\mathrm{G}_K, \mathcal{J}(\overline{K})\big)[n]$ with the group of homomorphisms:

$$\big\{\varphi \in \mathrm{Hom}\big(\mathrm{G}(L/K_n), \mathcal{J}(K_n)[n]\big) \text{ with } \varphi(\tau^{-1}) = Q \text{ and } \pi(Q) = qQ\big\} \enspace .$$

We use the explicit description of the Lichtenbaum-Tate pairing given in Example 2.23 – but for simplicity we do not distinguish between divisors on the curve and points on the Jacobian, i.e. divisors classes. Take $\gamma \in H^1\big(\mathrm{G}_K, \mathcal{J}(\overline{K})\big)[n]$ corresponding to φ with $\varphi(\tau^{-1}) = Q$ and $nQ = (f_Q)$. Take $P \in \mathcal{J}(K)$ (represented by a divisor) prime to (f_Q). Then $T_{L,n}(P, \gamma)$ is the class of cyclic algebra $(L, \tau, f_Q(P) \cdot N_{L/K_n} L^*)$.

Hence we get a non-degenerate pairing:

$$T_{n,0}\colon \mathcal{J}(K)/n\mathcal{J}(K) \times \mathcal{J}(\overline{K})[n]^{(q)} \to \mathbb{F}^*_{q^k}/\big(\mathbb{F}^*_{q^k}\big)^n \enspace ,$$

given by evaluation modulo the maximal ideal $\mathfrak{p} \subset O_K$ of a functions f_Q with $(f_Q) = nQ$ and $Q \in \mathcal{J}(\overline{K})[n]^{(q)}$ in $P \in \mathcal{J}(K)$.

We now want to apply these results to a curve defined over a finite field $\mathbb{F}_q$.

Unfortunately, a straightforward application is not possible: in fact, if $\mathcal{C}$ is a curve defined over $\mathbb{F}_q$, then both $H^1\big(\mathrm{G}_{\mathbb{F}_q}, \mathcal{J}(\overline{\mathbb{F}}_q)\big)[n]$ and $\mathrm{Br}(\mathbb{F}_q)$ are trivial, so the Lichtenbaum-Tate pairing is always degenerate.

The way to get it out is to start from a curve $\mathcal{C}_O$ defined over $\mathbb{F}_q$ ($q = p^m$) and consider its p-adic lifting.

Let O be the ring of holomorphic functions of an affine, non-singular curve $\mathcal{C}_O/\mathbb{F}_q$, with corresponding projective curve $\mathcal{C}$. Let $\mathcal{J}$ be the Jacobian variety of $\mathcal{C}$.

Let K be a local field with residue field $\mathbb{F}_q$. We can lift $\mathcal{C}_O$ to an affine, non-singular curve $\mathcal{C}^l_O$ defined over K, embedded in the projective curve $\mathcal{C}^l$ which is a lift of $\mathcal{C}$, such that all relevant data are preserved. In particular, for n prime to q, $\mathrm{Pic}(O^l)/n\,\mathrm{Pic}(O^l)$ is canonically isomorphic to $\mathrm{Pic}(O)/n\,\mathrm{Pic}(O)$.

Now, we can evaluate functions on $\mathcal{C}^l$ modulo the maximal ideal $\mathfrak{p} \subset O_K$ and get an explicit description of the Lichtenbaum-Tate pairing in the case of good reduction which only uses objects attached to the curve $\mathcal{C}_O$.

We can generalize the result stated in Corollary 2.28. For the sake of completeness, we report this final result in its complete form (cf. [Fre08]).

Theorem 2.34. *Assume that* $\mathcal{C}_O$ *is an affine curve with one singular point over* $\mathbb{F}_q$ *whose conductor is square free. Let* O *be the ring of holomorphic functions. Take* n *prime to* q *(and satisfying some "innocent" extra conditions).*

Then we get a non-degenerate pairing:

$$T_n \colon \operatorname{Pic}(O)/n\operatorname{Pic}(O) \times \mathcal{J}_{\mathcal{C}^l}(\overline{K})[n]^{(q)} \to \mathbb{F}_{q^k}^*/\left(\mathbb{F}_{q^k}^*\right)^n \ ,$$

given by evaluation modulo $\mathfrak{p}$ of a function f_Q with $(f_Q) = nQ$ and $Q \in \mathcal{J}_{\mathcal{C}^l}(\overline{K})[n]^{(q)}$ in $P \in \operatorname{Pic}(O)$.

If $\mathcal{C}_O$ is regular, Q and f_Q can be replaced by their reduction modulo $\mathfrak{p}$, i.e. by points and functions over $\mathbb{F}_q$, and we get a pairing:

$$T_{n,0} \colon \operatorname{Pic}(O)/n\operatorname{Pic}(O) \times \mathcal{J}(\overline{\mathbb{F}}_q)[n]^{(q)} \to \mathbb{F}_{q^k}^*/\left(\mathbb{F}_{q^k}^*\right)^n \ .$$

Before concluding this section and the chapter, it is important to make some additional considerations.

First, to get a bilinear structure on Jacobians of curves defined over $\mathbb{F}_q$ – or more precisely on the class group of rings of holomorphic functions of such curves – one has to show that the computation of the pairing is efficient. Because of high degrees n needed in practical applications, a naive approach to evaluate the function $f_Q(P)$ is not possible. The solution was provided by Miller [Mil86b, Mil04] for elliptic curves, later generalized. We will discuss in detail Miller's algorithm in Section 3.4.

Next, we need to bind the notation introduced in this chapter to the one previously seen in Section 1.4, that will be used in the next chapters too. Starting from the last description of the Lichtenbaum-Tate pairing in Theorem 2.34, we have already seen that if $n = \ell$ is prime, then $\operatorname{Pic}(O)/\ell\operatorname{Pic}(O)$ is isomorphic to $\mathcal{J}(\mathbb{F}_q)[\ell]$. Furthermore:

$$\mathcal{J}(\overline{\mathbb{F}}_q)[\ell] \subset \mathcal{J}(\mathbb{F}_{q^k})[\ell] \cong \mathcal{J}(\mathbb{F}_q)[\ell] \times \mathcal{J}(\mathbb{F}_{q^k})[\ell]^{(q)} \ .$$

Note that, by definition, the Frobenius endomorphism π acts on the two components of the ℓ-torsion group respectively as identity and multiplication times q. So, we can define $\overline{\mathbb{G}} := \mathcal{J}(\mathbb{F}_{q^k})[\ell]$,

$$\begin{aligned} \mathbb{G}_1 &:= \mathcal{J}(\mathbb{F}_q)[\ell] = \overline{\mathbb{G}} \cap \operatorname{Ker}(\pi - [1]) \ , \\ \mathbb{G}_2 &:= \mathcal{J}(\mathbb{F}_{q^k})[\ell]^{(q)} = \overline{\mathbb{G}} \cap \operatorname{Ker}(\pi - [q]) \ . \end{aligned}$$

Moreover, $\mathbb{F}_{q^k}^*/(\mathbb{F}_{q^k}^*)^\ell$ is isomorphic to the set of ℓ^{th} roots of unity $\mu_\ell \subset \mathbb{F}_{q^k}^*$. Recall that the isomorphism, realized with an exponentiation, also has the side effect to guarantee a unique representation of the elements, which is desirable for practical applications. Thus we get a non-degenerate (reduced) Lichtenbaum-Tate pairing – that from now on we shall simply call Tate pairing:

$$\hat{t} \colon \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T = \mu_\ell \ .$$

It is worth noticing that for practical applications it may be useful to exploit Weil's reciprocity and compute $f_P(Q)$ in place of $f_Q(P)$ as P is a $\mathbb{F}_q$-rational divisor class.

Last, but not least, we want to apply the construction of the Tate pairing to a TZV. From the theoretical perspective this is straightforward as a TZV $\mathcal{J}_r$ is isomorphic to a subgroup of $\mathcal{J}(\mathbb{F}_{q^r})$, hence the only change one has to made to the whole discussion is to replace $\mathcal{C}$ with its extension of scalars $\mathcal{C}_{\mathbb{F}_{q^r}}$ to $\mathbb{F}_{q^r}/\mathbb{F}_q$ that obviously poses no problems. The only tricky point is the computation of the embedding degree. It is known for supersingular abelian varieties that the embedding degree is invariant under extension of scalars. Unfortunately no technique has been developed to manage the embedding degree of ordinary TZV, and construction of a pairing-friendly (i.e. with constrained embedding degree) family of ordinary TZV is still an open problem.

Chapter 3

Pairing on Supersingular Trace Zero Varieties

Geometry is the art of correct reasoning
on incorrect figures.

In this chapter we review Trace Zero Varieties from a more application-oriented point of view and we discuss their efficient arithmetic that can be achieved exploiting the action of the Frobenius endomorphism. The core of the chapter is however related to pairing: we present a new algorithm to compute the Tate pairing over supersingular TZV that also makes use of the Frobenius endomorphism to improve on performance.

This chapter and the following are essentially built from [AC08a] and [AC08b]. Figures along the chapter come from a poster presented at Eurocrypt 2009, and available in Appendix A.

3.1 Hyperelliptic Curves and Trace Zero Varieties

In this section we introduce the necessary background on hyperelliptic curves, elliptic curves and their trace zero varieties. There are several books on the subject of elliptic and hyperelliptic curves. For reference material within the perspective of cryptographic of applications we refer to [AC+05]. We mention here only a few facts.

Throughout this chapter, $\mathbb{F}_q$ shall denote the Galois field of order q.

3.1.1 Hyperelliptic Curves

A *hyperelliptic curve* $\mathcal{C}$ of genus g over $\mathbb{F}_q$ having an $\mathbb{F}_q$-rational Weierstraß point is a non singular curve defined by the equation:

$$y^2 + h(x)y = f(x), \qquad f \text{ monic, } \deg f = 2g+1, \ \deg h \le g \ . \tag{3.1}$$

A hyperelliptic curve of genus 1 is called an *elliptic curve*. It can be written in the following form:

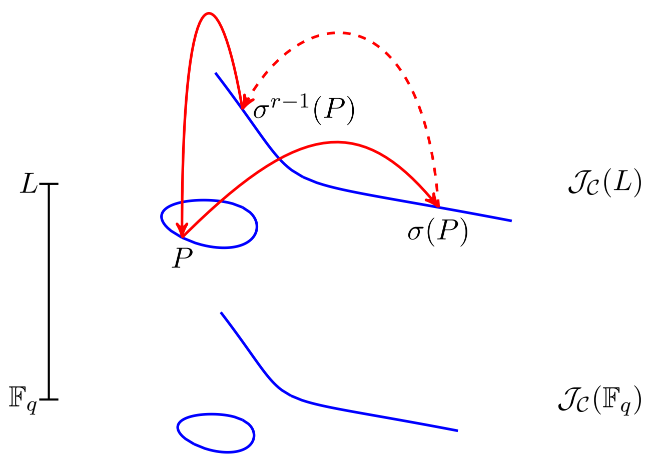
$$y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6 \ , \qquad \text{with} \quad a_i \in \mathbb{F}_q \ . \tag{3.2}$$

For every finite extension $L/\mathbb{F}_q$, the *Jacobian variety* $\mathcal{J}_{\mathcal{C}}(L)$ of $\mathcal{C}$ over L is isomorphic (as group) to the ideal class group over L. Hence we can represent the elements of $\mathcal{J}_{\mathcal{C}}(L)$

by a pair of polynomials with coefficients in L (Mumford's representation, [MWZ98]) and compute the group law using Cantor's algorithm [Can87, Kob89].

The *Frobenius endomorphism* σ operates on a element of $\mathcal{J}_\mathcal{C}(L)$, represented by a pair of polynomials $[u, v]$, by raising each coefficient of u and v to the power of q.

Jacobian variety and Frobenius endomorphism are the main ingredients for the construction of TZV. We give a pictorial representation in Figure 3.1.



$\sigma^{r-1}(P)$
L
$\mathcal{J}_\mathcal{C}(L)$
$\sigma(P)$
P
$\mathbb{F}_q$
$\mathcal{J}_\mathcal{C}(\mathbb{F}_q)$

Figure 3.1: Ingredients for the construction of a TZV

The Frobenius endomorphism σ satisfies the characteristic polynomial:

$$\chi(T) = T^{2g} + a_1 T^{2g-1} + \cdots + a_g T^g + \cdots + a_1 q^{g-1} T + q^g \quad , \tag{3.3}$$

where $a_i \in \mathbb{Z}$. The Hasse–Weil theorem states that from the complex roots τ_i of $\chi(T)$ we can obtain the group order over any extension:

$$|\mathcal{J}_\mathcal{C}(\mathbb{F}_{q^n})| = \prod_{i=1}^{2g} (1 - \tau_i^n) \quad ,$$

in particular $|\mathcal{J}_\mathcal{C}(\mathbb{F}_q)| = \chi(1)$.

If $\mathcal{E}$ is an elliptic curve over $\mathbb{F}_q$, we have:

$$|\mathcal{E}(\mathbb{F}_{q^n})| = (1 - \tau_1^n)(1 - \tau_2^n) = q^n + 1 - t_n \quad , \tag{3.4}$$

where the sequence $(t_n)_{n\in\mathbb{Z}}$ satisfies $t_0 = 2$, $t_1 = t$ and $t_{n+1} = t t_n - q t_{n-1}$, for $n \geq 1$.

Ordinary Elliptic Curves Over Binary Fields

An ordinary (i.e. non supersingular) elliptic curve $\mathcal{E}$ over a binary field $\mathbb{F}_q$ can always be brought to the following Weierstraß form:

$$y^2 + xy = x^3 + ax^2 + b, \qquad \text{with} \quad a \in \mathbb{F}_2,\ b \in \mathbb{F}_q \quad . \tag{3.5}$$

Supersingular Elliptic Curves

Supersingular elliptic curves represent a small set among all elliptic curves on a given field $\mathbb{F}_q$. In characteristic 2 or 3 they are precisely the curves with j-invariant equals to $0 = 1728$, and in characteristic $p > 3$ we have supersingular curves with $j = 0$ if $p \not\equiv 1 \bmod 3$ and with $j = 1728$ if $p \not\equiv 1 \bmod 4$ (see [Sch87] for more details).

This means that we can have only a few (1 or 2) isomorphism classes of supersingular curves over the algebraic closure of $\mathbb{F}_q$ or, in other words, we have only a few twists of a single supersingular curve over $\mathbb{F}_q$. For this reason, the equations of supersingular elliptic curves contains only a few parameters. For instance, over binary fields supersingular elliptic curves have equation:

$$y^2 + y = x^3 + x + b, \qquad \text{with} \quad b \in \mathbb{F}_2 \ .$$

In this work we work with supersingular elliptic curve already considered in [BK$^+$02], whose equations will be given in Lemmas 3.9, 3.10 and 3.11 below.

Ordinary Genus 2 HEC Over Binary Fields

In [CY02, BD04, LS04], ordinary genus 2 hyperelliptic curves over binary fields with a fixed rational point at infinity are classified in terms of the degree of the polynomial $h(x)$. In order to be able to use the Lange–Stevens' doubling formulae [LS04] we shall consider the so-called curves of Type II (cf. [LS04] or see Section 14.5 of [AC$^+$05]), which have an equation of the form:

$$y^2 + xy = x^5 + f_3x^3 + \varepsilon x^2 + f_0, \qquad \varepsilon \in \mathbb{F}_2,\ f_0, f_3 \in \mathbb{F}_q \ . \tag{3.6}$$

3.1.2 Trace Zero Varieties

Let $\mathcal{C}$ be a hyperelliptic curve of genus g over a finite field $\mathbb{F}_q$, given by equation (3.1); let r be a (small) prime number, let $\mathcal{J}_\mathcal{C}(\mathbb{F}_{q^r})$ be the Jacobian variety of $\mathcal{C}$ over $\mathbb{F}_{q^r}$ and let $D \in \mathcal{J}_\mathcal{C}(\mathbb{F}_{q^r})$.

We can formally define the *trace* of D, as for fields:

$$\mathrm{Tr}(D) = D + \sigma(D) + \cdots + \sigma^{r-1}(D) \ .$$

It is clearly an endomorphism of $\mathcal{J}_\mathcal{C}(\mathbb{F}_{q^r})$.

The *trace zero subgroup* of $\mathcal{J}_\mathcal{C}(\mathbb{F}_{q^r})$ is the kernel of this endomorphism

$$\mathbb{G} := \mathrm{Ker}\,\mathrm{Tr} = \{P \in \mathcal{J}_\mathcal{C}(\mathbb{F}_{q^r}) \mid \mathrm{Tr}(P) = \mathcal{O}\} \ ,$$

where $\mathcal{O}$ is the neutral element in $\mathcal{J}_\mathcal{C}(\mathbb{F}_{q^r})$, and as such it is a group. This is isomorphic to a codimension g subvariety $\mathcal{J}_r$ of the Weil restriction of scalars of $\mathcal{J}_\mathcal{C}$ from $\mathbb{F}_{q^r}$ to $\mathbb{F}_q$, whence the name *Trace Zero Variety* (TZV).

Now, the intersection of $\mathbb{G}$ and $\mathcal{J}_\mathcal{C}(\mathbb{F}_q)$ is formed by the r-torsion elements of $\mathcal{J}_\mathcal{C}(\mathbb{F}_q)$. Let $v = \gcd(r, |\mathcal{J}_\mathcal{C}(\mathbb{F}_q)|)$. If $v = 1$ then this intersection is empty and we can compute the group order as follows:

$$|\mathbb{G}| = \frac{|\mathcal{J}_\mathcal{C}(\mathbb{F}_{q^r})|}{|\mathcal{J}_\mathcal{C}(\mathbb{F}_q)|} \ .$$

If $v \neq 1$ then $\mathbb{G} \cap \mathcal{J}_\mathcal{C}(\mathbb{F}_q)[r] \neq \emptyset$, but, being interested in subgroups of large prime order, we could still take a subgroup of $\mathbb{G}$ not intersecting the r-torsion. However, we can restrict ourselves to the case $v = 1$ in what follows.

From the cryptographic point of view, only the following cases are relevant: $g = 1,\ r = 3$; $g = 1,\ r = 5$ and $g = 2,\ r = 3$, namely TZV of elliptic curves over extensions of degree 3 and 5, and TZV of genus 2 hyperelliptic curves over extensions of degree 3 (cf. Section 3.8).

The following proposition (cf. [AL04]) gives us the group orders and useful bounds.

Proposition 3.1. *For the group order of TZV in the considered cases we have:*

1. *For $g = 1$ and $r = 3$:*
$$|\mathbb{G}| = q^2 - q(1 + a_1) + a_1^2 - a_1 + 1 \tag{3.7}$$
and
$$|\mathbb{G}| \leq q^2 + 2q^{3/2} + 3q + 2q^{1/2} + 1 \ ; \tag{3.8}$$

2. *For $g = 1$ and $r = 5$:*
$$|\mathbb{G}| = q^4-(a_1+1)q^3+(a_1+1)^2q^2 +\bigl(5a_1-(a_1+1)^3\bigr)q-\bigl(5a_1(a_1^2+a_1+1)-(a_1+1)^4\bigr)$$
and
$$|\mathbb{G}| \leq q^q + 2q^{7/2} + 3q^3 + 4q^{5/2} + 5q^2 + 4q^{3/2} + 3q + 2q^{1/2} + 1 \ ;$$

3. *For $g = 2$ and $r = 3$:*
$$|\mathbb{G}| = q^4-a_1q^3+(a_1^2+2a_1-a_2-1)q^2 +(-a_1^2-a_1a_2+2a_1)q+a_1^2+a_2^2-a_1a_2-a_1-a_2+1$$
and
$$|\mathbb{G}| \leq q^4 + 4q^{7/2} + 10q^3 + 16q^{5/2} + 19q^2 + 16q^{3/2} + 10q + 4q^{1/2} + 1 \ .$$

Here, the integers a_1, a_2 are coefficients of the characteristic polynomial of the Frobenius endomophism (3.3).

We can already see an advantage in using TZV: to count the number of points on such a variety we need to determine the characteristic polynomial of a curve defined over a smaller field than we would have considering plain elliptic and hyperelliptic curves with similar group size. This makes counting points on trace zero varieties much faster and allows a faster generation of cryptographically suitable curves.

Moreover, in characteristic 2 the group of rational points of an ordinary elliptic curve always has even order, requiring to work in a subgroup of small index (usually 2 or 4). TZV (as well as Jacobians of higher genus ordinary hyperelliptic curves) do not suffer this problem, since their order can be prime.

For cryptographic applications, we want to work in a subgroup $\mathbb{G}_1$ of $\mathbb{G}$ of large prime order ℓ (that may be $\mathbb{G}$ itself).

3.2 Arithmetic in $\mathbb{G}_1$

Arithmetic in $\mathbb{G}_1$ is performed using formulae for the whole group $\mathcal{J}_\mathcal{C}(\mathbb{F}_{q^r})$. We may however speed up the scalar multiplication by making use of the Frobenius operation σ. It is still an open problem to find explicit (and faster) formulae for TZV.

If $\mathbb{G}_1$ is generated by D of order ℓ prime, then $\sigma(D) = sD$, for some integer s, that can be computed as $T - s = \gcd(T^r - 1, \chi(T))$ in $\mathbb{F}_\ell[T]$ (cf. [AL04, AC08b]):

1. For $g = 1$ and $r = 3$: $s \equiv \dfrac{q-1}{1-a_1} \bmod \ell$;

2. For $g = 1$ and $r = 5$: $s \equiv \dfrac{q^2 - q - a_1^2 q + a_1 q + 1}{q - 2a_1 q + a_1^3 - a_1^2 + a_1 - 1} \bmod \ell$;

3. For $g = 2$ and $r = 3$: $s \equiv -\dfrac{q^2 - a_2 + a_1}{a_1 q - a_2 + 1} \bmod \ell$,

where the integers a_1, a_2 are coefficients of the characteristic polynomial of the Frobenius endomophism (3.3).

Using this result we want to replace any scalar multiplication mD ($|m| \leq \ell/2$) with the computation of

$$m_0 D + m_1 \sigma(D) + \cdots + m_{r-1}\sigma^{r-1}(D) \ ,$$

where $m_i = O(\ell^{1/(r-1)}) = O(q^g)$. In this way we are reducing the size of the scalars, and hence the number of "doubling" operations in a double-and-add scalar multiplication algorithm.

Since in cryptographic algorithms m is usually random, we can achieve the reduction following two ways: generate a random integer m, then split it in opportune m_i; or directly generate some m_i, by making sure to avoid collisions, i.e. different sequences of m_i give different elements of $\mathbb{G}_1$.

The second approach has the advantage to save the time necessary to split the scalar, but in some context (for instance for digital signature verification) it is not applicable. Hence we are going to describe both.

3.2.1 Scalar Splitting

Given an integer m with, without loss of generality, $|m| \leq \frac{\ell}{2} = \frac{|\mathbb{G}_1|}{2}$, we want to compute some m_i of bounded size, such that:

$$mD = m_0 D + m_1 \sigma(D) + \cdots + m_{r-1}\sigma^{r-1}(D) \ .$$

First we write $m = n_0 + k_1 q^g$, with $|n_0| \leq q^{g/2}$. Then we use the fact that σ operate in $\mathbb{G}_1$ as multiplication by s and the following relations modulo the group size ℓ:

$$\chi(s) \equiv 0 \bmod \ell \quad \text{and} \quad s^{r-1} + \cdots + s + 1 \equiv 0 \bmod \ell \ .$$

Proceeding in this direction we can expand m as desired, bounding each m_i to $O(q^g)$.

Theorem 3.2. *For the three cases we consider, there exists an efficient technique for expressing a scalar m in the form $m \equiv \sum_{i=0}^{r-2} m_i s^i \bmod \ell$ where $m_i = O(q^g)$. We have:*

1. *For $g = 1$ and $r = 3$, we have $|m_i| < 4q$ if $k \geq 7$;*

2. *For $g = 1$ and $r = 5$, we have $|m_i| < 2q$ if $k \geq 17$;*

3. *For $g = 2$ and $r = 3$, we have $|m_i| < 4q^2$ if $k \geq 23$.*

Proof. The proof is essentially the same as in odd characteristic, as reported in [AL04], and so we limit here ourselves to sketch the approach for $g = 1$, $r = 3$ and to briefly remark the differences in the other two cases.

Let $g = 1$ and $r = 3$. Write first $m = n_0 + k_1 q$, with $|n_0| \leq q/2$. Now use the fact that $\chi(s) \equiv 0 \bmod \ell$, i.e. that $s^2 + a_1 s + q \equiv 0 \bmod \ell$ to get: $m \equiv n_0 + k_1(-a_1 s - s^2) \bmod \ell$. The term $-a_1 k_1$ is $O(q^{3/2})$, which is still too big, hence we need to reduce again: $-a_1 k_1 = n_1 + k_2 q$, with $|n_1| \leq q/2$. Then

$$m \equiv n_0 + n_1 s + k_2(-a_1 s^2 - s^3) - k_1 s^2 \bmod \ell \ .$$

All the coefficients are now $O(q)$.

At this time, we need to reduce the terms in s^2 and s^3, using the trace relation $s^2 + s + 1 \equiv 0 \bmod \ell$. We obtain:

$$m \equiv m_0 + m_1 s \quad \bmod \ell \ ,$$

with $m_0 = n_0 + k_1 - k_2 + a_1 k_2 = O(q)$ and $m_1 = n_1 + k_1 + a_1 k_2 = O(q)$. We may now explicitly bound m_0, m_1 using the estimation on the group size ℓ, given by (3.8), and finally obtain that $|m_0|$ and $|m_1|$ are both lower than $4q$ if $q > 73$, i.e. $k \geq 7$.

For $g = 1$, $r = 5$ we continue to reduce $n_i \equiv -a_1 k_i - k_{i-1} \bmod q$, $k_{i+1} := \frac{-a_1 k_i - k_{i-1} - n_i}{q}$, for $1 \leq i \leq 5$ (assuming $k_0 = 0$); The relation $\chi(s) \equiv 0 \bmod \ell$ is the same as above, and to reduce the powers of s greater than 4 the trace zero relation is $s^4 + s^3 + s^2 + s + 1 \equiv 0 \bmod \ell$. Proceeding as in the case $g = 1$, $r = 3$, we get a relation $m \equiv m_0 + m_1 s + m_2 s^2 + m_3 s^3 \bmod \ell$ where $|m_1| < 91q$. In order to get the sharp bounds stated in the theorem, we reduce m_1 again once $m_1 = m_1' + k_2'(-a_1 s - s^2)$ and thus we consider the coefficients $m_0, m_1', m_2 - a_1 k_2'$ and $m_3 - k_2'$.

For $g = 2$, $r = 3$, since the constant term of the characteristic polynomial is q^2 we reduce as in the previous cases, but by taking quotients and rests by q^2. In order to get the sharp bounds m_1 must be reduced again as in the case $g = 1$, $r = 5$. □

As already mentioned, to construct the m_i one computes quotients and takes remainders modulo q^g, and for trace zero varieties in characteristic 2 this amounts to simple bit shifting or masking operations, which are very efficient.

3.2.2 Multi–scalar Generation

We consider now the technique of beginning with $(r-1)$-tuples of scalars, instead of deriving them from a single scalar. To avoid collisions, i.e. in order to ensure that different tuples $(m_0, ..., m_{r-2})$ and $(m_0', ..., m_{r-2}')$ yield different elements of $\mathbb{G}_1$, or equivalently $\sum_{i=0}^{r-2} m_i s^i \not\equiv \sum_{i=0}^{r-2} m_i' s^i \bmod \ell$, we use the following theorem to bound the size of the scalars. The proofs of the three cases are immediate adaptations of the proofs found respectively in [Nau99, AL04, Lan01] for the case where the definition field has odd characteristic.

Theorem 3.3. *Let D be a generator of $\mathbb{G}_1$. Then the B^{r-1} elements $b_0 D + \cdots + b_{r-2}\sigma^{r-2}(D)$ are pairwise distinct for $b_i < B$, where:*

1. *For $g = 1$ and $r = 3$,*

$$B := \min\left\{\frac{\ell}{q - a_1}, \frac{q-1}{\gcd(q-1, a_1 - 1)}\right\} \ ;$$

2. *For $g = 1$ and $r = 5$,*

$$B := \min\left\{\frac{\ell}{(1+q+|a_1|q)M}, \frac{|q^2 - a_1^2 q + a_1 q - q + 1|}{\gamma}\right\} \quad ,$$

where $M = \max\left\{|q^2 - a_1^2 q + 3a_1 q - 2q - a_1^3 + a_1^2 - a_1 + 2|,\ |q^2 - a_1^2 q - a_1 q + a_1^3 - a_1^2 + a_1|\right\}$ *and* $\gamma = \gcd(q^2 - a_1^2 q - q + 1, 2a_1 q - q - a_1^3 + a_1^2 - a_1 + 1)$*;*

3. *For $g = 2$ and $r = 3$,*

$$B := \min\left\{\frac{\ell}{M}, \frac{q^2 - a_2 + a_1}{\gcd(q^2 - a_2 + a_1, a_1 q - a_2 + 1)}\right\} \quad ,$$

where $M = \max\left\{|q^2 + a_1 q - 2a_2 + a_1 + 1|,\ |q^2 + a_1 - a_1 q - 1|\right\}$.

Here, the integers a_1, a_2 are coefficients of the characteristic polynomial of the Frobenius endomophism (3.3)*.*

Once a $(r-1)$-tuple $(m_0, ..., m_{r-2})$ has been generated whose components are smaller than B, scalar multiplication can be performed by multi-exponentiation techniques. The problems arise when generating the curves for cryptographic purposes. In fact, we want B to be $O(q^g)$ with the implied constant as close to 1 as possible, and this requires testing more curves. If B is too small, less points on the curve can be generated than desired. We want to remark that in the case $g = 1$, $r = 5$ the parameter B is of the order of $\sqrt{q}$, which means that TZV on extensions of degree 5 are not "good" from this point of view.

In odd characteristic, the search for a proper parameter B makes generating good curves even more difficult, but scalar splitting is slower than generating the multi-scalars directly, so this can be the preferred way. On the contrary, in characteristic 2 scalar splitting is much faster and thus probably the preferred way: note, however, that it requires the implementation of a simple multiprecision integer library besides the binary arithmetic.

3.3 Supersingular Trace Zero Varieties and Tate Pairing

From now on, we assume $\mathcal{E}$ is an elliptic curve over $\mathbb{F}_q$, given by the equation (3.2). Let h be the embedding degree[1]of $\mathcal{E}$, i.e. the multiplicative order of q modulo $|\mathcal{E}(\mathbb{F}_q)|$.

Let k be the multiplicative order of q modulo ℓ. Rubin and Silverberg (cf. [RS09, Theorem 9.2]) show that if $\mathcal{E}$ is supersingular and $\gcd(r, 2qh) = 1$, then $k = rh$ and k is the embedding degree of $\mathcal{E}_r$. Moreover, they show (but see also [Nau99] and Section 4.3.3) that points of $\mathbb{G}$ can be compressed to elements of $(\mathbb{F}_q)^{r-1}$, allowing for a boost of the effective "security multiplier" by a factor $r/(r-1)$.

This motivates the use of supersingular TZV for pairing computation. For this, we need to also consider points of the TZV defined over $\mathbb{F}_{q^k}$. Let $\overline{\mathbb{G}}$ be the subgroup of $\mathcal{E}(\mathbb{F}_{q^k})$ whose points have trace $\mathcal{O}$ and order ℓ – from a geometrical point of view this is isomorphic to $\mathcal{E}_r[\ell]$.

[1]Rubin and Silverberg [RS09] define the cryptographic exponent and prove that it is a finer invariant than the embedding degree; however for the cases of our interest (see Lemmas 3.9, 3.10 and 3.11 below) the two measures coincide.

From Frey and Rück [FR94] (see also [AC+05, Prop. 6.12]) it follows that the Tate pairing induces a pairing

$$T\colon \overline{\mathbb{G}} \times \overline{\mathbb{G}} \to \mathbb{F}_{q^k}^*/(\mathbb{F}_{q^k}^*)^\ell$$

which is non-degenerate on the left, i.e., if $T(P,Q) = 1$ for all $Q \in \overline{\mathbb{G}}$ then $P = \mathcal{O}$.

Now, an alternative definition for $\mathbb{G}_1$ is to set $\mathbb{G}_1 = \overline{\mathbb{G}} \cap \mathrm{Ker}(\pi - [1])$, where $\pi = \sigma^r$ is the $(q^r)^{\text{th}}$ power Frobenius endomorphism; let $\mathbb{G}_2 = \overline{\mathbb{G}} \cap \mathrm{Ker}(\pi - [q^r])$ and let μ_ℓ be the set of ℓ^{th} roots of unity in $\mathbb{F}_{q^k}^*$.

Then there is a non-degenerate, bilinear (reduced) Tate pairing:

$$\hat{t}\colon \mathbb{G}_1 \times \mathbb{G}_2 \to \mu_\ell \subset \mathbb{F}_{q^k}^* \ .$$

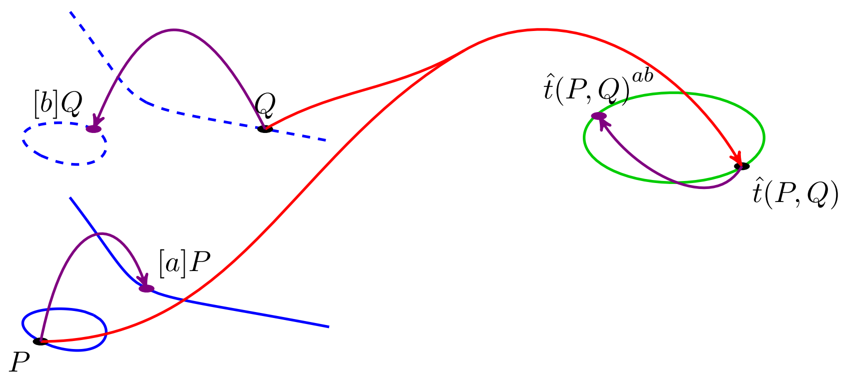


Figure 3.2: Bilinear (reduced) Tate pairing $\hat{t}(\cdot,\cdot)$

Figure 3.2 depicts the Tate pairing map, for simplicity over an elliptic curve. The point P is usually taken from the group $\mathbb{G}_1$, i.e. a subgroup of the group of rational points over the ground field. The point Q must be selected in the full ℓ-torsion group (otherwise the two points would be linearly dependent, and the pairing degenerates), i.e. on the group of rational points over the extension field, whose degree is given by the embedding degree. Finally, the pairing takes values in the subgroup of the ℓ^{th} roots of unity of the multiplicative group of the extension field. By bilinearity, $\hat{t}\left([a]P,[b]Q\right) = \hat{t}(P,Q)^{ab}$.

