UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Escola Tècnica Superior d'Enginyeria
de Telecomunicació de Barcelona

telecos
BCN

Recent Applications of Pairing-based Cryptography

By

Danae Argyriadou

Universitat Politécnica de Catalunya

Supervised by:

Jorge Villar Santos

A thesis submitted for the Master degree of

Cybersecurity

Barcelona, July 2022

This thesis is submitted to the Telecommunications Engineering Department, Universitat Politécnica de Catalunya in fulfilment of the requirements for the Master's degree in Cybersecurity.

Danae Argyriadou, July 2022

Declaration

I hereby state that this thesis represents my own original study and implementation and that no sources beyond those mentioned were used.

I further attest that the contributions and inputs of outside persons and sources are included, recognized and properly cited.

Acknowledgements

I want to start by expressing my gratitude to Dr. Jorge Villar Santos, who served as my master's thesis supervisor, for his assistance and direction. He consistently let me develop this project on my own, yet he gave me direction when I needed it.

I would also like to thank my family for the support and encouragement throughout this year.

Abstract

Pairing based cryptography was born near year 2000 with the paper "A One Round Protocol for Tripartite Diffie-Hellman" by A. Joux [1]. Since then, hundreds of research papers have been published, offering a wide range of improvements of existing cryptographic primitives and a lot of new functionalities.
However, deployment of pairing-based cryptosystems is very limited. As an example, the open source library OpenSSL provides only a very limited support for elliptic curves, and no support for pairings at all.
Recent practical scenarios, mainly requiring very space-efficient and versatile signature schemes and compact zero-knowledge proofs, focused the interest of the community on pairing based cryptosystems, like aggregatable BLS signatures or SNARKs.

This project aims to reviewing the most recent protocols using pairing-based cryptography, and the state-of-the-art of existing implementations. As a byproduct, the project will consider the possibility of extending well-known libraries like OpenSSL to add some basic support for pairing-friendly elliptic curves and pairing computations.

Contents

List of Abbreviations

PBC	Pairing-Based Cryptography
BDH	Bilinear Diffie Hellman
IBE	Identity-Based Encryption
PKI	Public Key Infrastructure
CA	Certification Authority
PKG	Private Key Generator
RIBE	Revocable Identity-Based Encryption
PKE	Public Key Encryption
BLS	Boneh-Lynn-Shacham
ZK	Zero Knowledge
NIZK	Non-Interactive Zero Knowledge
zk-SNARK	Zero Knowledge Non-Interactive Argument of Knowledge
EC	Elliptic Curve
DLP	Discrete Logarith Problem
BN	Barreto-Naehrig
PSI	Private Set Intersection
IoT	Internet of Things
AES	Advanced Encryption Standard
DES	Data Encryption Standard
RSA	Ron Rivest, Adi Shamir, Leonard Adleman
SSL	Secure Socket Layer
W3C	World Wide Web Consortium
TCG	Trusted Computer Group

Introduction

Cryptography is a broad field of research with a significant connection and overlap between mathematics and computer science. Fundamentally, secure protocols are based on mathematical frameworks and proved to be secure within different type of complexity theoretical security models. An emerging area of such secure protocols is **pairing-based cryptography**; essentially defined over elliptic curves and bilinear maps, its fundamental concept being pairing functions that map pairs of points on an elliptic curve into a finite field. Pairings were first used in cryptography to attempt to solve the discrete logarithm problem in the group of points of some particular elliptic curves, by offering a reduction to the discrete logarithm in finite fields. However, pairings are presently thought to be one of the most ideal mathematical tools for designing secure and effective cryptographic protocols. Numerous advancements and studies in several branches of cryptography have resulted from these pairing arrangements.

To pair sets of points on an elliptic curve, one must perform operations on elliptic curves as well as compute and evaluate functions. All cryptographic pairings now in use are based on pairings on elliptic curves. It is crucial to maximize pairing computation efficiency for a specific security level, and extensive research has been conducted in this area.

Since pairing based cryptography was introduces near year 2000 by A. Joux ("A One Round Protocol for Tripartite Diffie-Hellman"), hundreds of research papers have been published, offering a wide range of improvements of existing cryptographic primitives and a lot of new functionalities.
However, deployment of pairing-based cryptosystems is very limited. As an example, the open source library OpenSSL provides only a very limited support for elliptic curves, and no support for pairings at all.
Recent practical scenarios, mainly requiring very space-efficient and versatile signature schemes and compact zero-knowledge proofs, focused the interest of the community on pairing based cryptosystems, like aggregatable BLS signatures or SNARKs.

This project aims to reviewing the most recent protocols using cryptography based on pairings, and the state-of-the-art of existing implementations. As a byproduct, the project will consider the possibility of extending well-known libraries like OpenSSL to add some basic support for pairing-friendly elliptic curves and pairing computations.

Some basic background knowledge is a requirement of the reader as we will not proceed into a review of the fundamentals of abstract algebra. Nevertheless, we introduce sufficient elliptic curve theory and bilinear pairings basics for cryptographic applications.

Gantt Chart

	February	March	April	May	June	July
Preparation of thesis proposal	🟦					
Analysis		🟦				
Research		🟦	🟦			
Testing			🟦			
Implementation				🟦	🟦	
Report					🟦	
Presentation						🟦

The thesis has been mainly supported by the following publications:

[1] https://research.nccgroup.com/2022/02/03/estimating-the-bit-security-of-pairing-friendly-curves/

For the implementation part:

[2] The open source library OpenPairing was taken from the following github repository: https://github.com/dfaranha/OpenPairing

Chapter 1

Pairing based Protocols

Since pairings had been brought into the research on cryptography, a wide range of protocols have been proposed for use in various contexts, including encryption, key settlement, and digital signatures. Development of primitives that cannot be built using other approaches (such as Identity Based Encryption) and construction of primitives in which pairings attempt to enhance their utility (e.g. Three-party key agreement) are the two categories into which the protocols may be divided. In this phase we can discuss some of the most significant bilinear pairing-based cryptographic techniques.

Let us first introduce some basic definitions in favour of the reader. A pairing is a map $\hat{e} : G_1 \times G_2 \rightarrow G_T$, where G_1, G_2, G_T are cyclic groups, for which the following properties hold:

1. Bilinearity: $a,b \in$: $\hat{e}(aP,bQ) = \hat{e}(P,Q)^{ab}$, where P,Q are generators of G_1 and G_2 respectively.
2. Non-degeneracy: $\hat{e}(P,Q) \neq 1$.
3. $\hat{e}$ is efficiently computable.

We will also describe the Discrete Logarithm problem, a significant concept in cryptography that is required for the next chapters. Let us consider that G is a cyclic group and g is a generator of G. The Discrete Logarithm problem is as follows: *for $h \in G$, find x such as $h=g^x$* . The difficulty of the problem relies on the groups that we select.

1.1 Key Agreement and Key Exchange

1.1.1 Three-party Key Exchange

Let us first introduce a brief description of the Bilinear Diffie-Hellman problem. Let G_1, G_2 be two cyclic groups of prime order m and let P be a generator of G_1. Let $\hat{e} : G_1 \times G_2$ be a bilinear map. The BDH problem in (G_1,G_2, $\hat{e}$) is the following:

Given (P, aP, bP, cP) for some a, b, c $\in Z^*_m$ *compute* $v \in G_2$ *such that* $v = \hat{e}(P,P)^{abc}$.
In practice, we make use of the Weil pairing, which will be described later, as the bilinear map.

The Diffie-Hellman Key Exchange protocol may be prolonged to three parties, however this will require more than one rounds. Cryptographer Antoine Joux proposed a technique using a bilinear pairing where only one round is required to set up a secret key among three parties. The protocol presupposes that all parties determined in advance two groups G_1 and G_2, an element $P \in G_1$ and a bilinear map $e : G_1 \times G_2 \rightarrow G_T$ .Given the difficulty of the Bilinear Diffie-Hellman problem in these groups, key agreement is accomplished with the procedures described below:

1. Alice generates a random positive integer *a*. Then she makes *aP* publicly known.
2. Now Bob also generates the positive integer *b* at random and the information he releases is: *bP*.
3. An outside party creates a random positive integer *c* and publishes *cP*.

Everyone can calculate $\hat{e}(P,P)^{abc}$ after the steps mentioned above, because bilinearity gives us the following known property:

$$\hat{e}(P,P)^{abc} = \hat{e}(aP,bP)^{c} = \hat{e}(bP,cP)^{a} = \hat{e}(aP,cP)^{b}$$

An eavesdropper only has access to the following information, which is an example of the BDH problem: *G1, GT,* $\hat{e}$*, P, aP, bP,* and *cP*. This shared secret, $\hat{e}(P,P)^{abc}$, cannot be retrieved without solving the BDH.

