ELLIPTIC CURVES: IDENTITY-BASED SIGNING AND QUANTUM ARITHMETIC

by

Parshuram Budhathoki

A Dissertation Submitted to the Faculty of

The Charles E. Schmidt College of Science

in Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy

Florida Atlantic University

Boca Raton, FL

August 2014

ELLIPTIC CURVES: IDENTITY-BASED SIGNING AND QUANTUM ARITHMETIC

by

Parshuram Budhathoki

This dissertation was prepared under the direction of the candidate's dissertation co-advisors, Dr. Rainer Steinwandt, Department of Mathematical Sciences, and Dr. Thomas Eisenbarth, Department of Electrical and Computer Engineering, WPI, and has been approved by the members of his supervisory committee. It was submitted to the faculty of the Charles E. Schmidt college of Science and was accepted in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

SUPERVISORY COMMITTEE:

Rainer Steinwandt, Ph.D.
Dissertation Co-Advisor

Thomas Eisenbarth, Ph.D.
Dissertation Co-Advisor

Lee Klingler, Ph.D.
Chair, Department of Mathematical Sciences

Lee Klingler, Ph.D.

Spyros Magliveras, Ph.D.

Russell Ivy, Ph.D.
Interim Dean, The Charles E. Schmidt College of Science

Deborah L. Floyd, Ed.D.
Interim Dean, Graduate College

6/24/14
Date

ACKNOWLEDGEMENTS

To me, Ph.D was not an easy journey, and the most important person with me in this journey was my advisor, Dr. Rainer Steinwandt. I would never have been able to finish this dissertation without his guidance, support, and patience. I would like to express my deepest gratitude to him for giving me such an inspiring learning environment. In addition to the research and other academic work, he was always there to polish my written and verbal communication skills. I will always be in debt for all of his help.

I would like to thank my next advisor, Dr. Thomas Eisenbarth, who introduced Cryptography to me. Though he was physically far from me, he was always available to give me feedback in my academic and personal pursuits.

Similarly, I am grateful to my supervisory committee members: Professor Lee Klingler and Spyros Magliveras, for their helpful suggestions and guidance throughout my entire time at the Department of Mathematical Sciences.

I would like to express my sincere thanks to the Department of Mathematical Sciences at Florida Atlantic University for accepting me in the graduate program and giving me an opportunity to know myself as a student of Mathematics. I am very thankful to the former graduate director, Professor William Kalies for his help and support in making my earlier days easy in the world of Mathematics. He gave me an opportunity to interact with an excellent learning environment, and his classes turned on the light in my mind. I would like to thank, Professor Stephen Locke for helpful discussion on graph coloring and Brittanney Amento for giving her Python code to

generate quantum circuit for multiplication. Big thanks to Beth and Emily for their unconditional help and support. I also would like to thank Helen for her reminder emails about time sensitive official deadline. I am really thankful to all my office mates, classmates, graduate students and faculty in the Department who contributed in my learning process directly and indirectly.

I am very grateful to all of our Nepali friends in Boca Raton, who helped my wife and me by treating us as their family members. Thank you all for being awesome companions. My special thanks go to Tulasi, who polished my writing.

I would like to thank my five sisters and my parents, Aama Mithu and Buwa Dik Bahadur Budhathoki, for all of sacrifices that they have made for me. I am truly in debt to all of my family members for their love and support.

Finally, my utmost sincere and loving thanks go to my dear wife Anju. She has earned more appreciation than I can ever offer, for her care, inspiration, friendship, and tolerance during my graduate study.

ABSTRACT

Author:	Parshuram Budhathoki
Title:	Elliptic curves: identity-based signing and quantum arithmetic
Institution:	Florida Atlantic University
Dissertation Co-Advisors:	Dr. Rainer Steinwandt Dr. Thomas Eisenbarth
Degree:	Doctor of Philosophy
Year:	2014

Pairing-friendly curves and elliptic curves with a trapdoor for the discrete logarithm problem are versatile tools in the design of cryptographic protocols. We show that curves having both properties enable a deterministic identity-based signing with "short" signatures in the random oracle model. At PKC 2003, Choon and Cheon proposed an identity-based signature scheme along with a provable security reduction. We propose a modification of their scheme with several performance benefits. In addition to faster signing, for batch signing the signature size can be reduced, and if multiple signatures for the same identity need to be verified, the verification can be accelerated. Neither the signing nor the verification algorithm rely on the availability of a (pseudo)random generator, and we give a provable security reduction in the random oracle model to the (ℓ-)Strong Diffie-Hellman problem.