Most improvements in the pairing computation rely on the action of π in the two groups $\mathbb{G}_1$ and $\mathbb{G}_2$, namely identity and multiplication by q^r. We will present the most important ones in Section 3.5.

For TZV however another efficient endomorphism exists, σ, which acts in $\mathbb{G}_1$ as scalar multiplication by s.

It can easily be shown that σ acts on $\mathbb{G}_2$ as scalar multiplication by an integer S that can be computed as $T - S = \gcd(T^r - q^r, \chi(T))$ in $\mathbb{F}_\ell[T]$:

1. For $r = 3$: $S \equiv \dfrac{q-q^2}{q+t} \bmod \ell$;

2. For $r = 5$: $S \equiv -\dfrac{q^4 - q^3 + (1+t)q^2 - t^2 q}{q^3 + (1+t)q^2 - (t^2+2t)q + t^3} \bmod \ell$.

3.4 The Miller Function and Miller's Algorithm

Let $K/\mathbb{F}_q$ be a finite extension field and let $\mathcal{E}/\mathbb{F}_q$ be an elliptic curve. Here we shall set $K = \mathbb{F}_q$ to get results on $\mathcal{E}$, and $K = \mathbb{F}_{q^r}$ to obtain results that, by restriction, apply to $\mathcal{E}_r$.

For $P \in \mathcal{E}(K)$ and $n \in \mathbb{Z}$, the Miller function $f_{n,P}$ is a K-rational function with divisor:

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

The Miller function is defined up to a constant $c \in K^*$. For every point $Q \neq P$, $[n]P$, $\mathcal{O}$, we have $f_{n,P}(Q) \in K^*$. Let ℓ be a (large) prime dividing $|\mathcal{E}(K)|$. It is a known fact that, upon putting $f_P := f_{\ell,P}$, the function $W(P,Q) = f_P(Q)/f_Q(P)$ defines a non-degenerate, anti-symmetric bilinear pairing on $\mathcal{E}[\ell]$. Since this function for our purposes has a huge degree, to evaluate it efficiently Miller suggested an iterative approach.

For every $\lambda, \mu \in \mathbb{Z}$ the following properties hold:

$$f_{\lambda+\mu,P} = f_{\lambda,P} \cdot f_{\mu,P} \cdot \frac{l_{[\lambda]P,[\mu]P}}{v_{[\lambda+\mu]P}} \tag{3.9}$$

$$f_{\lambda\mu,P} = f^{\mu}_{\lambda,P} \cdot f_{\mu,[\lambda]P} \ , \tag{3.10}$$

where $l_{[\lambda]P,[\mu]P}$ and $v_{[\lambda+\mu]P}$ are resp. the line through $[\lambda]P$, $[\mu]P$ and the vertical line through $[\lambda+\mu]P$.

From these properties it is then easy to derive Miller's algorithm [Mil86b, Mil04], which we present as Algorithm 3.1.

Algorithm 3.1: Miller's algorithm for elliptic curves

```
Input: $P, Q \in \mathcal{E}[\ell], n = \sum_{i=0}^{L} n_i 2^i \in \mathbb{N}$   $(P \neq Q,\ n_i \in \{0,1\},\ n_L = 1)$
Output: $f_{n,P}(Q)$
1  $T \leftarrow P$ , $f \leftarrow 1$
2  for $j = L-1$ downto $0$ do
3      $f \leftarrow f^2 \cdot l_{T,T}(Q)/v_{[2]T}(Q)$
4      $T \leftarrow [2]T$
5      if $n_j = 1$ then
6          $f \leftarrow f \cdot l_{T,P}(Q)/v_{T+P}(Q)$
7          $T \leftarrow T + P$
8      end
9  end
10 return $f$
```

The property stated by equation (3.10) is simple to understand if we refer to Miller's algorithm: suppose we are calculating $f_{n,P}(Q)$ with $n = \lambda\mu$ and we already computed $f_{\lambda,P}(Q)$; then we still need to perform a loop on μ, starting from the point $[\lambda]P$, i.e. we still need $f_{\mu,[\lambda]P}(Q)$; moreover the quantity $f_{\lambda,P}(Q)$ will be iteratively accumulated during the loop, thus raising it to the power of μ.

Hence a single loop on $n = \lambda\mu$ can be "reduced" onto two independent loops on λ and μ (plus one exponentiation to μ). However, this property alone does not allow to reduce the complexity of the Miller's algorithm, since the two loops require to work with distinct points, resp. P and $[\lambda]P$.

For every $k \in \mathbb{N}$, from (3.10) one easily derives (cf. [BG+07, HSV06, both in Lemma 2]):

$$f_{\lambda^k,P} = f_{\lambda,P}^{\lambda^{k-1}} \cdot f_{\lambda,[\lambda]P}^{\lambda^{k-2}} \cdots f_{\lambda,[\lambda^{k-1}]P} \quad . \tag{3.11}$$

We now want to introduce the effect of the Frobenius endomorphism on the computation of the Miller function. Let $\sigma \in \mathrm{Gal}_{\bar{K}/K}$. Clearly we have $f_{n,P^\sigma} = f_{n,P}^\sigma$ and hence the Miller function is Galois invariant in the following sense:

$$f_{n,P^\sigma}(Q^\sigma) = (f_{n,P}(Q))^\sigma \quad . \tag{3.12}$$

The next proposition describes the relationship between endomorphisms and the Miller function. The proof is included for completeness.

Proposition 3.4. *Let $\phi \in \mathrm{End}_K(\mathcal{E})$ with $\mathrm{Ker}\,\phi = \{\mathcal{O}\}$. Then, up to a factor in K^*:*

$$f_{n,\phi(P)} \circ \phi = (f_{n,P})^{\deg \phi} \quad , \tag{3.13}$$

In particular:

(i) *If ϕ is purely inseparable of degree q, $f_{n,\phi(P)} \circ \phi = f_{n,P}^q$.*

(ii) *If ϕ is an automorphism, $f_{n,\phi(P)} = f_{n,P} \circ \phi^{-1}$.*

Proof. The left hand side of (3.13) is the pullback $\phi^* f_{n,\phi(P)}$. The equality up to elements in K^* holds because $(\phi^* f) = \phi^*(f)$, for every $f \in K(\mathcal{E})$ [Sil86, Ch. II, Prop. 3.6(b)]. Since $\mathrm{Ker}\,\phi = \{\mathcal{O}\}$, $P = \phi^{-1}(\phi(P))$ is unique and the pullback of $(f_{n,\phi(P)})$ is:

$$\phi^*\big(n(\phi(P)) - ([n]\phi(P)) - (n-1)(\mathcal{O})\big) = \deg\phi\big(n(P) - ([n]P) - (n-1)(\mathcal{O})\big) \quad ,$$

($[n]$ commutes with ϕ). The first statement holds. If ϕ is an automorphism, the right-side composition of (3.13) with ϕ^{-1} gives the second statement. □

Remarks 3.5.

1. *With a slightly misleading notation, Prop. 3.4.ii provides a way to compute $f_{n,\phi(P)}(Q)$ based on $f_{n,P}$, at the price of evaluating it in $\phi^{-1}(Q)$; the misleading notation is because if $\phi\colon \mathcal{E} \to \mathcal{E}$, then P is naturally taken from the "left" $\mathcal{E}$, while the point Q from the right one.*

2. *If $\phi(Q) = Q$, then Prop. 3.4.i allows to reduce $f_{n,\phi(P)}$ to a power of $f_{n,P}$. This has been used to define the twisted Ate pairing (see [HSV06] and Section 3.5.1).*

To round off this section, we go back to the notation previously introduced and let $K = \mathbb{F}_{q^r}$. We have:

$$\hat{t} \;:\; \mathbb{G}_1 \times \mathbb{G}_2 \to \mu_\ell \subset \mathbb{F}_{q^k}^* \qquad \text{with} \qquad \hat{t}(P,Q) = f_{\ell,P}(Q)^{\frac{q^k-1}{\ell}} \quad ,$$

where $f_{\ell,P}(Q)$ can be computed with Miller's algorithm, and the *final exponentiation* to $\frac{q^k-1}{\ell}$ allows a unique representation of the result, useful for practical applications.

Several improvements to the basic algorithm are possible (see, e.g., [BG+07]). For instance, for every integer n such that $\ell \mid n \mid q^k - 1$, we have:

$$\hat{t}(P,Q) = f_{n,P}(Q)^{\frac{q^k-1}{n}} \quad . \tag{3.14}$$

Hence, one can compute the Tate pairing choosing a multiple of ℓ with lower Hamming weight, to reduce the number of additions in the Miller loop.

For future reference we also describe another improvement, the *denominator elimination* [BK+02]: note that the final exponentiation eliminates terms defined over subfields, and hence these can be omitted from the calculations. Since we are dealing with supersingular varieties, there exists a distorsion map $\psi\colon \mathbb{G}_1 \to \mathbb{G}_2$ which can be used to transfer points in the first group to the second one[2]. If such a distortion map is chosen so that the x-coordinates always lies in a subfield, then all terms $v(Q)$ in Algorithm 3.1 may be eliminated. As a result, we no longer have any division in Miller's algorithm.

3.5 A Survey of Pairings

In this section we survey different algorithms to compute bilinear pairings. All of them require to compute $f_{a,X}(Y)^b$ for some integer a, b and some points X, Y. For the purpose of comparing them, we will refer to a as the *(Miller) loop length*. At least for moderate security levels, the exponentiation to b is negligible with respect to the computation of $f_{a,X}(Y)$ (cf. [KM05]). We note that Hess [Hes08] recently proposes a new framework which encompasses all known pairing functions based on the Tate and Weil pairings, including the ones mentioned here.

We begin by defining t_n as the algorithm that computes the Tate pairing as in (3.14); the key point is that we run a Miller loop on n or, rephrasing, this algorithm has a loop length of n.

Example 3.6. *Let $q = 2^m$ and $\mathcal{E}/\mathbb{F}_q : y^2 + y = x^3 + x + b$, with $b \in \mathbb{F}_2$.*

Let $\mathcal{E}_3$ be the TZV built upon $\mathcal{E}$ and an extension of degree $r = 3$, whose order according to (3.7) is:

$$N = q^2 - q(1-t) + t^2 + t + 1 \ , \qquad \text{with } t = \pm\sqrt{2q} \ .$$

Let ℓ be a large prime dividing N, i.e. $N = \ell c$, where c is a small cofactor. A real example is with $m = 103$ and $b = 1$, having ℓ a 192-bit prime (cf. [RS09]).

Let σ the q^{th} power Frobenius endomorphism; we have $\pi = \sigma^3$. Given a point $P' \in \mathcal{E}(\mathbb{F}_{2^{3m}})$, $P = c\,(P' - \sigma(P'))$ is either $\mathcal{O}$ or, as we assume, a generator of $\mathbb{G}_1$.

$\mathcal{E}$ has embedding degree $h = 4$, so the embedding degree of $\mathcal{E}_3$ is $k = rh = 12$. For every $\tilde{Q} \in \mathbb{G}_1$, $Q = \psi(\tilde{Q}) \in \mathbb{G}_2$ where ψ is a distortion map (an actual example is given in [BG+07] and in Section 4.3).

We define two algorithms for computing the Tate pairing:

$$\mathsf{t}_N(P,Q) := f_{N,P}(Q)^{\frac{q^k-1}{N}} \qquad\qquad \mathsf{t}_\ell(P,Q) := f_{\ell,P}(Q)^{\frac{q^k-1}{\ell}} \ .$$

The first one looks promising, since N has a low Hamming weight, at least in NAF representation; for both, the loop length is $O(q^2)$.

[2]Although on supersingular curves the distortion map allows to define a symmetric pairing, we prefer to keep distinct the two groups $\mathbb{G}_1$ and $\mathbb{G}_2$, in order to have a better understanding of the two components of the ℓ-torsion group.

3.5.1 Ate Pairing

The Ate pairing was defined in [HSV06]; we introduce it in a similar fashion as in [Ver08].

For every $L \in \mathbb{Z}$ such that $\ell \nmid L$,

$$f_{\ell L,P}(Q)^{\frac{q^k-1}{\ell}} = \hat{t}(P,Q)^L \tag{3.15}$$

is a non-degenerate, bilinear pairing.

Let $\lambda \equiv q^r \bmod \ell$, e.g. $\lambda = t_r - 1$ with the notation of (3.4). Then $\ell \mid \lambda^k - 1$ since $\ell \mid q^k - 1$. Let L such that $\ell L = \lambda^k - 1$ and note $\ell \nmid L$:

$$\hat{t}(P,Q)^L = f_{\ell L,P}(Q)^{\frac{q^k-1}{\ell}} = f_{\lambda^k-1,P}(Q)^{\frac{q^k-1}{\ell}} = f_{\lambda^k,P}(Q)^{\frac{q^k-1}{\ell}} \quad . \tag{3.16}$$

Using (3.11) and $[\lambda^i]P = [q^{ir}]P$:

$$f_{\lambda^k,P} = f_{\lambda,P}^{\lambda^{k-1}} \cdot f_{\lambda,[q^r]P}^{\lambda^{k-2}} \cdots f_{\lambda,[q^{(k-1)r}]P} \quad .$$

In order to achieve a better algorithm for computing a bilinear pairing, we would like to reduce the computation of $f_{\lambda,[q^{ir}]P}$ to the computation of $f_{\lambda,P}$. Unfortunately this is not possible for a general curve.

By exchanging the roles of P and Q, it is instead possible to exploit the Miller function Galois-invariance (3.12) against the $(q^r)^{\text{th}}$ power Frobenius π:

$$f_{n,[q^{ir}]Q}(P) = f_{n,\pi^i(Q)}\left(\pi^i(P)\right) = f_{n,Q^{\pi^i}}\left(P^{\pi^i}\right) = (f_{n,Q}(P))^{\pi^i} = (f_{n,Q}(P))^{q^{ir}} \quad .$$

In conclusion we have:

$$\hat{t}(Q,P)^L = f_{\lambda^k,Q}(P)^{\frac{q^k-1}{\ell}} = f_{\lambda,Q}(P)^{\frac{q^k-1}{\ell}\sum_{i=0}^{k-1}\lambda^{k-1-i}q^{ir}} \quad ,$$

and we define the algorithm:

$$\mathsf{a}_\lambda(Q,P) := f_{\lambda,Q}(P)^{\frac{q^k-1}{\ell}} \quad .$$

It computes a non-degenerate bilinear pairing, which is a fixed power of the Tate pairing. Such an algorithm, however, requires to perform a Miller loop on $Q \in \mathcal{E}(\mathbb{F}_{q^k})$, which is fairly less efficient than working with $P \in \mathcal{E}(K)$ (the latter is sometimes referred as Miller lite loop/algorithm).

By further generalization, we can take $\lambda \equiv q^{ir} \bmod \ell$, for any $1 \leq i < k$; we refer to [ZZH07] for more details.

3.5.2 Eta and Eta$_T$ Pairings

As already noted, the Ate pairing requires to switch between P and Q, which is not a good choice from the implementation perspective.

The twisted Ate pairing [HSV06] has been defined to overcome this problem for ordinary curves.

Supersingular curves allow to swap Q and P in a more natural way, and this in fact was described before the introduction of the Ate pairing, by defining the Eta and Eta$_T$ pairings (cf. [BG+07]). We prefer to introduce them, however, from an a-posteriori point of view.

Let $\mathcal{E}$ be supersingular. Denote $\hat{\pi}$ the dual of the $(q^r)^{\text{th}}$ power Frobenius π, also called Verschiebung. Since $\mathcal{E}$ is supersingular, $\mathcal{E}[q^r] = \{\mathcal{O}\}$ and $\hat{\pi}$ is purely inseparable. Since $\pi \circ \hat{\pi} = [q^r]$, $\hat{\pi}$ acts on $\mathbb{G}_1$, resp. $\mathbb{G}_2$, as π acts on $\mathbb{G}_2$, resp. $\mathbb{G}_1$.

We fit into the hypothesis of Prop. 3.4.i (setting $\phi = \hat{\pi}^i$), so we have:

$$f_{n,[q^{ir}]P}(Q) = f_{n,\hat{\pi}^i(P)}\left(\hat{\pi}^i(Q)\right) = f_{n,\hat{\pi}^i(P)} \circ \hat{\pi}^i\,(Q) = (f_{n,P}(Q))^{q^{ir}} \quad ,$$

and repeating the arguments of the previous section, we can define the algorithm:

$$\mathsf{a}^{\mathsf{t}}_{\lambda}(P,Q) := f_{\lambda,P}(Q)^{\frac{q^k-1}{\ell}} \quad ,$$

which computes a non-degenerate, bilinear pairing.

The Eta and Eta$_T$ pairings were defined in [BG+07]. The point of view is slightly different, but the final result almost coincides with the one we just achieved. We repeat the original argument, since we are going to use a similar approach in the proof of Theorem 3.14.

The starting point is a supersingular elliptic curve $\mathcal{E}$ with even embedding degree k and a distortion map ψ that admits denominator elimination [Sco04]. Let T such that $T^a + 1 = \ell L$, for some positive integers a and L, and $T \equiv q^r \bmod \ell$. Suppose there exists an automorphism γ of $\mathcal{E}$ such that $\gamma(P) = [T]P$ and $\gamma \circ \psi^{\pi} = \psi$, or equivalently $\gamma^{-1} \circ \psi = \psi^{\pi}$. We have:

$$\hat{t}(P,Q)^L = f_{\ell L,P}(Q)^{\frac{q^k-1}{\ell}} = f_{T^a+1,P}(Q)^{\frac{q^k-1}{\ell}} = f_{T^a,P}(Q)^{\frac{q^k-1}{\ell}} \quad .$$

Compare the last equality with the similar derivation done for the Ate pairing in (3.16): let $[n]P = \mathcal{O}$; for Ate we used $f_{n,P} = f_{n+1,P}$, which is always true; here $f_{n,P} = f_{n-1,P} \cdot v_P$ holds, where v_P is the vertical line through P, whose contribution cancels out assuming the distortion map permits denominator elimination.

Again using (3.11) we reduce the computation of $f_{T^a,P}$ to powers of $f_{T,[T^i]P}$ for $0 \le i < a$. We have $[T^i]P = \gamma^i(P)$ and by using Prop. 3.4.ii and Galois invariance (3.12):

$$\begin{aligned} f_{T,\gamma^i(P)}\,(Q) &= f_{T,P} \circ \gamma^{-i}\,(Q) = \\ &= f_{T,P} \circ \gamma^{-i}\,(\mathbb{Q}) = f_{T,P}\left(\psi^{\pi^i}(\tilde{Q})\right) = f_{T,P}\left(\mathbb{Q}^{\pi^i}\right) = \\ &= f_{T,P^{\pi^i}}(Q^{\pi^i}) = (f_{T,P}(Q))^{\pi^i} = (f_{T,P}(Q))^{q^{ir}} \quad . \end{aligned}$$

Finally, $T \equiv q^r \bmod \ell$ allows to replace the last exponent with T^i when raising to the power of $\frac{q^k-1}{\ell}$. In conclusion we have:

$$\hat{t}(P,Q)^L = f_{T^a,P}(Q)^{\frac{q^k-1}{\ell}} = f_{T,P}(Q)^{\frac{q^k-1}{\ell} a T^{a-1}} \quad ,$$

and we define the algorithms:

$$\eta_T(P,Q) := f_{T,P}(Q)^{\frac{q^k-1}{\ell}} \qquad \text{and} \qquad \eta(P,Q) := \eta_{q^r}(P,Q) \quad .$$

For $T = \lambda = t_r - 1$ (and $a = k/2$), the algorithms η_T and $\mathsf{a}^{\mathsf{t}}_{\lambda}$ coincide.

Example 3.7. *Let $\mathcal{E}_3$ be defined as in Example 3.6 and η, η_T as before. The loop length of η is 2^{3m}, which is worse than a direct computation from the definition of Tate pairing using t_N. The loop length of η_T is T, in particular for $T = t_3 - 1 = \mp 2^{(3m+1)/2} - 1$, we have an algorithm with a loop length $O(q^{3/2})$.*

3.5.3 Optimal (Twisted) Ate Pairing

Vercauteren [Ver08] defines the concept of *optimal pairing* and describes an algorithm to compute optimal Ate pairings. Hess [Hes08] extends this framework by (i) allowing for more general pairing functions and (ii) applying it to the Weil pairing; however no better explicit algorithm is given than in [Ver08].

The starting point is again (3.15). Let $\lambda = \ell L$ and write $\lambda = \sum_{i=0}^{a} c_i q^i$. Given the vector $[c_0, \ldots, c_a]$, define the following algorithm:

$$\mathsf{a}^{\mathrm{t}}_{[c_0,\ldots,c_a]}(P,Q) := \left(\prod_{i=0}^{a} f_{c_i,P}(Q)^{q^i} \cdot C(P,Q) \right)^{\frac{q^k-1}{\ell}} ,$$

where $C(P,Q)$ plays the role of a "correction" term and is given by:

$$C(P,Q) = \prod_{i=0}^{a-1} \frac{l_{[s_{i+1}]P,[c_i q^i]P}(Q)}{v_{[s_i]P}(Q)}, \qquad \text{with } s_i = \sum_{j=i}^{a} c_j q^j \ .$$

Theorem 1 in [Ver08] shows that $\mathsf{a}^{\mathrm{t}}_{[c_0,\ldots,c_a]}(P,Q)$ computes a bilinear pairing and states a condition for non-degeneracy. Furthermore, an algorithm to explicitly derive useful vectors $[c_0, \ldots, c_a]$ is given, based on finding short vectors in a proper lattice. Given such a small vector V, we define the algorithm[3]:

$$\mathsf{a}_{\mathrm{opt}}(P,Q) := \mathsf{a}^{\mathrm{t}}_{V}(P,Q) \ .$$

We illustrate it through an explicit example.

Example 3.8. *This example is similar to the one presented in [Ver08, Section 4], related to supersingular elliptic curves over $\mathbb{F}_{3^m}$, with $k = 6$. Let $\mathcal{E}_3$ be defined as in Example 3.6.*

The shortest vector is $V = [v_0, v_1] = [2^{(3m-1)/2}, 2^{(3m-1)/2} \mp 1]$, and "another nice choice" is $W = [2^{(3m+1)/2}, -1]$ that gives the η_T pairing.

We have $\ell \mid v_0 + v_1 q^3 = 2^{(3m-1)/2} \cdot \left|\mathcal{E}(\mathbb{F}_{q^3})\right|$. We consider the algorithm:

$$\mathsf{a}_{\mathrm{opt}} := \mathsf{a}_V = f_{v_0,P}(Q) \cdot f_{v_1,P}(Q)^{q^3} \cdot l_{[v_0]P,[v_1 q^3]P}(Q) \ ;$$

we note that $[v_0]P = -[v_1 q^3]P$, so $l_{[v_0]P,[v_1 q^3]P}$ is actually the vertical line through $[v_0]P$ and we can ignore it because the distortion map admits denominator elimination.

It remains to show how to efficiently compute the product of the two Miller functions: since $v_1 = v_0 - 1$, by (3.9) we have $f_{v_1,P}(Q)^{q^3} = f_{v_0,P}(Q)^{q^3} \cdot l_{[v_0]P,[v_0-1]P}$; here we can not avoid the final multiplication because $[v_0]P \neq \mathcal{O}$. In conclusion

$$\mathsf{a}_{\mathrm{opt}} = l_{[v_0]P,[v_0-1]P} \cdot f_{v_0,P}(Q)^{1+q^3} \ , \qquad \text{where } v_0 = 2^{(3m-1)/2} \ .$$

The loop length is $O(2^{(3m-1)/2})$, but a final correction term is required, thus this algorithm is essentially equivalent to η_T.

[3]Actually the property for V to be small is only a necessary condition for the related Ate pairing to be optimal; a detailed discussion is out of the scope of this work and we refer to the original paper for further details.

3.6 Pairing on Supersingular Trace Zero Varieties

We already presented through examples how to apply the current literature to a particular TZV. We stress the following facts:

- The presented pairings make use of the $(q^r)^{\text{th}}$ power Frobenius π. They apply not only to points of the TZV, but to the whole $\mathcal{E}(\mathbb{F}_{q^r})$.

- The supersingular TZV in the examples is perfectly equivalent from a security perspective to the supersingular elliptic curve in characteristic 3 (small characteristic field, effective security multiplier of 6, see also Lemma 3.9). Furthermore, since $q = 2^m$ and $t = 2^{(m+1)/2}$, most of the algorithms come with a very efficient (i.e. low Hamming weight) Miller loop; the two pairings η_T and a_{opt} achieve the shortest loops.

3.6.1 Preliminaries

We begin by looking at how to exploit the action of the q^{th} power Frobenius endomorphism σ. From previous sections we already know that σ acts on $\mathbb{G}_1$, resp. $\mathbb{G}_2$, as the scalar multiplication by an integer s, resp. S.

The next lemmas introduce three relevant families of supersingular TZV and shows that, in this context, the action of σ can be better made explicit and s, S are indeed very close to powers of q.

The starting point to build these TZV are the supersingular curves considered in [BK⁺02], for which there exist distortion maps that permit denominator elimination.

Lemma 3.9 (Supersingular $\mathcal{E}_3$ over $\mathbb{F}_{2^m}$)**.** *Let $\mathcal{E}/\mathbb{F}_{2^m}$ (m prime) be a supersingular elliptic curve defined by the Weierstraß equation:*

$$y^2 + y = x^3 + x + b \ , \qquad b \in \mathbb{F}_2 \ ,$$

with embedding degree $h = 4$.

Let $\mathcal{E}_3$ be the TZV built upon $\mathcal{E}$ and an extension of degree $r = 3$, with embedding degree $k = rh = 12$. As already mentioned, $P \in \mathbb{G}_1$ can be compressed to two elements in $\mathbb{F}_q$ (see also Section 4.3.3), hence $\mathcal{E}_3$ provides an effective security multiplier of $12/2 = 6$.

For every $P \in \mathbb{G}_1$ and $Q \in \mathbb{G}_2$, we have:

$$\sigma(P) = -\left[q^{\frac{r+1}{2}}\right] P = -\left[q^2\right] P \ , \qquad \sigma(Q) = \left[q^{\frac{r^2+1}{2}}\right] Q = \left[q^5\right] Q \ .$$

Proof. We have to prove that $s \equiv -q^{\frac{r+1}{2}}$ and $S \equiv q^{\frac{r^2+1}{2}}$ satisfy their respective defining polynomials, i.e. $s^r \equiv 1$ and $\chi(s) \equiv 0$, resp. $S^r \equiv q^r$ and $\chi(S) \equiv 0$ (all the equivalences are intended modulo ℓ if not otherwise specified).

Note h is exactly $r + 1$: this, togheter with the relationship $k = rh$, allows us to work out the expressions of s, resp. S, and to exploit the definition of embedding degree. Another key remark is that since k is the smallest integer such that $q^k \equiv 1$ and it is even, we have $q^{k/2} \equiv -1$ or equivalently $-q^{k/2} \equiv 1$.

By substituting $s \equiv -q^{\frac{r+1}{2}}$ into s^r, we have $s^r \equiv -q^{rh/2} = -q^{k/2} \equiv 1$. To prove $S^r \equiv q^r$, observe that $r \equiv -1 \bmod h$, hence $\frac{r^2+1}{2} \equiv 1 \bmod h$, or equivalently $\frac{r^2+1}{2} = 1 + ch$ for some integer c. Thus $S^r \equiv q^{(1+ch)r} = q^{r+ck} \equiv q^r$.

We now prove that both s and S are roots of $\chi(T)$. Write

$$\chi(s) \equiv \left(q^r + tq^{\frac{r-1}{2}} + 1\right) q \ , \qquad \chi(S) \equiv \left(q^{r^2} - tq^{\frac{r^2-1}{2}} + 1\right) q \ . \tag{3.17}$$

Using (3.4), we can compute $t_r = -tq^{\frac{r-1}{2}}$, resp. $t_{r^2} = tq^{\frac{r^2-1}{2}}$ (for $r = 3$, starting with $t = \sqrt{2q}$). By this, the expressions in parentheses in (3.17) are $|\mathcal{E}(\mathbb{F}_{q^r})|$, resp. $\left|\mathcal{E}(\mathbb{F}_{q^{r^2}})\right|$. Since $\ell \mid |\mathcal{E}(\mathbb{F}_q)| \mid |\mathcal{E}(\mathbb{F}_{q^r})| \mid \left|\mathcal{E}(\mathbb{F}_{q^{r^2}})\right|$, both expressions vanish modulo ℓ and $\chi(s) \equiv 0$, resp. $\chi(S) \equiv 0$. □

Lemma 3.10 (Supersingular $\mathcal{E}_5$ over $\mathbb{F}_{3^m}$)**.** *Let $\mathcal{E}/\mathbb{F}_{3^m}$ (m prime) be a supersingular elliptic curve defined by the Weierstraß equation:*

$$y^2 = x^3 - x \pm 1 \ ,$$

with embedding degree $h = 6$.

Let $\mathcal{E}_3$ be the TZV built upon $\mathcal{E}$ and an extension of degree $r = 5$, with embedding degree $k = 30$ and effective security multiplier 7.5.

For every $P \in \mathbb{G}_1$ and $Q \in \mathbb{G}_2$, we have:

$$\sigma(P) = -\left[q^{\frac{r+1}{2}}\right] P = -\left[q^3\right] P \ , \qquad \sigma(Q) = \left[q^{\frac{r^2+1}{2}}\right] Q = \left[q^{13}\right] Q \ .$$

Proof. The proof proceeds exactly as in Lemma 3.9. Here again $h = r + 1$, $\chi(s) \equiv q \cdot |\mathcal{E}(\mathbb{F}_{q^r})|$ and $\chi(S) \equiv q \cdot \left|\mathcal{E}(\mathbb{F}_{q^{r^2}})\right| \bmod \ell$. □

Lemma 3.11 (Supersingular $\mathcal{E}_3$ over $\mathbb{F}_p$)**.** *Let $\mathcal{E}/\mathbb{F}_p$ ($p > 3$ prime) be a supersingular elliptic curve with embedding degree $h = 2$.*

Let $\mathcal{E}_3$ be the TZV built upon $\mathcal{E}$ and an extension of degree $r = 3$, with embedding degree $k = 6$ and effective security multiplier 3.

For every $P \in \mathbb{G}_1$ and $Q \in \mathbb{G}_2$, we have: Then:

$$\sigma(P) = \left[p^{\frac{r+1}{2}}\right] P = \left[p^2\right] P \ , \qquad \sigma(Q) = \left[p^{\frac{r^2+1}{2}}\right] Q = \left[p^5\right] Q \ .$$

Proof. Since $t = 0$ we proceed by direct computation:

$$s \equiv p - 1 \equiv p^2 = p^{\frac{r+1}{2}} \ , \qquad S \equiv (p - p^2)/p \equiv 1 - p \equiv p^5 = p^{\frac{r^2+1}{2}} \ .$$

□

In what follows, we assume that $\mathcal{E}$ is one of the curves defined in Lemma 3.9, 3.10 or 3.11 and put $\Sigma = \frac{r^2+1}{2}$. This allows to adopt the following common notation:

$$\sigma(P) = [s]\, P = \left[\pm q^{\frac{r+1}{2}}\right] P \ , \qquad \sigma(Q) = [S]\, Q = \left[q^{\Sigma}\right] Q \ .$$

The next lemma describes the action of $\hat{\sigma}$, the dual of the Frobenius endomorphism σ, called the Verschiebung. Our main theorem will then follow.

Lemma 3.12. *Let $\mathcal{E}_r$ be a TZV as in Lemma 3.9, 3.10 or 3.11. Let $\hat\sigma$ be the dual of the Frobenius endomorphism σ. Then:*

$$\hat\sigma^i(P) = \left[\left(qs^{-1}\right)^i\right] P \ , \qquad \hat\sigma^i(Q) = \left[q^{i(1-\Sigma)}\right] Q \ .$$

Moreover $\hat\sigma^{r+2}(P) = [q]P$.

Proof. Let $X \in \mathcal{E}[\ell]$, let $z \in \mathbb{Z}$t, $0 < z < \ell$ and suppose $\sigma(X) = [z]X$. As immediate consequence of $\hat\sigma\sigma = [\deg\sigma]$, we obtain $\hat\sigma(X) = \left[qz^{-1}\right] X$ and thus $\hat\sigma^i(X) = \left[\left(qz^{-1}\right)^i\right] X$. This proves the first result on $\hat\sigma^i(P)$, resp. $\hat\sigma^i(Q)$, setting $X = P$, $z = s$, resp. $X = Q$, $z = S = q^\Sigma$.

By this, we have $\hat\sigma^{r+2}(P) = [q]P$ if and only if $\left(qs^{-1}\right)^{r+2} \equiv q \bmod \ell$ if and only if $q^{r+1} \equiv s^{r+2} \equiv s^2 \bmod \ell$, and this holds by Lemma 3.9, 3.10 and 3.11. □

Theorem 3.13. *Let $\mathcal{E}_r$ be a TZV as in Lemma 3.9, 3.10 or 3.11. Then there exist an integer j, $0 \le j < k$, such that:*

$$\left(\hat\sigma^{r+2} \circ \sigma^j\right)(Q) = Q \qquad \textit{for every } Q \in \mathbb{G}_2 \ ,$$

and:

$$f_{n,[s]P}(Q) = f_{n,P}\left(Q^{\sigma^{-1}}\right)^\sigma = f_{n,P}\left(\left[q^{k-\Sigma}\right] Q\right)^q \tag{3.18}$$

$$f_{n,[q]P}(Q) = f_{n,P}\left(Q^{\sigma^j}\right)^{q^{r+2}} = f_{n,P}\left(\left[q^{j\Sigma}\right] Q\right)^{q^{r+2}} \tag{3.19}$$

for every $P \in \mathbb{G}_1$ and $Q \in \mathbb{G}_2$.