1.1.2 Identity-based Encryption

Everybody can encrypt a message using the public key in public key encryption methods, but only the party with access to the associated secret key will be able to decrypt and read the original message. However, the majority of the time, one wishes to deliver a message to an entity rather than a "public key." This is what public key infrastructures are meant to achieve (PKI). In essence, a PKI certificate's function is to establish a connection between a public key and its owner through a signature provided by one (or more) certified trustworthy certification authority (CA).However, the biggest issue with PKI is how complicated it is to deploy and manage. Another option is to use identity-based cryptography, an alternative method where the identity of the receiver substitutes the public key that is required to encrypt a message.

Identity-based encryption (IBE), which Adi Shamir first proposed in 1984, is likely the most well-known use of pairing-based cryptography. It makes use of public-key encryption, but the public key is really just a random string (for instance phone number, email address etc.). As long as the string indicates the user, it can be any string. Without any prior key exchange between them, parties to an Identity based encryption can cipher messages or verify signatures. Its principal benefit is that it eliminates the requirement for digital certificates that link public keys to the identities of the relevant users.

Review of cryptographic operations:

Identity-based cryptography depends on the Private Key Generator, a dependable third party (PKG). The PKG must create a public/private keypair (abbreviated pkPKG and skPKG) and make pkPKG accessible to consumers of its services before operation can start. The master public key and master private key are the two names for these keys. The following is how encryption and decryption work:

1. Alice prepares plaintext message *M* for Bob. To be able to encrypt M and acquire the encrypted message C, she uses Bob's identification ID1 and the PKG's public key pkPKG. Alice then transmits C to Bob. Alice

may encrypt a message for Bob without any prior collaboration or planning on his part because she already knew ID1 and pkPKG before starting the encryption procedure.

2. Bob gets ciphertext *C* from Alice. Nearly all architectures presumptively include plaintext instructions for contacting the Private Key Generator to acquire the private key needed to decipher C. Bob's private key is afterwards transmitted to him, over a protected connection after Bob authenticates with it, thus sending enough evidence that IDBob belongs to him. The PKG could send a nonce to the email address belonging to ID1, for example, and if it was successfully returned, it could be possible to reasonably conclude that the owner of ID1 was the one who contacted the PKG. A safe connection for getting Bob's private key was provided by the return of this nonce over an SSL hypertext link. Bob might need to physically show his identification in order to obtain skIDBob and acquire a higher level of assurance. To recover the text format of the message (M), Bob utilizes his private key to decipher C.

1.1.3 Applications of Identity-based Encryption

i. **Revocation of Public Keys:**

A public key is replaced with an identity string in Revocable Identity-based Encryption (RIBE), which also supports key revocation. RIBE is an extension of Identity-based Encryption. Since a certificate links a person's public key and identity, a credential of a user in a public key encryption (PKE) scheme can be revoked via a certificate revocation process. Given that an IBE system lacks a certificate, it is challenging to offer key revocation for an IBE scheme. A user's credential can be revoked using one of two methods: either directly, in which case a sender specifies a recipient set in ciphertext, or indirectly, in which case a trusted center periodically provides fresh (updated) keys for non-revoked users.

A predetermined expiration date is included in public key certificates. In an IBE system, the public key "bob@email.com / present year" can be used by Alice to encrypt emails addressed to Bob. By doing this, Bob can only use his private key for the remainder of the current year. Bob must request a new private key from the PKG once a year. As a result, we see annual private key expiration. Notably, Alice does not have to request a new certificate from Bob each time Bob updates his private key, in contrast to the traditional PKI. Alice does not need to communicate with any third party certificate directories in order to obtain Bob's daily public key, which is an intriguing property. Therefore, identity-based encryption is a very effective method for implementing transient public keys.

ii. Delegation of keys

Delegating decryption abilities is another use for IBE systems. Security systems that manage numerous of public keys can be made simpler by identity-based encryption. Rather than retaining a large database of public keys the system can either deduce these keys from the names of the users, or just utilize the numbers 1, . . . ,k as different public keys.

In the Revocable Delegated Identity-based encryption, a cloud server may be used as part of an encryption system to generate an update key for RIBE. A trusted center controls a revocation list and only makes use of a master private key to generate secret keys. Using a master update key it has obtained from the trusted center, a cloud server regularly generates an update key for the revocation list before broadcasting it to users who are not on the list of those whose access has been revoked. At this point, anybody can openly confirm that the update key generated by the cloud server is legitimate on the revocation list.

1.2 Signature Schemes

Digital signatures are a significant primitive in cryptography and are constantly being deployed in daily life. Pairings have provided numerous implementations regarding signatures. In this phase, some of the most well-known and widely used signature schemes will be introduced.

1.2.1 Boneh-Lynn-Shacham (BLS) signature scheme

A little before the BLS curve family was established, BLS signatures were proposed. In 2001, the suggestion of a short signature scheme based on bilinear pairings was introduced by Boneh, Lynn and Shacham (BLS) [26]. The most well-known application of the BLS digital signing system is the convergence of several signatures in order to reduce file size. We will first describe how the BLS signature scheme technique works.

Let's consider that H is a hash function and $\hat{e}$: $G \times G \rightarrow G_T$ is a bilinear pairing and G, G_T are cyclic groups. Also we will take γ to be a generator of G. An element in G is a signature σ. The BLS protocol follows the procedure below:
Generating the key : Calculate $V = \gamma^x$ using an integer x of your choice. The public key is $V \in G$, while the secret key is x.
Signing : Provided a private key x and a message M, calculate $Q = H(M) \in G$ and $\sigma = xQ$. The signature is $\sigma \in G$.

Verification : Given a public key $V \in G$, a message M and a signature $\sigma \in G$, calculate $H(M) \in G$ and verify that $(\gamma, V, H(M), \sigma)$ is a proper DH tuple. That stands if we confirm that: $\hat{e}(\sigma,\gamma) = \hat{e}(H(M),\gamma^x)$ If it is, generate result valid; otherwise, invalid.

Pairing friendly curves are used in BLS signature aggregation. It implements a hash function separate from the curve that hashes straight to the elliptic curve. The most straightforward method is to hash a message as usual and use the output as the first coordinate of a point. There exist two places with a positive and a negative y-coordinate for every valid x-coordinate (simply because since (x,y) is on the curve $y^2=x^3+\alpha x+\beta$ it follows that $(x,-y)$ is also on the curve). This indicates that our hash has an approximately 50/50 chance of locating two points for some *x* and a 50/50 chance of locating none. By adding a number to the message and incrementing it when it fails, we can attempt hashing any message multiple times in order to discover a point. We try H(m0), H(m1), H(m2) (where H is the hash function), and so forth until at last we discover a point that makes a match. Next, we decide which of the two matching points, let's say the one with smaller *y*, to use.

1.2.2 Properties

Threshold BLS Signatures. A threshold signature is a technique to generate a cryptographic signature by using a common private key that makes use of a distributed group of users. Usually, the parameters are specified so that n of k (for instance, 2 of 4) are necessary to produce a signature. Any fewer than n individuals are prohibited from learning about the shared secret key and from generating a signature, which is how the security is established. The difficulty of an attack is increased by dispersing the signature through different sites, requiring a minimum number of users to be affected in order to obtain the common secret key. Threshold BLS signatures are a rather simple combination of group operations and secret trading. The key generation procedure remains the same, with the variation of using Shamir’s secret sharing to divide the secret key into shares where a predefined number of them can recreate the private key. The signing process is pretty much identical to that of standard BLS signatures; however, one signs with the share of the secret key rather than the secret key itself. Once n signatures have been obtained, they can be merged into one signature that is protected by the secret key.

Signature Aggregation. It is possible to combine numerous signatures created using various public keys for various messages into a single signature. Therefore, instead of the 2n pairs you may anticipate to need, we would only need two pairings to verify a single message signed by n parties, or n+1 pairings to verify n separate messages signed by n parties. Given how expensive it is to compute pairings, this trait is crucial.

Unique and deterministic. There is only one legitimate signature for a specific key and message.

1.2.3 BLS Signature in Ethereum

Ethereum was first proposed in 2014 by Vitalik Buterin [27] and is now the second largest cryptocurrency around. To enable safe cryptography within the protocol, Ethereum uses the BLS signature technique. With this technique, communications can be signed by validators, and the resulting signatures are then collected and massively validated. The entirety of the set of (message, public key) pairings that the single signature represents can be validated simultaneously. In other words, if there exist: a set of private keys $x_1,\ldots,x_n$ (held by different users), the corresponding public keys $V_1,\ldots,V_n$, and messages $m_1,\ldots,m_n$ where m_i is signed by the corresponding x_i, and the signatures are $\sigma_i = x_i * H(V_i,m_i)$. The final step is to create the aggregated signature $S = \sigma_1 + \sigma_2 \ldots + \sigma_n$, which can be checked against the set of messages and public keys to ensure that it is a perfectly legitimate aggregate of signatures for those key and message combinations. S has a fixed size (often 32–96 bytes depending on setup).

Because of this, a comprehensive Proof-of-Stake system with a large number of validators can operate effectively in real-world settings.