Implementing the group arithmetic is a cost-critical task when designing quantum circuits for Shor's algorithm to solve the discrete logarithm problem. We introduce a tool for the automatic generation of addition circuits for ordinary binary elliptic

curves, a prominent platform group for digital signatures. Our `Python` software generates circuit descriptions that, without increasing the number of qubits or T-depth, involve less than 39% of the number of T-gates in the best previous construction. The software also optimizes the (CNOT) depth for $\mathbb{F}_2$-linear operations by means of suitable graph colorings.

DEDICATION

To my parents, sisters and wife.

ELLIPTIC CURVES: IDENTITY-BASED SIGNING AND QUANTUM ARITHMETIC

LIST OF FIGURES

Chapter 1

Introduction

In early days, cryptography was basically concerned with converting messages into unreadable form to protect message content during the time the message being carried from one place to another. Nowadays, its application has grown. Cryptography, which is also called cryptology, at its core, deals with protecting information from unauthorized access. Besides providing confidentiality, cryptography offers techniques for ensuring integrity and authenticity. One of the central tools for achieving the latter is the idea of a digital signature. Basically, a digital signature is a way to ensure that an electronic document is authentic. Authentic means that we know who created the document and we know that it has not been altered in any way since that person created it. In various practical applications, such as e-commerce, online banking, and transmission of sensitive digital data, where security of the information and transfer times are considered vital, cryptography has become an important tool to use.

The process of locking up information using cryptography is called encryption and the locked up message is called encrypted message or ciphertext. The process of unlocking the encrypted message is called decryption. The secret, like a password, which is used to lock (or unlock) the message (or encrypted message) is called key. If a sender and receiver use the same key to lock (or unlock) the message (or encrypted message), then that is called symmetric key cryptography (which is also called private

key cryptography), if they use two different types of key for the locking (or unlocking) process then it is called asymmetric key cryptography (which is also called public key cryptography).

Public key cryptography is concerned with cryptographic algorithms that require two separate keys, one being private and another being public. Public-key algorithms build on the perceived hardness of certain algorithmic problems, such as factorization and the discrete logarithm problem (DLP). Factorization means finding the decomposition of a composite number into smaller non-trivial divisors, which when multiplied together equal the original number. To understand the DLP, in general, take a group G, a finite order element g of G, $k \in \{1, \ldots, \text{order of } g\}$ and $h = g^k$, then for given g and h, finding a value of k is called DLP [cf. Section 2.1].

In 1977, Rivest et al. [52] proposed one of the first practical public key cryptosystems, which is also called the RSA cryptosystem. RSA-based encryption is one of the most commonly used techniques for public-key encryption. In this system, a user creates and then publishes the product of two large prime numbers, along with an auxiliary value, as their public key. The prime factors must be kept secret. Anyone can use the public key to encrypt a message, but with currently published methods, if the public key is large enough, only someone with knowledge of the prime factors can feasibly unlock the encrypted message. Breaking RSA encryption is known as the RSA problem. It is an open question whether it is as hard as the factoring problem. The US National Institute of Standards and Technology (NIST) has recommended 2048-bit or stronger keys for RSA [43]. Because of the large key length, RSA slows down the system.