Proof. Equation (3.18) comes from $[s]P = P^\sigma$ and (3.12).

For (3.19) we have:

$$f_{n,[q]P}(Q) = f_{n,\hat\sigma^{r+2}(P)}\left((\hat\sigma^{r+2} \circ \sigma^j)(Q)\right) = f_{n,\hat\sigma^{r+2}(P)} \circ \hat\sigma^{r+2}\left(Q^{\sigma^j}\right) =$$
$$= f_{n,P}\left(Q^{\sigma^j}\right)^{q^{r+2}} = f_{n,P}\left(\left[q^{j\Sigma}\right] Q\right)^{q^{r+2}} \ ,$$

where the first equality follows by Lemma 3.12 and the third by repeatedly applying Prop. 3.4 (i), setting $\phi = \hat\sigma$.

We now proceed to explicitly compute the integer j. Using the previous lemmas, we can rewrite $\left(\hat\sigma^{r+2} \circ \sigma^j\right)(Q) = Q$ as: $q^{(r+2)(1-\Sigma)} \cdot q^{j\Sigma} \equiv 1 \bmod \ell$, which holds if and only if $(r+2)(1-\Sigma) + j\Sigma \equiv 0 \bmod k$ (here we are tacitly assuming that k is the effective embedding degree, i.e. ℓth roots of unity are not defined over a smaller extension of $\mathbb{F}_q$ than $\mathbb{F}_{q^k}$; this is always true for supersingular curves).

In our setting Σ is invertible modulo k and thus such a j can be found. Explicitly we get $j \equiv 4 \bmod k$ when $r = 3$ resp. $j \equiv 18 \bmod k$ when $r = 5$. □

3.6.2 A New Algorithm for the Tate Pairing

We now use the last result to derive a new formula for the Tate pairing. Figure 3.3 gives a graphical representation of the setting of Theorem 3.14: we build a TZV on a supersingular elliptic curve and over an extension of degree r; it has embedding degree $k = rh$, where h is the embedding degree of the underlying curve. As the curve is

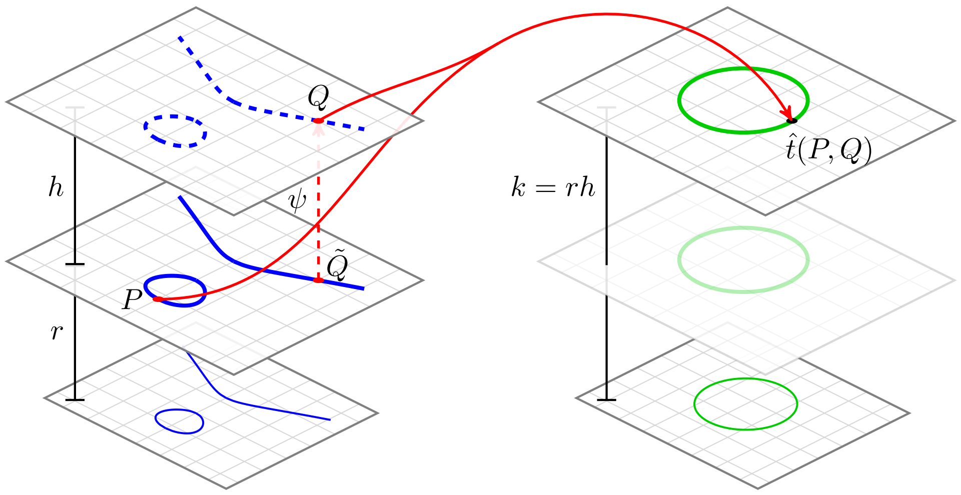


Figure 3.3: Reduced Tate pairing $\hat{t}(\cdot, \cdot)$ over supersingular TZV

supersingular, there exists a distortion map $\psi \colon \mathbb{G}_1 \to \mathbb{G}_2$, $\tilde{Q} \mapsto Q$ and we further assume the distortion map admits denominator elimination (cf. Section 3.4). We want to efficiently compute the Tate pairing $\hat{t}(P, Q)$, exploiting the action of the q^{th} power Frobenius endomorphism σ, available on the TZV (see also Figure 3.1).

Theorem 3.14. *Let $\mathcal{E}_r$ be a TZV as in Lemma 3.9, 3.10 or 3.11. In particular the embedding degree k is even and there exists a distortion map ψ that admits denominator elimination (cf. Section 3.4).*

Then, for every $P \in \mathbb{G}_1$ and $Q \in \mathbb{G}_2$, the Tate pairing can be computed as:

$$\hat{t}(P, Q) = \left(\prod_{i=0}^{r-1} f_{q,P} \left(Q^{\sigma_i}\right)^{q^{i(r+1)}} \right)^{M \frac{a}{r} q^{a-1}} , \tag{3.20}$$

where $\sigma_i = \sigma^{ij}$ (where j is given in Theorem 3.13), $a = k/2$ and $M = q^{k/2} - 1$.

Proof. We proceed by proving the following two equalities:

$$\hat{t}(P, Q) = f_{q^a,P}(Q)^M = \left(\prod_{i=0}^{r-1} f_{q,[q^i]P} \left(Q\right)^{q^{-i}} \right)^{M \frac{a}{r} q^{a-1}} , \tag{3.21}$$

then we will exploit Theorem 3.13 and thus the action of the q^{th} power Frobenius σ to prove the thesis.

We note that the first part of the proof has many similarities with the proof of Theorem 1 in [BG+07]. First, we use the special form of the curve to reduce $\hat{t}(P, Q)$ to the evaluation of $f_{q^a,P}$: the scalar is now a power of q. Then we use the action of the $(q^r)^{\text{th}}$ power Frobenius π to change $f_{q^a,P}$ into a product of functions $f_{q,[q^i]P}$.

We begin by the first equality in (3.21). Since k is even, we have $\ell \mid q^k - 1 = (q^{k/2} - 1)(q^{k/2} + 1)$ and, by the minimality of k, $\ell \mid q^{k/2} + 1$. Hence:

$$\hat{t}(P, Q) = f_{\ell,P}(Q)^{(q^k-1)/\ell} = f_{q^{k/2}+1,P}(Q)^{q^{k/2}-1} = f_{q^{k/2},P}(Q)^{q^{k/2}-1} .$$

The last equality deserves more attention (see also [BG+07, Lemma 3]): we know that $\left[q^{k/2}+1\right] P = \mathcal{O}$, which implies $\left[q^{k/2}\right] P = -P$; by (3.9) we have

$$f_{q^{k/2}+1,P} = f_{q^{k/2},P} \cdot v_P \ ,$$

where v_P is the vertical line through P (and $-P$). Let $\tilde{Q} \in \mathbb{G}_1$ such that $Q = \psi(\tilde{Q})$. Since ψ admits denominator elimination, $v_P(\psi(\tilde{Q})) \in \mathbb{F}^*_{q^{k/2}}$, thus its contribution is canceled by raising to the power of $q^{k/2}-1$.

Set $a = k/2$, $M = q^{k/2}-1$. We now prove the second equality in (3.21). Exploiting (3.11) we change $f_{q^a,P}$ in the product of a functions $\left\{f_{q,[q^i]P}\right\}_{i=0}^{a-1}$; moreover, the action of the $(q^r)^{\text{th}}$ power Frobenius π allows to group together the functions $f_{q,[q^i]P}$ and $f_{q,[q^{i+r}]P}$, since $f_{q,[q^{i+r}]P}(Q)^M = f_{q,[q^i]P}(Q)^{Mq^r}$ (use Prop. 3.4 (i) with $\phi = \hat{\pi}$, the dual of π; see also [BG+07, Lemma 1]). In detail:

$$\begin{aligned} f_{q^a,P}(Q)^M &= \left(f_{q,P}(Q)^{q^{a-1}} \cdot f_{q,[q]P}(Q)^{q^{a-2}} \cdots f_{q,[q^{a-1}]P}(Q)\right)^M = \\ &= \left(f_{q,P}(Q)^{\frac{a}{r}q^{(a-1)}} \cdot f_{q,[q]P}(Q)^{\frac{a}{r}q^{(a-2)}} \cdots f_{q,[q^{r-1}]P}(Q)^{\frac{a}{r}q^{(a-r)}}\right)^M = \\ &= \left(f_{q,P}(Q) \cdot f_{q,[q]P}(Q)^{q^{-1}} \cdots f_{q,[q^{r-1}]P}(Q)^{q^{-(r-1)}}\right)^{M\frac{a}{r}q^{a-1}} \ . \end{aligned}$$

So far we have proved (3.21). Now, using (3.19) with q^i instead of q, from Theorem 3.13 we have:

$$f_{q,[q^i]P}\left(Q\right)^{q^{-i}} = f_{q,P}\left(Q^{\sigma^{ij}}\right)^{q^{-i+i(r+2)}} = f_{q,P}\left(Q^{\sigma_i}\right)^{q^{i(r+1)}} \ ,$$

and the result follows. □

The previous theorem yields a new method to compute the Tate pairing over supersingular TZV.

We first perform a single Miller loop from P to $[q]P$ and evaluate it at the r points Q^{σ_i} (raising each evaluation to the proper power $q^{i(r+1)}$). At the end compute the final exponentiation to $M\frac{a}{r}q^{a-1}$. This is summarized in Algorithm 3.2.

Algorithm 3.2: Computing $\mathrm{t}_{\mathrm{TZV}}$ on $\mathcal{E}_r$

```
    Input: $P \in \mathbb{G}_1$ , $Q \in \mathbb{G}_2$
    Output: $\mathrm{t}_{\mathrm{TZV}}(P,Q)$
1   $f \leftarrow 1$
2   for $i = 0$ to $r-1$ do
3       $f \leftarrow f \cdot \left(f_{q,P}\left(Q^{\sigma_i}\right)\right)^{q^{i(r+1)}}$
4   end
5   $f \leftarrow \left(f^{\frac{a}{r}}\right)^{q^{a-1}}$
6   return $f^{M+1}/f$
```

Each iteration in the Miller loop requires: (i) a point doubling – and possibly an addition – in $\mathcal{E}(\mathbb{F}_{q^r})$; (ii) r evaluations at Q^{σ_i}; (iii) accumulating the r results, i.e. r multiplications in $\mathbb{F}_{q^k}$; we omit from this operation count the exponentiations, which are computed by Frobenius operations and thus essentially negligible.

We now deal with the final exponentiation to $M\frac{a}{r}q^{a-1}$. (i) Powering to the M is computed as a power of the (r^{th} power of the) Frobenius followed by a division (whose cost is small with respect to the cost of the Miller loop); (ii) Raising to the power of q^{a-1} is again done exploiting the Frobenius; (iii) Exponentiation to $\frac{a}{r}$ is also efficient, since it is resp. 2, 3, 1 for the TZV of our interest. If we avoid the two last exponentiations, we still get a bilinear pairing.

A number of techniques have been described to carry out efficiently these operations, see for instance [BG+07, BB+08] and Section 4.3.

The algorithm is suitable for a parallel implementation, requiring r processors and achieving a Miller loop of length q. Moreover, both in a parallel and in a sequential model, an implementation with precomputation of the multiples of P requires the storage of only $\log_2 q$ points.

Remark 3.15. *Equation (3.18) might also be exploited to derive an algorithm reducing a loop on $s^{r-1} + \cdots + s + 1$ in $r - 1$ loops on s. The result is similar to the use of the endomorphism in the so-called NSS curves [Sco05]. This approach does not seem interesting for TZV because s is generally too big. The best case is in characteristic 2 for $r = 3$: $s = O\left(q^{3/2}\right)$, and the resulting algorithm (even assuming the two loops can be "packed" in some way) cannot be better, for instance, than η_T (cf. [BG+07], see also Section 4.3.5).*

3.6.3 Pairing Compression

To reduce storage or bandwidth requirements, the values of pairing can be compressed using traces or working on algebraic tori [GPS06b, SB04]. For an elliptic curve $\mathcal{E}/\mathbb{F}_q$ with embedding degree h, pairing takes values in the algebraic torus T_h over $\mathbb{F}_q$, and these values can be compressed – at least in theory – to $\mathbb{F}_{q^{\varphi(h)}}$; however, finding explicit and optimal compression algorithms for general h and in all characteristics is still an open problem [vDG+05]. Besides compression, the direct computation of compressed pairing is an interesting topic, in particular for constrained environments [NBS08].

For TZV, pairing compression is particular interesting in force of a result presented in [MRS07, Section 5]. Roughly speaking, $\mathcal{E}_r$ can be seen as a twist of the underlying $\mathcal{E}$ and pairing preserves twisting. Hence, pairing takes values in an algebraic torus T_k of dimension k over $\mathbb{F}_q$. Clearly, to achieve compression, one can restrict the attention to any torus $T_{k/d}$ over $\mathbb{F}_{q^d}$, and for $d = r$ we get back results of the underlying elliptic curve, namely compression on the torus T_h over $\mathbb{F}_{q^r}$, i.e. onto $\mathbb{F}_{q^{r\times\varphi(h)}}$.

Thus, in general with TZV we can expect to achieve additional compression of the pairing values than only considering the underlying curve, in the best case by a factor $r/\varphi(r)$.

For TZV in characteristic $p > 3$ (cf. Lemma 3.11), we can consider the full torus T_6 over $\mathbb{F}_p$ and compress pairing values to $\mathbb{F}_{p^2}$ [GPS06b] (or directly compute compressed values as in [NBS08]), and this improves of the naïf use of T_2 over $\mathbb{F}_{p^3}$, whose elements can be compressed to $\mathbb{F}_{p^3}$.

Characteristic 3 (cf. Lemma 3.10) is an interesting case, since it is represent another actual application of the torus T_{30}, already considered in literature for torus-based cryptography [vDG+05]. Unfortunately, the best available result for compressing elements of T_{30} maps them onto $\mathbb{F}_{q^{10}}$ (see [vDG+05], for characteristic p), but for TZV the same compression can be easily achieved by considering the torus T_6 over $\mathbb{F}_{q^5}$.

Finally, for TZV in characteristic 2 (cf. Lemma 3.9) the algebraic torus T_{12} over $\mathbb{F}_q$, or T_6 over $\mathbb{F}_{q^2}$ could be exploited, in theory to compress to $\mathbb{F}_{q^4}$. However we were not able to find any explicit compression better than the one achieved with T_2 over $\mathbb{F}_{q^6}$, hence onto $\mathbb{F}_{q^6}$, that can be easily achieved by computing the trace of the pairing result (cf. [BG+07]).

3.7 Construction of Varieties and Generators

For cryptographic applications we need to be able to construct:

- TZV whose order is prime or almost prime, and
- A generator of the subgroup $\mathbb{G}_1$ of large group order.

For ordinary curves, in order to find a good curve we generate random ones with random coefficients until we find one with the correct order. By the Hasse-Weil Theorem, we thus need only to fix the field size in order to get groups whose orders are in a relatively narrow interval. To compute the characteristic polynomial of an elliptic curve we use MAGMA, and to find good curves of genus two we modified a software package written by Vercauteren.

For supersingular curves, because of the constrained embedding degree, the security of the field where the pairing takes values is fundamental in force of the MOV attack. Hence, we select a field big enough for the security we wish to obtain, then we search for a curve with a large prime order subgroup, that in general has quite a big index in $\mathbb{G}$.

We need to distinguish between two cases: characteristic 2 or 3, and characteristic $p > 3$. The former case is straightforward: we have two curves for every field by selecting the parameter b in their equation (cf. Lemmas 3.9 and 3.10) and we check if at least one is suitable for cryptography, i.e. its order is divisible by a large prime.

In characteristic $p > 3$, for elliptic curves we select a prime ℓ, then search for a random cofactor c such that $p = c\ell - 1$ is a prime of the desired size ($c\ell = p + 1$ is the order of the curve). On the contrary, for TZV (over an extension of degree $r = 3$) we select a prime p, compute the order of the TZV according to (3.7), and check if it is divisible by a large prime ℓ. The order of the TZV is in general quite big, so the factorization, and thus this search method, is not really efficient.

Once a curve has been found, we need only to take a random divisor D' on it and observe that $D = D' - \sigma(D')$ lies in the TZV. To ensure that it has the right order we replace D by its multiple by the index of $\mathbb{G}_1$ in $\mathbb{G}$: if the result is the neutral element of the group, we try with another choice of D'.

We now present examples of curves (ordinary and supersingular) suitable for cryptographic application. We select two distinct levels of security, namely 80 and 96-bit symmetric security.

For supersingular curves, 70-bit security was also considered in [BK+02, BG+07]. A security level of 70 bits is really poor, but we add it for comparison purposes. As noted in [BK+02], the supersingular elliptic curves over $\mathbb{F}_{3^{127}}$ are not secure (i.e. their group orders are not divisible by large enough primes). For this reason we also introduced the 96-bit security level, according to other works in literature (e.g. [KAT08]).

The equivalent security levels for finite fields and curves, according to ECRYPT II recommendations [ECR09, Table 7.2 and Section 6.2.1], are presented in Table 3.1.

Security (bits)	RSA	DLOG		EC
		Field Size	Subfield	
70	960	960	141	141
80	1248	1248	160	160
96	1776	1776	192	192
128	3248	3248	256	256
256	15424	15424	512	512

Table 3.1: Security levels and related key-size equivalence

In Appendix B we give explicit examples of ordinary curves suitable for cryptographic applications, also used in our experiments in Chapter 4.

In Table 3.2 we give examples of supersingular elliptic curves and TZV suitable for pairing-based cryptography.

Curve equation	Field of def.	Order (bits)	FF Security (bits)	Point (bits)
$\mathcal{E}: y^2 = x^3 + x$	$\mathbb{F}_{p_A}$, $p_A \approx 2^{480}$	140	$480 \times 2 = 960$	481
$\mathcal{E}_3: y^2 = x^3 + x$	$\mathbb{F}_{p_B}$, $p_B \approx 2^{160}$	140	$160 \times 6 = 960$	322
$\mathcal{E}: y^2 + y = x^3 + x + b$, $b = 0$	$\mathbb{F}_{2^{239}}$	200	$239 \times 4 = 956$	239
$\mathcal{E}_3: y^2 + y = x^3 + x + b$, $b = 0$	$\mathbb{F}_{2^{79}}$	141	$79 \times 12 = 948$	160
$\mathcal{E}: y^2 = x^3 - x + b$, $b = 1$	$\mathbb{F}_{3^{97}}$	151	$\log_2(3^{97 \times 6}) = 922$	$155 \div 195$
$\mathcal{E}_5: y^2 = x^3 - x + b$, $b = -1$	$\mathbb{F}_{3^{19}}$	112 *	$\log_2(3^{19 \times 30}) = 903$	insecure
$\mathcal{E}: y^2 = x^3 + x$	$\mathbb{F}_{p_C}$, $p_C \approx 2^{624}$	160	$624 \times 2 = 1248$	625
$\mathcal{E}_3: y^2 = x^3 + x$	$\mathbb{F}_{p_D}$, $p_D \approx 2^{208}$	160	$208 \times 6 = 1248$	418
$\mathcal{E}: y^2 + y = x^3 + x + b$, $b = 0$	$\mathbb{F}_{2^{307}}$	268	$307 \times 4 = 1228$	307
$\mathcal{E}_3: y^2 + y = x^3 + x + b$, $b = 1$	$\mathbb{F}_{2^{103}}$	192	$103 \times 12 = 1236$	208
$\mathcal{E}: y^2 = x^3 - x + b$, $b = -1$	$\mathbb{F}_{3^{127}}$	83 *	$\log_2(3^{127 \times 6}) = 1208$	insecure
$\mathcal{E}_5: y^2 = x^3 - x + b$, $b = -1$	$\mathbb{F}_{3^{29}}$	104 *	$\log_2(3^{29 \times 30}) = 1379$	insecure
$\mathcal{E}: y^2 = x^3 + x$	$\mathbb{F}_{p_E}$, $p_E \approx 2^{888}$	192	$888 \times 2 = 1776$	889
$\mathcal{E}_3: y^2 = x^3 + x$	$\mathbb{F}_{p_F}$, $p_F \approx 2^{296}$	192	$296 \times 3 = 1776$	594
$\mathcal{E}: y^2 + y = x^3 + x + b$, $b = 1$	$\mathbb{F}_{2^{457}}$	457	$457 \times 4 = 1828$	457
$\mathcal{E}_3: y^2 + y = x^3 + x + b$, $b = 1$	$\mathbb{F}_{2^{157}}$	203	$157 \times 12 = 1884$	316
$\mathcal{E}: y^2 = x^3 - x + b$, $b = 1$	$\mathbb{F}_{3^{193}}$	272	$\log_2(3^{193 \times 6}) = 1835$	$307 \div 387$
$\mathcal{E}_5: y^2 = x^3 - x + b$, $b = -1$	$\mathbb{F}_{3^{43}}$	265	$\log_2(3^{43 \times 30}) = 2044$	$279 \div 347$

* Insecure: *Order* too small w.r.t. *Finite Field Security*

Table 3.2: Explicit curves and related security parameters

The column *Order* refers to the size of a prime ℓ dividing the full group order, while *FF Security* refers to the size of the finite field where the pairing takes values.

For a fixed field, both in characteristic 2 and 3, two curves are available by choosing the parameter b in the curve equation. In case both curves are suitable for cryptography, we selected the one with the smallest ℓ.

We marked as insecure the curves whose order ℓ is too small, if compared with the security provided by the finite field where the pairing takes values.

The last column, *Point*, contains the number of bits required to store or transmit a point of the curve, where we took into account compression techniques. For curves in characteristic 3 we provide a range of bits, since the actual size depends upon the binary representation of elements in $\mathbb{F}_{3^m}$. The lower bound is the information theoretical $\log_2(3^m)$, while the upper bound is $2m$, obtained by representing each element in $\mathbb{F}_3$ by 2 bits. Probably the best trade-off is to store a 5-tuple of $\mathbb{F}_3$ elements in 1 byte.

We draw attention to the following remarks:

- In all cases, TZV improve on elliptic curves as they allows a smaller field of definition. This is a direct consequence of the boost of the effective security multiplier (cf. [RS09]).

- The TZV in characteristic 2 is defined over a field which is twice as small as the TZV in characteristic $p > 3$ and 3 times smaller than the elliptic curves; this clearly implies smaller storage or transmission costs. Curves in characteristic 3 are even better in this sense, but unfortunately they are weak at 80-bit security.

3.8 Security Considerations

In this section we make some considerations about the security of the proposed varieties.

To prevent Weil Descent attacks [GHS02b, GHS02a] on the elliptic and hyperelliptic curves which we use to construct the TZV, the degree m of the definition field $\mathbb{F}_q$ over $\mathbb{F}_2$ must be chosen to be a prime.

We now turn the attention to the security of TZV. The case $g = 1,\ r = 3$ is simple: a consideration on the dimension shows that the group of rational points of a TZV defined over a field $\mathbb{F}_q$ with $r = 3$ could be contained in the Jacobian of a curve of genus at least 2 (in fact, the first step of the Weil descent attack is done by the construction of the TZV itself), which means that the best known attack has complexity not worse than $O(q)$ and this meets the square-root bound.

Let us consider now the case of dimension 4, i.e. $g = 1,\ r = 5$ and $g = 2,\ r = 3$. By the arguments developed in [Lan04], and in small characteristic under the pessimistic assumption that the results in [DS03] can also be adapted to fields of characteristic 2 and 3 (or that similar results hold), we cannot exclude that $\mathbb{G}$ is contained (up to isomorphism) in the Jacobian of a hyperelliptic curve of genus 6. In this case the fastest known attack is the double large prime variation of the index calculus algorithm [GT+07, iN04], that has complexity $\widetilde{O}(q^{5/3})$. Thus we have an asymptotic reduction of the security of at most 1/6 of the bit length. Note that these algorithms compute in the whole Jacobian and the size of the latter determines the complexity. However the estimated loss is asymptotic, and in case of high security applications $|\mathbb{G}| \sim 2^{256}$ we can estimate that the security is reduced by at most 6 bits.

At this point it is important to observe that the situation is different than in the case of low genus hyperelliptic curves. In [ATW08], the group sizes for the Jacobians of hyperelliptic curves of genus 3 and 4 had to be increased in order to correctly compare to curves of genus 1 and 2. In fact, for curves of genus 3 and 4, the fastest known attack is the already mentioned double large prime variation of the index calculus algorithm. Ignoring logarithmic terms, this attack requires $O(q^{2-2/g})$ group operations for a genus g over $\mathbb{F}_q$. For a curve of genus 3, resp. 4 over $\mathbb{F}_q$, this means $O(q^{4/3})$, resp. $O(q^{3/2})$ group operations. Ignoring also the constants, we see that to compare with an elliptic curve over a field of m bits, we need a field of $m/2$ bits for genus two, $3m/8$ bits for genus three and $m/3$ bits for genus four. At the current state of research, such losses do not seem to affect TZV.

Finally, the group $\mathbb{G}$ may be also contained in the Jacobian of a non-hyperelliptic curve, the worst case being a $C_{3,5}$ curve, in which case one may want to apply Diem's attack on planar curves, that has a complexity $\widetilde{O}(q^{4/3})$. A counting argument shows that

$\mathbb{G}$ will be contained in Jacobian of a non-hyperelliptic planar curve only with probability $O(1/q^2)$, so in practically all cases this attack can not be applied.

Furthermore, when we consider supersingular TZV for pairing applications we are using larger groups anyway and the complexity of an attack is dominated by the complexity of attacking the system in the finite field in which the pairing function takes its values.

For instance, take $\mathcal{E}_5/\mathbb{F}_{3^{43}}$ from Table 3.2. The order is divisible by a 265-bit prime, hence because of the Pollard's ρ attack its security is 132.5 bits. With the index calculus attack the security is reduced to $\log_2(3^{43\times 5/3}) = 113$ bits. However $\mathcal{E}_5$ is still secure, since it has been chosen to provide 96-bit security. To be more accurate, the pairing take values in $\mathbb{F}_{3^{43\times 30}}$, which offers 2044-bit security and the corresponding security level according to [ECR09] is 103 bits.

Chapter 4

Implementation

Two is a very odd number.

In this chapter we discuss implementation details, starting from the lower level of finite fields (ground fields and extensions) up to Jacobians, TZV and pairing.

The key-word of this chapter is “the rule of the two”.

We pursue essentially 2 goals: to speed-up the arithmetic in the ideal class group of a TZV, and to efficiently compute the Tate pairing over supersingular TZV. The former was started in [Ces04], further developed in [AC08b] and extended in this work by taking into account several coordinate systems, for ordinary and supersingular TZV. The latter is described in [AC08a]. For both, we make use of the Frobenius endomorphism which is available for TZV and, for emotional reasons, most of the chapter is devoted to the case of characteristic 2.

Our implementation supports 2 compilers, the GNU Compiler Collection (`gcc`) and the Intel C/C++ Compiler (`icc`), and 2 architectures at 32 and 64-bits.

We perform experiments on 2 different platforms:

Intel: Intel quad-core 2.8 GHz, OS Fedora GNU/Linux 2.6.30 x86_64, with compiler `gcc` ver. 4.3.2.

PowerPC: Twin dual-core 2.7 GHz G5 PowerMac, OS Darwin Kernel 8.11.0, with compiler `gcc` ver. 4.2.1.

Through the chapter, we shall denote operations in finite fields as follows:

- In the base field $\mathbb{F}_q$: square (`s`), square root (`r`), multiplication (`m`), double multiplication ($\mathtt{m}_2$, see Section 4.1.1), inversion (`i`), solution of quadratic equation of the form $y^2 + y + c = 0$, i.e. a half-trace computation (`h`).
- In the extension field $\mathbb{F}_{q^r}$: square (`S`), multiplication (`M`), vector and double multiplications (`V`, $\mathtt{M}_2$, see Section 4.1.1), inversion (`I`), pseudo-inversion (`P`, see Section 4.1.2), solution of quadratic equation (`H`).

4.1 Finite Fields

4.1.1 Base Fields

Our implementation of binary fields is the one described in the paper [AT07] (see also [ATW08]). Binary fields are represented in polynomial basis and explicit polynomials are given in Appendix B. These polynomials are so-called "*square root friendly*" and allow efficient algorithms for square root extraction [Ava07].

Another relevant aspect of our implementation is the availability of *sequential multiplications* (cf. [AT07]), i.e. sets of multiplications with a common operand. In particular, in the sequel we shall exploit *double multiplications* (denoted by $\mathtt{m}_2$, respectively $\mathtt{M}_2$ in extensions), i.e. the concurrent computation of $r_1 = ab_1$ and $r_2 = ab_2$, and *vector multiplications* in extension fields (denoted by $\mathtt{V}$), i.e. the multiplication of an element $B \in \mathbb{F}_{q^r}$, seen as a vector over $\mathbb{F}_q$, by an element a in the ground field $\mathbb{F}_q$.

For prime fields, the implementation is described in [Ava04], which is used also in [AL04].

4.1.2 Small Degree Extensions of Binary Fields

For binary fields the construction of small extensions follows a different approach with respect to the odd characteristic case, as described for instance in [AL04].

If α is an algebraic element of degree r over $\mathbb{F}_2$, it is also an algebraic element of degree r over $\mathbb{F}_{2^m}$ for every m prime to r. This is our case, m is prime (to avoid Weil descent attacks) and we need extensions of degree 3 and 5 to build TZV and extensions of degree 4 to compute pairings.

Hence we can construct $\mathbb{F}_{q^r}$ with an irreducible polynomial over $\mathbb{F}_2$, which is fixed and independent from the ground field $\mathbb{F}_q$. We used $f(X) = X^3 + X + 1$ for $r = 3$, $f(X) = X^4 + X + 1$ for $r = 4$ and $f(X) = X^5 + X^2 + 1$ for $r = 5$. In other words, we construct $\mathbb{F}_{q^r}$ as the composite of the two fields $\mathbb{F}_q$ and $\mathbb{F}_{2^r}$ over $\mathbb{F}_2$. Since the polynomial is fixed, we can derive explicit formulae for modular reduction, which in the end requires only to perform a few additions in $\mathbb{F}_q$, and no multiplications.

For multiplications, we considered three algorithms: the naïve (schoolbook), the Karatsuba and, for $r = 3$, the Toom-Cook [Too63] (see also [Bod07]). We did not take into account the Schönhage-Strassen [SS71, Sch77] nor the newer algorithm by Fürer [F07] since these are especially designed to be asymptotically fast and become useful only for very big operand sizes.

For inversion, we considered two methods: Kobayashi et al. [KM+99] and Itoh-Tsujii [IT88, GP02]. Both methods reduce the computation of an inversion in $\mathbb{F}_{q^r}$ to a single inversion in $\mathbb{F}_q$.

By omitting such inversion in $\mathbb{F}_q$, instead of the inverse of an element $a \in \mathbb{F}_{q^r}$ we obtain a *pseudo-inverse* $\iota(a) \in \mathbb{F}_{q^r}$ and a *co-pseudo-inverse* $\mathfrak{c}(a) \in \mathbb{F}_q^*$, such that:

$$a^{-1} = \frac{\iota(a)}{\mathfrak{c}(a)} \ .$$

We denote by $\mathtt{P}$ the *pseudo-inversion*, thought as an operation that returns both $\iota(a)$ and $\mathfrak{c}(a)$. Thinking to computational costs, we have that $1\mathtt{I}$ is equivalent to $1\mathtt{i} + 1\mathtt{V} + 1\mathtt{P}$ where $\mathtt{V}$, we recall, is a vector multiplication, i.e. the multiplication of an element in $\mathbb{F}_{q^r}$ – the pseudo-inverse – by an element in $\mathbb{F}_q$ – the inverse of the co-pseudo-inverse. We are going to use pseudo-inversion in Section 4.2.2 to build compressed López-Dahab coordinates for TZV.

In Table 4.1 we point out the cost of operations in the extension field in terms of operations in the base field, both in even and odd characteristic and for the degrees of our interest. In this work extensions of degree 4 are only relevant in characteristic 2 (for pairing computation).

$r = 3$		
	$q = 2^m$	$q = p > 3$
S	3s	3s + 3m
M	5 ÷ 6m	5 ÷ 6m
I	3s + 9m + 1i	3s + 9m + 1i
σ	negl.	2m

$r = 4$	
	$q = 2^m$
S	4s
M	9m
I	25m + 1i
σ	negl.

$r = 5$		
	$q = 2^m$	$q = p > 3$
S	5s	5s + 10m
M	14m	14m
I	39m + 1i	50m + 1i
σ	negl.	4m

Table 4.1: Costs of operations in $\mathbb{F}_{q^r}$ in terms of operations in $\mathbb{F}_q$. Operations considered are square (S), multiplication (M), inversion (I) and Frobenius automorphism (σ).

We refer to [AC08b] for details about the implementation of extensions of degree 3 and 5. For extensions of degree 4 we used Karatsuba for multiplication and we derive an explicit version of the method by Kobayashi et al. for inversion, that saves one multiplication with respect to Itoh-Tsujii method and also allows to exploit sequential multiplications for further gains. Since inversion is not particularly relevant for pairing computation, we omit further details.

4.1.3 Experimental Results

In this section we present some experimental results related to the implementation of binary fields, including small degree extensions.

The focus is not to discuss which is the better algorithm for a specific operation (this, of course, has been done), but to compare performance in finite fields that will be used to build cryptographic primitives, notably scalar multiplication and pairing.

The results are summarized in Tables 4.2, C.1, C.2 and C.3, timings are in microseconds. We compared small extensions and base fields performance: the row *Field* denotes the degree m or $m \times r$ respectively of the base field $\mathbb{F}_{2^m}$ or the extension $\mathbb{F}_{2^{m \times r}}$. Operations considered are Frobenius automorphism (σ), square (S), multiplication (M), double-multiplication ($\mathtt{M}_2$) and inversion (I).