1.2.4 Blind Signature Scheme

Applications where sender privacy is critical frequently utilize blind signatures. This includes numerous electronic payment systems and voting procedures, both of which place a high priority on anonymity. A blind signature protocol aims to make it possible for a user to receive a signature from a verifier without the verifier being aware of anything about the message it signed or for the user to acquire more than one valid signature from the signer in a single interaction. The Blind Signature Scheme is done as follows:

First, let *H* be a hash function. The secret key is $x \in Z_P^*$ and the shared key is *pubkey* = *xP*.

Blind signature Issuing protocol : Given secret key *x* and a message *m*:

1. (*Blinding*) The user picks an arbitrary element *r*, specifically a non zero integer , calculates $M' = r\,H(m)$ and transmits M' to the verifier.
2. (*Signing*) The verifier then calculates $\sigma' = x\,M'$ and sends back σ' to the user.
3. (*Unblinding*) The user finally calculates the signature $\sigma = r^{-1}\,\sigma'$and has as a result the (m, σ).

4. *(Verifying)* Provided the shared key pubkey, a message *m* and a signature *s*, confirm that: $\hat{e}(pubkey, H(m)) = \hat{e}(P, \sigma)$.

1.3 Zero-Knowledge (ZK)

Zero-Knowledge is a protocol that enables one party, referred to as the prover, to persuade another party, known as the verifier, that a specific statement is true without disclosing any additional data or evidence. It stands to reason that obtaining such an evidence that an assertion is true is equivalent to learning it from a reliable source. They are regarded as one of the basic cryptographic primitives. Although it was mostly a theoretical tool for a long time, it has been enhanced with new pairing-based cryptography approaches during the past ten years, making it an order of magnitude more effective. In modern blockchain applications, it serves as a practical verification of accurate computations, as well as in various use cases like online voting and authentication.

In mathematics, a proof is a series of congruent claims that are logically deduced using some principles from the axioms, premises, and conclusion. These arguments are regarded as immovable facts and are crucial for believing that a conclusion—like a theorem—is true. Real-world proofs have a dynamic interpretation; they are seen as a method for demonstrating the truth of a proposition. In both situations, a prover—an entity that offers the proof—and a verifier—an entity that carries out the shorter verification procedure—are involved. Zero-Knowledge Proofs were first introduced in 1989 by Goldwaser, Micali, and Rackoff [28]. The prover's objective, given a language L that describes a class of issues, is to persuade the verifier that a public assertion x is in this language without disclosing the witness's secret information. They exchange messages back and forth until the verifier is sure of the truth of the assertion, $x \in L$ or the opposite, at which point it chooses whether to accept or reject the evidence. Ideally, the verifier should accept if both parties are sincere and the relation is satisfied.

1.3.1 Fundamental Properties

Two key qualities are demanded from all zero knowledge proofs; *completeness* and *soundness*. If the assertion is true, as is indicated by the definition of completeness, the proof should be convincing to the verifier. When both parties are honest, this is the preferred behavior. The *soundness* property assures that deceitful witnesses cannot persuade the verifier of any false assertion. In addition to these properties, the *Zero Knowledge* quality is required, which ensures that no other information besides the legitimacy of the statement is revealed from the messages involved in the proof.

1.3.2 Pairing-based Non-Interactive Zero-Knowledge Proof

The initial Zero Knowledge proofs were presented as an interactive debate between the verifier and the prover, in which the verifier randomly selects items to pose questions to the prover and anticipates persuasive responses.On the contrary, this exchange of messages is replaced in non-interactive proofs by a single message from the prover to the verifier, which represents the proof and can be verified off-line by the verifier. Some non-interactive contracts only require the verifier to send one message from the provider to the verifier; others require the verifier to produce some setting details that can be made available to the public beforehand and separately from the assertion to be proved afterwards. In order to uphold safety and avoid fraud from the verifier, these setting details are typically produced by a reliable third party.

In the last decades, this field suffered a big change with the development of cryptography using pairings. The bilinear structure is very suitable to develop efficient constructions of NIZK proofs with efficient public verification.

1.3.3 zk-SNARKs

The idea of SNARK, or a *succinct non-interactive argument of knowledge*, is one of the more intriguing concepts in the family of non-interactive proofs for demonstrating the integrity of outcomes for complex computations. By this phrase, we refer to a proof scheme that is:
succinct: The size of the proof is very small compared to the size of the statement or the witness.

non-interactive: There is no need for the prover and the verifier to interchange claims in rounds.
argument: it is considered secure for provers with constrained computational resources, which means that provers with sufficient computing power may persuade the verifier that a statement was false.
knowledge-sound: Without knowing a specific so-called witness for the statement, the prover cannot create a proof; formally, for any prover able to produce a verifiable evidence, there is an extractor capable of extracting a witness ("the knowledge") for the statement.

A *zero-knowledge* property can be further added to the SNARK systems. This attribute makes it possible to carry out the proof without disclosing any information about the preliminary stages (the witness). These schemes are called **zk-SNARKs**.
A significant development in the realm of zero knowledge is represented by Zk-SNARKs. The combination of the succinctness property with the fact that they are defined for very general statements, makes them very useful to work in a variety of scenarios. The secret to their effectiveness lies in the fact that zk-SNARKs are not only succinct but also highly effective for communication and verification. Verifiable computation schemes involve a party delegating a computation to a party with more resources, who then receives the outcome of the computation and a zk-SNARK demonstrating the accuracy of the computation.

1.3.4 Zk-SNARKs in Blockchain

Zk-SNARKs have been implemented in the field of cryptocurrencies, like Zcash, Ethereum, Monero, where zk-SNARKs guarantee the correctness of the transactions, in the sense of preventing double-spending and offering anonymity. They are also implemented in smart contracts and anonymous identification systems.
Zcash is a cryptocurrency with the assurance of privacy for all users, and the first cryptocurrency to apply zero-knowledge tests to guarantee the security of

users. The privacy guarantee offered by Zcash depends heavily on its ability to fully encrypt every shielded transaction. In addition, given zkSNARKs and the network's consensus procedures, the shielded transactions are simple to verify. All of this may be done privately without disclosing the identity of the sender, the receiver, or the amount of the transfer. This kind of shielded transactions is the opposite of what we see in, for example, the Bitcoin blockchain, where sender and receiver addresses—as well as the value of a transaction—are available for anyone to see.
The Zero Knowledge protocol in **Ethereum** blockchain enables multiple parties to verify any computation, and the deployment of Zk-SNARKs makes this feasible rapidly. ZoKrates is a toolbox for zk-SNARKs on Ethereum that enables developers to build and validate zero-knowledge proofs using Solidity contracts. Some improvements to Ethereum's cryptography were introduced in the Byzantium hard fork, including curve addition, scalar multiplication, and pairing checks on the elliptic curve alt bn128 in order to carry out zk-SNARK verification.

1.3.5 Zk-SNARKs and pairings

The bilinear pairing that is utilized to protect the information is the zk-SNARKs engine. Pairings are unique maps that disguise data while still enabling you to perform some rudimentary arithmetic on it.

Verifying arithmetic
These pairs are utilized in zk-SNARKs as a means of monitoring that the arithmetic has been done correctly.
Lets suppose we have the quadratic equation x^2-x-42 = 0. We could convince someone that we know x by solving the equation and revealing the solution to them. Or we could keep the number x secret and use a pairing on an elliptic curve;
Let $\hat{e} : G_1 \times G_1 \rightarrow G$ be a symmetric pairing.
Notice that if $\hat{e}(G,G)^k = 1$, then k is either a 0 or a multiple of the order of the target group G. Thus if the following equation holds, then we can be sure the

quadratic equation is satisfied.

$$\hat{e}(G,G)^{x^2-x-42}$$

Pairing is a unique map built over elliptic curves. Generally, an elliptic curve is defined so that pairing is not efficiently computable since elliptic curve cryptography is broken if pairing is efficiently computable. As the significance of the pairing grows, elliptic curves where pairing is efficiently computable are researched and the special curves known as pairing-friendly curves are proposed.

Although in theory there are pairings for any elliptic curve, in practice there are curves whose pairings cannot be appropriately applied for cryptographic purposes. Associated to each elliptic curve, there is a parameter that can be calculated and is known as the embedding degree *k*. This embedding degree represents the difficulty of converting an elliptic curve system into a classical discrete logarithm system.

Using bilinearity the equation can be rewritten as:

$$\hat{e}(G,G)^{x^2}\hat{e}(G,G)^{-x}\,\hat{e}(G,G)^{-42} = 1$$

And further:

$$\hat{e}(xG,xG)\ \hat{e}(xG,-G)\ \hat{e}(G,-42G) = 1$$

Now, in order to check that our secret number satisfies the quadratic equation, one would just need to check the pairing equation above. Given a certain point G on the elliptic curve, G and –42G can be computed using elliptic curve arithmetic. Also, given *x*G (*x* cannot be figured out because of the elliptic curve discrete logarithm problem), one can compute the three necessary pairings and verify that they multiply up to 1, *without knowing the value of x*.