Elliptic Curve Cryptography (ECC) [19] explores public key cryptosystems that are based on the algebraic structure of elliptic curves over finite fields. Finding the DLP of a random elliptic curve element with respect to a publicly known base point

is considered computationally infeasible when the curve is properly chosen. This forms the basis for elliptic curve-based protocols. Due to small key sizes, reduced transmission and storage requirements, and efficient computation features, ECC has become very popular in cryptographic applications today. To achieve the same level of security as with 3072-bits parameters in RSA, 256-bits parameters in ECC are considered sufficient [44], for details see Section 2.2. The National Security Agency has decided to move to elliptic curve based public key cryptography to protect both classified and unclassified National Security information [1]. So, designing a protocol in ECC and its cryptanalysis are active areas of research today.

Identity-based cryptography (IBC) [33] is an extension of the public-key paradigm, which was initially proposed by Adi Shamir [57] at CRYPTO'84. An identity of a user, such as physical address, name, phone number or IP addresses can be used as public key for encryption or signature verification in IBC. Thus, IBC significantly reduces the system complexity and the cost for establishing and managing the public key authentication [cf. Section2.5]. Although Shamir [57] easily constructed an Identity Based Signature (IBS) scheme using the existing RSA function, he was unable to construct an Identity-based encryption (IBE) scheme. IBE was later solved independently by Boneh & Franklin [12] and Cocks [18]. Boneh & Franklin first used a pairing in IBC. Pairing-based cryptography (PBC) is the use of a pairing between elements of two groups to a third group to construct cryptographic systems [cf. Section 2.4]. Sometimes a pairing can be used to reduce a hard problem in one group to a more feasible problem in a different group, such as the Menezes-Okamoto-Vanstone (MOV) attack using the pairing [40]. Therefore PBC is an active area of research today [7, 15, 17, 29, 48, 56].

For classical computers the discrete logarithmic problem is assumed to be infeasible for suitably chosen cyclic groups. However, Shor's algorithms [58] offers an

efficient solution on a quantum computer. One of the critical tasks in implementing quantum algorithms is the identification of efficient quantum circuits that capture the task at hand. This is an area with significant research activity [3, 4, 34, 38, 55].

1.1 OUR CONTRIBUTION

In this thesis, there are two contributions to ECC. We propose a cryptographic scheme based on suitable elliptic curves and we offer quantum circuits that contribute to the cryptanalyis of elliptic curve based schemes.

In Chapter 3, a signature scheme with security proof is introduced. Our focus is on the first short identity-based signature scheme in the random oracle model, which assumes the availability of a pairing-friendly group with discrete logarithm trapdoor [cf. Section 2.6]. That means a group where pairing computation is easy, but DLP is only easy to someone with the knowledge of some secret information of the group. Our signature scheme brings data integrity, authenticity, and non-repudiation, by using elliptic curves over a finite field. To design this protocol, we have used a pairing, which is a bilinear map. One of the most expensive operations in the proposed scheme is the evaluation of pairing. Scalar multiplication on the elliptic curve is assumed to be the next expensive operation. Our signature scheme is the first short identity-based signature scheme. Unlike other signature schemes, it needs only a single group element for representing a signature, fewer pairing operations for repeated verification of messages by the same signer, and a single scalar multiplication operation for each signing and verification process. Our scheme builds on the strong Diffie-Hellman assumption (SDH). To understand this assumption let g be an element of prime order p in an abelian group and $\alpha \in \mathbb{Z}_p$ (integer modulo p). Then, for given ℓ terms $g, g^2, \ldots, g^\alpha$ to find $g^{\alpha+1}$ is called ℓ-SDH problem.

When solving the discrete logarithm problem on a quantum computer, one faces the task of expressing the pertinent computation as a quantum circuit. For an efficient quantum circuit, only a small number of gates and qubits should be needed, and the circuit should have a small depth. On a more fine-grained level, one also should distinguish between the types of gates used. To solve the discrete logarithmic problem using Shor's algorithm, one important subtask that occurs in a typical implementation is the addition of a fixed group element to an arbitrary group element from the elliptic curve. To compute one such addition operation, we need to perform various operations in a finite field, such as squaring, multiplication by a constant, and multiplication.

In Chapter 4, we introduce a software implementation for generating a complete quantum circuit for the addition operation by using Python. This program takes an irreducible polynomial to fix the representation of the underlying finite field and an elliptic curve along with a generator of a subgroup as inputs, and it gives a complete circuit as output. This circuit uses optimized circuits for the squaring and the multiplication by constant operations.