	80-bit security							96-bit security						
Curve:	ord. $g=1$			ord. $g=2$		supers.		ord. $g=1$			ord. $g=2$		supers.	
Field:	163	83×3	41×5	83	41×3	307	103×3	191	97×3	47×5	97	47×3	457	157×3
σ	–	0.01	0.01	–	0.01	–	0.01	–	0.01	0.01	–	0.01	–	0.02
S	0.03	0.05	0.04	0.02	0.03	0.05	0.05	0.03	0.04	0.04	0.01	0.03	0.19	0.10
M	0.22	0.52	0.48	0.09	0.20	0.71	0.69	0.28	0.54	0.51	0.09	0.21	1.90	1.24
$\mathtt{M}_2$	0.38	0.97	0.95	0.16	0.37	1.17	1.21	0.56	0.92	1.01	0.15	0.40	3.79	2.26
I	1.60	1.30	1.44	0.54	0.48	4.93	1.77	1.97	1.36	1.57	0.60	0.55	11.87	3.38
$\mathtt{M}_2/\mathtt{M}$	1.77	1.88	2.00	1.80	1.83	1.66	1.75	2.00	1.70	2.00	1.68	1.88	2.00	1.83
I/M	7.37	2.52	3.03	6.00	2.39	7.00	2.55	6.97	2.50	3.09	6.77	2.54	6.24	2.73

Table 4.2: Comparison among operations in finite fields – Intel (32-bit), timings in μs. Operations considered are Frobenius automorphism (σ), square (S), multiplication (M), double-multiplication ($\mathtt{M}_2$) and inversion (I).

We have grouped together fields suitable to define curves or TZV with the same characteristics. First, we distinguish in terms of security provided (80 or 96-bit security). Then, we regroup by geometrical properties: *ord.* $g = 1$ refers to ordinary elliptic curves and TZV with $g = 1$, $r = 3$ and $r = 5$; *ord.* $g = 2$ refers to ordinary genus 2 hyperelliptic curves and TZV with $g = 2$, $r = 3$; *supers.* refers to supersingular elliptic curves and TZV with $g = 1$, $r = 3$.

We make the following remarks:

- Double-multiplication is 10% faster on average than two independent multiplications. In some fields (e.g. extensions of degree 5) the ratio is exactly 2.00, which means double-multiplication is not implemented.
- In the extensions, the ratio `I/M` is really low. This suggests that affine coordinates for TZV defined over such extensions could outperform more sophisticate coordinate systems. In practical applications, extensions (and TZV) may be used to reduce the loss of performance due to inversion, which is often a critical operation in particular on constrained devices.
- If we consider fields related to ordinary curves, in each group extensions are bigger than the corresponding base field by a factor $r/(r-1)$. This clearly has an impact on multiplication, which is always slower in a extension than in the corresponding base field. On the contrary inversion is usually faster, at least comparable for small fields (i.e. the $g = 2$ cases).
- Fields related to supersingular curves, on the contrary, have comparable sizes (fixed the security level) and this reflects on timings of multiplication, which are now quite close each others. In some cases, e.g. for 32-bit implementations, multiplication in extensions is even faster. Clearly the same benefit is also evident for inversion.

4.2 Scalar Multiplication on TZV in Characteristic 2

4.2.1 Coordinate Systems

The first thing to take into account in the implementation of curve-based cryptography is the choice of the coordinate system, and usually the balance is the cost of the inversion in the ground field.

In Section 4.1.3 we have seen that for TZV the ratio `I/M` is very low and this motivated us, in previous works [Ces04, AC08b], to only consider affine coordinates for TZV. However several alternative coordinate systems have been proposed in literature and in the sequel we are going to provide a detailed comparison.

In this section we introduce coordinate systems with focus on ordinary elliptic curves. An excellent source on this topic is the Explicit-Formulas Database [BL07], which now covers characteristic 2. We shall make some considerations about supersingular elliptic curves in Section 4.2.6, while for genus 2 curves we still considered only affine coordinates.

Let $\mathcal{E}/\mathbb{F}_q$ be an ordinary elliptic curve given by equation 3.5. In *affine coordinates* ($\mathcal{A}$) a point is represented by two coordinates $(x, y) \in \mathbb{F}_q \times \mathbb{F}_q$ and the formulae for doubling and addition require one inversion. For instance, let $P_1 = (x_1, y_1)$ and $P_3 = 2P_1 = (x_3, y_3)$. The explicit formulae for doubling are:

$$\begin{aligned} \lambda &= x_1 + y_1/x_1 \\ x_3 &= \lambda^2 + \lambda + a_2 \\ y_3 &= \lambda(x_1 + x_3) + y_1 + x_3 \quad . \end{aligned} \tag{4.1}$$

In *projective coordinates* ($\mathcal{P}$) a point is represented by the homogeneous coordinates (X, Y, Z) that correspond to $(x, y) = (X/Z, Y/Z)$. Formulae for doubling and addition no longer require inversions, at the price of an additional coordinate – and several multiplications.

To achieve formulae faster than the projective and still without inversions one has to move to weighted projective coordinates. In *Jacobian coordinates* ($\mathcal{J}$) the tuple (X, Y, Z) corresponds to $(x, y) = (X/Z^2, Y/Z^3)$ [AC⁺05] and in *López-Dahab coordinates* ($\mathcal{LD}$) to $(x, y) = (X/Z, Y/Z^2)$ [LD99, AC⁺05]. In these systems, additions are a bit more expensive but doublings get considerably cheaper.

Finally, if memory occupation is not a concern, one can store within the coordinates some computation done during an operation that is reused for next operations. In *extended López-Dahab coordinates* (e$\mathcal{LD}$) a point is represented by (X, Y, Z, Z^2) if $a = 1$ in the curve equation and by (X, Y, Z, Z^2, XZ) if $a = 0$ [KK07, BLF08].

Explicit algorithms for all these coordinates systems can be found in the Explicit-Formulas Database. As an example, we show how to derive López-Dahab doubling formulae. By substituting $(x_i, y_i) = (X_i/Z_i, Y_i/Z_i^2)$, $i \in \{1, 3\}$, into (4.1), one get $\lambda = \frac{X_1^2+Y_1}{X_1Z_1}$ and by properly defining Z_3 and Λ:

$$Z_3 = (X_1Z_1)^2 \qquad \Lambda = X_1^2 + Y_1 \tag{4.2a}$$
$$X_3 = \Lambda^2 + \Lambda(X_1Z_1) + a_2Z_3 \tag{4.2b}$$
$$= X_1^4 + Y_1^2 + X_1^3Z_1 + X_1Y_1Z_1 + a_2X_1^2Z_1^2 \tag{4.2c}$$
$$= X_1^4 + a_6Z_1^4 = (X_1^2 + \sqrt{a_6}Z_1^2)^2 \tag{4.2d}$$
$$Y_3 = \Lambda(X_1Z_1)\big(X_1^2(X_1Z_1) + X_3\big) + Y_1X_1^2Z_3 + X_3Z_3 \tag{4.2e}$$
$$= X_1^4(X_1Z_1)^2 + (\Lambda(X_1Z_1) + Z_3)X_3 \tag{4.2f}$$
$$= X_1^4(X_1Z_1)^2 + (X_1Y_1Z_1 + X_1^3Z_1 + X_1^2Z_1^2)X_3 \tag{4.2g}$$
$$= \big(X_1^2(X_1Z_1) + (Y_1 + \sqrt{a_6}Z_1^2)(X_1^2 + \sqrt{a_6}Z_1^2)\big)^2 \qquad \text{if } a_2 = 1 \tag{4.2h}$$
$$= \big(\sqrt{a_6}Z_1^2(X_1Z_1) + (Y_1 + \sqrt{a_6}Z_1^2)(X_1^2 + \sqrt{a_6}Z_1^2)\big)^2 \qquad \text{if } a_2 = 0 \tag{4.2i}$$

We note that this derivation, i.e. equations (4.2a), (4.2d), (4.2h) and (4.2i), is useful only if the multiplication times $\sqrt{a_6}$ is faster than a multiplication in the field. This holds, for instance, if $r > 1$ (e.g. for TZV) or if $\sqrt{a_6}$ is known to have low Hamming weight. If this is not the case, then using (4.2a), (4.2b) and (4.2f) results in a slightly faster algorithm.

4.2.2 Compressed López-Dahab Coordinates

In this section we introduce compressed López-Dahab coordinates for ordinary elliptic curves over binary fields, that are the corresponding of compressed Jacobian coordinates in characteristic p, firstly proposed by Hoshino et al. [HKA06] (see also [AL04]).

Compressed coordinate systems apply when dealing with curves over extension fields, so for instance TZV. The idea is to use weighted projective coordinates and to exploit pseudo-inversion to continue avoiding inversion in the ground field (for performance) but reducing the size of the Z-coordinate (for memory consumption).

Let $\mathcal{E}/\mathbb{F}_q$ be an elliptic curve given by equation 3.5. We focus on points of $\mathcal{E}(\mathbb{F}_{q^r})$.

In *compressed López-Dahab coordinates* (c$\mathcal{LD}$) a $\mathbb{F}_{q^r}$-rational point is represented by the tuple $(X, Y, z) \in \mathbb{F}_{q^r} \times \mathbb{F}_{q^r} \times \mathbb{F}_q$ that corresponds to $(x, y) = (X/z, Y/z^2)$.

Similarly as for $\mathcal{LD}$ in (4.2), we substitute $(x_i, y_i) = (X_i/z_i, Y_i/z_i^2)$, $i \in \{1, 3\}$, into (4.1), and get $\lambda = \frac{X_1^2+Y_1}{X_1 z_1}$. Now we want to define z_3 such that it is in $\mathbb{F}_q$ but still cancels out the denominator in λ. The idea is to use pseudo-inversion: define $\overline{X}_1 = \iota(X_1)$ and $x_1 = \mathfrak{c}(X_1)$. Then

$$\lambda = \frac{\overline{X}_1(X_1^2 + Y_1)}{x_1 z_1}$$

and hence:

$$z_3 = (x_1 z_1)^2 \qquad \Lambda = \overline{X}_1(X_1^2 + Y_1) \tag{4.3a}$$

$$X_3 = \Lambda^2 + \Lambda(x_1 z_1) + a_2 z_3 \tag{4.3b}$$

$$Y_3 = \Lambda(x_1 z_1)\big(X_1 x_1(x_1 z_1) + X_3\big) + Y_1 x_1^2 z_3 + X_3 z_3 \tag{4.3c}$$

$$= X_1^2 x_1^2 (x_1 z_1)^2 + (\Lambda(x_1 z_1) + z_3) X_3 \tag{4.3d}$$

We note that these new formulae contain two additional operations: one is pseudo-inversion, which is also quite expensive, the other one is the "correction" of Λ via multiplication by $\overline{X}_1$. The good is that now some multiplications become vector multiplications, i.e. by elements in $\mathbb{F}_q$, or even better multiplications in $\mathbb{F}_q$. In Section 4.2.4 we shall experimentally compare c$\mathcal{LD}$ with the other coordinate systems.

From (4.3a), (4.3b) and (4.3d) – that corresponds to (4.2a), (4.2b) and (4.2f) – we can derive an algorithm to compute doubling in compressed López-Dahab coordinates, that we report as Algorithm 4.1. In a similar way one can derive an algorithm for addition. We omit details and present it as Algorithm 4.2. In this algorithm, we put in evidence the optimization in case $z_2 = 1$, i.e. the second input is an affine point.

Algorithm 4.1: Doubling in Compressed López-Dahab Coordinates

```
Input: $P_1 = (X_1 : Y_1 : z_1) \in \mathcal{E}(\mathbb{F}_q^r)$
Output: $P_3 = 2P_1 = (X_3 : Y_3 : z_3)$
1 $\overline{X}_1 \leftarrow \iota(X_1)$    $x_1 \leftarrow \mathfrak{c}(X_1)$        1P
2 $a \leftarrow x_1 z_1$                                   1m
3 $B \leftarrow X_1^2$                                     1S
4 $\Lambda \leftarrow \overline{X}_1(B + Y_1)$             1M
5 $C \leftarrow \Lambda a$                                 1V
6 $z_3 \leftarrow a^2$                                     1s
7 $X_3 \leftarrow \Lambda^2 + C + a_2 z_3$                 1S
8 $Y_3 \leftarrow B x_1^2 z_3 + (C + z_3) X_3$   1s + 1m + 1V + 1M
                        Total: 2s + 2m + 2S + 2V + 2M + 1P
```

4.2.3 Scalar Multiplication Techniques

The basic algorithm to compute a scalar multiplication, i.e. to evaluate mD, is the well known left-to-right double-and-add algorithm.

Advanced techniques consist in writing the scalar m in the form $m = \sum_{i=0}^{r} d_i 2^i$, where the digits d_i belong to a suitable digit set $\mathcal{D}$, and in the evaluation of mD as the sum $\sum_{i=0}^{r} d_i 2^i D$ via a Horner scheme. The recodings of the scalar (which differ in

Algorithm 4.2: Addition in Compressed López-Dahab Coordinates

```
    Input: $P_1 = (X_1 : Y_1 : z_1)$ , $P_1 = (X_2 : Y_2 : z_2) \in \mathcal{E}(\mathbb{F}_q^r)$
    Output: $P_3 = P_1 + P_2 = (X_3 : Y_3 : z_3)$
 1  $o_1 \leftarrow z_1^2$ ; $o_2 \leftarrow z_2^2$                          2s (-1s)
 2  $(X_1', Y_1') \leftarrow (X_1 z_2, Y_1 o_2)$                          2V (-2V)
 3  $(X_2', Y_2') \leftarrow (X_2 z_1, Y_2 o_1)$                               2V
 4  $\Delta \leftarrow X_1' + X_2'$
 5  $\overline{\Delta} \leftarrow \iota(\Delta)$      $\delta \leftarrow \mathfrak{c}(\Delta)$                               1P
 6  $\Lambda \leftarrow \overline{\Delta}(Y_1' + Y_2')$                               1M
 7  $a \leftarrow z_1 z_2 \delta$ ; $b \leftarrow a\delta$                          3m (-1m)
 8  $C \leftarrow \Lambda a$                                        1V
 9  $z_3 \leftarrow a^2$                                         1s
10  $X_3 \leftarrow \Lambda^2 + C + \Delta b + a_2 z_3$                          1S + 1V
11  $(X_2'', Y_2'') \leftarrow (X_2' b, Y_2' b^2)$                          1s + 2V
12  $Y_3 \leftarrow C(X_2'' + X_3) + Y_2'' + X_3 z_3$                      1V + 1M
                          Total:  4s + 3m + 1S + 9V + 2M + 1P
                                  3s + 2m + 1S + 7V + 2M + 1P
```

the digit set and in the algorithm used to generate the expansion) are the binary representation of the scalar, the Non-Adjacent Form (NAF) [Rei62], and signed windowing methods (w-NAF) [MOC97].

For TZV we also split the scalar using the method described in Theorem 3.2, and performed the multiplication mD by computing $\sum_{i=0}^{n-2} m_i s^i \bmod \ell$ via interleaved scalar multiplications (described by Möller in [Möl01], but in fact an older idea which has been rediscovered several times: see [Lim00], a two base case appears in [SL00], and indeed the principles can be traced back to the work of Pippenger [Pip76]; see also [Ber]). To recode the smaller scalars m_i we used the NAF and w-NAF representations, and in the case of curves with $r = 3$ also Solinas' Joint Sparse Form (JSF) [Sol01].

4.2.4 Comparison Between Ordinary Curves and TZV

In this section we compare performance between ordinary curve and TZV and we extend the results described in [AC08b].

In Tables 4.3, C.4, C.5 and C.6 we report timings of basic operations and scalar multiplication in different groups, all from ordinary curves: for $g = 1$ we have elliptic curves over $\mathbb{F}_{2^m}$ (EC) and TZV over $\mathbb{F}_{2^{m \times r}}$; for $g = 2$ we have genus 2 hyperelliptic curves over $\mathbb{F}_{2^m}$ (HEC) and TZV over $\mathbb{F}_{2^{m \times r}}$. Explicit curves are given in Appendix B. Operations considered are negation $(-)$, Frobenius endomorphism (σ), doubling (2) and addition $(+)$. Scalar multiplication is performed in affine or López-Dahab (original, extended and compressed) coordinates, without (bin, NAF/JSF) or with (w-NAF) precomputation. In case of w-NAF, the default windows size is $w = 4$ and the best size for each experiment, if different, is reported in brackets. The scalar splitting technique (cf. Theorem 3.2) is used for TZV, except for affine/bin. For each curve, the best timings are put in bold.

From the results, we can do the following remarks:

- TZV with $g = 1$, $r = 5$ are the fastest groups. On Intel (32-bit) at 80-bit security

		80-bit security					96-bit security				
		$g = 1$			$g = 2$		$g = 1$			$g = 2$	
	Curve:	EC	TZV	TZV	HEC	TZV	EC	TZV	TZV	HEC	TZV
	Field:	163	83×3	41×5	83	41×3	191	97×3	47×5	97	47×3
operation	$-$	0.01	0.01	0.02	0.01	0.02	0.01	0.02	0.02	0.01	0.02
	σ	–	0.01	0.02	–	0.02	–	0.02	0.02	–	0.02
	2	2.13	2.45	2.48	1.07	1.65	2.58	2.92	2.68	1.29	1.77
	$+$	2.13	2.45	2.48	2.34	4.72	2.58	2.92	2.68	2.73	5.03
affine	bin	660.9	747.1	721.2	426.5	672.6	920.7	1021.1	965.8	640.9	823.3
	NAF/JSF	601.2	337.7	**253.0**	**364.0**	**351.6**	858.9	512.9	**311.4**	**539.5**	**415.1**
	w-NAF (4)	542.5	318.8	**200.1**	**309.7**	**313.5**	771.4	474.2	**242.0**	**451.7**	**370.4**
Lopez-Dahab	NAF/JSF	287.4	351.9	302.6	–	–	410.0	529.0	364.0	–	–
	w-NAF (4)	251.0	(5) 309.4	209.7	–	–	357.2	454.8	248.3	–	–
	NAF/JSF	**282.4**	344.8	312.2	–	–	**403.1**	**483.9**	376.6	–	–
	w-NAF (4)	**244.9**	300.0	208.1	–	–	**354.4**	**409.7**	248.3	–	–
	NAF/JSF	–	**323.6**	275.4	–	–	–	487.1	336.6	–	–
	w-NAF (4)	–	**297.6**	209.7	–	–	–	435.5	252.5	–	–

Table 4.3: Comparison between ordinary curves and TZV – Intel (32-bit), timings in μs. Operations considered are negation ($-$), Frobenius endomorphism (σ), doubling (2) and addition ($+$). Scalar multiplication is performed in affine or Lopez-Dahab (original, extended and compressed) coordinates, without (bin, NAF/JSF) or with (w-NAF) precomputation. The scalar splitting technique is used for TZV, except for affine/bin.

they are about 10% faster (20% using precomputation) and at 96-bit security they are about 22% faster (30% using precomputation) than elliptic curves with extended López-Dahab coordinates. On PowerPC (32-bit) we observe even better speedups, while on 64-bit platforms performance are worse because the ground field size in less than a single machine word.

- If we limit to consider affine coordinates, TZV with $g = 1$ are always faster than elliptic curves (by factors about 1.5 for $r = 3$ and more than 2 for $r = 5$), while TZV with $g = 2$ are faster than genus 2 hyperelliptic curves only on Intel platform and with small gain. This last result on Intel is however an improvement with respect to [AC08b].

- The row affine/bin – that we recall is the sole without scalar splitting for TZV– shows that, because of the poor multiplication available in extensions, TZV can not compete with curves without exploiting the Frobenius endomorphism or, in other words, the good performance previously observed are due to the Frobenius and the scalar splitting technique.

- Affine coordinates are very bad for elliptic curves, but for TZV they are close to the best coordinates, or even the best ones e.g. in the case $g = 1$, $r = 5$. Choosing to work in affine coordinates simplifies the implementation and also requires less memory when performing the computation.

- The new c$\mathcal{LD}$ coordinates are on average $8 \div 10\%$ faster than $\mathcal{LD}$. In case of TZV with $g = 1$, $r = 3$, these are even better than affine coordinates – despite such TZV are not the fastest group. We want to also observe that compressed coordinates become so much more effective as worst the inversion is, thus they are attractive for devices with constrained resources

4.2.5 Comparison Between Even and Odd Characteristic

In this section we make a comparison among ordinary curves and TZV defined over binary fields and over fields of characteristic $p > 3$, as described in [AL04].

Besides affine coordinates, in characteristic p we considered the followings: Jacobian ($\mathcal{J}$), Chudnovsky Jacobian ($\mathcal{C}$) and Cohen's modified Jacobian (m$\mathcal{J}$) for elliptic curves [CMO98], weighted ($\mathcal{W}$) for genus 2 hyperelliptic curves [Lan05, referred as "new coordinates"] and compressed Jacobian (c$\mathcal{J}$) for TZV with $g = 1$ [HKA06, AL04]. The code in characteristic p is a courtesy of Avanzi and original results are reported in [Ava04, AL04]. Here we simply run the code on Intel (32-bit) platform for the sake of comparison.

In Table 4.4 we report timings. Scalar multiplication is performed in affine or the best available coordinates, without (bin, NAF/JSF) or with (w-NAF) precomputation. The scalar splitting technique is used for TZV, except for affine/bin. We skip (–) coordinates if they are not better than affine ones.

			80-bit security					96-bit security				
			$g=1$			$g=2$		$g=1$			$g=2$	
		Curve:	EC	TZV	TZV	HEC	TZV	EC	TZV	TZV	HEC	TZV
		Field:	160	80×3	40×5	80	40×3	192	96×3	48×5	96	48×3
affine	char 2	bin	660.9	747.1	721.2	426.5	672.6	920.7	1021.1	965.8	640.9	823.3
		NAF/JSF	601.2	337.7	253.0	364.0	351.6	828.9	512.9	311.4	539.5	415.1
		w-NAF (4)	542.5	318.8	200.1	309.7	313.5	771.4	474.2	242.0	451.7	370.4
	char p	bin	889.9	842.5	944.6	667.4	1961.9	1427.6	1173.0	1164.3	856.4	2379.3
		NAF/JSF	784.1	424.8	463.4	596.9	988.4	1277.7	589.9	568.2	765.1	1195.5
		w-NAF (4)	721.4	409.7	361.6	544.6	939.8	1153.2	561.1	435.5	693.1	1117.8
best	char 2		e$\mathcal{LD}$	c$\mathcal{LD}$				e$\mathcal{LD}$	e$\mathcal{LD}$			
		NAF/JSF	282.4	323.6	–	–	–	403.1	483.9	–	–	–
		w-NAF (4)	244.9	297.6	–	–	–	354.4	409.7	–	–	–
	char p		m$\mathcal{J}$	c$\mathcal{J}$		$\mathcal{W}$		m$\mathcal{J}$	c$\mathcal{J}$		$\mathcal{W}$	
		NAF/JSF	340.4	386.8	–	540.7	–	603.7	503.8	–	668.0	–
		w-NAF (4)	(3) 309.9	371.1	–	490.8	–	535.1	481.3	–	601.8	–

Table 4.4: Comparison on ordinary curves and TZV between characteristic 2 and $p > 3$ – Intel (32-bit), timings in μs. Scalar multiplication is performed in affine or the best available coordinates, without (bin, NAF/JSF) or with (w-NAF) precomputation. The scalar splitting technique is used for TZV, except for affine/bin.

From the data we can observe similar results for the curves, both over binary or prime fields. Surprisingly, the new code in characteristic 2 outperforms prime fields on Intel (32-bit), while our previous results presented in [AC08b] reported better performance on Intel-like platforms for odd characteristic (this is still the case with 64-bit code, whose results however are not included here).

4.2.6 Comparison Between Supersingular Elliptic Curves and TZV

In this section we apply scalar multiplication techniques to supersingular elliptic curves and TZV with $g = 1$, $r = 3$. These will be the subject of study in next sections where we will focus on pairing.

However actual protocols and applications based on pairing can not be reduced to the sole computation of the pairing itself, but the operations in the group of rational points remain of paramount importance. For instance, in the broadcast encryption scheme presented in Section 1.5.2 we have seen that the computation time both to encrypt and to decrypt messages is dominated by operations in the group.

The main difference between ordinary and supersingular elliptic curves is that for the latter doubling is almost for free, requiring only a few squares (no multiplications nor inversions) in the field. Thus scalar multiplication is, at least in principle, extremely faster. The penalty is that, because of pairing and the MOV attack, they require bigger definition fields. Furthermore, supersingular curves are rare, so their group size may be a bit bigger than the strict necessary, and clearly this slow down scalar multiplication because scalars grow. For instance, referring to Table 3.2 the supersingular TZV $\mathcal{E}_3/\mathbb{F}_{2^{103}}$ offers the same 80-bit security of an ordinary TZV with $g = 1$, $r = 3$ over $\mathbb{F}_{2^{83}}$, but it is defined over a bigger field and its group order is a 192-bit prime.

All these considerations motivate an in-depth analysis of scalar multiplication on supersingular curves. Besides affine coordinates, we considered using Jacobian and López-Dahab coordinates also for supersingular curves (deriving explicit formulae is an exercise, we omit details here), but these require a multiplication in doubling formulae.

We used the three sets of curves defined in Table 3.2. The first one includes an elliptic curve over $\mathbb{F}_{2^{239}}$ and a TZV over $\mathbb{F}_{2^{79}}$, offering 70-bit security[1]. In the second set we have an elliptic curve over $\mathbb{F}_{2^{307}}$ and a TZV over $\mathbb{F}_{2^{103}}$, providing 80-bit security. Finally, the third set includes an elliptic curve over $\mathbb{F}_{2^{457}}$ and a TZV over $\mathbb{F}_{2^{157}}$, offering 96-bit security.

Experimental results are given in Tables 4.5, C.7, C.8 and C.9. Operations considered are negation ($-$), Frobenius endomorphism (σ), doubling (2) and addition ($+$). Scalar multiplication is performed in affine or other (Jacobian and López-Dahab) coordinates, without (bin, NAF/JSF) or with (w-NAF) precomputation. In case of w-NAF, the default windows size is $w = 4$ and the best size for each experiment, if different, is reported in brackets. The scalar splitting technique (cf. Theorem 3.2) is used for TZV, except for affine/bin. For each curve, the best timings are put in bold.

		70-bit security		80-bit security		96-bit security	
	Curve:	EC	TZV	EC	TZV	EC	TZV
	Field:	239	79×3	307	103×3	457	157×3
operation	$-$	0.01	0.01	0.01	0.02	0.02	0.02
	σ	–	0.01	–	0.02	–	0.02
	2	0.13	0.15	0.21	0.20	0.81	0.40
	$+$	4.36	2.11	6.40	3.27	16.11	5.94
affine	bin	528.7	188.3	1030.5	378.3	4073.2	701.0
	NAF/JSF	**376.4**	**96.3**	**757.6**	**189.2**	**2911.7**	**380.1**
	w-NAF (4)	**237.2**	**83.5**	**456.0**	**160.1**	(5) **1702.5**	(3) **320.9**
other	NAF/JSF	500.4	194.7	843.8	385.6	4208.4	768.6
	w-NAF (4)	367.7	164.8	617.6	320.1	(5) 2887.8	(5) 625.0
	NAF/JSF	563.5	214.0	933.6	432.9	4391.4	853.0
	w-NAF (4)	393.8	175.5	649.9	(5) 345.6	(5) 2927.6	(5) 667.2

Table 4.5: Comparison between supersingular curves and TZV – Intel (32-bit), timings in μs. Operations considered are negation ($-$), Frobenius endomorphism (σ), doubling (2) and addition ($+$). Scalar multiplication is performed in affine or other (Jacobian and Lopez-Dahab) coordinates, without (bin, NAF/JSF) or with (w-NAF) precomputation. The scalar splitting technique is used for TZV, except for affine/bin.

[1] As already mentioned, this is really poor security, but we kept it for comparison with literature.

We make the following remarks:

- Comparing supersingular elliptic curve with the corresponding TZV, already looking at row affine/bin (which is without scalar splitting for TZV), we note a consistent difference: TZV are 2, 3 or even more time faster. The reason is twofold: first, we have already noted that definition fields are similar in size, therefore we no longer see a loss of performance for TZV due to the slow multiplication (as for ordinary curves). Second, by Table 3.2 we know that the order of the group for TZV is less than for elliptic curves, then smaller are scalars.

- Jacobian and López-Dahab coordinates are really slower than affine coordinates, since they both require a multiplication in doubling formulae.

- Comparing supersingular with ordinary TZV (cf. Section 4.2.4) we see that supersingular TZV outperform ordinary ones (about 25% without precomputation) at 80-bit security on Intel (32-bit), but this is no longer true at 96-bit security or on other platforms. As already discussed, this is the contributory cause of a bigger definition field and a bigger group size.

4.3 Pairing on Supersingular TZV in Characteristic 2

In this section we detail the new algorithm for the Tate pairing in the case of supersingular curves over binary fields (cf. Lemma 3.9). We first discuss the benefits of parallelization then sketch some tricks that can be applied to the serial algorithm. We also show an improvement over the point compression algorithm of Rubin and Silverberg [Sil05, RS09].

Beuchat et al. [BB+08] make an in-depth analysis of the η_T algorithm in even characteristic. Most of their results can be applied to the general Miller's algorithm. Here we specialize them to the case of Algorithm 3.2. For simplicity we only consider the direct approach without square root computation (but we will compare the new algorithm also with the fastest η_T in Section 4.3.5).

Let $\mathcal{E}_3/\mathbb{F}_{2^m}$ be the supersingular TZV where $\mathcal{E}\colon y^2 + y = x^3 + x + b$, $b \in \mathbb{F}_2$. As in [BG+07] and [BB+08] we represent $\mathbb{F}_{q^{12}}$ as $\mathbb{F}_{q^3}(s,t)$ with $s^2 = s + 1$, $t^2 = t + s$. In the implementation we also use a more conventional polynomial basis and we found that switching between the two representations allows some further optimizations (see Section 4.3.4 for more details).

Let $P = (x_P, y_P)$, $\tilde{Q} = (x_{\tilde{Q}}, y_{\tilde{Q}}) \in \mathbb{G}_1$. Let the distortion map $\psi : \mathbb{F}_{q^3} \to \mathbb{F}_{q^{12}}$ be given by $\tilde{Q} \mapsto Q = (x_Q, y_Q) = (x_{\tilde{Q}} + s^2, y_{\tilde{Q}} + s x_{\tilde{Q}} + t)$.

We discuss here the computation of $f_{q,P}(Q_i)$, which is the core of Algorithm 3.2: since we perform a loop on $q = 2^m$, the only group operations in the Miller loop are doublings. This is in clear contrast with the case of odd characteristic, where the loop on the prime p also requires point additions. In characteristic 3 one can perform a triple-and-add variant of the Miller loop, which, in analogy with characteristic 2, requires only point triplings.

Let $T = (x_T, y_T) \in \mathbb{G}_1$ be the point which is accumulated during the loop. Let λ be the slope of the line $l_{T,T}$ (for the curve of interest, $\lambda = x_T^2 + 1$). Point doubling is done as $[2]T = (\lambda^2, y_T^4 + x_{[2]T} + 1)$. At each iteration we also compute $l_{T,T}(Q_i) = \lambda(x_T + x_i) + y_T + y_i$, where $(x_i, y_i) = \psi(\tilde{Q}^{\sigma^i})$. As noted in [BB+08], replacing λ and using the curve equation we can rewrite $l_{T,T}(Q_i) = \lambda x_i + y_T^2 + y_i + b$ which costs essentially 1m (note that y_T^2 is already computed in the doubling of T). Due to the special choice of the distortion map, $l_{T,T}(Q_i) = a_i + b_i s + t$ for some $a_i, b_i \in \mathbb{F}_{q^3}$. The multiplication $f \leftarrow f^2 \cdot l_{T,T}(Q)$ is the core of the computation and can be performed with 6m as shown by Beuchat et al. [BB+08, Algorithm 12]. In Section 4.3.4 we give a slightly improved version of this algorithm.

4.3.1 Parallel Computation

We already noted that Algorithm 3.2 is very suitable for parallelization, by simply performing the r computations of $\left(f_{q,P}\left(Q^{\sigma_i}\right)\right)^{q^{i(r+1)}}$ independently (cf. line 3). Actually the accumulation of the r partial results and the final exponentiation need to be computed sequentially.

In the following we detail some features of the parallel algorithm and compare it with an alternative approach available for $\mathcal{E}_3/\mathbb{F}_{2^m}$.

We remark that $Q^\sigma = (x_Q^q, y_Q^q) = [q^5]Q = \psi(x_{\tilde{Q}}^q + 1, y_{\tilde{Q}}^q + x_{\tilde{Q}}^q + 1)$, hence the computation of Q^{σ_i} only requires 2 applications of the Frobenius automorphism and one addition in $\mathbb{F}_{q^r}$. Similarly the computation of the power to $q^{i(r+1)}$ is obtained by applying the Frobenius in $\mathbb{F}_{q^k}$, i.e. 4 Frobenius operations in $\mathbb{F}_{q^r}$ and a few additions.

Apart from a few operations for computing Q^{σ_i} and for raising the result to the $q^{i(r+1)}$ power, the parallel threads carry out exactly the same operations, namely evaluations of $f_{q,P}$ at some point. This algorithm is hence suitable not only for a parallel implementation, but could also be implemented using SIMD operations.

One can also exploit the special form (3.7) of the order N of $\mathcal{E}_3/\mathbb{F}_{2^m}$ to achieve a dual-core computer friendly algorithm which requires only $2 = r - 1$ processors. Since $t^2 = 2q$, we have $N = (q+1)(q+t) + 1$ and thus:

$$f_{N,P}(Q)^M = f_{N-1,P}(Q)^M = f_{q+1,P}(Q)^{(q+t)M} \cdot f_{q+t,[q+1]P}(Q)^M \ ,$$

where $M = (q^{12}-1)/N = (q^6-1)(q^2+1)\big(q^2+q+1+t(q+1)\big)$: the first equality holds because the distortion map admits denominator elimination (cf. proof of Theorem 3.14) and the second one comes from (3.10).

It is then possible to compute the evaluations of the two different Miller functions in parallel. This is summarized in Algorithm 4.3.

We note that $[q+1]P$ can be computed exploiting Koblitz curve techniques (see, e.g., [BG+07]), in fact $[q]P = \phi^k\left((\sigma^2(P)\right)$, where $\phi(P) = (x_P + 1, y_P + x_P)$. More, the final exponentiation can be computed with a single division even if t is negative. With respect to Algorithm 3.2, the drawback of this algorithm is that the two parallel computations carry out different tasks.