Chapter 2

Mathematical Preliminaries

Following the introduction of RSA cryptosystem, which was the first asymmetric public key cryptosystem, researchers started examining other mathematically based cryptographic alternatives other than algorithms based on factoring. Then, elliptic curve cryptography was suggested. Finite fields and elliptic curves are the cornerstones of pairing-based cryptography. In this chapter, we shortly introduce the fundamental concepts behind elliptic curves and pairings. It is assumed that the reader is familiar with basic abstract algebra.

2.1 A Brief Introduction to Elliptic Curves

2.1.1 Definition

An elliptic curve E over a field F is a curve given by an equation of the form:

$$y^2 + \alpha_1 xy + \alpha_3 y = x^3 + \alpha_2 x^2 + \alpha_4 x + \alpha_6$$

for some α_i that belong to the field F. This is referred to as the Weierstrass equation for an elliptic curve. We must identify what set α_i, x and y belong to. Usually, they will be taken to be elements of a field, for example, the real numbers $\mathbb{R}$, the complex numbers $\mathbb{C}$, one of the finite fields $F_p (= \mathbb{Z}_p)$ for a prime p, etc. If K is a field and $\alpha_i \in K$, then we say that E *is defined over K.*

Additionally, there is a necessary assumption that the discriminant

$$\Delta = 4A^3+27B^2$$

is nonzero.

Equivalently, the polynomial x^3+Ax+B has different roots. This guarantees the curve's nonsingularity, which means that the curve has no self-intersections. We also include an additional point, $\mathcal{O}$,that is a point "at infinity", so E is the set: $E = \{(x,y) : y^2 = x^3 + Ax + B\} \cup \{\mathcal{O}\}$. It is simplest to think of it as a point (∞,∞), or simply ∞, perched above the y-axis.

Note: We only provided one equation of an elliptic curve. One has to understand that an elliptic curve is an abstract object that can take many forms, a model given by a Weierstrass equation being one. (Other models would include for example the Hessian model: $x^3 + y^3 + z^3 = dxyz$)

2.1.2 Adding points on an Elliptic Curve

Let P,Q be two points on the elliptic curve E, L be the line connecting P and Q (tangent to E if P=Q), and R be the third point of intersection of L with E.

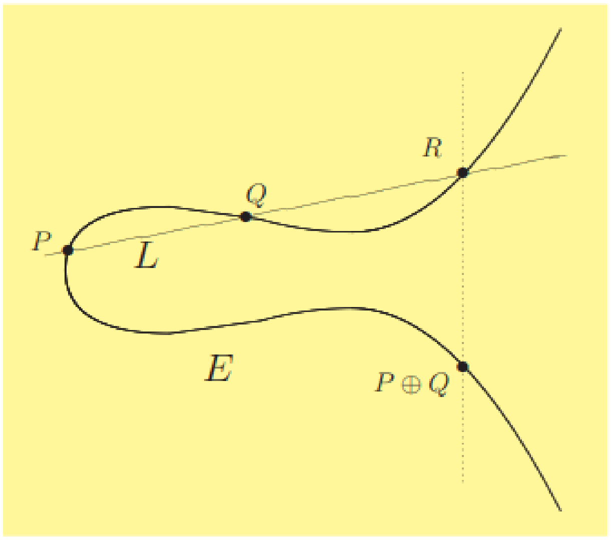


Figure 2.1: Addition of two points

We define the sum of P and Q on E to be the reflected point. It is denoted by P $\oplus$Q or just P + Q.

2.1.3 Vertical Lines and the Extra Point "At Infinity"

Let P ∈ E. We denote the reflected point by –P.

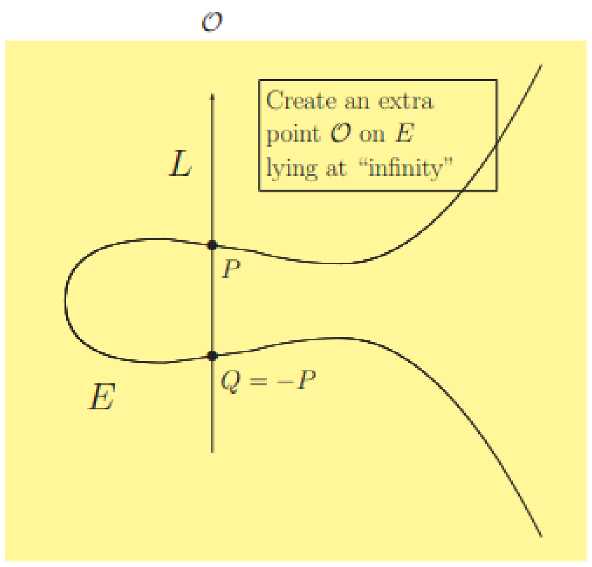

$\mathcal{O}$
Create an extra point $\mathcal{O}$ on E lying at "infinity"
L
P
$Q = -P$
E

Figure 2.2: Addition of two vertical points

We need a third point to define P + (-P). Since the vertical line L through P and –P does not intersect E in a third point, we create a point $\mathcal{O}$ at "Infinity".

Note: $\mathcal{O}$ is a point in every vertical line.

2.1.4 The Group Law

The addition of points on an Elliptic curve E satisfies the following properties:

i. (commutativity) $P_1+P_2 = P_2+P_1$ for all P_1 , P_2 on E.
ii. (existence of identity) $P + \mathcal{O} = P$ for all points P on E.
iii. (existence of inverses) Given P on E, there exists P' with $P + P' = \mathcal{O}$. This point P' will usually be denoted –P.
iv. (associativity) $(P_1 + P_2) + P_3 = P_1 + (P_2 + P_3)$ for all P_1,P_2,P_3 on E.

It follows that the points on E form an abelian group under addition with $\mathcal{O}$ being the identity element.

2.1.5 Divisors

Definition

Let E be an elliptic curve defined over a field K. A *divisor* D on E is a finite linear combination with integer coefficients:

$$D = \sum_j \alpha_j (P_j)$$

for $\alpha_j \in \mathbb{Z}$ and $\alpha \neq 0$ for only a finite number of points and P_j are projective points of the curve (some P_j could be points at infinity).
A divisor is therefore an element of the free abelian group generated by points on the curve. The group of divisors is denoted Div(E).
Define the *degree* of a divisor to be the sum of its coefficients:

$$deg(\sum_j \alpha_j (Pj)) = \sum_j \alpha_j \in \mathbb{Z}$$

2.1.6 Torsion Points on Elliptic Curves

Every point on an elliptic curve is one of two kinds: a point of *finite* order or a point of *infinite* order. For P to be a point of finite order means there exist a smallest integer n such that $nP = \mathcal{O}$. If no such n exists then P is of infinite order. To put it another way, P being of infinite order means you can never get the point at infinity by adding P to itself, no matter how many times you do it. The following derives from the distinction between finite and infinite points:

Definition : Let us consider an elliptic curve E over a field K. We take a point and add it *n* times and it gets mapped to the point at infinity $\mathcal{O}$. We define the following subgroup:

$$E[n] = \{ P \in E \,/\, nP = \mathcal{O} \}$$

E[n] is called the n torsion subgroup.

2.2 Introduction to Pairings

2.2.1 Definition

An *admissible bilinear pairing* is a mapping $\hat{e} : G_1 \times G_2 \rightarrow G_T$, where G_1, G_2, G_T are cyclic groups of large prime order r, that satisfies the following conditions:

- *Bilinearity:* for all $P,Q \in G_1$ and $R, S \in G_2$, $\hat{e}(P+Q, R) = \hat{e}(P,R) \cdot \hat{e}(Q,R)$ and $\hat{e}(P, R+S) = \hat{e}(P,R) \cdot \hat{e}(P,S)$.
- *Non-degeneracy:* $\hat{e}(P,R) \neq 1$ for some $P \in G_1$ and $R \in G_2$. Or, equivalently, $\hat{e}(P,R) = 1$ for all $R \in G_2$ if and only if $P = \mathcal{O}$; and $\hat{e}(P,R) = 1$ for all $P \in G_1$ if and only if $R = \mathcal{O}$. Non-degeneracy means that the mapping cannot be the trivial map which sends every pair of elements of G_1 and G_2 to the identity element of G_T.

It immediately follows that $\hat{e}(aP,bR)= \hat{e}(bP,aR)=e(P,R)^{ab}$, for any two integers a, b. All groups are of prime order, so consequently if P is a generator of G_1 and Q is a generator of G_2, then e(P,Q) is a generator of G_T. A mapping is said to be *computable* if an algorithm exists which can efficiently compute $\hat{e}(P,Q)$ for any $P,Q \in G_1$. If $G_1=G_2$ then the pairing is said to be *symmetric*. Otherwise, it is said to be *asymmetric.*

Let $\hat{e}$ be a bilinear, non-degenerate pairing and E an elliptic curve. Then for all $P,Q \in E$, and $a,b \in \mathbb{Z}$, it follows from the bilinear property that:

i. $\hat{e}(P, \mathcal{O}) = \hat{e}(\mathcal{O},P) = 1$
ii. $\hat{e}(-P,Q) = \hat{e}(P,-Q) = \hat{e}(P,Q)^{-1}$
iii. $\hat{e}(aP,Q) = \hat{e}(P,aQ) = \hat{e}(P,Q)^a$
iv. $\hat{e}(aP,bQ) = \hat{e}(P,Q)^{ab}$

2.2.2 Types of Pairings

There are several ways to describe a pairing, but the most efficient ones are defined when the groups G_1 and G_2 are elliptic curves and G_T is the multiplicative group of a finite field. Below are defined three types of pairings:

- When $G_1 = G_2$.;
- When $G_1 \neq G_2$ but an efficiently computable isomorphism $\varphi : G_2 \rightarrow G_1$ is known, while none is known in the opposite direction;
- When $G_1 \neq G_2$ and there appears to be no efficiently computable isomorphism known between G_1 and G_2, in either direction.

2.2.3 The Weil and Tate Pairings

The Weil and Tate pairing of algebraic curves were two earlier bilinear pairings that were deployed in cryptography for the MOV attack using Weil pairing and the FR attack using Tate pairing. Through these attacks, the Discrete Logarithm Problem in some elliptic curves was minimized to the Discrete Logarithm Problem in a finite field. Bilinear pairings have been effectively used in a range of cryptographic concepts to develop new cryptographic techniques in recent years.

Before describing the aforementioned pairings, at this point should be noted Miller's Algorithm, which was introduced by Victor Miller in 1986 in his unreleased work. Miller's Algorithm calculates the Weil pairing on an elliptic curve and has become ever since the cornerstone of pairing based cryptography.

The Weil and Tate pairings take r-torsion points as input, and in the case of the Weil pairing, both inputs are r-torsion points. They can be defined using rational functions.

The Weil Pairing

The Weil pairing on the n-torsion points is a major tool in the study of elliptic curves and it also has important applications in cryptography. Let E be an elliptic curve over a field K and let n be an integer not divisible by the characteristic of K. Then $E[n] \cong (Z/nZ)^2$. Let $\mu_n = \{x \in K \mid x^n = 1\}$ be the group of n-th roots of unity in K. It is a cyclic group of order n and any generator ζ of μ_n is called a primitive n-th root of unity.

Definition

The pairing

$$\hat{e}_n : E[n] \times E[n] \rightarrow \mu_n$$

is called the *Weil pairing*. It satisfies the following properties:

1. $\hat{e}_n$ is bilinear in each variable. This means that
   $$\hat{e}_n(S_1 + S_2, T) = \hat{e}_n(S_1, T)\, \hat{e}_n(S_2, T)$$
   and
   $$\hat{e}_n(S, T_1 + T_2) = \hat{e}_n(S, T_1)\, \hat{e}_n(S, T_2).$$

2. $\hat{e}_n$ is alternated: $\hat{e}_n(T, T) = 1$ for all $T \in E[n]$ and $\hat{e}_n(T, S) = \hat{e}_n(S, T)^{-1}$ for all $S, T \in E[n]$.

3. $\hat{e}_n$ is non-degenerate. This means that if $\hat{e}_n(S, T) = 1$ for all $T \in E[n]$ then $S = \mathcal{O}$ and also that if $\hat{e}_n(S, T) = 1$ for all $S \in E[n]$ then $T = \mathcal{O}$.

The Tate Pairing

The Tate pairing is quicker than the Weil pairing, not only because it only requires one application of Miller's algorithm as opposed to being used twice, but also because it allows a host of optimizations.

Definition

Let E be an elliptic curve over a finite field F_q. We write $\mathcal{O}_E$ for the point at

infinity on E. Let l be a positive integer which is co-prime to q. Let k be a positive integer such that the field F_{q^k} contains the lth roots of unity (in other words, $l \mid (q^k-1)$). Let $G = E(F_{q^k})$ and write $G[l]$ for the subgroup of points of order l, and G/lG for the quotient group.
Then the *Tate pairing* is a mapping:

$$< . , . > : G[l] \times G/lG \rightarrow F^*_{q^k} / (F^*_{q^k})^l \text{ .}$$

The quotient group on the right side can be thought of as the set of equivalence classes of $F^*_{q^k}$ under the equivalence relation $a \equiv b$ if and only if exists a $c \in F^*_{q^k}$ such that $a=bc^l$.

The Tate pairing satisfies the following properties:

1. (*Well defined*). $< \mathcal{O}, Q > = 1$, for all $Q \in G$ and $< P,Q > \in (F^*_{q^k})^l$ for all $P \in G[l]$ and all $Q \in lG$.
2. (*Non-degeneracy*). For each point $P \in G[l] - \{0\}$, there is some point $Q \in G$ such that $< P,Q > \notin (F^*_{q^k})^l$.
3. (*Bilinearity*). For any integer n, $< nP,Q > \equiv < P,nQ > \equiv < P,Q >^n$.

Chapter 3

Pairing friendly Curves

Pairing is a unique map built over elliptic curves. Generally, an elliptic curve is defined so that pairing is not efficiently computable since elliptic curve cryptography could be compromised if the pairing is efficiently computable. As the significance of the pairing grows, elliptic curves where pairing is efficiently computable are researched and the special curves known as pairing-friendly curves are proposed.

Although in theory there are pairings for any elliptic curve, in practice there are curves whose pairings cannot be appropriately applied for cryptographic purposes. Associated to each elliptic curve, there is a parameter that can be calculated and is known as the embedding degree *k*. This embedding degree represents the difficulty of converting an elliptic curve system into a classical discrete logarithm system.

3.1 Curve Selection

Definition Let E be an elliptic curve defined over $K = F_q$. Let $G \subseteq E(Fq)$ be a cyclic group of order r. Let k be the smallest positive integer such that $r \mid q^k - 1$. Then we say that the *embedding degree* of G is *k*.

The Weil and the Tate pairing can be employed if the embedding degree of the elliptic curve is *sufficiently small*. However, with overwhelming probability, the embedding degree of a randomly selected curve is excessively large. Hence, if we want to find curves suitable for pairing-based applications, we must take into consideration special categories of curves.
One of these special classes is the class of *supersingular elliptic curves*. The embedding degree of supersingular curves is less than or equal to 6, which is sufficiently small for efficient computation of the pairings. Moreover, supersingular curves have a rich structure, which makes the existence of

distortion maps possible. These distortion maps map a point on the curve to a linearly independent point of the same order. Hence, they can be used to modify the pairings such that they satisfy the strong non-degeneracy property, which often comes in handy in cryptographic applications. On the other hand, the rich structure potentially provides cryptanalysts with tools to attack cryptosystems defined on these curves.

3.2 Barreto-Naehrig (BN) Curves

Barreto and Naehrig devised a method in 2005 [29] to generate pairing-friendly elliptic curves over a prime field, with prime order and embedding degree $k = 12$. The equation of the curve is $E : y^2 = x^3 + b$, with $b \neq 0$. The trace of the curve, the curve order and the characteristic of F_p are parameterised this way respectively:

$$t(x) = 6x^2 + 1$$
$$n(x) = 36x^4 - 36x^3 + 18x^2 - 6x + 1$$
$$p(x) = 36x^4 - 36x^3 + 24x^2 - 6x + 1\ .$$

The integer x is the parameterization of the size of the curve. Such a curve represents a Barreto-Naehrig or BN curve. BN curves have embedding degree $k = 12$, therefore pairings are calculated over points in $E(F_{p^{12}})$.

3.2.1 Bit Security of BN-Curves

A BN-curve over a 256-bit prime field F_q has, being an elliptic curve, a 256-bit group attached to it, say of order N. The best known attacks take $\approx \sqrt{N}$ time, so this provides us 128-bits of protection from discrete logarithm attacks.

The curves also are of embedding degree 12. Thus, we can apply a pairing to map a discrete logarithm problem to Fp^{12}.Considering that $p \approx 2^{256}$, we know that $p^{12} \approx 2^{3072}$. Hence Fp^{12} is a 3072-bit finite field, and finding a solution to the DLP there should take around 2^{128} effort. This would mean that the curves give 128-bit security level. Although they did at the time BN-curves were proposed, subsequent attacks have demonstrated that finite fields of size

23072 no longer genuinely offer 128-bit security. The security level is thought to be roughly 110 bits, according to recent studies.

It is possible to construct a BN-curve that targets 128-bit security, by selecting a curve closer to 2^{384}. The larger group order, however, degrades the performance of cryptographic operations and decelerates computation time.

3.3 Barreto-Lynn-Scott (BLS) Curves

A class of pairing-friendly curves called BLS curves was introduced in 2002 [30]. A BLS curve, in contrast to BN curves, lacks a prime order but has a big parameterized prime r that divides its order, and the pairing is determined on the r-torsions points. The most popular embedding degrees for BLS curves are 12, 24, and 48, and they are a solid option for pairings at the high-security 256-bit level.