4.3.2 Sequential Computation

As an alternative to parallel evaluation, computing the r Miller loops in a sequential way allows to share some variables and reuse some intermediate computations. Notably only one copy of the points T and $\tilde{Q}$ is needed, and the point doubling computation can be shared. For supersingular curves in general this is not a great advantage, since the doubling formulae are quite simple. However, in very constrained environments the precomputation of the multiples of T might be competitive with respect to their

Algorithm 4.3: Computing $\mathrm{t}_{\mathcal{E}_3}$ on $\mathcal{E}_3/\mathbb{F}_{2^m}$

```
Input: $P \in \mathbb{G}_1$ , $Q \in \mathbb{G}_2$
Output: $\mathrm{t}_{\mathcal{E}_3}(P,Q)$ $\left( = \mathrm{t}_{\mathrm{TZV}}(P,Q) \right)$
1 $f_0 \leftarrow \left(f_{q+1,P}\left(Q\right)\right)^{q+t}$
2 $f_1 \leftarrow f_{q+t,[q+1]P}\left(Q\right)$
3 $f \leftarrow f_0 \cdot f_1$
4 $f \leftarrow f^{q^2+q+1+t(q+1)}$
5 return $f^{(q^6-1)(q^2+1)}$
```

repeated computation: in this case Algorithm 3.2 improves over previous algorithms, requiring to store only $\log_2 q$ points.

The details of the computation have already been discussed, and Algorithm 4.4 represents the sequential version of Algorithm 3.2 for $\mathcal{E}_3/\mathbb{F}_{2^m}$.

Algorithm 4.4: Computing $\mathrm{t}_{\mathrm{TZV}}$ on $\mathcal{E}_3/\mathbb{F}_{2^m}$

```
Input: $P, \tilde{Q} \in \mathcal{E}_3[\ell](\mathbb{F}_{2^m})$
Output: $\mathrm{t}_{\mathrm{TZV}}(P,Q)$
1  $T \leftarrow P$ , $f \leftarrow 1$
2  for $j = m-1$ downto $0$ do
3      $f \leftarrow f^2$
4      $\lambda \leftarrow x_T^2 + 1$ ; $\bar{y}_T \leftarrow y_T^2$
5      for $i = 0$ to $2$ do
6          $a_i \leftarrow \lambda\left(x_{\tilde{Q}}^{q^i} + 1\right) + \bar{y}_T + y_{\tilde{Q}}^{q^i} + b$
           $b_i \leftarrow \lambda + x_{\tilde{Q}}^{q^i}$
7          $g_i \leftarrow a_i^{q^i} + b_i^{q^i} s + t$
8          $f \leftarrow f \cdot g_i$
9      end
10     $x_T \leftarrow \lambda^2$ ; $y_T \leftarrow \bar{y}_T^2 + x_T + 1$
11 end
12 $f \leftarrow (f^2)^{q^5}$
13 return $f^{q^6}/f$
```

It is worth noting that the computation of a_i on line 6 requires the multiplication of a fixed λ times several different values $x_{\tilde{Q}}^{q^i} + 1$. Hence, unrolling the loop allows to use a sequential multiplication, which is usually faster than several multiplications, or in other words to store the precomputations associated to λ in the comb-based binary polynomial multiplication, and possibly use a larger comb.

4.3.3 Point Compression on TZV

In [RS02], Rubin and Silverberg describe an algorithm for point compression over TZV, which is detailed by Silverberg in [Sil05]. Naumann already considered point compression for TZV in odd characteristic [Nau99].

Classical point compression over elliptic curves compresses a point $P = (x_P, y_P)$ by dropping the y-coordinate and then recovers it by solving a quadratic equation. For supersingular curves in characteristic 2 (cf. Lemma 3.9), this requires to compute $C = x_P^3 + x_P + b$ and solve $y_P^2 + y_P = C$ for y_P (one additional bit of information may be necessary to determine the right solution). The total cost is $1\mathtt{S} + 1\mathtt{M} + 1\mathtt{H}$.

For a TZV, $P \in \mathbb{G}_1 \subset \mathcal{E}(\mathbb{F}_{q^r})$. So $x_P \in \mathbb{F}_{q^r}$ can be seen as a vector in $\mathbb{F}_q^r$, whose coordinates are not independent; it is possible to exploit such a dependency to compress x_P to a vector in $\mathbb{F}_q^{\varphi(r)}$. We briefly introduce the algorithm of Rubin and Silverberg (see [Sil05, RS09] for more details) and then analyze it in characteristic 2, leading to our improvement.

Since $P + \sigma(P) + \cdots + \sigma^{r-1}(P) = \mathcal{O}$, there exists a function $F(X, Y)$ with zeros of order 1 in $P, \sigma(P), \ldots, \sigma^{r-1}(P)$ and a pole of order r at $\mathcal{O}$. Let $\tilde{F}(X, Y) = -F(-P)$. The function $\prod_{i=0}^{r-1}\left(X - \sigma^i(s)\right)$ vanishes at $\pm P, \pm\sigma(P), \ldots, \pm\sigma^{r-1}(P)$, hence we have:

$$\gamma F(X, Y)\tilde{F}(X, Y) = \prod_{i=0}^{r-1}\left(X - \sigma^i(s)\right) , \qquad \gamma \in \mathbb{F}_q^* . \tag{4.4}$$

We can make (4.4) explicit for a TZV constructed from an elliptic curve given by an equation of the form (3.2) and $r = 3$, choosing a representation of $\mathbb{F}_{q^3}$ as $\mathbb{F}_q[T]/(T^3 + T + 1)$. The following system of three equations results:

$$\begin{aligned} \alpha_1^2 + a_1\alpha_1 + a_2 &= s_0 \\ a_1\alpha_0 + a_3\alpha_1 + a_4 &= s_0^2 + s_1^2 + s_1 s_2 + s_2^2 \\ \alpha_0^2 + a_3\alpha_0 + a_6 &= s_0^3 + \left(s_1^2 + s_1 s_2 + s_2^2\right) s_0 + s_1^3 + s_1 s_2^2 + s_2^3 . \end{aligned} \tag{4.5}$$

After some manipulations and taking into account that the curve is supersingular, hence $a_1 = a_2 = 0$, (4.5) becomes:

$$s_0^4 + s_0 a_3^2 + s_1^4 + s_1^2 s_2^2 + s_2^4 + a_4^2 = 0 ,$$

which is a bi-quadratic equation in s_1 (or s_2).

Now Rubin and Silverberg [Sil05] consider s_1 as an unknown and solve

$$s_1^2 + s_2 s_1 + K , \qquad \text{with} \qquad K = (s_0 + s_2)^2 + a_3\sqrt{s_0} + a_4 \qquad \text{for} \qquad K .$$

Unfortunately this equation is not in the form $y^2 + y + C = 0$, which would make its solution very fast [FH+03, Ava07], but instead requires extra work to compute the solutions. In fact the cost is $1\mathtt{s} + 1\mathtt{r} + 1\mathtt{m}$ to compute the constant term K, $1\mathtt{s} + 1\mathtt{m} + 1\mathtt{i}$ to transform to an equation of the form $y^2 + y + C$, $1\mathtt{h}$ to solve the equation and finally $1\mathtt{m}$ to recover the solution s_1.

We can improve the algorithm, in particular avoid the inversion, by taking instead s_0 as unknown (denoted x), under the assumption that $a_3 = 1$, satisfied for the curve defined in Lemma 3.9. This leads to the equation:

$$x^4 + x + C , \qquad C = \left(s_1^2 + s_1 s_2 + s_2^2 + a_4\right)^2 , \tag{4.6}$$

that can be reduced to a system of two quadratic equations: $x^2 + x = y$ and $y^2 + y + C = 0$, since $x^4 + x + C = \left(x^2 + x\right)^2 + (x^2 + x) + C$. Note that in $\mathbb{F}_q$, with $q = 2^m$ and m odd, equation (4.6) only admits two solutions.

The compression of the point $P = (s, t)$ is done by (i) dropping the coordinate s_0 and (ii) computing the extra bit needed to distinguish s_0 from the other solution of (4.6): if s_0 is some solution of $x^4 + x + C$, then $s_0 + 1$ is the other solution and as the extra bit we can use LSB in the binary representation of s_0. The compression algorithm is thus for free.

Decompressing a pair $(s_1, s_2) \in \mathbb{F}_q^2$ can be done by solving equation (4.6): first compute a solution $\bar{y}$ of $y^2 + y + C$; then solve $x^2 + x + \bar{y} + \mathrm{Tr}(\bar{y})$ whose solutions are the two candidates for s_0; finally select the solution having the LSB equal to the extra bit received. The total cost is $2\mathtt{s} + 1\mathtt{m}$ to compute the constant term C and $2\mathtt{h}$ to solve the quartic equation (reduced as two quadratic equations). We omit from the count the computation of $\mathrm{Tr}(\bar{y})$ since this is a negligible operation [FH+03, Ava07].

4.3.4 Extension Fields of Degree 4 for Pairing Computation

A natural representation for $\mathbb{F}_{2^{4m}}$, with $(m, 4) = 1$ is via the polynomial basis $\{\alpha^i\}_{i=0}^3$ with $\alpha^4 = \alpha + 1$. In [BG+07] the basis $\{1, s, t, st\}$ with $s^2 = s + 1$, $t^2 = t + s$ is suggested as an efficient alternative. Let $\mathbb{F}_{2^{4n}} \ni x = a_3\alpha^3 + a_2\alpha^2 + a_1\alpha + a_0 = x_0 + x_1 s + x_2 t + x_3 st$. Basis change is easily done:

$$\begin{aligned} x_0 &= a_0 + a_1 & a_0 &= x_0 + x_1 + x_3 \\ x_1 &= a_1 + a_3 & a_1 &= x_1 + x_3 \\ x_2 &= a_2 + a_1 & a_2 &= x_1 + x_2 + x_3 \\ x_3 &= a_3 & a_3 &= x_3 \end{aligned}$$

Since this change is linear, the two representations lead to multiplications in the extension field which require the same number of multiplications in the small field. But, the number of field additions may vary, resulting in slight overall improvements. Beuchat et al. [BB+08] show that the second basis is better for pairing computation. However their focus is on building a cryptographic pairing accelerator, and we think that the first basis can be more suitable for a software library which implements full arithmetic in an extension of degree 4.

In our implementation we decided to use the basis $\{1, s, t, st\}$ for the Miller loop computation but we prefer to return a result in basis $\{\alpha^i\}_{i=0}^3$. Since squarings require two additions less in the latter basis, in some algorithms, like η_T we perform the basis change before finishing the final exponentiation, saving around m additions.

We conclude with the following remark concerning [BB+08]. In the Miller loop, the core of the computation is the product of a generic element $f = f_0 + f_1 s + f_2 t + f_3 st$ for an element of the particular form $g_0 + g_1 s + t$. Beuchat et al. describe it in Algorithm 12. We point out that: (i) the 6 multiplications in lines 2 and 3 can be computed as 3 *sequential multiplications* [AT07], which leads to performance improvements since one operand is common in two multiplications; (ii) in line 5 the sum $f_2 + f_3$ can be substituted by the previously computed a_2; (iii) as the result is computed, it is not possible to directly accumulate it in f, while an additional copy is necessary. Algorithm 4.5 below fixes these problems.

4.3.5 Experimental Results

In order to put the performance of the new algorithm in a broader perspective, we implemented it alongside all the algorithms available in the literature that are applicable

Algorithm 4.5: Computing $(g_0 + g_1 s + t) \cdot (f_0 + f_1 s + f_2 t + f_3 st)$

```
Input: $g = g_0 + g_1 s + t \in \mathbb{F}_{2^{4m}}$ and $f = f_0 + f_1 s + f_2 t + f_3 st \in \mathbb{F}_{2^{4m}}$
Output: $w = f \cdot g$ ($w$ can be $f$)
1 $a_0 \leftarrow g_0 + g_1$; $a_1 \leftarrow f_0 + f_1$; $a_2 \leftarrow f_2 + f_3$;
2 $[m_0, m_2] \leftarrow g_0 \cdot [f_0, f_2]$; $[m_1, m_3] \leftarrow g_1 \cdot [f_1, f_3]$;
3 $[m_4, m_5] \leftarrow a_0 \cdot [a_1, a_2]$;
4 $w_2 \leftarrow f_0 + f_2 + m_2 + m_3$;
5 $w_0 \leftarrow f_3 + m_0 + m_1$;
6 $w_3 \leftarrow f_1 + f_3 + m_2 + m_5$;
7 $w_1 \leftarrow m_0 + m_4 + a_2$;
8 return $w_0 + w_1 s + w_2 t + w_3 st$
```

to supersingular TZV $\mathcal{E}_3/\mathbb{F}_{2^m}$, where $\mathcal{E}\colon y^2 + y = x^3 + x + b$, $b \in \mathbb{F}_2$. This curve has been chosen since it is the one used in [BG+07, RS09, BB+08].

We used the same curves presented in Section 4.2.6 and Table 3.2. The first two sets are the same adopted in [BG+07].

We now describe the considered algorithms (see also Section 3.5).

- t_N computes a straight Tate pairing with a Miller loop on an integer N which is a low Hamming weight multiple of ℓ (cf. [BK+02]). For the TZV we used $N = q^2 - q(1-t) + t^2 + t + 1$ with $t = \pm\sqrt{2q}$, whose NAF-representation is very sparse.

- The η and η_T pairings [BG+07]; here for the TZV the parameter $T = t_3 - 1 = \mp 2^{(3m+1)/2}$ has been chosen, with the notation of (3.4). For these two pairings we provide two implementations: the first uses essentially the same Miller loop as the other algorithms, while the second one, labeled (HLV), uses a point halving based iteration in the Miller loop, exploiting the techniques described in [BG+07] and [BB+08].

- $\mathsf{a}_{\mathrm{opt}}$ is the optimized (twisted) Ate pairing [Ver08]. For our varieties we adapt the example of supersingular elliptic curves over $\mathbb{F}_{3^m}$ from [Ver08, Section 4] using the *shortest vector* $V = [2^{(3m-1)/2}, 2^{(3m-1)/2} \mp 1]$ instead; we note that the loop is shorter than in η_T but, because of the ∓ 1 in the second component, an additional final multiplication is required.

- t_σ computes the Tate pairing exploiting the q^{th} power Frobenius σ adapting the approach from [Sco05] as explained in Remark 3.15.

- $\mathsf{t}_{\mathrm{TZV}}$ is the sequential version of our new algorithm (cf. Algorithm 4.4).

- $\mathsf{t}_{\mathrm{TZV}}$ (Par) is a parallel version of our algorithm, where the three evaluation of $f_{q,P}(Q^{\sigma_i})$ (see Algorithm 3.2, line 3) are executed as three threads, each one running on a different core.

- $\mathsf{t}_{\mathcal{E}_3}$ (Par) is the dual-core computer friendly algorithm available for $\mathcal{E}_3$ (cf. Algorithm 4.3).

All implementations except the ones labeled $\mathsf{t}_{\mathrm{TZV}}$ (Par) and $\mathsf{t}_{\mathcal{E}_3}$ (Par) run on a single core. For parallel algorithms the OpenMP library has been used.

Tables 4.6, C.10, C.11 and C.12 show our experimental results compared with their theoretical complexity given by the size of the iteration in the Miller loop (exponentiations are almost always negligible for these algorithms).

		70-bit security		80-bit security		96-bit security	
		EC	TZV	EC	TZV	EC	TZV
Pairing	Loop Length	239	79×3	307	103×3	457	157×3
t_N	$N = O(q^2)$	987.1	591.7	1576.4	1104.1	7250.2	3009.4
η	q^3	946.0	817.7	1513.3	1543.1	7094.7	4302.2
η_T	$2^{(3m+1)/2} - 1$	516.5	444.8	819.8	829.5	3785.2	2291.3
a_{opt}	$2^{(3m-1)/2}$	527.5	440.0	833.8	828.1	3788.1	2252.1
η (HLV)	q^3	1093.3	822.3	1696.4	1684.2	6690.2	4401.3
η_T (HLV)	$2^{(3m+1)/2} - 1$	506.3	428.2	791.9	798.9	3609.0	2195.1
t_{TZV}	$3 \times q$	–	751.3	–	1418.0	–	3985.8
t_σ	$2 \times s$	–	923.3	–	1749.3	–	4674.3
t_{TZV} (Par)	$3 \times q$	–	300.6	–	553.3	–	1494.7
$\mathsf{t}_{\mathcal{E}_3}$ (Par)	$2 \times O(q)$	–	339.0	–	609.7	–	1643.6

Table 4.6: Comparison among pairings – Intel (32-bit), timings in μs

We conclude with the following remarks:

- As expected, our new algorithm t_{TZV} in its sequential form has performance similar to η (3 loops on q against a single loop on q^3). As already noted, it has indeed better properties for parallelization and/or storage requirements.
- The parallel version of t_{TZV} is on average $25 \div 30\%$ faster than any previous pairing algorithm – notably η_T (HLV) – on the same curve.
- A comparison between t_{TZV} and $\mathsf{t}_{\mathcal{E}_3}$ confirms the expectation that the SIMD-like approach to the parallezation of the former produces a lower overhead (i.e. better performance, about 10% on average, slightly more on PowerPC) than the latter, but at the price of an additional computational unit.
- In general, with 32-bit code TZV performance are better than the corresponding elliptic curve (the only exception is $\mathcal{E}_3/\mathbb{F}_{2^{103}}$ on Intel, but with minimal loss). On the contrary, 64-bit code suffer from less flexibility in the implementation caused by the larger granularity.

Chapter 5

Real-world Applications

*In theory there is no difference
between theory and practice.
In practice there is.*

In this chapter we discuss real-world applications that often use pairing as a black-box, to build efficient cryptographic primitives like broadcast encryption or anonymous authentication schemes.

In the previous chapter we have seen practical examples of very efficient symmetric pairings that can be useful to deploy applications in constrained environment. For higher security requirements, asymmetric pairing over ordinary curves is probably a better choice (cf. Section 1.4), and nowadays the best available solution is R-ate pairing over Barreto-Naehrig curves [BN05]. Unfortunately it is still an open problem to find pairing-friendly ordinary TZV, for instance “Barreto-Naehrig TZV”. It is quite easy to find very small toy examples, but searching for curves of cryptographic size seems to be infeasible, at least with currently available techniques [FST06].

Independently from the choice of the best pairing, the mathematical solution provides security only in some critical phases of a real application, and it is increasingly common to observe that vulnerabilities come from poor implementations or from problems related to users'/servers platforms. In this chapter we present the Trusted Computing technology as a possible solution to mitigate some common security issues. We provide two concrete examples of applications: Trusted Broadcast Encryption [CRV08b] and an anonymous authentication system based on TLS and Direct Anonymous Attestation [CL+10].

5.1 Trusted Computing and OpenTC

Trusted Computing (TC) is a new emerging technology specified by the Trusted Computing Group [Tru09] aiming to enforce, by hardware and software, the specific behaviour of a computer system. The core of this technology is a tamper-resistant chip called Trusted Platform Module (TPM) that offers built-in cryptographic and core security functions.

The TPM is a low-cost chip whose main goal is to provide three hardware Root of Trusts: 1) for Storage (RTS) to securely store keys and data; 2) for Measurement (RTM) to collect and store the integrity measurements (used to represent the system configurations); 3) for Reporting (RTR) to allow remote verification of such integrity measurements.

Integrity measures are securely kept by the TPM within the Platform Configuration Registers (PCRs). PCRs are shielded locations able to accumulate the measurements of the system components. TPM guarantees that the values of PCRs can only be updated by adding a new measurement through the "PCR extend" operation:

$$\mathrm{PCR}_{\mathrm{new}} = \mathrm{SHA\text{-}1}(\mathrm{PCR}_{\mathrm{old}}||\mathrm{Measurement}) \enspace ,$$

where Measurement is typically the hash value of the component's binary or configuration file.

In the context of TC, virtualization is used to enforce strong isolation (i.e. memory isolation and communications subject to flow control policy) among components with different security requirements that run on a system. Two paradigms have emerged: full virtualization and para-virtualization. The former implies hardware emulation to run an unmodified guest Operating System (OS). The latter requires a modified guest OS, but exhibits better performance: indeed the virtual resources can be implemented in a more efficient way than barely emulate the hardware. Many virtualization engines, also called Virtual Machine Monitors (VMMs) or hypervisors, are available for PC class platforms: for example VMware, VirtualBox, QEmu offer full virtualization, Fiasco, a micro-kernel of L4 family, provides para-virtualization while Xen [BD+03] offers both.

Virtualization has been used as enabling technology to ensure strong isolation in different research projects like Terra [GP+03], EMSCB [SSP04] and OpenTC [Ope].

Particularly, OpenTC is a research project funded by the European Commission and aiming at the development of an open-source security framework built upon TCG's security concepts. OpenTC has a layered architecture that consists of: 1) a virtualization layer capable of running multiple Virtual Machines – also called domains or compartments – ensuring strong isolation among them; 2) a layer that implements trusted services; 3) an application layer. Services and applications run in different compartments. Currently OpenTC distributes a proof-of-concept implementation that supports two different virtualization engines: Xen and L4/Fiasco.

Coming back to the TPM, among the new functionality introduced and now currently available, sealing and remote attestation deserve more attention.

Sealing provides secure storage and allows to cryptographically bind confidential data to a specific system configuration in a way that data can only be accessed (unsealed) if the system runs on the same configuration the data was sealed for. One of the main issues of sealing is that once a piece of data is sealed against a system state, it is not possible to update that system – thus changing the values of PCRs – without loosing availability of the data. While undesirable from a design perspective, this is also true if the system after the update exhibits better security properties than the ones owned at sealing time, for instance when the update fixes some vulnerabilities. A solution to overcome this problem using a Sealing-Proxy has been presented in [CRV08a].

Remote attestation is a procedure to report integrity measurements to an external entity, called verifier. The measurements can be reported through a specific TPM command that generates a RSA digital signature over a subset of PCR values (and a nonce) by means of an Attestation Identity Key (AIK). The AIK is protected by TPM and can be certified by a special Certification Authority (Privacy CA), or through the Direct Anonymous Attestation (DAA) protocol, whose implementation is embedded in the TPM since version 1.2.

5.2 Trusted Broadcast Encryption

A Broadcast Encryption (BE) system allows a center to send encrypted messages over a public broadcast channel towards many users. The use of BE has been proposed for different scenarios: multimedia broadcasting, encrypted file systems, secure mailing lists and peer-to-peer applications.

The notion of BE was formally defined by Fiat and Naor [FN94], and the first practical BE scheme, called Subset Difference, was proposed by Naor et al. [NNL02]. This scheme is based on symmetric encryption and has been improved by many authors (e.g. [HS02, GST04, JY$^+$05]) to achieve smaller broadcast messages or keys on the receivers. The transition to public-key cryptography was initially done by Dodis and Fazio [DF03], then improved by Boneh et al. [BGW05].

BE protects the communication channel, but not the platforms where the contents are created, distributed or received; in this section we outline a system where the protection of the communication channel offered by the BE is extended to the platforms using Trusted Computing (TC) techniques. We call such coupling of TC and BE Trusted Broadcast Encryption (TBE).

5.2.1 Broadcast Encryption

In this section we review the main aspects of BE, defining the involved actors, describing a generic BE system and presenting some scenarios. To simplify the exposition, we limit our discussion to the asymmetric BE; we follow the notation presented in [BGW05].

Actors

In classical BE the actors are a set of N users and a center. With abuse of language we will often refer to actors meaning their platforms.

Users are the receivers of broadcast messages; each message is addressed only to a subset $\mathcal{S}$ of *legitimate* users. Users removed from $\mathcal{S}$ are called *revoked*.

Center is the entity responsible for 1) issuing users' credentials, 2) creating and 3) delivering messages to the users. This definition is not suitable for real scenarios; we hence map the three center's functionalities onto three distinct roles: *issuer*, *content provider* and *sender*.

Notation

A BE system is composed of three phases.

Setup. The issuer creates N private keys $SK_1, \dots, SK_N$ and one public key PK. Each user i is provided with his own private key[1] SK_i.

Encryption. The sender needs (from the content provider) the message M and the set of legitimate users $\mathcal{S}$ as input. It creates the broadcast message (Hdr, C_M) where Hdr is called header and C_M body.

C_M is the encryption of M under a symmetric session key K. Hdr contains a description of $\mathcal{S}$ and the encryption of K with the public key PK; the details of the header generation are the core of the papers on BE and are out of our scope.

[1] In symmetric BE the issuer must send the keys also to the sender.

Decryption. An user $i \in \mathcal{S}$ takes as input the broadcast message (Hdr, C_M). Using his private key SK_i and the public key, he retrieves K from the header (again this is the core of a BE algorithm). Finally, he decrypts M from C_M using K.

Examples

BE literature describes four main scenarios that we will use as examples in our discussion. In the following, we briefly give an overview of the scenarios.

Multimedia broadcasting. A video producer wants to broadcast its content to a set of customers. Here, the video producer is the center and the customers are the users of the BE system. The scenario can be extended by distinguishing the three roles of the center as three distinct services.

Encrypted file system. An user of an operating system (OS user for short) wants to securely store a file so that only a set of authorized OS users may access it. In this scenario, the OS user is the content provider (because he provides the file to be encrypted) and the others are the users of the BE system; the OS plays the role of issuer and "sender".

Secure mailing list. The subscribers of a mailing list want to securely exchange e-mails relying on a mail server. The writer acts as content provider, while the recipients are the users; the mail server is issuer and sender.

Peer-to-peer application. The peers of a peer-to-peer application (e.g. video conferencing or file sharing) want to communicate in a secure way and they use BE to protect their messages; we assume the peers belonging to a predefined and closed group of users. At each time, the peer who "speaks" is both content provider and sender, while the listeners are the users. Moreover the system requires an external entity acting as issuer.

5.2.2 Issues and Motivations

In this section we present the main issues arising in BE literature. We also introduce new problems that are not covered in literature, but arise when thinking of real implementations; such points represent the motivation for our work and will be investigated in the next section.

The goal of BE is to provide the confidentiality of a message broadcasted from a center to many users. The message is created on the center, transmitted over an insecure channel, then received and processed on the users' platforms. We aim to extend the protection that BE offers to the communication channel, by also including the platforms. For instance, we will guarantee the confidentiality of the message on the users' platforms by enforcing their correct behaviour (applying TC technology).

BE has been analysed in the literature from a theoretical point of view, identifying the following problems: efficiency of encryption and decryption algorithms, transmission costs (i.e. size of the header), and size of public and private keys.

By focusing the attention onto real applications, other problems arise that are usually not considered in literature. We cope with the followings.

Platforms for BE. Generally, applications are designed to perform the critical tasks (e.g. storing and using keys) using a smart card. Its usage has some disadvantages: distributing keys is done by delivering preconfigured smart cards, updating private keys is usually not feasible and smart cards have limited storage and computational power.

These considerations apply to users' platforms, while the center is considered as a complex and powerful entity, hence its platform is not taken into account.

Trustworthiness of the user. In BE the most critical information is the user's private key, therefore the only attack that arises is private key extraction: the session key is too short-lived to be critical and malicious handling of the decrypted content is not considered because the focus is only on the communication channel. Therefore the most critical point is the key storage on the user's platform.

An orthogonal problem to BE is traitor tracing: a set of malicious users extract their private keys and cooperate to build a pirate decoder; some BE algorithms (e.g. [NP00, DF$^+$05, BW06]) are designed to be able to identify and revoke traitors. Traitor tracing is a countermeasure to the key exposure attack.

Trustworthiness of the center infrastructure. The first problem from the users' perspective is the key escrow: the center, more precisely the role of the issuer, knows all the secrets used to generate users' keys[2]. Another problem arises when the users want to have evidence of the correct behaviour of the sender (e.g. in the peer-to-peer application scenario). Finally, the users have to trust the content provider: we do not consider this problem from the technical point of view and we assume such trustworthiness being provided in other ways (e.g. by contract).

We can also consider the relationships among the different roles of the center. We limit our discussion to the relationship between the content provider and the sender: the latter can modify or disclose the content[3] compromising the integrity or confidentiality of the message.

5.2.3 Applying TC Techniques to BE

In this section we show how to apply TC techniques to BE to face the problems discussed in the previous section. We call the coupling of TC and BE as Trusted Broadcast Encryption (TBE).

After presenting the relevant aspects of a suitable architecture, we define a simple TBE system where we pursue the sole objective of extending the confidentiality of the broadcast message up to the users' platforms, by enforcing their correct behaviour. As a next step, we outline an extended TBE system, where we also take into account the trustworthiness of the center infrastructure.

Architecture

We base our design on TCG's principles and OpenTC as a reference architecture: different execution environments (often called compartments) run isolated on top of a Trusted Computing Base (TCB) which offers basic security mechanism and services, including TC primitives. For this purpose, a TC-enabled hardware provided with a TPM is required; we note that current platforms are not resistant to physical attacks, which are out of the scope of TCG models. In addition, even if we are aware of the limits of the current implementations, we assume the availability of a set of security services, including: a trusted channel to establish secure channels among trusted parties (e.g. user and issuer); secure paths between TCB, compartments and I/O devices (e.g. to display contents preventing unauthorized disclosure); a run-time monitor capable of tracking

[2]This also applies in symmetric BE.

[3]Note that in symmetric BE, the sender also stores users' keys, so it can also expose them.

changes to the platform configuration. We will come back to these issues in Section 5.2.5.

The isolation of compartments is suitable for building secure and modular applications. For instance, the user's decoder application implementing the decryption phase can be split into three different compartments that perform respectively: 1) session key extraction, 2) content decryption and 3) content visualisation. The first compartment is critical because directly accesses the user's key, hence it should be placed within the TCB. The other two do not deal with user's key and they can be confined according to policy enforced by the TCB, allowing flexibility: an user could run his preferred viewer in a completely isolated domain that is only allowed to receive data from the content decryptor and has no other network communication channel.

An advantage of the proposed architecture is to rely only on general-purpose hardware as opposed as custom hardware using smart cards. The former overcomes the limitation in storage and computational power. Furthermore, moving keys from smart cards to TPMs allows on-line distribution, instead of physically delivering preconfigured smart cards to a large number of users. Finally, going in the direction of an open implementation naturally tends to avoid security through obscurity.

Trusted Broadcast Encryption

We begin by defining a basic version of a TBE system, where we focus on the confidentiality of the message within the users' platforms.

In this simple TBE system we limit our scope to the multimedia broadcasting scenario where a single infrastructure plays all the roles of the center. We introduce TC techniques in the setup and decryption phases, while encryption remains unchanged. In the following, we describe the modifications of setup and decryption with respect to the original phases (cf. Section 5.2.1).

Setup. Before sending the private key to a user, the issuer needs a guarantee about the integrity of the user's platform: this is usually done through remote attestation. In some scenarios the trustworthiness of the user's platform can be assumed, as in the encrypted file system where the user's platform is the same where the operating system (i.e. the issuer) runs. If the user's platform is considered untrusted, the issuer aborts the setup phase.

When the user receives his private key, he seals it against the current configuration; notice that the user can not prevent sealing, because the correct behaviour of his platform is guaranteed by the integrity verification.

Decryption. In order to decrypt the message, the user needs to unseal his private key. The unsealing, and hence the access to the content, is only possible if his platform is in the same state the issuer judged trusted.

As already noted, this basic version only extends the confidentiality of the message up to the users' platforms, by guaranteeing their correct behaviour. Actually, in the setup phase we used a scheme composed of remote attestation followed by sealing; while this has already been exploited in the literature (e.g. [KS07]), to the best of our knowledge it has never been properly designed nor implemented[4]; moreover, in

[4] Exploiting TPM v1.2 sealing: 1) the scheme can be implemented as a module within the TCB; 2) after remote attestation, sealing is done against the reported configuration C_r; 3) after unsealing, the module verifies that the configuration at sealing time C_s is equal to the current one (which is C_r because of successful unsealing), and aborts in case the check fails.

its current implementation sealing offers a protection that does not endure in time, unless we assume a run-time monitor able to detect changes in the configuration of the user's platform and consequently able to deny the access to unsealed data; we further investigate this aspect in Section 5.2.4.

We finally remark that in this simple system any consideration on the trust of the center is trivial: it can be either assumed trustworthy as in BE literature, or subject to integrity verification; the latter case is not realistic because the center as a whole is a complex infrastructure.

We now turn the attention to a complete TBE system: we cover all the scenarios and split the center into distinct roles to discuss its trustworthiness. To keep the treatment simple and short, we introduce the improvements step-by-step.

First, we improve the concept of configuration of the user's platform. In the setup phase, the issuer can request the user to seal data against a different configuration than the one reported; this allows the user's platform to perform the decryption phase in a distinct configuration (e.g. in the encrypted file system scenario, the setup phase may happen during the installation, while the decryption during the life time of the system).

Next, we consider the relationship user-issuer. Unless the issuer is assumed trustworthy, the user needs to verify its integrity in the setup phase, before accepting his key. In the encryption file system scenario, we consider the operating system a trustworthy issuer if the setup phase occurs during the installation, because a freshly installed operating system is, by definition, in a good state. On the other hand, in the secure mailing list scenario, the user may perform a remote attestation against the mail server before joining the list.

We now continue with the relationship user-sender. For scalability reasons, we assume no return-channel exists (i.e. it is not feasible for all the users to remotely attest the sender). Nonetheless, the sender can convince the users of its good state by embedding the attestation data in the broadcast message. This integrity report takes place in the encryption phase and the verification in the decryption. Note that a mechanism for freshness is needed to avoid reply attacks; we will provide further details in Section 5.2.4.

Finally, we consider the relationship sender-content provider. The content provider needs to verify the integrity of the sender in the encryption phase, before providing its content M. This can be done by exploiting remote attestation, e.g. in the secure mailing list scenario. Applying the scheme composed of remote attestation followed by sealing allows reducing the number of remote attestations, and notably the workload on the content provider for the verifications. The sealed data can be a symmetric key that the content provider will use to encrypt the message M; in this case the amount of sealed data on the sender grows with the number of content providers. This scheme is suitable, for instance, in the multimedia broadcasting scenario.

Taking into account all the previous considerations, we covered the issues introduced in Section 5.2.2.

5.2.4 New TC Techniques

In our TBE system, we applied sealing and remote attestation to protect critical data and verify integrity of platforms. However, in their current implementations, these techniques suffer from an important drawback: they assess trust only in the instant of time in which they are performed, and if the platform turns into a bad state it is not guaranteed that a proper countermeasure will be taken, e.g. access to unsealed data will be denied, or the change in the configuration will be reported to the peer of a trusted channel.