3.3.1 The BLS12-381 Curve

Early in 2017, curve BLS12-381 was created [22] as the framework for an improvement to the Zcash protocol. It works well for constructing zkSnarks and is pairing-friendly, making it useful for digital signatures. Short digital signatures that may be quickly aggregated or thresholded are valued highly by a number of blockchain systems. The preferred curve for these techniques is typically BLS12-381 due to its characteristics.

Naming. The curve's embedding degree is 12, which is just right—neither too low nor too high. The field modulus, or 381, is the amount of bits required to express coordinates on the curve. A prime number with a width of 381 bits makes up the finite field from which a point's coordinates are drawn.

Curve equation. BLS12-381 curve is given by the equation $y^2 = x^3 + 4$. A single parameter x (different from the x in the curve equation) that is chosen to provide the curve good implementation qualities is used to set the key parameters for a BLS curve.

Desired Properties. Design goals of the BLS12-381 curve are:

1. "low hamming weight" of x, suggesting that very few of its bits are set to 1. This is of great significance for the effectiveness of the Miller's Algorithm for pairing calculation.
2. The field modulus q stated above is prime and has 383 bits or less, which makes 64-bit or 32-bit arithmetic on it more functional.
3. The security target is 128 bits.

3.3.2 Bit Security of BLS12-381

BLS12-381 cannot be accurately advertised as offering 128-bit security. According to a report by the NCC Group, it is estimated to achieve between 117 and 120 bits, falling short of the initially stated target level of 128 bits. In particular, for 128-bit security with a BLS12 curve, the base field should have order of at least 460 bits. However, many assertions are supported that it is still substantially stronger than BN-254, which is already unbreakable by existing algorithms.

A BLS curve stating 128-bit security is discussed in report [14], which is a BLS12 curve over a 461-bit finite field. The security of BLS12-461 is calculated to be between 134 and 135 bits.

3.4. Implementations of Pairing-friendly Curves

At this point we briefly describe the pairing-friendly elliptic curves that are selected by existing standards, applications, and cryptographic libraries. *ISO/IEC* standard specifies public-key cryptographic techniques based on elliptic curves. It uses BN curves of the size of 256 for 100-bit security and of size 384 for security of 128 bits. *TCG* supports the BN curve of 256 bits and of 638 bits, while *W3C* standard organization adopts BN256, BN512 and BN638 curves proposed by TCG.
There are several cryptographic libraries that support pairing computations

using different elliptic curves. *PBC* is a famous library for pairings that supports BN curves, along with other pairing-friendly curves like MNT curves, Freeman curves. *MCL* is another library for pairing-based cryptography that uses four BN curves and BLS12-381. The BN curves within this library include BN254, BN_SNARK1, suitable for SNARK applications, and BN462. *RELIC* is a research-oriented library that uses various types of elliptic curves that include six BN curves (BN158, BN254R, BN256R, BN382R, BN446, BN638), where R is the relic parameter, which makes the curve different than, e.g. the BN254 of MCL library.
Several applications have adopted the use of pairing-friendly curves like BN and BLS curves. *Zcash* employs the BN128 curve in its library libsnark. After the proposal of the exTNFS algorithm for the discrete logarithm problem, an attack that affected many BN curves, Zcash proposed and published the use of the curve BLS12-381. *Ethereum* also supports the BLS12-381 and uses it for the implementations.

Chapter 4

Feasible Pairing-based Cryptography Applications A brief Review

Pairing-based cryptography has developed a wide range of intriguing applications during the past ten years, both in the field of cryptography and in computer/network security. If not always, it produces the most efficient but also most elegant methods. This chapter provides a brief overview of a few subjects where pairing-based approaches have either not been used or have not been fully utilized. The Internet of Things is the first area where pairs are practically applicable. Pairing-based encryption has some qualities that make it a desirable option for situations with limited resources, such as the Internet of Things. The second subject is "privacy-preserving set operations," which includes protocols for private set intersection (PSI). Despite considerable past work, state-of-the-art Private Set Intersection is based in less complex, non-pairing-based number theoretic environments.

4.1 Pairing-based Cryptography in IoT

One of the most common and well-known uses of pairings is in identity-based cryptography, as was previously discussed in the chapters. The main advantage of Identity-Based Cryptography is that it enables message encryption without the requirement of previously distributed keys. In IoT use scenarios where pre-distribution of keys is somewhat problematic and occasionally poses security problems, such a capability is appealing. For instance, when the same key is shared among all, the impairment of a single device can compromise the security of the entire network; or when a dedicated key is established for each couple of “things”, the solution cannot be scaled. Another advantage is that IBC provides the feature of including date information to the identity which entails revocation support without the usage of certificate revocation lists. It follows that pairings appear to be a promising solution for enabling cryptographic protocols to secure IoT devices. Previous work that has been carriet out includes using BN-254 and BN-256 elliptic curves to implement hardware on low power embedded devices. Implementations supporting the BLS12-381 curve, which provides strong security along with high computational complexity, have been proposed but have received very little attention.

4.2 Pairings in Private Set Intersection

Private Set Intersection (PSI) is a cryptographic protocol that involves two players, each of whom has a private set. Their aim is to compute the intersection of their respective sets, such that minimal information is revealed in the process. In other words, Alice and Bob should learn the elements (if any) common to both sets and nothing (or as little else as feasible) else. This can be a mutual process where, ideally, neither party has any advantage over the other. In another version of Private Set Intersection, called one-way PSI, the intersection of the two sets is revealed to Alice, however, Bob learns almost nothing. A PSI protocol has been described in the paper published in [25], where the initial message is unaffected by the set sizes and only two rounds are required within the protocol (cited as *laconic* protocol). Specifically, the technique presented was based on pairing friendly elliptic curves and the test outcomes of the experiment demonstrated that the aforementioned protocol's performance appeared to be better than earlier existing Private Set Intersection techniques.

Chapter 5

Implementation

The implementation of pairing-based cryptography is supported by a number of published cryptographic libraries. The PBC (Pairing-based Cryptography) library, a well-known library that performs the mathematical operations associated to pairings, is a free open source C library that was constructed on the GMP library. Despite being written in C, pairing calculation time is manageable thanks to GMP. Another pairing-based encryption package, MCL, is based on GMP and supports both the optimal Ate pairing over BN curves and curve BLS12-381.

5.1. Considering the Extension of OpenSSL

OpenSSL is an open source cryptographic library that provides implementations of the industry's best known and highest regarded algorithms, including encryption algorithms such as 3DES (sometimes known as 'Triple DES'), AES, RSA, RC4, to name some, as well as message digest algorithms and message authentication codes. OpenSSL also provides, even though in a limited extent, command line tools suitable for Elliptic Curve (EC) algorithms. The only elliptic curve algorithms supported are Elliptic Curve Diffie Hellman for key agreement and Elliptic Curve Digital Signature Algorithm for signing and verification operations. However, despite of the implementation of elliptic curves, there is presently no support for deployment of pairings and pairing computations within OpenSSL.

The final part of this thesis consists of discussing the possibility of extending OpenSSL cryptographic library so that pairing-based cryptography can be implemented.

5.2 OpenPairing Library

The methodology used to achieve the thesis's ultimate objective involved searching for appropriate libraries that enable pairing-based cryptography calculations and are compatible with OpenSSL.

Openpairing is an open source library developed in C language, that supports pairing implementation over a BarretoNaehrig curve, using OpenSSL as the arithmetic backend. After meticulous investigation, it emerged as the only library for the deployment of pairings within OpenSSL and appeared to be a good fit for the purpose of this project.

5.3 Openpairing Modifications

Openpairing is a free software, hence it is permissible to be redistributed and modified in accordance with the terms of GNU Lesser General Public Licence as published by the Free Software Foundation. The library provided via github appeared to contain errors that hindered it from being properly integrated within OpenSSL, and specifically with the OpenSSL crypto library (*libcrypto*). Also, the library is missing some include files that makes the build come to a failure and stops it from being compiled. As a result, some alterations had to be made in order to perfom the integration successfully.

The table as shown below includes the adjustments that were introduced on the github version of openpairing library.