To keep the discussion simple when using TC primitives, in Section 5.2.3 we overpassed these issues assuming the availability of a run-time monitor. In addition, we supposed such monitor capable not only to track changes, but also to carry out countermeasures (cf. Section 5.2.3, remarks after Decryption phase).

In this section we outline two new TC schemes, forward sealing and continual shared attestation, which simplify the requirements of the run-time monitor, because they allow countermeasures to be embodied within the protocol layer.

Before starting, it is worth to define the meaning of "platform turns into a bad state". For simplicity, we only focus on user's platform and key exposure attack, which is tracked by the run-time monitor as a change in the configuration. Our architecture model considers the TCB trustworthy, therefore the configuration change may reflect only a change within the TBE compartments. Note that in Section 5.2.3 we placed the compartment handling the user's key within the TCB (i.e. the TCB of a TBE system is composed of a general purpose TCB plus a TBE-specific compartment); here we relax the assumption on this compartment assuming it can be subject of the attack.

Forward Sealing

For simplicity, we restrict the discussion to the simpler multimedia broadcasting scenario.

Our goal is to extend the security provided by the sealing over time, by reducing the time window within which the private key persists unsealed in memory.

We adopt the following notation: the total life time of the BE system is divided into T windows and the attack happens at the beginning of window $t \in \{1 \dots T\}$.

In the setup phase, the center generates T secrets and N sets of T private keys and sends (after integrity verification) the secrets and his set of private keys to every user. Each user i seals his keys: every private key $SK_i^{(j)}$, $j = 1 \dots T$ is sealed against a configuration $C = (\mathrm{Cnf}_i, S^{(j)})$, where Cnf_i is the current (proved good) configuration, and $S^{(j)}$ is the secret received from the center. After sealing, the user deletes all the secrets.

At the beginning of a window j, the center distributes the secret $S^{(j)}$, the user i unseals the new key $SK_i^{(j)}$ and deletes the previous ones (both the sealed key and the copy unsealed that persists in memory); note that the user can not unseal any future user key until the center broadcasts the corresponding secret.

We turn now the attention on the properties that this scheme grants. First it offers forward security, i.e. it is not possible to decrypt any content received before the attack. Next, if the attack modifies the configuration, the TPM will not unseal any future key. Finally, the scheme offers a partial protection even against attacks that do not modify the configuration, but requires a time τ (comparable with the window length) to succeed: at any new time window the attacker is forced to waste time to re-run the attack; note that this assumption is realistic when thinking to collaborative attacks, like traitors' attacks.

Continual Shared Attestation

The following scheme is suitable for the peer-to-peer application scenario.

Our goal is to allow peers to continually verify the configuration of the sender during the application life time.

We recall that for simplicity we consider a single closed group, where each user is provided with a broadcast channel towards the other users; furthermore we assume the configurations of users' platforms being not confidential, so that we can avoid any consideration on their privacy.

In the encryption phase the sender embeds its attestation data within the broadcast message, which is verified in the decryption phase by the receivers. Two important issues must be considered: guarantee freshness of the integrity measurement and ensure that the attested platform is exactly the sender's platform.

We sketch possible solutions and let the detailed design as a future work. The first problem is specific to this scheme: the integrity report requires a nonce for freshness that we propose to be the previous attestation data[5]; such mechanism needs a starting nonce that may be provided by the issuer (which is trusted by all users). On the contrary the latter problem is common to all trusted channels, and it has already addressed in literature [GPS06a].

We turn now the attention to the properties granted by the scheme. It allows to continually attest the sender (which changes during the application life time), so that if a peer becomes rogue, it is detected by all the other peers as soon as it sends a message, and therefore revoked. Furthermore we exploit the broadcast channel to share a single remote attestation within the group. This turns out to be an efficient and scalable technique to lower the workload of the TPM.

Note that both forward sealing and continual shared attestation rely on a run-time monitor that is only required to detect changes in the platform configuration, and not to take any countermeasure.

5.2.5 Open Issues

As stated in Section 5.2.3, TBE allows using general-purpose hardware based on a TPM, rather than custom based on smart cards, bringing advantages with respect to key distribution and openess; however this architecture has still open issues related to hardware attacks, mainly because the TCG model does not cover them. Among the physical attacks, we claim that the most dangerous in the TBE context are the so called "open-case" (cf. [KSP05, Kau07]) that allows breaking the security provided by TPM and are relatively easy to mount having physical access to the platform; on the contrary, more sophisticated attacks are affordable for a small number of users, and the TBE already provides countermeasures through revocation and traitor tracing.

Despite the current issues, it is possible to foresee an actual TBE system. We note in fact that hardware manufacturers are making efforts to mitigate these open-case attacks (e.g. [Gro06]). Another simple solution, where possible, would be providing users' platforms with a tamper-resistant case protecting from physical attacks; this could be particularly realistic in the multimedia broadcasting scenario where set-top-boxes are alraedy shipped to the users.

Other critical points of our architecture are the software components we put in evidence within the TCB, notably the trusted channel and the run-time monitor. Similarly

[5] In principle any public and "random enough" data is suitable as nonce; we claim that the previously exchanged attestation data that in the peer-to-peer application is easily to be provided by a different peer, has both these characteristics.

to the hardware, there is ongoing work that allows foreseeing an actual implementation, which is missing at the moment. The trusted channel is the most studied and mature, for instance [AG+08] provides an implementation proposal that extends the TLS protocol. Finally, for the run-time monitor, we have shown in Section 5.2.4 that its capabilities may be simplified if suitable TC schemes carry out some of its requirements.

5.3 Direct Anonymous Attestation

Anonymous credential systems and other schemes for anonymous authentication provide privacy-preserving solutions where authentication of users is required, but where it is not necessary to identify individual users. In these systems, copying and sharing credentials can be a serious issue. As this cannot be prevented in a software-only solution, these problems form a major obstacle for the use of fully anonymous authentication systems in practice.

In this section, we propose a solution for anonymous authentication that is based on a hardware security module to prevent sharing of credentials. Our protocols are based on the standard protocols Transport Layer Security (TLS) and Direct Anonymous Attestation (DAA). We present a detailed description of our approach based on a Trusted Platform Module (TPM) as hardware security module. Moreover, we discuss drawbacks and alternatives, and present an implementation for the TPM-based solution, as well as a pure software implementation.

Many credential systems (e.g. CL credentials [CL01]) have been proposed in the scientific literature, and recently, idemix – a very flexible anonymous credential system – has also been implemented [BB+09]. Comparing to idemix, we make use of a hardware security module and we also provide report of client's integrity. Beside, Amit et al. [ALP05] provide pseudonymous authentication in peer-to-peer networks by employing DAA with TLS and IPsec, but they only sketch how such results can be achieved. In contrast, we provide a detailed design and implementation.

Bichsel et al. [BC+09] present an implementation of CL credentials that uses a JavaCard as hardware module, providing portable credentials and multi-application support. Particularly noteworthy are also smartphones that offer hardware security solutions (e.g. Mobile Trusted Module (MTM), ARM TrustZone) which can be used to protect credentials. Nokia On-board Credentials are an example of a (not necessarily anonymous) system protecting credentials in this way. Our framework is flexible enough to accomodate different security modules.

Finally, we note that some vulnerabilities have been found in DAA which may lead to privacy violation (e.g. [SRC07]), and fixes have been proposed. However, since we focus on the design of a general framework that allows to use a generic DAA scheme together with TLS, any improvement of DAA that fixes these vulnerabilities can be included in our framework as well.

5.3.1 Anonymous Authentication

Scenarios.

Let us consider an online subscription service where users can access contents with their computer at home and with a mobile phone while traveling. Examples for such a service include news sites and services for real-time information about stock market prices.

Service provider and users have different objectives, which intuitively may seem to be in conflict: the service provider requires that only authorized users (i.e. subscribed users) access the service; the users want to be anonymous because details of content accesses are personal and sensitive information (e.g. which articles they read, which stocks they are interested in).

We aim to design a system that allows a content provider to collect payments from users for subscription and enables only legitimate users to access the contents while being anonymous. More, one legitimate user should not be able to share access to the service with illegitimate users.

We also want to support a variant of this business where it is possible to trace different accesses of the same users, while still protecting the actual identity of the user. This can be useful, for instance, for offering targetted advertising in change of a reduced price. To support this feature, our system must also support pseudonymity. To completely achieve real user anonymity (or pseudonymity), traceability must be prevented at all network layers, however this section only focuses on the transport layer.

Moreover, we want to handle the cases where the server wants to enforce the usage of a particular software on the client (e.g. a specific viewer for the content that, for instance, prevents non-authorized copies). For this, the clients must be capable of measuring and reporting their integrity state.

Requirements.

To address the above scenarios, our solution provides authentication based on anonymous credentials. More specifically, we have to satisfy the following security requirements:

R1. Only clients that obtained a valid credential must be able to authenticate (anonymously) to the server.

R2. Users (both legitimate and illegitimate) must not be able to forge authentication credentials.

R3. It must be possible to have unlinkable sessions (full anonymity).

R4. The solution must offer the possibility to link sessions (pseudonymity).

R5. Credentials must be unclonable (hardware TPM only).

R6. (optional requirement) Nodes must be able to provide evidence of the integrity of their (hardware and software) state (i.e. PCR signature).

To obtain a realistic solution that can be readily deployed in practice, the protocols should be based on well-established standards whenever possible implemented by means of widely used software libraries.

Overview of our TLS-Based Solution.

Our solution is based on the usage of the TLS protocol together with DAA, and the latter allows either for anonymous or pseudonymous authentication (see Section 5.3.2).

Figure 5.1 presents a high-level overview of our architecture which comprehends the following roles: a client platform, an issuer and a verifier. In our scenario, the user owns

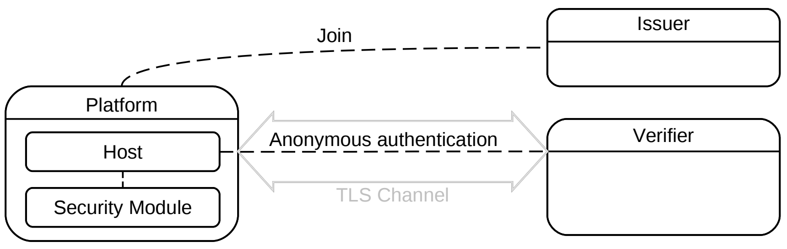


Figure 5.1: Architecture for anonymous authentication based on TLS and DAA

the client platform, and the service provider acts as issuer for credentials and as verifier during authentication.

The issuer is in charge of collecting the payments and supplying a valid credential to the user. This is referred to as Join protocol and done only when the client subscribes to the service. Further details are provided in Section 5.3.3.

The verifier accepts user connections and delivers contents to the user's platform. To support the DAA protocol, the platform consists of two components: a host and a security module. According to DAA design, the security module carries out the security critical operations, while the host computes the CPU-intensive operations.

We designed our framework to be flexible enough to support different variants of DAA and different security modules, both hardware and software. In the following we give a conceptual overview, whereas the detailed protocols are specified in Section 5.3.3.

Solution 1. Our principal solution is based on a TPM as a hardware security module, and fulfills requirements R1 to R5. The credentials can be used only within the TPM, and are always encrypted when stored outside. This solution conforms to the TPM specifications, and TLS is enhanced in accordance with the TLS specifications to use DAA for authentication.

In principle, this solution is suitable for the online journal scenario since it provides anonymity to the users and prevents sharing of credentials. More, if pseudonymity is used within the DAA protocol, the service provider can trace different accesses from the same user.

However this solution presents some drawbacks: TPM is not present in all potential clients, the DAA implementation cannot be changed and it is slow. Alternative hardware modules could be slow as well (e.g. smart cards) or expensive (e.g., crypto-accelerators). These limitations can be addressed with pure software modules instead of the TPM; this approach integrates well with existing web infrastructures, can be updated easily and could be deployed immediately. Nonetheless, without countermeasures, this solution only satisfies requirements R1 to R4, because copying of credentials cannot be prevented by software alone.

Solution 2. When combined with integrity measurements reports (which are provided by TCG-compliant platforms), Solution 1 can additionally ensure that only clients with correct hard- and software configurations may establish the TLS channel (similar to, for instance, TCG attestation). Here, we rely on a complete Trusted Computing infrastructure (hard- and software) including an integrity measurement and reporting architecture. With this solution, the TPM is used as both Root of Trust for Reporting and security module for DAA[6].

[6]A variant of this approach can be using pure software Solution 1 and protecting the execution environment of the security module by hardware means like virtualization, to prevent the cloning of the

This solution satisfies all the security requirements presented in Section 5.3.1 and therefore is more suitable for a scenario with strong security requirements; however, it requires a platform with a fully featured integrity measurement and reporting architecture (cf. Section 5.3.3) and databases of reference measurements which implies scalability and flexibility problems. Even if they are subject of research, these systems are not yet available for mainstream.

5.3.2 Background

To let understand our proposal, we present some internals of the TLS protocol with its enhancements and introduce the DAA protocol.

Transport Layer Security (TLS).

TLS [DR08] is a protocol that provides a secure channel (data authentication, integrity and confidentiality) between two parties: a Client initiating the communication and a Server listening for incoming connections. TLS is composed of five sub-protocols: the Handshake to establish the secure channel, the Change Cipher to set the ciphersuite, the Alert to exchange error messages and the Record to carry all previous protocols' data: it provides fragmentation, encryption, integrity and optionally compression.

We will only focus on the Handshake. It consists of four ordered sets of messages: (1) the Client sends messages to the Server, (2) the Server then replies, (3) again the Client sends messages and (4) finally the Server concludes.

As an example an overview of two sets is given. (1) only includes the message `ClientHello`: the Client initiates the Handshake and proposes some parameters to the server (e.g. known ciphersuites). (2) contains: `ServerHello` returns the parameters negotiated with Client (e.g. the ciphersuite to use); `Certificate` transports the Server's digital certificate for authentication; `ServerKeyExchange` provides additional material for key exchange depending on the ciphersuite (e.g. group parameters for Ephemeral Diffie-Hellman); (optional) `CertificateRequest` requires the Client authentication and `ServerHelloDone` tells the Client that the server completed its part.

To let exchange additional data during the Handshake, *Hello Extensions* [BWN+06] have been standardized. They are carried over Client and Server Hello messages. The Client can propose one or more extensions and the Server may accept them or not. Extensions are "backward compatible", because the specification requires the Server to ignore an Extension it does not know. Since hello extensions may deeply change Handshake's behavior and affect its security, the new ones must be defined via RFC to be strongly validated. Unfortunately they can carry just a limited amount of data and can be only exchanged at the beginning of the Handshake and in single two ways Client-Server interaction.

To overcome these limits the new Handshake message *Supplemental Data* [San06] has been standardized. On Server side, the new message belongs to set (2), sent soon after `ServerHello`, while on Client side it is the first message of set (3).

Supplemental Data are strongly constrained: they must be negotiated through an Extension, i.e. there can not be a Supplemental Data message without an Extension; moreover they can not interfere with the Handshake, so Supplemental Data must be processed after the Handshake finishes.

credentials: the TPM would be only used to remotely attest the protection mechanisms.

Direct Anonymous Attestation (DAA).

DAA is a very flexible anonymous credential system that has been designed specifically to encapsulate security-critical parts in a (low-cost, low-efficiency) secure hardware module. DAA possesses many different features, such as tagging rogue participants. In this presentation, we concentrate on the algorithms and protocols that are most relevant for our purposes and omit the other features.

A DAA system consists of different parties: a DAA issuer who issues DAA credentials, clients who can generate DAA signatures usually from a RSA credential, and verifiers who verify DAA signatures. A client is composed of a security module (TPM) and the host software. In our scenario, the server plays the role of both issuer and verifier. After the issuer set up a public/private key pair and generated parameters for the system, DAA provides two main protocols: (1) The DAA Join between issuer, host and TPM, used by the client to obtain a DAA credential from the issuer; and (2) the DAA Sign/Verify between host, TPM and verifier used by the client to prove possession of the DAA credential by computing a DAA signature. The latter protocol can be better specified:

- $(\texttt{basename}, n_v) \leftarrow \texttt{DAA\_Verifier\_Init}()$ is a two-party protocol between host and verifier, executed to obtain the verifier's nonce n_v (used for freshness) and basename `basename`. The basename can be either fixed by the verifier for pseudonimity, or the empty string for full anonymity.

- $\sigma_{\text{DAA}} \leftarrow \texttt{DAA\_Sign}(\mathit{Cred}_{DAA}, \texttt{basename}, m)$ is a two-party protocol between host and TPM, executed to obtain an anonymous (or pseudonymous) DAA signature σ_{DAA} from the DAA credential[7] Cred_{DAA} on a message m with respect to a basename `basename`.

- $\mathit{OK} \leftarrow \texttt{DAA\_Verify}(\sigma_{\text{DAA}}, \texttt{basename}, m)$ is an algorithm executed by the verifier to verify that a valid DAA signature σ_{DAA} on a message m has been generated with respect to a given basename `basename`, using a DAA credential issued by a given issuer.

Different DAA variants have been proposed. For the purpose of this papea,r two are particularly relevant and will be considered in Section 5.3.5: the original DAA scheme based on the strong RSA assumption [BCC04] – which has been standardized by the TCG and implemented in TPM v1.2 – and the newer scheme based on asymmetric bilinear maps [CMS08][8].

5.3.3 TLS-Based Anonymous Authentication Scheme

In this section, we describe the DAA enhancement to TLS based on TLS Hello Extensions and Supplemental Data. We present the two protocols Join and Authenticate in detail, using a TPM as security module. The behavior of issuer, host, and TPM for DAA is exactly as specified by the TCG, however we present it in a simplified way in this section.

[7]Note that in reality, Cred_{DAA} is distributed between host and TPM. For simplicity, we just show it as one variable here.

[8]We are aware that security flaws have been noticed in this scheme, and a preprint of a fixed version is available at `eprint.iacr.org/2009/198`. Currently our implementation is still based on [CMS08].

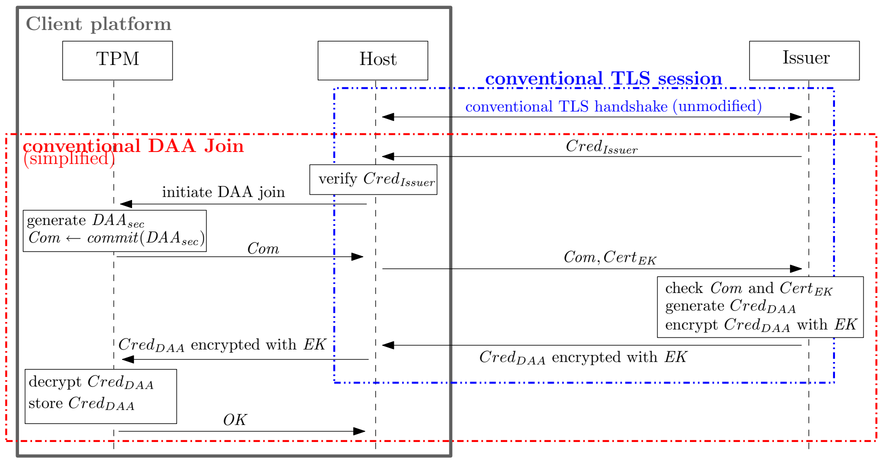


Figure 5.2: Join protocol with a TPM as security module: a conventional TLS session is used to protect the communication between client and issuer during the (unmodified) DAA join protocol. For clarity, a simplified abstract version of DAA join is shown.

Join.

Join is a protocol between a client (in our scenario a customer) and DAA issuer (in our scenario the service provider), where the client gets a DAA credential. This protocol is executed only once, then multiple anonymous TLS sessions with possibly distinct servers (e.g. different services provided by the same provider) can be based on this credential. To protect the communication during the DAA join protocol, (e.g. to keep the EK certificate confidential from eavesdroppers, and to ensure integrity of the DAA join messages), we use a conventional TLS session, without any modification. Note that in this phase anonymity is not required (in fact, for a subscription service, the user must be identified and payments could be collected).

Our protocol is shown in Figure 5.2. First, a conventional TLS session is initiated to protect all subsequent messages from outside adversaries (i.e. attackers that do not compromise server or client). Then the client retrieves the issuer credential $Cred_{Issuer}$ and verifies its validity.

Then, client and issuer execute the DAA join protocol (presented abstractly, in a simplified way): the client generates a DAA credential request Com based on the DAA secret of its TPM and sends it to the issuer, together with the TPM's EK certificate $Cert_{EK}$. The issuer checks the correctness of the request and of the EK certificate and releases the DAA credential, encrypted with the endorsement key EK from the certificate. The client receives the DAA credential, and the TPM decrypts it. Whenever the DAA credential is stored outside the TPM, it is encrypted by a key that cannot be accessed by the host software.

Remarks. During the join protocol, the issuer must verify that the client has a genuine TPM that will provide unclonability of credential; this is done by verifying the endorsement credential $Cert_{EK}$, and encrypting the DAA credential with EK, such that only the TPM can decrypt it (cf. Section 5.3.2). Since EK is unique to a specific TPM, it is privacy-sensitive data which must not be disclosed to outsiders. In our protocol, the $Cert_{EK}$ (which includes EK) is protected by TLS, as are all messages of the DAA join protocol.

Client Anonymous Authentication.

For our solution, we combine DAA with the TLS protocol by defining appropriate TLS Hello Extensions and Supplemental Data for client authentication. The TLS server plays the role of the DAA verifier to anonymously authenticate the client.

At a high level, client and server negotiate the usage of the anonymous authentication via TLS Hello Extentions, then the client authenticates itself by computing and sending –through Supplemental Data– a DAA signature, which is verified by the server. The anonymity (or pseudonymity) properties of a DAA signature lead to anonymity (or pseudonymity) for the client in our protocol. The fact that the (secret part of the) DAA credential is protected by the TPM, ensures that users cannot clone the credential.

In a standard TLS session, the client authenticates the messages exchanged based on digital signatures (usually RSA signatures). To be compliant with the TLS specification, we keep the client authentication using a conventional key pair, but to anonymize the process, we generate a fresh key pair for each session. We use a DAA signature to authenticate the X.509 certificate of the key pair used. This guarantees that it is not possible to identify the client, but only to tell that it owns a valid DAA credential; using a different key pair for each session prevents tracking of the user.

Our anonymous authentication protocol is shown in Figure 5.3. The client starts

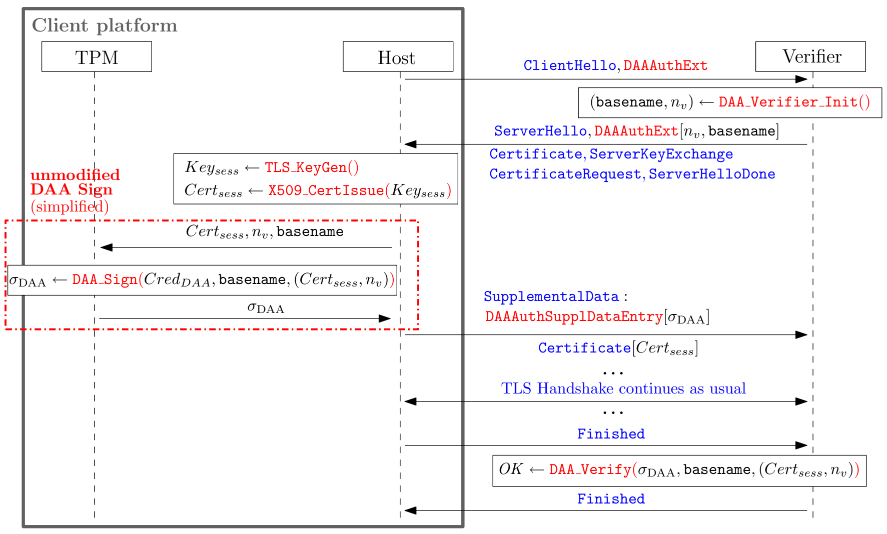

Client platform
TPM
Host
Verifier
ClientHello, DAAAuthExt
(basename, n_v) ← DAA_Verifier_Init()
ServerHello, DAAAuthExt[n_v, basename]
Certificate, ServerKeyExchange
CertificateRequest, ServerHelloDone
Key_{sess} ← TLS_KeyGen()
$Cert_{sess}$ ← X509_CertIssue(Key_{sess})
unmodified
DAA Sign
(simplified)
$Cert_{sess}$, n_v, basename
σ_{DAA} ← DAA_Sign($Cred_{DAA}$, basename, ($Cert_{sess}$, n_v))
σ_{DAA}
SupplementalData :
DAAAuthSupplDataEntry[σ_{DAA}]
Certificate[$Cert_{sess}$]
TLS Handshake continues as usual
Finished
OK ← DAA_Verify(σ_{DAA}, basename, ($Cert_{sess}$, n_v))
Finished

Figure 5.3: The anonymous authentication protocol based on TLS and DAA. TLS enhancements for DAA authentication are highlighted. For clarity, the (conventional, unmodified) DAA sign protocol is shown in a simplified abstract way.

the TLS handshake by sending a `ClientHello` message containing a TLS Hello Extension `DAAAuthExt` which informs the server to use DAA for anonymous authentication. The server uses the function `DAA_Verifier_Init`() to generate the nonce and the basename (for pseudonymity) for DAA, which are sent to the client within the extension `DAAAuthExt` in the `ServerHello` message. For full anonymity, the basename is left empty. Technically, `DAAAuthExt` is a TLS Hello Extension that contains one byte reserved for the version and optional parameters, including the version of the DAA enhancement, the nonce, the basename or an error message. Moreover, the server requests the TLS client authentication by sending a `CertificateRequest` message.

Then the client prepares for the anonymous authentication by generating a new (RSA) key pair for TLS authentication, and issues a self-signed certificate $Cert_{sess}$ for Key_{sess} (i.e. it signs the public key with the corresponding private key). Further, the host runs the `DAA_Sign` protocol[9] with the TPM to obtain a signature σ_{DAA} on $Cert_{sess}$, and sends σ_{DAA} to the server in a `DAAAuthSupplDataEntry` carried by the client `SupplementalData` message. The client sends $Cert_{sess}$ in a `ClientCertificate` message (as in standard TLS).

After that, the TLS handshake continues as usual. In a TLS session, and hence in our protocol, the client authenticates by computing a signature over all the messages exchanged by the client and the server, in our case using Key_{sess}.

After the `Finished` messages have been exchanged[10], the server verifies the DAA signature to validate the anonymous authentication using `DAA_Verify`.

Remarks. Note that it is possible to precompute and store several keys Key_{sess} with their certificates $Cert_{sess}$ for use in later sessions. If pseudonymity is in use, it is possible to optimize the process by generating only one single Key_{srvr} and $Cert_{srvr}$ for each server instead of for each session.

This protocol allows *client* anonymous authentication only. However, other scenarios like peer to peer applications may require mutual anonymous authentication. Extending this protocol to support server side anonymous authentication can be achieved similarly to the client authentication, because of the symmetric behaviour of TLS Hello extensions and supplemental data.

Software-Only Anonymous Authentication.

The same two protocols described above can also be implemented in software alone. For this, the host software additionally performs the operations of the TPM. In Section 5.3.5, we describe our pure software implementation based on a very efficient DAA version that is not available in hardware TPM. However, without additional measures, a software solution cannot prevent cloning of credentials, and thus fails to fulfill requirement R5.

Anonymous Trusted Channels.

A trusted channel is an extension of a secure channel like TLS (providing confidentiality and integrity of data), where an endpoint can also get assurance about the integrity of the other endpoint. To address requirement R6, we propose to enhance our TPM-based anonymous authentication with attestation (cf. Section 5.3.2), such that the server can verify the integrity of the client. We call the result an anonymous trusted channel.

The TCG specifications allow the client to measure the integrity state of the software that is loaded during system boot and securely store these measurements in the PCRs of the TPM, thus creating a chain of trust. Afterwards, the client state can be attested

[9] `DAA_Sign` is an interactive protocol, which may take more than one round, and where both parties input their portion of the DAA credential $Cred_{DAA}$ (the details depend on the actual DAA scheme). However, for the sake of clarity, we represent `DAA_Sign` as a simple signature generated by the TPM, because the essential property (besides anonymity) is that the host cannot create such a signature without the TPM.

[10] The verification of σ_{DAA} is delayed until here to comply with RFC 4680: To prevent a modification of the normal protocol flow, RFC 4680 mandates that the supplemental data are ignored until the TLS handshake finishes, and any action involving the data carried by `SupplementalData` must be performed after the handshake is completed.

by the TPM function `TPM_Quote`: the TPM signs selected PCRs and a nonce with an AIK.

Our anonymous trusted channel is based on the join protocol described above, while the authentication protocol is modified as follows: In addition to the TLS key Key_{sess} and self-signed certificate $Cert_{sess}$, the client creates an AIK –within the TPM–, and signs it with DAA (instead of signing $Cert_{sess}$) to get a DAA signature σ'_{DAA}. After that, the host computes a hash $n_H \leftarrow H(n_v|Cert_{sess})$, which is then used as the nonce for `TPM_Quote`. With this function, the TPM generates an RSA signature σ_{Quote} on n_H and a set of PCRs selected by the host, using the secret key of AIK. The public key of AIK, σ'_{DAA}, and σ_{Quote} are sent to the verifier using the `SupplementalData` message. At the end of the handshake, the verifier checks σ'_{DAA} and σ_{Quote}, and verifies that σ_{Quote} is computed on a set of PCRs indicating an acceptable state of the client.

With this protocol, the server can verify the client's integrity state, assuming that: (1) the attestation includes all relevant software (e.g. bootloader, OS kernel, protocol implementation), and (2) the software included in the attestation provides a secure execution environment. For instance, it must not be possible for another application to interfere with the protocol implementation, or replace/corrupt software at runtime without changing the PCRs.

Developing a realistical software architecture suitable for attestation is an orthogonal line of research and beyond the scope of this chapter. Recent efforts in this area include the OpenTC project

Note that an AIK can only be used to sign PCR values, and cannot be used as a TLS key. Furthermore, our protocol is compliant with TCG specifications and requires the computation of only a single DAA signature.

5.3.4 Security Analysis

The security of our solutions is directly based on the security of DAA and TLS. For both protocols, formal security proofs exist (see, e.g. [BCC04, GM+08]). In this section, we give an informal analysis of our protocols with respect to the requirements listed in Section 5.3.1, based on the assumption that DAA and TLS are secure. A formal proof of security is left for future work.

Anonymous Authentication.

Our protocols fulfill requirements R1 and R2, because authentication is successful only when the DAA signature can be verified correctly. The DAA signature is used to authenticate the certificate used for TLS, hence it is bound to the TLS channel. Thus, the unforgeability of DAA signatures implies that only clients with a valid DAA credential can authenticate successfully to the server (R1). Breaking requirement R2 implies forging a DAA credential, which would also break the security of the underlying DAA scheme.

Unlinkability (requirement R3) follows from the unlinkability of DAA signatures and from the fact that freshly generated keys and certificates are used for distinct TLS sessions and no other data that allows linking is transmitted.

The possibility of DAA to provide pseudonymity instead of full anonymity means that, in that case, DAA signatures can be linked to a pseudonym. This implies that our protocols offer the possibility to provide pseudonymous authentication (requirement R4).

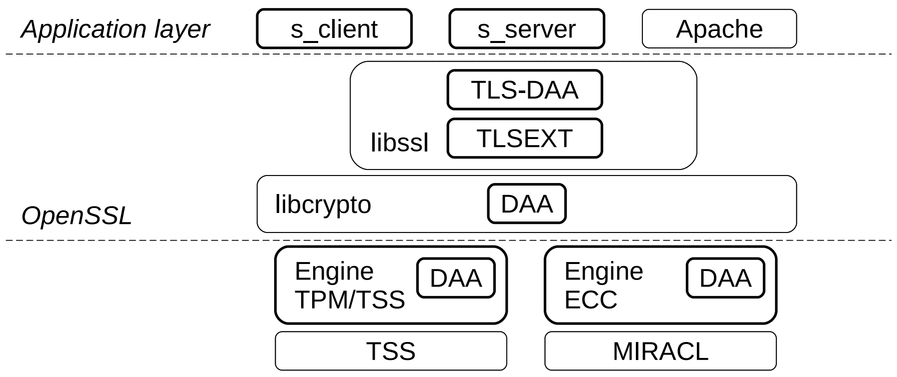


Figure 5.4: Implementation design for anonymous authentication based on TLS and DAA

Unclonability of credentials (requirement R5) is achieved by our solution based on a hardware security module. When using a TPM, the DAA credential is protected by the TPM (i.e. it is always encrypted when stored outside the chip), and unless the TPM can be attacked successfully (e.g. by hardware attacks), the credential is never disclosed to the user and thus cannot be copied. In a pure software based solution, copying of credentials cannot be prevented, of course. Therefore, requirement R5 can be achieved only by solutions that use a hardware module.

Anonymous Trusted Channel.

Requirements R1 to R5 are fulfilled similarly as above, except that the DAA signature does not sign the TLS certificate directly, but signs a freshly generated AIK, which then signs the TLS certificate via `TPM_Quote`. This means that, although the authentication procedure works differently now, the authentication is still based on DAA, and the arguments above can be carried out analogously for the anonymous trusted channel.

In addition, requirement R6 can be fulfilled under the following assumption: the platform must attest that it loaded a TCB capable to protect sensitive code and data; this is done by providing a chain of trust from the RTM to the TCB and signing it via the `TPM_Quote`. For instance, the TCB must guarantees that an attacker cannot inject false certificates from another platform as parameter to the `TPM_Quote` – because this would allow relay attacks. The chain of trust, the security of the TPM, DAA, and RSA signatures (`TPM_Quote`) imply that a server can securely verify the client integrity.

5.3.5 Implementation and Experimental Results

The implementation is based on OpenSSL: `libssl` and `libcrypto` have been extended to support respectively DAA and the DAA enhancement to TLS. OpenSSL also supports engines as a way to provide alternative implementations for a cryptographic primitive, usually to exploit cryptographic hardware accelerators. We developed two engines to provide two different implementations of the DAA protocol. Figure 5.4 presents the overall architecture.

DAA: is an interface to implement the Sign and Verify algorithms of DAA within the OpenSSL `libcrypto`. The interface is designed to provide DAA support in any context `libcrypto` is available, not only for the DAA enhancement to TLS.

A default implementation is included and the use of engines allows to override such default implementation. Currently this interface exposes three functions: `DAA_Verifier_Init`, `DAA_Sign` and `DAA_Verify`, that are used by the DAA enhancement to TLS as detailed in Section 5.3.3.

TLSEXT: is a framework to support TLS Hello Extensions and Supplemental Data into OpenSSL `libssl`, both needed for the DAA enhancement to TLS.

DAA-TLS: is the core of the implementation to enhance OpenSSL `libssl`. It is build upon **TLSEXT** and uses primitives offered by **DAA**. It provides an interface for applications that want to use anonymous authentication. Furthermore, we provide a special compile-time option to build it in *legacy mode*, that allows also unmodified applications to exploit the DAA enhancement; legacy applications may change settings via environment variables or a configuration file.

Engine TPM/TSS: is an OpenSSL engine that implements the **DAA** interface according to the TSS Specification [Tru], namely: `Tspi_DAA_Verifier_Init`, `Tspi_TPM_DAA_Sign`, `Tspi_DAA_VerifySignature`. It requires a TSS capable of supporting the DAA protocol and we used TrouSerS[11], with a patch provided by Finney [Fin09]. The engine is also in charge of the conversion between OpenSSL data structures and TSS' ones.

Engine ECC: is an OpenSSL engine that implements a purely software version of the ECC-based DAA [CMS08]. According to the authors' suggestion, our implementation uses asymmetric pairing over Barreto-Naehrig [BN05] elliptic curves at 256-bit security. The engine relies on MIRACL[12] as a software cryptographic accelerator to compute pairing.

The implementation has been done with OpenSSL v0.9.9 TrouSerS v0.3.0 and MIRACL v5.4, and it currently supports only client anonymous authentication.

Figure 5.4 also shows the application layer, that is the collection of software used to demonstrate our solution: we provide modified versions of the OpenSSL tools `s_client` and `s_server` (we stress that the modifications required are very small, only a few lines of code), and we demonstrate the use with legacy applications running an unmodified Apache web server (version 2.2.13).

We performed experiments on two HP Compaq DC7700 with Intel Core2 Duo 2.13GHz CPU and 2GB of RAM, using OpenSSL `s_server` on the server and OpenSSL `s_time` on the client to measure the number of connections per second. We remark that measures are taken from a client perspective and we are not benchmarking the server, which is left as future work.

In Figure 5.5 we present two sets of data to support the feasibility of our solution: the number of connections per second (table on the left) and the total number of bytes transmitted during a handshake (chart on the right). We compare TLS without and with client authentication and the DAA enhancement to TLS (TLS-DAA), using the Engine ECC and the Engine TPM/TSS.

For the number of connections we distinguish between initiating new connections (column new/s) and resuming previous TLS sessions (res/s). The DAA enhancement

[11]The open-source TCG Software Stack (`http://trousers.sourceforge.net/`)

[12]Multiprecision Integer and Rational Arithmetic C/C++ Library (`www.shamus.ie`)

Protocol	**Connections**	
	new/s	res./s
TLS (no client auth.)	91.62	690.8
TLS (with client auth.)	80.23	495.3
TLS-DAA (ECC)	3.22	493.1
TLS-DAA (TPM/TSS)	0.03	456.3

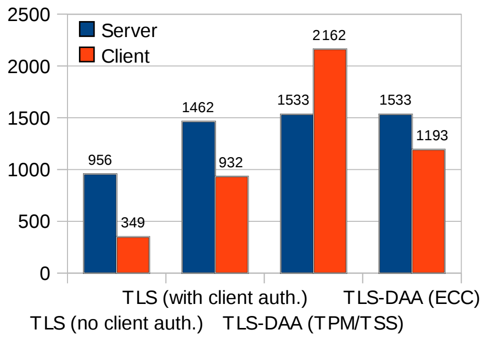


Figure 5.5: (Left) Number of connections per second for TLS and TLS enhanced with DAA: new connections and session resumed. (Right) Total number of bytes transmitted during a handshake: client and server side.

introduces a considerable latency in the TLS handshake: the TPM is really slow as the computation of a DAA signature takes about 37s; however the pure software implementation provides reasonable timings (around 30ms for a connection). Furthermore, the use of session resumption guarantees almost no loose in performance for all the following accesses.

We conclude with an analysis of the data transmitted during the handshake. As we are performing only client anonymous authentication, the transmission on the server is almost unchanged, compared to TLS with client authentication. On client side, the TPM/TSS version implements the RSA-based DAA whose signature is 1225 bytes long, and this has the effect to double the total number of bytes transmitted. We have to notice that we performed experiments in a pessimistic case, where the client's certificate is only 512 bytes long, and no certificate chain is transmitted. Nevertheless, the ECC version provides a very efficient solution, as the DAA signature in this case is only 256 bytes long.

Bibliography

[AC$^+$05] R. M. Avanzi, H. Cohen, C. Doche, G. Frey, T. Lange, K. Nguyen, and F. Vercauteren. *Handbook of Elliptic and Hyperelliptic Curve Cryptography*. CRC Press, 2005.

[AC08a] R. M. Avanzi and E. Cesena. Pairing on supersingular trace zero varieties. Cryptology ePrint Archive, Report 2008/404, 2008. http://eprint.iacr.org/2008/404.

[AC08b] R. M. Avanzi and E. Cesena. Trace Zero Varieties over Fields of Characteristic 2 for Cryptographic Applications. In J. Hirschfeld, J. Chaumine, and R. Rolland, editors, *Algebraic geometry and its applications*, volume 5 of *Number Theory and Its Applications*, pages 188–215. World Scientific, 2008. Proceedings of the first SAGA conference, 2007, Papeete.

[AG$^+$08] F. Armknecht, Y. Gasmi, A.-R. Sadeghi, P. Stewin, M. Unger, G. Ramunno, and D. Vernizzi. An efficient implementation of trusted channels based on OpenSSL. In *Proceedings of the 2008 ACM workshop on Scalable trusted computing*, pages 41–50, New York – USA, 2008. The Association for Computing Machinery.

[AKS09] M. Ak, K. Kaya, and A. A. Selçuk. Optimal subset-difference broadcast encryption with free riders. *Inf. Sci.*, 179(20):3673–3684, 2009.

[AL04] R. M. Avanzi and T. Lange. Cryptographic Applications of Trace Zero Varieties. Unpublished manuscript., 2004.

[ALP05] S. B. Amit, A. D. Lakhani, and K. G. Paterson. *Securing Peer-to-Peer Networks Using Trusted Computing*, chapter 10, pages 271–298. IEEE Press, 2005.

[ANS05] ANSI X9.62. Public key cryptography for the financial services industry, the elliptic curve digital signature algorithm (ECDSA). Technical report, American National Standard Institute, 2005.

[AT07] R. M. Avanzi and N. Thériault. Effects of Optimizations for Software Implementations of Small Binary Field Arithmetic. In *WAIFI 2007*, volume 4547, pages 69–84. Springer, 2007.

[ATW08] R. M. Avanzi, N. Thériault, and Z. Wang. Rethinking low genus hyperelliptic jacobian arithmetic over binary fields: Interplay of field arithmetic and explicit formulae. *Journal of Mathematical Cryptology*, 2:227–255, 2008.

[Ava04] R. M. Avanzi. Aspects of hyperelliptic curves over large prime fields in software implementations. In M. Joye and J.-J. Quisquater, editors, *Cryptographic Hardware and Embedded Systems – CHES 2004*, volume 3156 of *LNCS*, pages 148–162. Springer, 2004.

[Ava07] R. M. Avanzi. Another look at square roots (and other less common operations) in fields of even characteristic. In *Selected Areas in Cryptography*, volume 4876 of *LNCS*, pages 138–154, 2007.

[Bar] P. S. L. M. Barreto. The pairing-based crypto lounge. Available at http://www.larc.usp.br/~pbarreto/pblounge.html.

[BB$^+$08] J.-L. Beuchat, N. Brisebarre, J. Detrey, E. Okamoto, and F. Rodríguez-Henríquez. A comparison between hardware accelerators for the modified tate pairing over $\mathbb{F}_{2^m}$ and $\mathbb{F}_{3^m}$. Cryptology ePrint Archive, Report 2008/115, extended version of the Pairings 2008 paper of the same name (Volume 5209 of Lecture Notes in Computer Science, Springer, 2008, 297–315), 2008.

[BB$^+$09] P. Bichsel, C. Binding, J. Camenisch, T. Groß, T. Heydt-Benjamin, D. Sommer, and G. Zaverucha. Cryptographic protocols of the identity mixer library. Technical Report RZ 3730 (#99740), IBM Research, 2009.

[BC$^+$09] P. Bichsel, J. Camenisch, T. Groß, and V. Shoup. Anonymous credentils on a standard java card. In *Proceedings of the 16th ACM Conference on Computer and Communications Security (CCS'09)*. ACM Press, 2009.

[BCC04] E. Brickell, J. Camenisch, and L. Chen. Direct anonymous attestation. In *Proceedings of the 11th ACM Conference on Computer and Communications Security (CCS'04)*. ACM Press, 2004.

[BD95] M. Burmester and Y. Desmedt. A secure and efficient conference key distribution system. In *Advances in Cryptology – Eurocrypt 94*, volume 950 of *LNCS*, pages 275–286. Springer, Berlin, 1995.

[BD$^+$03] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield. Xen and the art of virtualization. In *ACM Symposium on Operating Systems Principles (ASOSP)*, pages 164–177, Bolton Landing, New York, USA, 2003.

[BD04] B. Byramjee and S. Duquesne. Classification of genus 2 curves over $\mathbb{F}_{2^n}$ and optimization of their arithmetic. Cryptology ePrint Archive, Report 2004/107, 2004. http://eprint.iacr.org/2004/107.

[Ber] D. J. Bernstein. Pippenger's exponentiation algorithm. http://cr.yp.to/papers.html. Note: to be incorporated into author's *High-speed cryptography* book.

[BF01] D. Boneh and M. Franklin. Identity-based encryption from the Weil pairing. *LNCS*, 2139:213–??, 2001.

[BG$^+$03] D. Boneh, C. Gentry, H. Shacham, and B. Lynn. Aggregate and verifiably encrypted signatures from bilinear maps. In *Eurocrypt '03*, volume LNCS, pages 416–432, 2003.

[BG$^+$07] P. S. Barreto, S. D. Galbraith, C. Ó' Héigeartaigh, and M. Scott. Efficient pairing computation on supersingular Abelian varieties. *Des. Codes Cryptography*, 42(3):239–271, 2007.

[BGW05] D. Boneh, C. Gentry, and B. Waters. Collusion resistant broadcast encryption with short ciphertexts and private keys. In *Crypto '05*, volume LNCS, pages 258–275, 2005.

[BK$^+$02] P. S. L. M. Barreto, H. Y. Kim, B. Lynn, and M. Scott. Efficient algorithms for pairing-based cryptosystems. In *Advances in Cryptology - CRYPTO 2002, 22nd Annual International Cryptology Conference, Santa Barbara, California, USA, August 18-22, 2002, Proceedings*, volume 2442 of *LNCS*, pages 354–368. Springer, 2002.

[BL07] D. J. Bernstein and T. Lange. Explicit-Formulas Database. Available at `http://www.hyperelliptic.org/EFD/`, 2007.

[Bla02] G. Blady. Die Weil-Restriktion elliptischer Kurven in der Kryptographie. Master's thesis, Universität-Gesamthochschule Essen, 2002.

[BLF08] D. J. Bernstein, T. Lange, and R. R. Farashahi. Binary edwards curves. Cryptology ePrint Archive, Report 2008/171, 2008. `http://eprint.iacr.org/2008/171`.

[BLS02] D. Boneh, B. Lynn, and H. Shacham. Short signatures from the Weil pairing. In *Advances in Cryptology – Asiacrypt 2001*, volume 2248 of *LNCS*, pages 514–532. Springer, Berlin, 2002.

[BN05] P. S. L. M. Barreto and M. Naehrig. Pairing-friendly elliptic curves of prime order. In *Proceedings of SAC 2005, volume 3897 of LNCS*, pages 319–331. Springer, 2005.

[Bod07] M. Bodrato. Towards optimal Toom-Cook multiplication for univariate and multivariate polynomials in characteristic 2 and 0. In *WAIFI'07 proceedings*, volume 4547 of *LNCS*, pages 116–133. Springer, 2007.

[BW06] D. Boneh and B. Waters. A fully collusion resistant broadcast, trace, and revoke system. In *Eurocrypt '06*, volume LNCS, pages 573–592, 2006.

[BWN$^+$06] S. Blake-Wilson, M. Nystrom, D. Hopwood, J. Mikkelsen, and T. Wright. Transport Layer Security (TLS) Extensions. RFC 4366 (Proposed Standard), 2006. Obsoleted by RFC 5246.

[Can87] D. G. Cantor. Computing in the Jacobian of a hyperelliptic curve. *Math. Comp.*, 48:95–101, 1987.

[Ces04] E. Cesena. Varietà a traccia zero su campi binari – applicazioni crittografiche. Master's thesis, Università degli Studi di Milano, 2004.

[CL01] J. Camenisch and A. Lysyanskaya. An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In *Advances in Cryptology – EUROCRYPT'01*, number 2045 in LNCS. Springer, 2001.

[CL$^+$10] E. Cesena, H. Löhr, G. Ramunno, A.-R. Sadeghi, and D. Vernizzi. Anonymous Authentication with TLS and DAA, 2010. Submitted to TRUST 2010.

[CMO98] H. Cohen, A. Miyaji, and T. Ono. Efficient elliptic curve exponentiation using mixed coordinates. In *Advances in Cryptology – ASIACRYPT'98*, volume 1514 of *LNCS*, pages 51–65. Springer, 1998.

[CMS08] L. Chen, P. Morrissey, and N. Smart. Pairings in trusted computing. In *Proceedings of Pairing 2008, volume 5209 of LNCS*, pages 1–17. Springer, 2008.

[CRV08a] E. Cesena, G. Ramunno, and D. Vernizzi. Secure storage using a sealing proxy. In *EUROSEC '08: Proceedings of the 1st European workshop on system security*, pages 27–34, New York, NY, USA, 2008. ACM.

[CRV08b] E. Cesena, G. Ramunno, and D. Vernizzi. Towards trusted broadcast encryption. *Young Computer Scientists, International Conference for*, 0:2125–2130, 2008.

[CY02] Y. Choie and D. Yun. Isomorphism classes of hyperelliptic curves of genus 2 over $\mathbb{F}_q$. In *ACISP 2002*, volume 2384 of *LNCS*, pages 190–202. Springer, Berlin, 2002.

[DBS04] R. Dutta, R. Barua, and P. Sarkar. Pairing-based cryptographic protocols : A survey. Cryptology ePrint Archive, Report 2004/064, 2004. http://eprint.iacr.org/2004/064.

[DF03] Y. Dodis and N. Fazio. Public key broadcast encryption for stateless receivers. *Digital Rights Management*, pages 61–80, 2003.

[DF$^+$05] Y. Dodis, N. Fazio, A. Kiayias, and M. Yung. Scalable public-key tracing and revoking. *Distrib. Comput.*, 17(4):323–347, 2005.

[DH76] W. Diffie and M. E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976.

[DR08] T. Dierks and E. Rescorla. The Transport Layer Security (TLS) Protocol Version 1.2. RFC 5246 (Proposed Standard), 2008.

[DS03] C. Diem and J. Scholten. Cover Attacks – A report for the AREHCC project, 2003.

[ECR09] ECRYPT II NoE. ECRYPT2 Yearly Report on Algorithms and Keysizes (2008-2009). ECRYPT II deliverable D.SPA.7 revision 1.0, 2009.

[ElG85] T. ElGamal. A public key cryptosystem and signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, pages 473–481, 1985.

[F07̈] M. Fürer. Faster integer multiplication. In *STOC '07: Proceedings of the thirty-ninth annual ACM symposium on Theory of computing*, pages 57–66, New York, NY, USA, 2007. ACM.

[FH$^+$03] K. Fong, D. Hankerson, J. Lopez, and A. Menezes. Field inversion and point halving revisited. *IEEE Transactions on Computers*, 53:1047–1059, 2003.

[Fin09] H. Finney. Private communication, 2009.

[FN94] A. Fiat and M. Naor. Broadcast encryption. pages 480–491, 1994.

[FR94] G. Frey and H.-G. Rück. A remark concerning m-divisibility and the discrete logarithm in the divisor class group of curves. *Mathematics of Computation*, 62(206):865–874, 1994.

[Fre98] G. Frey. How to disguise an elliptic curve. Talk at Waterloo workshop on the ECDLP, 1998. http://www.cacr.math.uwaterloo.ca/conferences/1998/ecc98/slides.html.

[Fre01] G. Frey. Applications of arithmetical geometry to cryptographic constructions. In *Finite fields and applications (Augsburg, 1999)*, pages 128–161. Springer, Berlin, 2001.

[Fre08] G. Frey. Discrete logarithms, duality, and arithmetic in Brauer groups. In J. Hirschfeld, J. Chaumine, and R. Rolland, editors, *Algebraic geometry and its applications*, volume 5 of *Number Theory and Its Applications*, pages 241–272. World Scientific, 2008. Proceedings of the first SAGA conference, 2007, Papeete.

[FST06] D. Freeman, M. Scott, and E. Teske. A taxonomy of pairing-friendly elliptic curves. Cryptology ePrint Archive, Report 2006/372, 2006. http://eprint.iacr.org/2006/372 – To appear in Journal of Cryptology (2009).

[GHS02a] S. Galbraith, F. Hess, and N. Smart. Extending the GHS Weil-descent attack. In *Eurocrypt 2002*, volume 2332 of *LNCS*, pages 29–44, Berlin, 2002. Springer.

[GHS02b] P. Gaudry, F. Hess, and N. P. Smart. Constructive and destructive facets of Weil descent on elliptic curves. *Journal of Cryptology*, 15(1):19–46, 2002. Online publication: 29 August 2001.

[GM$^+$08] S. Gajek, M. Manulis, O. Pereira, A.-R. Sadeghi, and J. Schwenk. Universally composable security analysis of TLS. pages 313–327, 2008.

[GP02] J. Guajardo and C. Paar. Itoh-tsujii inversion in standard basis and its application in cryptography and codes. *Des., Codes and Cryptography*, 25:207–216, 2002.

[GP$^+$03] T. Garfinkel, B. Pfaff, J. Chow, M. Rosenblum, and D. Boneh. Terra: a virtual machine-based platform for trusted computing. In *ACM Symposium on Operating Systems Principles (ASOSP)*, pages 193–206, Bolton Landing, New York, USA, 2003.

[GPS06a] K. Goldman, R. Perez, and R. Sailer. Linking remote attestation to secure tunnel endpoints. In *STC '06: Proceedings of the first ACM workshop on Scalable trusted computing*, pages 21–24, 2006.

[GPS06b] R. Granger, D. Page, and M. Stam. On small characteristic algebraic tori in pairing-based cryptography. *The LMS JCM*, 9:64–85, 2006.

[GPS08] S. D. Galbraith, K. G. Paterson, and N. P. Smart. Pairings for cryptographers. *Discrete Appl. Math.*, 156(16):3113–3121, 2008.

[Gro06] D. Growrock. *The Intel Safer Computing Initiative – Building Blocks for Trusted Computing*. Intel press, 2006.

[GST04] M. T. Goodrich, J. Z. Sun, and R. Tamassia. Efficient tree-based revocation in groups of low-state devices. *Advances in Cryptology – CRYPTO 2004*, pages 511–527, 2004.

[GT$^+$07] P. Gaudry, E. Thomé, N. Thériault, and C. Diem. A double large prime variation for small genus hyperelliptic index calculus. *Math. Comp.*, 76:475–492, 2007.

[Hes08] F. Hess. Pairing Lattices. In S. D. Galbraith and K. G. Paterson, editors, *Pairing 2008*, volume 5209 of *LNCS*, pages 211–224. Springer, 2008.

[HKA06] F. Hoshino, T. Kobayashi, and K. Aoki. Compressed jacobian coordinates for OEF. In *Progress in Cryptology - VIETCRYPT 2006*, volume 4341 of *LNCS*, pages 147–156. Springer, 2006.

[HS02] D. Halevy and A. Shamir. The lsd broadcast encryption scheme. *Advances in Cryptology - CRYPTO 2002: 22nd Annual International Cryptology Conference Santa Barbara, California, USA, August 18-22, 2002. Proceedings*, pages 145–161, 2002.

[HSV06] F. Hess, N. P. Smart, and F. Vercauteren. The Eta Pairing Revisited. *IEEE Trans. Inform. Theory*, 52:4595–4602, 2006.

[iN04] K. ichi Nagao. Improvement of thériault algorithm of index calculus for jacobian of hyperelliptic curves of small genus. Cryptology ePrint Archive, Report 2004/161, 2004. `http://eprint.iacr.org/2004/161`.

[IT88] T. Itoh and S. Tsujii. A fast algorithm for computing multiplicative inverses in GF(2^m) using normal bases. *Information and Computation*, 78(3):171–177, 1988.

[Jou00] A. Joux. A one round protocol for tripartite Diffie–Hellman. In *Algorithmic Number Theory, ANTS-IV*, volume 1838 of *LNCS*, pages 385–394. Springer, 2000.

[JY$^+$05] N.-S. Jho, E. S. Yoo, J. H. Cheon, and M.-H. Kim. New broadcast encryption scheme using tree-based circle. pages 37–44, 2005.

[KAT08] Y. Kawahara, K. Aoki, and T. Takagi. Faster Implementation of η_T Pairing over GF(3^m) Using Minimum Number of Logical Instructions for GF(3)-Addition. In *Proceedings of Pairing 2008, volume 5209 of LNCS*, pages 282–296, 2008.

[Kau07] B. Kauer. OSLO: Improving the security of Trusted Computing. In *Proceedings of the 16th USENIX Security Symposium, Boston, MA, USA*, 2007.

[KK07] K. H. Kim and S. I. Kim. A new method for speeding up arithmetic on elliptic curves over binary fields. Cryptology ePrint Archive, Report 2007/181, 2007. http://eprint.iacr.org/2007/181.

[KM$^+$99] T. Kobayashi, H. Morita, K. Kobayashi, and F. Hoshino. Fast elliptic curve algorithm combining Frobenius map and table reference to adapt to higher characteristic. In *Theory and Application of Cryptographic Techniques*, pages 176–189, 1999.

[KM05] N. Koblitz and A. Menezes. Pairing-based cryptography at high security levels. *Cryptography and Coding*, pages 13–36, 2005.

[Kob87] N. Koblitz. Elliptic curve cryptosystems. *Math. Comp.*, 48(177):203–209, 1987.

[Kob89] N. Koblitz. Hyperelliptic cryptosystems. *Journal of Cryptology*, 1:139–150, 1989.

[Kob91] N. Koblitz. Constructing elliptic curve cryptosystems in characteristic 2. In *Advances in Cryptology – Crypto 90*, volume 537 of *LNCS*, pages 156–167. Springer, Berlin, 1991.

[KS07] N. Kuntze and A. U. Schmidt. Protection of DVB Systems by Trusted Computing. In *IEEE International Symposium on Broadband Multimedia Systems and Broadcasting 2007, 28-29 March 2007 at the Orange County Convention Center, Orlando, FL, USA*, 2007.

[KSP05] K. Kursawe, D. Schellekens, and B. Preneel. Analyzing trusted platform communication. In *ECRYPT Workshop, CRASH - CRyptographic Advances in Secure Hardware*, 2005.

[Lan01] T. Lange. *Efficient arithmetic on hyperelliptic curves*. PhD thesis, University Essen, 2001.

[Lan04] T. Lange. Trace zero subvarieties of genus 2 curves for cryptosystems. *J. Ramanujan. Math. Soc.*, 19:15–33, 2004.

[Lan05] T. Lange. Formulae for arithmetic on genus 2 hyperelliptic curves. *Applicable Algebra in Engineering, Communication and Computing*, 15(5):295–328, 2005.

[LD99] J. López and R. Dahab. Improved algorithms for elliptic curve arithmetic in gf(2n). In *SAC '98: Proceedings of the Selected Areas in Cryptography*, volume 1556 of *LNCS*, pages 201–212. Springer, 1999.

[Lim00] C. H. Lim. Efficient multi-exponentiation and application to batch verification of digital signatures. Unpublished manuscript. http://dasan.sejong.ac.kr/~chlim/english_pub.html, 2000.

[LLP09] E. Lee, H.-S. Lee, and C.-M. Park. Efficient and generalized pairing computation on abelian varieties. *IEEE Transactions on Information Theory*, 55(4):1793–1803, 2009.

[Lor96] D. Lorenzini. *An invitation to arithmetic geometry*, volume 9 of *Graduate studies in mathematics*. AMS, 1996.

[LS04] T. Lange and M. Stevens. Efficient doubling on genus two curves over binary fields. In *Selected Areas in Cryptography*, pages 170–181, 2004.

[Mil86a] V. S. Miller. Use of elliptic curves in cryptography. In *Advances in cryptology – crypto '85*, volume 218 of *LNCS*, pages 417–426. Springer, Berlin, 1986.

[Mil86b] V. S. Miller. Short programs for functions on curves. Unpublished manuscript. http://crypto.stanford.edu/miller/, 1986.

[Mil04] V. S. Miller. The weil pairing, and its efficient calculation. *Journal of Cryptology*, 17(4):235–261, 2004.

[MK$^+$07] S. Matsuda, N. Kanayama, F. Hess, and E. Okamoto. Optimised versions of the Ate and Twisted Ate Pairings. In *The 11th IMA International Conference on Cryptography and Coding*, volume 4887 of *LNCS*, pages 302–312. Springer, 2007.

[MOC97] A. Miyaji, T. Ono, and H. Cohen. Efficient elliptic curve exponentiation. In *ICICS '97*, LNCS 1334, pages 282–290, 1997.

[Möl01] B. Möller. Securing elliptic curve point multiplication against Side-Channel Attacks. In *Information Security – ISC 2001*, pages 324–334. Springer, 2001.

[MOV93] A. J. Menezes, T. Okamoto, and S. Vanstone. Reducing elliptic curve logarithms to a finite field. *IEEE Trans. on Inform. Theory*, 39:1639–1646, 1993.

[MRS07] B. Mazur, K. Rubin, and A. Silverberg. Twisting commutative algebraic groups. *Journal of Algebra*, 314(1):419 – 438, 2007.

[MWZ98] A. J. Menezes, Y.-H. Wu, and R. Zuccherato. An elementary introduction to hyperelliptic curves. In N. Koblitz, editor, *Algebraic aspects of cryptography*, pages 155–178. Springer, Berlin, 1998.

[Nau99] N. Naumann. Weil-Restriktion abelscher Varietäten. Master's thesis, University Essen, 1999.

[NBS08] M. Naehrig, P. S. L. M. Barreto, and P. Schwabe. On compressible pairings and their computation. In *Progress in Cryptology – AFRICACRYPT 2008*, pages 371–388, 2008.

[NNL02] D. Naor, M. Naor, and J. Lotspiech. Revocation and tracing schemes for stateless receivers. *Advances in Cryptology - CRYPTO 2001: 21st Annual International Cryptology Conference, Santa Barbara, California, USA, August 19-23, 2001, Proceedings*, pages 41–, 2002.

[NP00] M. Naor and B. Pinkas. Efficient trace and revoke schemes. 2000.

[Ope] Open Trusted Computing (OpenTC). http://www.opentc.net/.

[PH78] S. Pohlig and M. Hellmann. An improved algorithm for computing logarithms over GF(p) and its cryptographic significance. *IEEE Trans. Inform. Theory*, IT-24:106–110, 1978.

[Pip76] N. Pippenger. On the evaluation of powers and related problems. In *SFCS '76: Proceedings of the 17th Annual Symposium on Foundations of Computer Science*, pages 258–263, Washington, DC, USA, 1976. IEEE Computer Society.

[Rei62] G. Reitwiesner. Binary arithmetic. *Advances in Computers*, 1:231–308, 1962.

[RS02] K. Rubin and A. Silverberg. Supersingular abelian varieties in cryptology. In *CRYPTO '02: Proceedings of the 22nd Annual International Cryptology Conference on Advances in Cryptology*, pages 336–353, London, UK, 2002. Springer.

[RS09] K. Rubin and A. Silverberg. Using abelian varieties to improve pairing-based cryptography. *Journal of Cryptology*, 22(3):330–364, 2009.

[RSA78] R. L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signature and public key cryptosystems. *Comm. ACM*, 21:120–126, 1978.

[San06] S. Santesson. TLS Handshake Message for Supplemental Data. RFC 4680 (Proposed Standard), 2006.

[SB04] M. Scott and P. S. L. M. Barreto. Compressed pairings. In *Advances in Cryptology – CRYPTO 2004*, pages 140–156, 2004.

[Sch77] A. Schönhage. Schnelle multiplikation von polynomen über körpern der charakteristik 2. *Acta Informatica*, 7:395–398, 1977.

[Sch87] R. Schoof. Nonsingular plane cubic curves. *Journal of Combinatorial Theory, Series A*, 46:183–211, 1987.

[Sch95] R. Schoof. Counting points on elliptic curves over finite fields. *J. de Théo. des Nombres de Bordeaux*, 7:219–254, 1995.

[Sco04] M. Scott. Faster identity based encryption. *Electronics Letters*, 40(14), 2004.

[Sco05] M. Scott. Faster Pairings Using an Elliptic Curve with an Efficient Endomorphism. In *INDOCRYPT 2005*, volume 3797 of *LNCS*, pages 258–269. Springer, 2005.

[Sil86] J. H. Silverman. *The arithmetic of elliptic curves*, volume 106 of *Graduate texts in mathematics*. Springer, 1986.

[Sil05] A. Silverberg. Compression for Trace Zero Subgroups of Elliptic Curves. In *Trends in Mathematics*, volume 8, pages 93–100, 2005. Proceedings of the Daewoo Workshop on Cryptography.

[SL00] S. G. Sim and P. J. Lee. An efficient implementation of two-term exponentiation in elliptic curves. In *Japan–Korea Joint Workshop on Information Security and Cryptology (JW-ISC 2000)*, pages 61–68, 2000.

[Sol00] J. A. Solinas. Efficient arithmetic on Koblitz curves. *Des. Codes Cryptogr.*, 19:195–249, 2000.

[Sol01] J. A. Solinas. Low-weight binary representations for pairs of integers. Combinatorics and Optimization Research Report CORR 2001-41, University of Waterloo, 2001.

[SRC07] B. Smyth, M. Ryan, and L. Chen. Direct anonymous attestation (daa): Ensuring privacy with corrupt administrators. In *ESAS*, pages 218–231, 2007.

[SS71] A. Schönhage and V. Strassen. Schnelle multiplikation großer zahlen. *Computing*, 7:281–292, 1971.

[SSP04] A.-R. Sadeghi, C. Stüble, and N. Pohlmann. European multilateral secure computing base, 2004.

[Too63] A. L. Toom. The complexity of a scheme of functional elements realizing the multiplication of integers. *Soviet Mathematics*, 3:714–716, 1963.

[Tru] Trusted Computing Group. TCG Software Stack Specification Version 1.2, Level 1, Errata A. Available at https://www.trustedcomputinggroup.org/.

[Tru09] Trusted Computing Group. TCG website. https://www.trustedcomputinggroup.org, last accessed Sep. 2009.

[vDG$^+$05] M. van Dijk, R. Granger, D. Page, K. Rubin, A. Silverberg, M. Stam, and D. Woodruff. Practical cryptography in high dimensional tori. In *Advances in Cryptology – EUROCRYPT 2005*, pages 234–250, 2005.

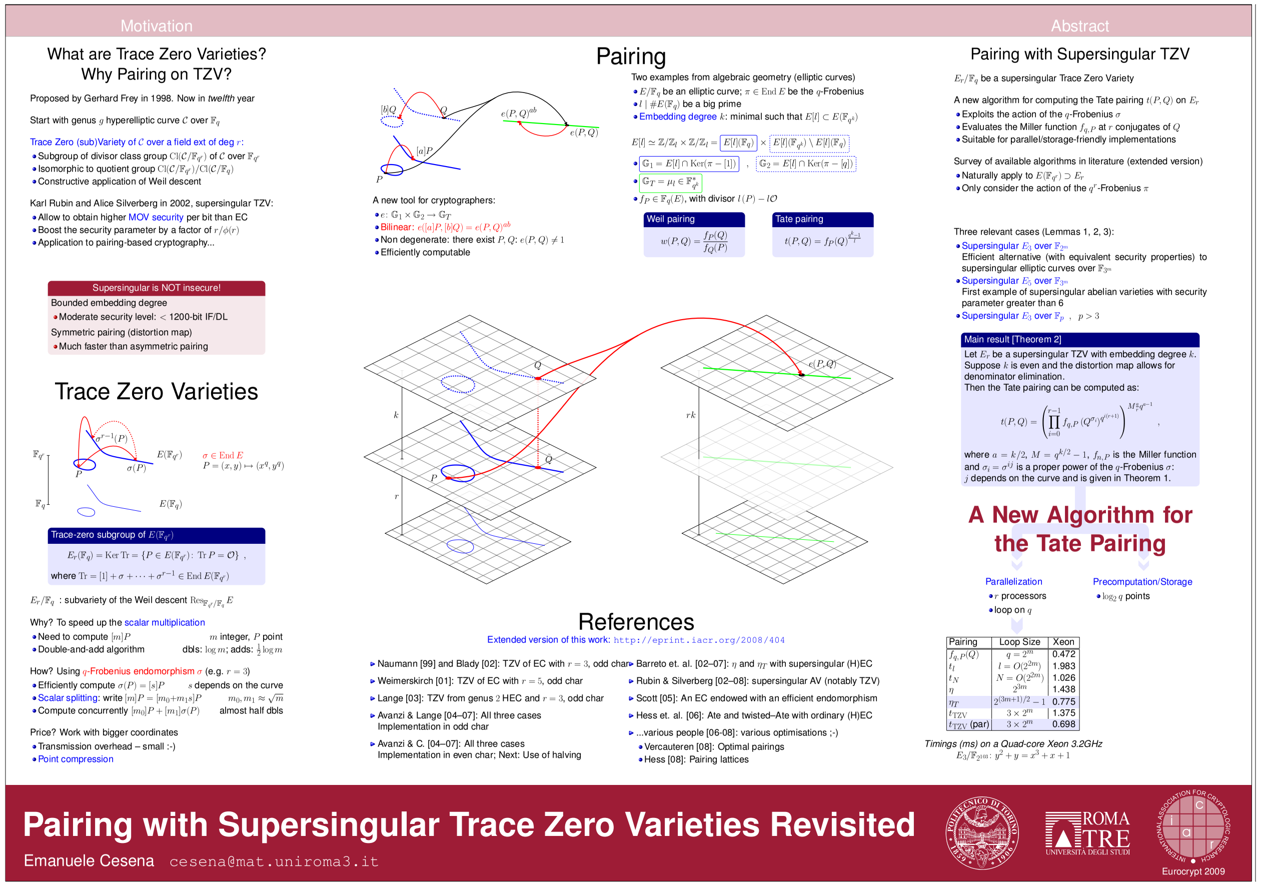
[Ver08] F. Vercauteren. Optimal Pairings. Cryptology ePrint Archive, Report 2008/096, 2008. http://eprint.iacr.org/2008/096.