File	Modification
Ec_lcl.h	Delete ec_lcl.h file
Op.h	1) include this header : obj_mac.h add structs from ec_lcl.h (for EC_GROUP, see "pairing_group_st" declaration) ec_method_st ec_extra_data_st ec_group_st ec_key_st ec_point_st 2) Copy op.h to directory: ./openssl1.0.1.u/include/openssl
Op_core.c	change include op header : #include <openssl/op.h>
Op_fp2.c	change include op header : #include <openssl/op.h> add missing return statement in line: 216, 287, 324, 358, 412, 480, 509, 549, 618, 650, 667
Op_fp5.c	change include op header : #include <openssl/op.h> add missing return statement in line: 227, 334, 421, 448, 468, 578, 666, 764
Op_fp12	change include op header : #include <openssl/op.h> add missing return statement in line: 157, 218, 259, 311, 343, 380, 427, 544, 676
Op_mac.c	change include op header : #include <openssl/op.h> add missing return statement in line: 208, 320, 386, 506, 636
Makefile	rewrite the makefile so it will

	be compatible with openssl config file

Makefile	add op folder in line 148 like this:
	
	`# dirs in crypto to build`
	`SDIRS=   \`
	`                    objects \`
	`md4 md5 sha mdc2 hmac ripemd whrlpool \`
	`des aes` **`op`** `rc2 rc4 idea bf cast camellia seed modes \`
	`bn ec rsa dsa ecdsa dh ecdh dso engine \`
	`buffer bio stack lhash rand err \`
	`evp asn1 pem x509 x509v3 conf txt_db pkcs7 pkcs12 comp ocsp ui krb5 \`
	`cms pqueue ts srp cmac`

Makefile.org	In line 146 perform the same as in Makefile

Config
 	
*** NOTICE: "config" NOT "configure"**
	add op in line 886 like this:
	
	`for i in aes bf camellia cast des dh dsa ec hmac idea md2 md5 mdc2 op rc2 rc4 rc5 ripemd rsa seed sha`
	`do`
	`  if [ ! -d crypto/$i ]`
	`  then`
	`options="$options no-$i"`
	`    fi`
	`        done`

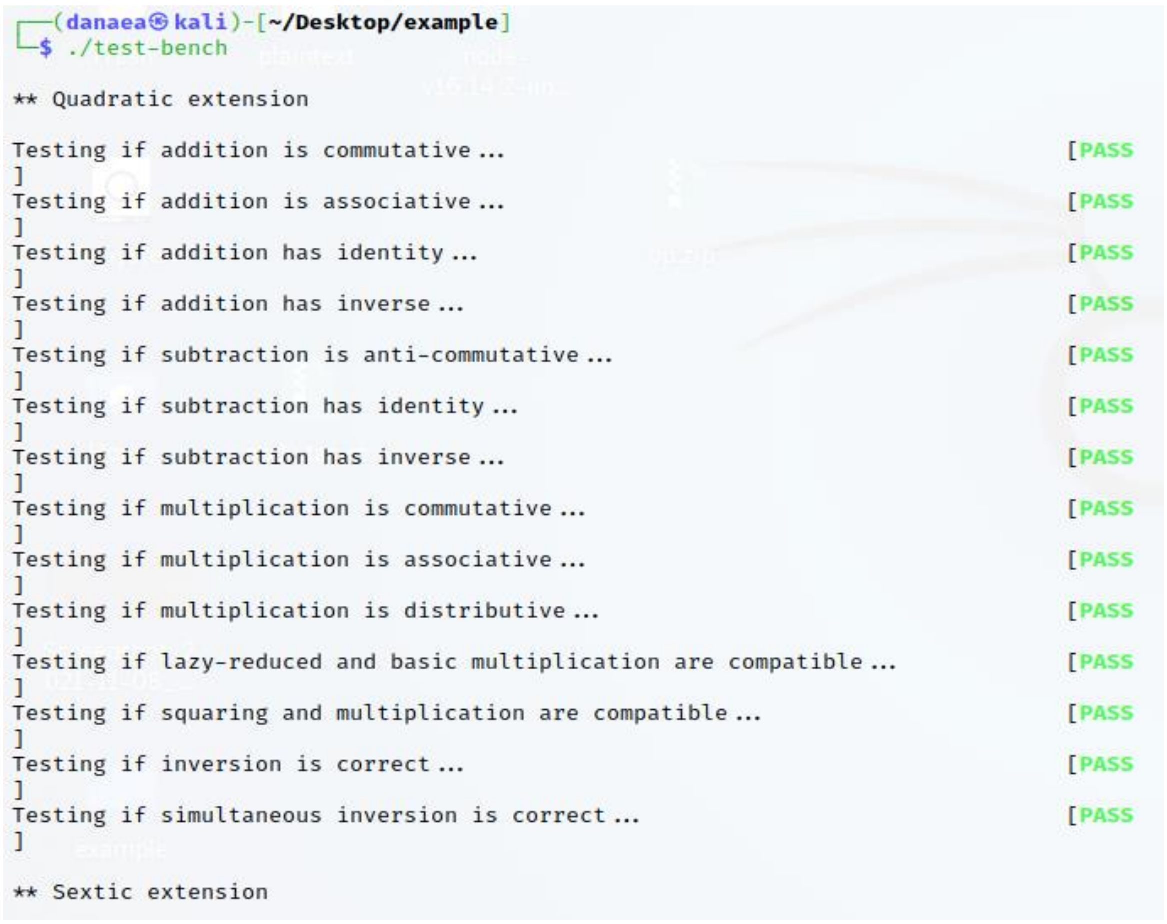
5.4 Results

In the github repository of OpenPairing, one may find all the files required to build and test the library. This includes the C files and their relevant header files required to declare constants, variables and functions. In order to trial the library, it is necessary to include in a folder (in our case the folder was randomly named *example*) the files mentioned below:

`Makefile, op_arch, op_bench(C file), op_bench (H file), op_bench (C file), op_test (H), op_test (C), test-bench (C)`

It is important to include file `Makefile` because in order to test the library, `test-bench` needs to be compiled with `Makefile`.

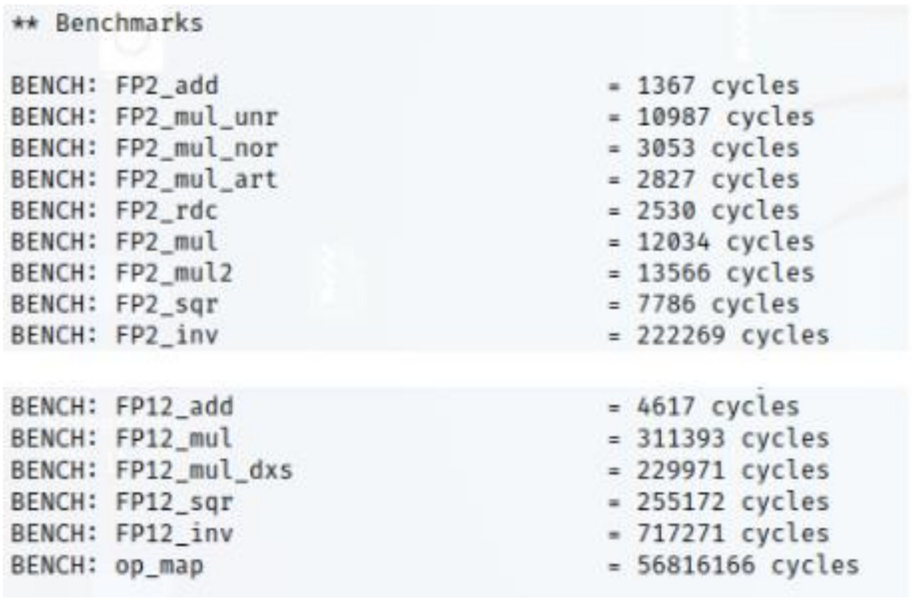
This precedes the execution of `make` command in linux terminal within this specific folder. When running the test-bench C file, the success or failure of the testing of the library is evident when the color of the string is green or red, respectively.



```
┌──(danaea㉿kali)-[~/Desktop/example]
└─$ ./test-bench

** Quadratic extension

Testing if addition is commutative ...                                  [PASS
]
Testing if addition is associative ...                                  [PASS
]
Testing if addition has identity ...                                    [PASS
]
Testing if addition has inverse ...                                     [PASS
]
Testing if subtraction is anti-commutative ...                          [PASS
]
Testing if subtraction has identity ...                                 [PASS
]
Testing if subtraction has inverse ...                                  [PASS
]
Testing if multiplication is commutative ...                            [PASS
]
Testing if multiplication is associative ...                            [PASS
]
Testing if multiplication is distributive ...                           [PASS
]
Testing if lazy-reduced and basic multiplication are compatible ...     [PASS
]
Testing if squaring and multiplication are compatible ...               [PASS
]
Testing if inversion is correct ...                                     [PASS
]
Testing if simultaneous inversion is correct ...                        [PASS
]

** Sextic extension
```

Figure 5.1: Indication of successful testing of the library

In Chapter 3, the reader became familiar with Barreto-Naehrig curves (BN) and their properties. A BN curve has embedding degree 12. Consequently, the pairings are computed in the elliptic curve built over the field Fp^{12}. As mentioned before, Openpairing library was constructed to implement pairings over a BN curve. Therefore we can see the computation cycles on the field Fp^{12} and also on Fp^2 (namely in Figure 5.2). This is because, for the information of the reader, a Barreto-Naehrig curve always have a twist that is difined over the finite field. (defining the definition of twist on an elliptic curve is beyond the scope of this project).



```
** Benchmarks

BENCH: FP2_add                                  = 1367 cycles
BENCH: FP2_mul_unr                              = 10987 cycles
BENCH: FP2_mul_nor                              = 3053 cycles
BENCH: FP2_mul_art                              = 2827 cycles
BENCH: FP2_rdc                                  = 2530 cycles
BENCH: FP2_mul                                  = 12034 cycles
BENCH: FP2_mul2                                 = 13566 cycles
BENCH: FP2_sqr                                  = 7786 cycles
BENCH: FP2_inv                                  = 222269 cycles

BENCH: FP12_add                                 = 4617 cycles
BENCH: FP12_mul                                 = 311393 cycles
BENCH: FP12_mul_dxs                             = 229971 cycles
BENCH: FP12_sqr                                 = 255172 cycles
BENCH: FP12_inv                                 = 717271 cycles
BENCH: op_map                                   = 56816166 cycles
```

Figure 5.2: Computation cycle counts

Fp^2 : finite field of the twisted curve
BN curves always have order 6 twists.
Fp^{12}: finite field of the elliptic curve

5.5 Limitations

Despite the fact that the integration of Openpairing library within OpenSSL has been successfully completed, there still exist some remaining flaws and limitations worth mentioning to the reader. Openpairing was built for OpenSSL version 1.0.1, so the principal step one has to do in order to put the library into function is to downgrade OpenSSL from current version to version 1.0.1u (download OpenSSL 1.0.1u). Attempts to incorporate openpairing within the newest version of OpenSSL were made, but compiling against the latest version had multiple errors as a result, therefore the testing of the library failed repeatedly.