[Wei01] A. Weimerskirch. The application of the Mordell–Weil group to cryptographic systems. Master's thesis, Worchester Polytechnic Institute, 2001.

[ZZH07] C.-A. Zhao, F. Zhang, and J. Huang. A Note on the Ate Pairing. Cryptology ePrint Archive, Report 2007/247, 2007. http://eprint.iacr.org/2007/247.

Appendix A

Poster (Eurocrypt 2009)



Motivation

What are Trace Zero Varieties? Why Pairing on TZV?

Proposed by Gerhard Frey in 1998. Now in *twelfth* year

Start with genus g hyperelliptic curve $\mathcal{C}$ over $\mathbb{F}_q$

Trace Zero (sub)Variety of $\mathcal{C}$ over a field ext of deg r:

- Subgroup of divisor class group $\mathrm{Cl}(\mathcal{C}/\mathbb{F}_{q^r})$ of $\mathcal{C}$ over $\mathbb{F}_{q^r}$
- Isomorphic to quotient group $\mathrm{Cl}(\mathcal{C}/\mathbb{F}_{q^r})/\mathrm{Cl}(\mathcal{C}/\mathbb{F}_q)$
- Constructive application of Weil descent

Karl Rubin and Alice Silverberg in 2002, supersingular TZV:

- Allow to obtain higher MOV security per bit than EC
- Boost the security parameter by a factor of $r/\phi(r)$
- Application to pairing-based cryptography...

Supersingular is NOT insecure!

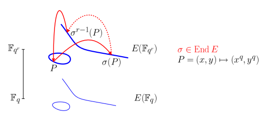
Bounded embedding degree

- Moderate security level: $<$ 1200-bit IF/DL

Symmetric pairing (distortion map)

- Much faster than asymmetric pairing

Trace Zero Varieties



Trace-zero subgroup of $E(\mathbb{F}_{q^r})$

$$E_r(\mathbb{F}_q) = \mathrm{Ker}\,\mathrm{Tr} = \{P \in E(\mathbb{F}_{q^r}) \colon \mathrm{Tr}\, P = \mathcal{O}\} ,$$

where $\mathrm{Tr} = [1] + \sigma + \cdots + \sigma^{r-1} \in \mathrm{End}\, E(\mathbb{F}_{q^r})$

$E_r/\mathbb{F}_q$: subvariety of the Weil descent $\mathrm{Res}_{\mathbb{F}_{q^r}/\mathbb{F}_q} E$

Why? To speed up the scalar multiplication

- Need to compute $[m]P$ — m integer, P point
- Double-and-add algorithm — dbls: $\log m$; adds: $\frac{1}{2}\log m$

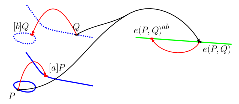
How? Using q-Frobenius endomorphism σ (e.g. $r = 3$)

- Efficiently compute $\sigma(P) = [s]P$ — s depends on the curve
- Scalar splitting: write $[m]P = [m_0+m_1 s]P$ — $m_0, m_1 \approx \sqrt{m}$
- Compute concurrently $[m_0]P + [m_1]\sigma(P)$ — almost half dbls

Price? Work with bigger coordinates

- Transmission overhead – small :-)
- Point compression

Pairing



Two examples from algebraic geometry (elliptic curves)

- $E/\mathbb{F}_q$ be an elliptic curve; $\pi \in \mathrm{End}\, E$ be the q-Frobenius
- $l \mid \#E(\mathbb{F}_q)$ be a big prime
- Embedding degree k: minimal such that $E[l] \subset E(\mathbb{F}_{q^k})$

$$E[l] \simeq \mathbb{Z}/\mathbb{Z}_l \times \mathbb{Z}/\mathbb{Z}_l = \boxed{E[l](\mathbb{F}_q)} \times \boxed{E[l](\mathbb{F}_{q^k}) \setminus E[l](\mathbb{F}_q)}$$

- $\boxed{\mathbb{G}_1 = E[l] \cap \mathrm{Ker}(\pi - [1])}$, $\boxed{\mathbb{G}_2 = E[l] \cap \mathrm{Ker}(\pi - [q])}$
- $\boxed{\mathbb{G}_T = \mu_l \in \mathbb{F}^*_{q^k}}$
- $f_P \in \mathbb{F}_q(E)$, with divisor $l\,(P) - l\mathcal{O}$

Weil pairing

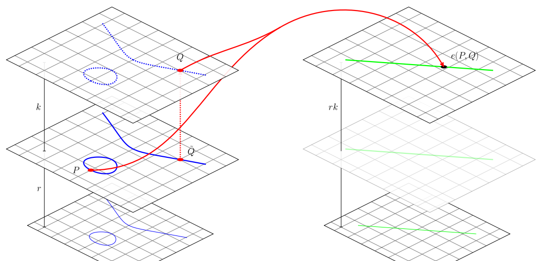
$$w(P,Q) = \frac{f_P(Q)}{f_Q(P)}$$

Tate pairing

$$t(P,Q) = f_P(Q)^{\frac{q^k-1}{l}}$$

A new tool for cryptographers:

- $e \colon \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$
- Bilinear: $e([a]P, [b]Q) = e(P,Q)^{ab}$
- Non degenerate: there exist P, Q: $e(P,Q) \neq 1$
- Efficiently computable



References

Extended version of this work: http://eprint.iacr.org/2008/404

- Naumann [99] and Blady [02]: TZV of EC with $r = 3$, odd char
- Weimerskirch [01]: TZV of EC with $r = 5$, odd char
- Lange [03]: TZV from genus 2 HEC and $r = 3$, odd char
- Avanzi & Lange [04–07]: All three cases
  Implementation in odd char
- Avanzi & C. [04–07]: All three cases
  Implementation in even char; Next: Use of halving
- Barreto et. al. [02–07]: η and η_T with supersingular (H)EC
- Rubin & Silverberg [02–08]: supersingular AV (notably TZV)
- Scott [05]: An EC endowed with an efficient endomorphism
- Hess et. al. [06]: Ate and twisted–Ate with ordinary (H)EC
- ...various people [06-08]: various optimisations ;-)
  - Vercauteren [08]: Optimal pairings
  - Hess [08]: Pairing lattices

Abstract

Pairing with Supersingular TZV

$E_r/\mathbb{F}_q$ be a supersingular Trace Zero Variety

A new algorithm for computing the Tate pairing $t(P,Q)$ on E_r

- Exploits the action of the q-Frobenius σ
- Evaluates the Miller function $f_{q,P}$ at r conjugates of Q
- Suitable for parallel/storage-friendly implementations

Survey of available algorithms in literature (extended version)

- Naturally apply to $E(\mathbb{F}_{q^r}) \supset E_r$
- Only consider the action of the q^r-Frobenius π

Three relevant cases (Lemmas 1, 2, 3):

- Supersingular E_3 over $\mathbb{F}_{2^m}$
  Efficient alternative (with equivalent security properties) to supersingular elliptic curves over $\mathbb{F}_{3^m}$
- Supersingular E_5 over $\mathbb{F}_{3^m}$
  First example of supersingular abelian varieties with security parameter greater than 6
- Supersingular E_3 over $\mathbb{F}_p$, $p > 3$

Main result [Theorem 2]

Let E_r be a supersingular TZV with embedding degree k. Suppose k is even and the distortion map allows for denominator elimination.
Then the Tate pairing can be computed as:

$$t(P,Q) = \left(\prod_{i=0}^{r-1} f_{q,P}\left(Q^{\sigma_i}\right)^{q^{i(r+1)}} \right)^{M\frac{a}{r}q^{a-1}} ,$$

where $a = k/2$, $M = q^{k/2} - 1$, $f_{n,P}$ is the Miller function and $\sigma_i = \sigma^{ij}$ is a proper power of the q-Frobenius σ:
j depends on the curve and is given in Theorem 1.

A New Algorithm for the Tate Pairing

Parallelization

- r processors
- loop on q

Precomputation/Storage

- $\log_2 q$ points

Pairing	Loop Size	Xeon
$f_{q,P}(Q)$	$q = 2^m$	0.472
t_l	$l = O(2^{2m})$	1.983
t_N	$N = O(2^{2m})$	1.026
η	2^{3m}	1.438
η_T	$2^{(3m+1)/2} - 1$	0.775
t_{TZV}	3×2^m	1.375
t_{TZV} (par)	3×2^m	0.698

Timings (ms) on a Quad-core Xeon 3.2GHz
$E_3/\mathbb{F}_{2^{103}} \colon y^2 + y = x^3 + x + 1$

Pairing with Supersingular Trace Zero Varieties Revisited

Emanuele Cesena cesena@mat.uniroma3.it

ROMA TRE UNIVERSITÀ DEGLI STUDI

Eurocrypt 2009

Appendix B

Explicit Examples of Curves and Trace Zero Varieties

In this appendix we list the actual fields and curves used in our experiments. The construction methodology is discussed in Section 3.7. Notice that, in all the examples, finding good groups required only a few minutes of computation on old workstations.

Elliptic, respectively hyperelliptic, curves have equations of the form (3.2), respectively (3.1). We denote with ℓ the group order, with c the cofactor and with s the constant such that $sD = \sigma(D)$ on the TZV.

Field elements are given in hexadecimal form, the least significant digits being to the right. For instance, let $K = \mathbb{F}_{2^7} = \mathbb{F}_2(\beta)$ with $\beta^7 + \beta + 1 = 0$. The hexadecimal string `0x23` is (0010 0011) in binary notation, and represents the field element $\beta^5 + \beta + 1 = \beta^{19}$.

B.1 Binary Fields

In Table B.1 we list the polynomials defining the binary fields used in our experiments.

Some of them differs from NIST's standard polynomials and are so-called square-root friendly, in the sense that they allow fast computation of the square root operation [Ava07], that may be desirable for instance to implement point-halving techniques.

Binary field	Defining polynomial
$\mathbb{F}_{2^{41}}$	$X^{41} + X^3 + 1$
$\mathbb{F}_{2^{47}}$	$X^{47} + X^5 + 1$
$\mathbb{F}_{2^{79}}$	$X^{79} + X^9 + 1$
$\mathbb{F}_{2^{83}}$	$X^{83} + X^{29} + X^{25} + X^3 + 1$
$\mathbb{F}_{2^{97}}$	$X^{97} + X^{33} + 1$
$\mathbb{F}_{2^{103}}$	$X^{103} + X^9 + 1$
$\mathbb{F}_{2^{157}}$	$X^{157} + X^{55} + X^{47} + X^{11} + 1$
$\mathbb{F}_{2^{163}}$	$X^{163} + X^{57} + X^{49} + X^{29} + 1$
$\mathbb{F}_{2^{191}}$	$X^{157} + X^9 + 1$
$\mathbb{F}_{2^{239}}$	$X^{239} + X^{36} + 1$
$\mathbb{F}_{2^{307}}$	$X^{307} + X^{113} + X^{81} + X^{25} + 1$
$\mathbb{F}_{2^{457}}$	$X^{457} + X^{61} + 1$

Table B.1: Binary fields and their defining polynomials

B.2 80-bit Security Level Groups

Elliptic curve over $\mathbb{F}_{2^{163}}$

$$\begin{aligned} a_2 &= \texttt{1} \\ a_6 &= \texttt{0x3B4A2A47C6CF50931A85F1DF23A3E5501E4DDCF5D} \\ \ell &= 5846006549323611672814742442876390689256843201587 \quad \text{(162-bit prime)} \\ c &= 2 \end{aligned}$$

This elliptic curve is isomorphic to the standard `SECG-163r2` curve.

TZV over $\mathbb{F}_{2^{83}}$ with parameters $g = 1$, $r = 3$

$$\begin{aligned} a_2 &= \texttt{0} \\ a_6 &= \texttt{0x7E068A86F57E189F1579F} \\ \ell &= 93536104789211454038017047439473721230904196835799 \quad \text{(166-bit prime)} \\ s &= 52717605029603870685346358367337123607533507700722 \quad (\text{mod } \ell) \end{aligned}$$

TZV over $\mathbb{F}_{2^{41}}$ with parameters $g = 1$, $r = 5$

$$\begin{aligned} a_2 &= \texttt{0} \\ a_6 &= \texttt{0x8200040000} \\ \ell &= 23384024072061117236215132194916117478003941329701 \quad \text{(164-bit prime)} \\ s &= 8596898397316878370683469747519719961741471215582 \quad (\text{mod } \ell) \end{aligned}$$

Genus 2 hyperelliptic curve over $\mathbb{F}_{2^{83}}$

$$\begin{aligned} f_0 &= \texttt{0x396F428E376D7A8890383} \\ f_2 &= \texttt{1} \\ f_3 &= \texttt{0x19DCA0E8D1C92F9264B24} \\ \ell &= 46768052394608457006025733480294790398034747555811 \quad \text{(165-bit prime)} \\ c &= 2 \end{aligned}$$

TZV over $\mathbb{F}_{2^{41}}$ with parameters $g = 2$, $r = 3$

$$\begin{aligned} f_0 &= \texttt{0x19A020B045} \\ f_2 &= \texttt{1} \\ f_3 &= \texttt{0x1FD89650906} \\ \ell &= 23384039272790760689682700973748651773762499208657 \quad \text{(164-bit prime)} \\ s &= 2891357468045729639561564946934067632888417875314 \quad (\text{mod } \ell) \end{aligned}$$

B.3 96-bit Security Level Groups

Elliptic curve over $\mathbb{F}_{2^{191}}$

$a_2 =$ `1`
$a_6 =$ `0x2F0F22E6C4915EDBE905DD0A0C714E1D1413EE7AF8AED970`
$\ell = 1569275433846670190958947355829117289761686103911895757549$ (190-bit prime)
$c = 2$

TZV over $\mathbb{F}_{2^{97}}$ with parameters $g = 1$, $r = 3$

$a_2 =$ `1`
$a_6 =$ `0x661CA973864F807D23C4CC09`
$\ell = 25108406941546760434751930836434834976964168002830535418893$ (194-bit prime)
$s = 10669767213033396095159100125257086651731213520935893527279$ (mod ℓ)

TZV over $\mathbb{F}_{2^{47}}$ with parameters $g = 1$, $r = 5$

$a_2 =$ `0`
$a_6 =$ `0x20840001`
$\ell = 392318871423095243169862765890601877920346380738406349101$ (188-bit prime)
$s = 183867226781893594594157220308313210869565837762023539\text{19}$ (mod ℓ)

Genus 2 hyperelliptic curve over $\mathbb{F}_{2^{97}}$

$f_0 =$ `0x4E822D0AE44FEAEB13E11479`
$f_2 =$ `1`
$f_3 =$ `0x1A12EFA11ADD61DD3A24955D`
$\ell = 6277101735386666829310889747862416984303814218548215922379$ (192-bit prime)
$c = 4$

TZV over $\mathbb{F}_{2^{47}}$ with parameters $g = 2$, $r = 3$

$f_0 =$ `0x3FC1AB44F431`
$f_2 =$ `1`
$f_3 =$ `0x52035DAC7331`
$\ell = 392318938973239036261891895279452669713699615804265346389$ (188-bit prime)
$s = 143307738370408000753254266660332591522050481755159582620$ (mod ℓ)

Appendix C

Additional Experimental Results

In this appendix we include additional experimental results that constitute the completion of those presented in Chapter 4.

C.1 Comparison Among Operations in Finite Fields

	80-bit security							96-bit security						
Curve:	ord. $g = 1$			ord. $g = 2$		supers.		ord. $g = 1$			ord. $g = 2$		supers.	
Field:	163	83×3	41×5	83	41×3	307	103×3	191	97×3	47×5	97	47×3	457	157×3
σ	–	0.01	0.01	–	0.01	–	0.01	–	0.01	0.01	–	0.01	–	0.01
S	0.03	0.05	0.05	0.02	0.03	0.04	0.05	0.02	0.05	0.04	0.01	0.03	0.06	0.07
M	0.14	0.41	0.29	0.07	0.13	0.36	0.48	0.15	0.42	0.32	0.07	0.14	0.85	0.80
M_2	0.25	0.74	0.58	0.12	0.26	0.59	0.83	0.27	0.75	0.63	0.12	0.29	1.51	1.42
I	1.14	1.04	0.94	0.43	0.38	2.94	1.29	1.38	1.11	1.04	0.49	0.41	6.10	2.28
M_2/M	1.86	1.80	2.00	1.75	2.00	1.64	1.72	1.80	1.77	2.00	1.70	2.00	1.79	1.77
I/M	8.34	2.56	3.21	6.21	2.86	8.23	2.69	9.08	2.61	3.26	6.82	2.91	7.20	2.84

Table C.1: Comparison among operations in finite fields – Intel (64-bit), timings in μs. Operations considered are Frobenius automorphism (σ), square (S), multiplication (M), double-multiplication (M_2) and inversion (I). Refer to Section 4.1.3.

	80-bit security							96-bit security						
Curve:	ord. $g = 1$			ord. $g = 2$		supers.		ord. $g = 1$			ord. $g = 2$		supers.	
Field:	163	83×3	41×5	83	41×3	307	103×3	191	97×3	47×5	97	47×3	457	157×3
σ	–	0.02	0.02	–	0.02	–	0.02	–	0.02	0.02	–	0.02	–	0.02
S	0.05	0.09	0.07	0.04	0.05	0.09	0.08	0.05	0.07	0.07	0.04	0.05	0.27	0.13
M	0.35	0.68	0.67	0.12	0.28	1.18	0.81	0.40	0.89	0.69	0.14	0.29	2.18	1.84
M_2	0.65	1.30	1.34	0.21	0.48	2.04	1.43	0.81	1.43	1.37	0.23	0.51	4.36	3.33
I	2.35	1.65	2.01	0.65	0.68	8.16	2.08	2.91	2.04	2.11	0.80	0.76	21.00	4.68
M_2/M	1.86	1.92	2.00	1.83	1.69	1.73	1.76	2.00	1.60	2.00	1.62	1.75	2.00	1.80
I/M	6.70	2.44	3.01	5.51	2.39	6.94	2.56	7.28	2.28	3.07	5.69	2.62	9.62	2.54

Table C.2: Comparison among operations in finite fields – PowerPC (32-bit)

	80-bit security							96-bit security						
Curve:	ord. $g = 1$			ord. $g = 2$		supers.		ord. $g = 1$			ord. $g = 2$		supers.	
Field:	163	83×3	41×5	83	41×3	307	103×3	191	97×3	47×5	97	47×3	457	157×3
σ	–	0.02	0.02	–	0.02	–	0.02	–	0.02	0.02	–	0.02	–	0.02
S	0.05	0.08	0.08	0.04	0.05	0.08	0.09	0.05	0.09	0.08	0.04	0.05	0.12	0.12
M	0.17	0.46	0.47	0.08	0.21	0.48	0.53	0.22	0.50	0.48	0.09	0.22	1.04	0.98
M_2	0.34	0.83	0.93	0.14	0.39	0.86	0.89	0.35	0.85	0.96	0.14	0.43	1.88	1.94
I	1.67	1.46	1.55	0.74	0.66	4.28	1.74	1.97	1.63	1.68	0.91	0.73	10.01	3.09
M_2/M	2.00	1.79	2.00	1.72	1.85	1.80	1.68	1.60	1.69	2.00	1.58	2.00	1.80	1.97
I/M	9.70	3.14	3.29	8.87	3.11	8.95	3.30	9.06	3.26	3.41	9.95	3.35	9.61	3.15

Table C.3: Comparison among operations in finite fields – PowerPC (64-bit)

C.2 Comparison Between Ordinary Curves and TZV

		80-bit security					96-bit security				
		$g=1$			$g=2$		$g=1$			$g=2$	
	Curve:	EC	TZV	TZV	HEC	TZV	EC	TZV	TZV	HEC	TZV
	Field:	163	83×3	41×5	83	41×3	191	97×3	47×5	97	47×3
operation	$-$	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01
	σ	–	0.01	0.01	–	0.01	–	0.01	0.01	–	0.01
	2	1.46	1.92	1.57	0.88	1.19	1.70	2.17	1.76	1.00	1.29
	$+$	1.46	1.92	1.57	1.85	3.24	1.70	2.17	1.76	2.07	3.57
affine	bin	451.4	519.6	468.9	331.7	474.4	621.3	683.9	639.2	445.6	591.5
	NAF/JSF	399.8	262.2	**166.5**	**283.2**	**246.1**	551.8	351.6	**199.9**	**373.2**	**309.0**
	w-NAF (4)	362.3	248.0	**129.7**	**240.9**	**216.7**	499.0	319.4	**157.8**	**317.6**	**270.4**
Lopez-Dahab	NAF/JSF	168.0	278.7	196.9	–	–	218.2	361.3	229.3	–	–
	w-NAF (4)	(3) **146.8**	245.6	134.5	–	–	(3) 191.8	306.5	162.0	–	–
	NAF/JSF	**167.0**	274.0	201.7	–	–	**214.0**	**329.0**	235.6	–	–
	w-NAF (4)	(3) 147.8	236.2	132.9	–	–	(3) **189.0**	**277.4**	159.9	–	–
	NAF/JSF	–	**255.1**	176.1	–	–	–	332.3	210.4	–	–
	w-NAF (4)	–	**233.8**	132.9	–	–	–	296.8	162.0	–	–

Table C.4: Comparison between ordinary curves and TZV – Intel (64-bit), timings in μs. Operations considered are negation ($-$), Frobenius endomorphism (σ), doubling (2) and addition ($+$). Scalar multiplication is performed in affine or Lopez-Dahab (original, extended and compressed) coordinates, without (bin, NAF/JSF) or with (w-NAF) precomputation. The scalar splitting technique is used for TZV, except for affine/bin. Refer to Section 4.2.4.

		80-bit security					96-bit security				
		$g=1$			$g=2$		$g=1$			$g=2$	
	Curve:	EC	TZV	TZV	HEC	TZV	EC	TZV	TZV	HEC	TZV
	Field:	163	83×3	41×5	83	41×3	191	97×3	47×5	97	47×3
operation	$-$	0.02	0.03	0.03	0.02	0.03	0.02	0.03	0.03	0.02	0.03
	σ	–	0.03	0.03	–	0.03	–	0.03	0.04	–	0.03
	2	3.13	3.18	3.46	1.34	2.34	3.75	4.08	3.61	1.67	2.49
	$+$	3.13	3.18	3.46	3.01	6.46	3.75	4.08	3.61	3.81	6.75
affine	bin	781.4	786.5	854.9	486.1	903.7	1085.5	1177.4	1014.1	594.7	1105.1
	NAF/JSF	712.6	**406.2**	**341.0**	**411.0**	**476.7**	982.6	**612.9**	**416.6**	**492.5**	**571.1**
	w-NAF (4)	645.7	**382.6**	**269.0**	**342.0**	**426.5**	888.1	**567.7**	**324.0**	**418.0**	**509.0**
Lopez-Dahab	NAF/JSF	426.1	472.4	413.1	–	–	558.7	722.6	507.0	–	–
	w-NAF(4)	426.1	(5) 408.6	288.2	–	–	475.3	612.9	345.0	–	–
	NAF/JSF	**424.1**	460.6	429.1	–	–	**550.4**	654.8	521.8	–	–
	w-NAF (4)	**360.3**	(5) 387.3	285.0	–	–	**475.3**	(5) 551.6	342.9	–	–
	NAF/JSF	–	432.2	379.4	–	–	–	654.8	462.9	–	–
	w-NAF (4)	–	394.4	288.2	–	–	–	587.1	345.0	–	–

Table C.5: Comparison between ordinary curves and TZV – PowerPC (32-bit)

		80-bit security					96-bit security				
		$g = 1$			$g = 2$		$g = 1$			$g = 2$	
	Curve:	EC	TZV	TZV	HEC	TZV	EC	TZV	TZV	HEC	TZV
	Field:	163	83×3	41×5	83	41×3	191	97×3	47×5	97	47×3
operation	$-$	0.02	0.02	0.02	0.02	0.03	0.02	0.02	0.02	0.02	0.03
	σ	–	0.02	0.02	–	0.03	–	0.02	0.03	–	0.03
	2	2.03	2.48	2.62	1.26	2.07	2.47	2.79	2.77	1.50	2.13
	$+$	2.03	2.48	2.62	2.36	5.15	2.47	2.79	2.77	2.62	5.37
affine	bin	515.2	628.2	635.6	413.6	763.1	713.0	809.7	770.0	360.6	918.7
	NAF/JSF	458.5	318.8	**259.4**	**350.6**	**387.3**	643.5	409.7	**324.0**	**305.1**	**464.3**
	w-NAF (4)	416.0	292.9	**208.1**	**299.2**	**349.3**	576.8	383.9	**248.3**	**267.1**	**409.0**
Lopez-Dahab	NAF/JSF	218.6	333.0	310.6	–	–	307.2	416.1	385.0	–	–
	w-NAF (4)	191.3	285.8	219.3	–	–	269.6	364.5	256.7	–	–
	NAF/JSF	**213.6**	328.3	317.0	–	–	**301.6**	**377.4**	393.4	–	–
	w-NAF (4)	**189.3**	278.7	214.5	–	–	**265.5**	**332.3**	254.6	–	–
	NAF/JSF	–	**302.3**	273.8	–	–	–	383.9	342.9	–	–
	w-NAF (4)	–	**274.0**	212.9	–	–	–	348.4	252.5	–	–

Table C.6: Comparison between ordinary curves and TZV – PowerPC (64-bit)

C.3 Comparison Between Supersingular Curves and TZV

		70-bit security		80-bit security		96-bit security	
	Curve:	EC	TZV	EC	TZV	EC	TZV
	Field:	239	79×3	307	103×3	457	157×3
operation	$-$	0.01	0.01	0.01	0.01	0.01	0.01
	σ	–	0.01	–	0.01	–	0.01
	2	0.11	0.16	0.18	0.19	0.25	0.29
	$+$	2.60	1.83	3.72	2.31	7.82	4.02
affine	bin	332.9	158.4	667.9	272.8	2251.4	498.3
	NAF/JSF	**235.0**	**83.5**	**459.6**	**145.5**	**1511.5**	**253.4**
	w-NAF (4)	**148.0**	**83.5**	**283.7**	**116.4**	(5) **851.2**	**211.1**
other	NAF/JSF	256.7	171.2	452.4	287.4	1710.4	489.9
	w-NAF (4)	198.0	143.4	344.7	(5) 225.5	(5) 1209.2	405.4
	NAF/JSF	372.1	190.5	484.7	309.2	1853.6	557.4
	w-NAF (4)	206.7	154.1	355.5	(5) 243.7	(5) 1257.0	439.2

Table C.7: Comparison between supersingular curves and TZV – Intel (64-bit), timings in μs. Operations considered are negation ($-$), Frobenius endomorphism (σ), doubling (2) and addition ($+$). Scalar multiplication is performed in affine or other (Jacobian and Lopez-Dahab) coordinates, without (bin, NAF/JSF) or with (w-NAF) precomputation. The scalar splitting technique is used for TZV, except for affine/bin. Refer to Section 4.2.6.

		70-bit security		80-bit security		96-bit security	
	Curve:	EC	TZV	EC	TZV	EC	TZV
	Field:	239	79×3	307	103×3	457	157×3
operation	$-$	0.02	0.03	0.02	0.03	0.03	0.03
	σ	–	0.03	–	0.03	–	0.04
	2	0.19	0.28	0.30	0.35	1.07	0.53
	$+$	6.03	2.89	10.74	3.85	25.68	8.52
affine	bin	644.0	235.4	1493.7	363.8	6157.5	929.1
	NAF/JSF	**450.4**	**128.4**	**1034.1**	**156.4**	**4343.7**	**498.3**
	w-NAF (4)	**287.2**	**109.1**	**628.4**	**134.6**	(5) **2521.9**	**413.9**
other	NAF/JSF	578.8	278.2	1353.7	327.4	4932.4	1140.2
	w-NAF (4)	428.6	229.0	983.8	276.5	(5) 3548.1	(5) 903.7
	NAF/JSF	633.2	308.2	1436.3	363.8	5179.0	1258.4
	w-NAF (4)	450.4	244.0	1044.9	301.9	(5) 3580.0	(5) 945.9

Table C.8: Comparison between supersingular curves and TZV – PowerPC (32-bit)

		70-bit security		80-bit security		96-bit security	
	Curve:	EC	TZV	EC	TZV	EC	TZV
	Field:	239	79×3	307	103×3	457	157×3
operation	$-$	0.02	0.02	0.02	0.02	0.02	0.03
	σ	–	0.02	–	0.02	–	0.03
	2	0.19	0.31	0.27	0.37	0.48	0.47
	$+$	3.30	2.39	5.31	2.95	12.31	5.31
affine	bin	374.2	214.0	764.8	229.2	2959.4	633.4
	NAF/JSF	**267.6**	**115.63**	**535.0**	**174.6**	**2052.5**	**329.4**
	w-NAF (4)	**178.4**	**98.4**	**348.3**	**156.4**	(5) **1177.4**	(3) **270.3**
other	NAF/JSF	313.3	222.6	624.8	341.9	2227.5	658.8
	w-NAF (4)	243.7	186.2	477.6	(5) 283.7	1591.1	(5) 532.1
	NAF/JSF	332.9	239.7	664.3	367.4	2330.9	726.4
	w-NAF (4)	248.0	194.7	488.3	(5) 298.3	(5) 1654.7	(5) 549.0

Table C.9: Comparison between supersingular curves and TZV – PowerPC (64-bit)

C.4 Comparison Among Pairings

		70-bit security		80-bit security		96-bit security	
		EC	TZV	EC	TZV	EC	TZV
Pairing	Loop Length	239	79×3	307	103×3	457	157×3
t_N	$N = O(q^2)$	489.4	496.9	831.6	766.2	2890.5	1893.5
η	q^3	463.2	683.2	790.9	1069.7	2797.8	2707.0
η_T	$2^{(3m+1)/2} - 1$	258.8	377.0	438.3	582.0	1494.2	1451.7
a_{opt}	$2^{(3m-1)/2}$	263.8	369.7	447.4	573.6	1512.7	1418.7
η (HLV)	q^3	519.8	729.0	847.7	1104.7	2910.3	2865.2
η_T (HLV)	$2^{(3m+1)/2} - 1$	245.9	356.5	411.1	553.1	1435.7	1382.6
t_{TZV}	$3 \times q$	–	614.1	–	968.3	–	2466.8
t_σ	$2 \times s$	–	791.0	–	1185.4	–	2962.1
t_{TZV} (Par)	$3 \times q$	–	269.9	–	385.8	–	956.1
$\mathsf{t}_{\mathcal{E}_3}$ (Par)	$2 \times O(q)$	–	289.1	–	423.6	–	1043.5

Table C.10: Comparison among pairings – Intel (64-bit), timings in μs. Refer to Section 4.3.5.

		70-bit security		80-bit security		96-bit security	
		EC	TZV	EC	TZV	EC	TZV
Pairing	Loop Length	239	79×3	307	103×3	457	157×3
t_N	$N = O(q^2)$	1148.2	782.9	2569.6	1340.5	8526.0	4325.4
η	q^3	1096.9	1080.5	2486.8	1871.8	8230.4	6240.4
η_T	$2^{(3m+1)/2} - 1$	619.8	601.1	1361.7	1022.6	4452.4	3311.6
a_{opt}	$2^{(3m-1)/2}$	633.8	585.1	1352.7	1003.2	4484.0	3290.7
η (HLV)	q^3	1196.9	1165.4	2728.4	1962.1	7787.7	6457.8
η_T (HLV)	$2^{(3m+1)/2} - 1$	583.5	567.1	1287.5	958.9	4219.4	3170.2
t_{TZV}	$3 \times q$	–	959.6	–	1716.1	–	5719.3
t_σ	$2 \times s$	–	1257.3	–	2081.9	–	6781.6
t_{TZV} (Par)	$3 \times q$	–	389.7	–	669.8	–	2154.0
$\mathsf{t}_{\mathcal{E}_3}$ (Par)	$2 \times O(q)$	–	446.7	–	745.9	–	2406.9

Table C.11: Comparison among pairings – PowerPC (32-bit)

		70-bit security		80-bit security		96-bit security	
		EC	TZV	EC	TZV	EC	TZV
Pairing	Loop Length	239	79×3	307	103×3	457	157×3
t_N	$N = O(q^2)$	613.9	581.3	1208.0	894.5	3654.1	2621.8
η	q^3	579.1	800.1	1133.0	1246.5	3511.3	3873.2
η_T	$2^{(3m+1)/2} - 1$	324.2	455.0	632.2	701.2	1916.6	2024.4
a_{opt}	$2^{(3m-1)/2}$	334.2	433.9	639.2	671.1	1928.8	1991.7
η (HLV)	q^3	569.9	840.2	1122.5	1263.5	3540.2	3661.9
η_T (HLV)	$2^{(3m+1)/2} - 1$	303.2	416.1	586.1	642.0	1788.7	1954.6
t_{TZV}	$3 \times q$	–	698.7	–	1095.4	–	3463.4
t_σ	$2 \times s$	–	930.1	–	1386.8	–	4065.0
t_{TZV} (Par)	$3 \times q$	–	290.3	–	445.6	–	1305.9
$\mathsf{t}_{\mathcal{E}_3}$ (Par)	$2 \times O(q)$	–	337.1	–	507.3	–	1486.6

Table C.12: Comparison among pairings – PowerPC (64-bit)