An additional drawback to be noted here is that the openpairing library cannot be launched and controlled through the linux command line terminal. One would have to develop a file written in C language and run it via the command line, in order to utilize the functionalities and features of the library. To compile a file that employs openpairing, some specific steps that are listed below are required to be executed.

```
//============================================
#To compile an application that uses openpairing:
        -) include the directory in the application:
                #include <openssl/op>


        -) define CFLAGS in the applications's Makefile:
                -I/usr/local/test_pbc/include


        -) link the library:
                /usr/local/test_pbc/lib/libcrypto.a
```

Chapter 6

6.1 Conclusions

Pairing based cryptography can be considered as a subcategory of elliptic curve cryptography which has been an area of research for many years and has enabled the feasibility of numerous cryptographic schemes and protocols. It is characterized by a variety of favourable features and can offer a satisfying level of security, which is why pairing based cryptography began to get adopted commercially by companies and organizations. One might wonder why, in spite of having , pairings are not that as much widespread as other cryptosystems. The main reason lies behind the complexity of pairings whose computation can be challenging to comprehend for engineers. Additionally, the pairing based cryptography extensive group operations somewhat reduce the benefits of smaller key sizes. Realistically, considering these difficulties one could come to the conclusion that pairing based cryptography will likely not replace other asymmetric cryptosystems, like RSA or El Gamal, anytime soon.

The primary goal of this thesis was to examine a review of the most current implementations as well as the most well-known protocols using pairing-based cryptography. With that said, and having introduced sufficient background details and information for the reader, I introduced an extended version of OpenSSL, including openpairing library, that makes it feasible to implement pairings. However, a lot of work and research for supporting and deploying pairings is yet to be undertaken. After all, let us not forget the advantages that come with pairings and their implementation within cryptography; very small proof sizes (one proof can be as small as 128 bytes), fast verifier as one verification can be as fast as two pairing computations, solid standardization (BN curves, BLS curves, MNT curves), to name some of them. Pairings can generate finite fields that are broad enough to increase the hardness of the dicrete logarithm problem, yet small enough for feasible calculations. Thus, their study ought to be continued progressively.

Bibliography

[1] A. Joux, *"A One Round Protocol for Tripartite Diffie-Hellman"*, SCSSI 18, https://cgi.di.uoa.gr/~aggelos/crypto/page4/assets/joux-tripartite.pdf

[2] Lawrence C. Washington, *"Elliptic Curves, Number Theory And Cryptography"*, Second Edition, University of Maryland, 2008.

[3] N. El Mrabet, M. Joye, *"Guide to Pairing-based Cryptography"*, Chapman and Hall/CRC, Edition Nº1, June 2016.

[4] J. Groth, *"Short Pairing-based Non Interactive Zero-Knowledge Arguments"*, University College London, ASIACRYPT 2010.

[5] R. Barua, R. Dutta, P. Sarkar, *"Pairing Based Cryptographic Protocols : A Survey"*, Cryptology Research Group, India 700108. https://eprint.iacr.org/2004/064.pdf

[6] Ian F. Blake, G. Seroussi, and N. P. Smart, *"Elliptic curves in cryptography"*, Cambridge University Press, London Mathematics Society, Lecture Note Series 265.

[7] D. Freeman, M. Scott, and E. Teske, *"A taxonomy of pairing-friendly elliptic curves"*, Preprint, 2006.

[8] F. Hess, *"A note on the tate pairing of curves over finite fields"*, Arch. Math 82, 28-32 (2004).

[9] B. Waters, *"Efficient Identity-based Encryption without random oracles"*, Advances in Cryptology, https://eprint.iacr.org/2004/180.pdf

[10] J. H. Silverman, *"The arithmetic of elliptic curves"*, Springer-Verlag, Berlin, 1995.

[11] D. Jao, K. Yoshida, *"Boneh–Boyen signatures and the strong Diffie-Hellman problem"*, 2009, https://eprint.iacr.org/2009/221.pdf

[12] R. Sakai, M. Kasahara, *"Cryptosystems Based on Pairing over Elliptic Curves"*, SCIS 2003.

[13] R. Sakai, K. Ohgishi, M. Kasahara, *"Cryptosystems Based on Pairings"*, Symposium on Cryptography and Information Secu-rity-SCIS'00, 2000, Okinawa, Japan, 26-28.

[14] R. Barbulescu, S. Duquesne - *"Updating Key Size Estimations for Pairings"*, 2018, https://eprint.iacr.org/2017/334.pdf

[15] M. Barreto, M. Naehrig, *"Pairing-friendly elliptic curves of prime order"*, https://eprint.iacr.org/2005/133.pdf

[16] Utsav Banerjee, Anantha P. Chandrakasan, *"A Low Power BLS12-381 Pairing Crypto Processor for Internet of Things Security Applications"*, https://arxiv.org/pdf/2201.07496.pdf

[17]Lindermann,*"What are zk-SNARKS?"*, https://z.cash/technology/zksnarks/

[18] V. Cakulev, G. Sudaram, I. Broustis,*"Identity-based Authenticated Key Exchange"*, March 2012, https://www.rfc-editor.org/rfc/pdfrfc/rfc6539.txt.pdf

[19] M. Joye, A. Miyaji, A. Otsuka *"Pairing-based Cryptography – Pairing 2010"*, 4th International Conference, LNCS 6487, Japan, 2010.

[20] A. Nitulescu, *"zk-SNARKS; A gentle Introduction"* https://www.di.ens.fr/~nitulesc/files/Survey-SNARKs.pdf

[21] J. Groth, R. Ostrovsky, A. Sahai, *"New Techniques for Non-Interactive Zero Knowledge"*, Journal of the ACM, 59(3), 2012.

[22] Sean Bowe, *"BLS12-381: New zk-SNARK Elliptic Curve Construction"*, 2017 https://electriccoin.co/blog/new-snark-curve/

[23] A. Menezes, P. Sarkar, S. Singh, *"Challenges with Assesing the Impact of NFS Advances on the Security of Pairing-based Cryptography"* https://eprint.iacr.org/2016/1102.pdf

[24] I. Karantaidou, T. Halkidis, L. Mamatas, S. Petridou, G. Stephanides, *"Pairing-based Cryptography on the Internet of Things; a Feasibility study"*, https://hal.inria.fr/hal-02269719/document

[25] D. Aranha, C. Lin, C. Orlandi, M. Simkin, *"Laconic Private-Set Intersection from Pairings"*, https://eprint.iacr.org/2022/529.pdf

[26] D. Boneh, B. Lynn, H. Shacham, *"Short Signatures from the Weil Pairing"*, Advances in Cryptology, ASIACRYPT '01, LNCS2248, pp 514-532, Springer-Verlag, 2001.

[27] Vitalik Buterin, *"Ethereum: A Next-Generation Smart Contract and Decentralized Application platform"*, 2014. https://ethereum.org/669c9e2e2027310b6b3cdce6e1c52962/Ethereum_Whitepaper_-_Buterin_2014.pdf

[28] S. Goldwasser, S. Micali, C. Rackoff, *"The Knowledge Complexity of Interactive Proof Systems"* , Vol.18, No. 1, pp 186-208, February 1989.

[29] P. Barreto, M. Naehrig, *"Pairing-friendly Elliptic Curves of Prime Order"* https://eprint.iacr.org/2005/133.pdf

[30] P. Barreto, P. Lynn, M. Scott, *"Constructing Elliptic Curves with Prescribed Embedding Degrees"*, Security in Communication Networks pp 257-267, DOI 10.1007/3-540-36413-7_19, 2003.

[31] R. Sakai, K. Ohgishi, and M. Kasahara, *"Cryptosystems based on pairings"*, The 2000 Symposium on Cryptography and Information Security, Okinawa, Japan, 2000.

[32] J. H. Silvermann, *"An Introduction to the Theory of Elliptic Curves"*, Brown University and NTRU Cryptosystems Inc., July 2006.