1.2 OUTLINE

In Chapter 2, we review some basic cryptographic tools, such as discrete logarithm problem, elliptic curve cryptography, digital signature schemes, pairings, strong Diffie-Hellman assumptions, identity based cryptography, trapdoor for discrete logarithmic problem, hash functions, random oracle, Shor's algorithm, and quantum arithmetics. Thereafter, in Chapter 3, we present a deterministic identity-based signature scheme in the random oracle model, which assumes the availability of a pairing-friendly group with discrete logarithm trapdoor. This identity-based signature scheme affords "short" signatures in the sense that the signature consists of a single group element.

While it is known how to create such a signature in the public key setting [13], no such construction is known in the identity-based setting. Moreover, the number of group and pairing operations in the described scheme compares favorably to existing schemes, and the verification cost can be reduced when verifying multiple messages that are presumably signed by the same identity. The security reduction relies on the strong Diffie-Hellman assumption, which comes at the usual cost: Brown and Gallant [14], Cheon [16], and by Jao and Yoshida [31] indicate that for many groups the strong Diffie-Hellman problem is easier to solve than the discrete logarithm problem.

In Chapter 4, we show how a quantum circuit for certain arithmetic tasks can be generated automatically. Section 4.1 describes the choice of a suitable (polynomial-basis) representation of the underlying finite field and shows how edge colorings can be used to find efficient circuits for squaring, constant multiplication, and square root computation. In Section 4.2, we combine such circuits with Al-Daoud et al.'s addition formula for ordinary binary elliptic curves to derive a new quantum circuit for point addition with improved T-gate complexity. Complementing the theoretical discussion, we discuss concrete examples of circuits that have been synthesized with our software. Chapter 5 concludes the thesis.

Chapter 2

Preliminaries

This Chapter discusses the necessary cryptographic primitives to understand the upcoming chapters. We review discrete logarithm problem, elliptic curve cryptography, digital signature schemes, pairings, strong Diffie-Hellman assumptions, identity based cryptography, trapdoor for discrete logarithmic problem, hash functions, random oracle, Shor's algorithm, and finally quantum arithmetic. Our discussions are based on: *Handbook of Elliptic and Hyperelliptic Curve Cryptography* [19], *Introduction to Modern Cryptography* [36], *Handbook of Applied Cryptography* [41], *Identity-Based Cryptography* [33], *Understanding Cryptography* [49], *Elliptic Curves and Their Application to Cryptography* [23], *Quantum Computing* [28] and numerous papers on the topics.

2.1 DISCRETE LOGARITHM PROBLEM

The DLP has been extensively studied since the discovery of public-key cryptography in 1975. Let us assume that $\mathbb{G}$ is an additive cyclic group of order n. If P is a generator of $\mathbb{G}$, then generally we write $\mathbb{G} = \langle P \rangle$. For given P and $Q = r \cdot P$, where $r \in [1, n-1]$, the DLP is the problem of finding the value of r.

The DLP was first proposed as a hard problem in cryptography by Diffie and Hellman [22]. Since then, together with factorization, it has become one of the two

major pillars of public key cryptography. It is believed to be intractable for certain (carefully chosen) groups including the certain multiplicative groups of a finite field, and suitable subgroups of elliptic curves defined over a finite field.

2.2 ELLIPTIC CURVE CRYPTOGRAPHY

In 1985, Neal Koblitz [35] and Victor Miller [42] independently proposed ECC, which is a method to realize public-key cryptography based on the mathematics of elliptic curves over finite fields. For ECC, it is assumed that finding the DLP of a random elliptic curve element with respect to a publicly known base point is computationally infeasible. This infeasibility is also called Elliptic Curve Discrete Logarithm Problem (ECDLP). Elliptic curves have cryptographic value, because scalar multiplication serves as a candidate one-way function. For a suitably chosen curve, the size of the cyclic group generated by the public base point determines the difficulty of the corresponding ECDLP instance.

Elliptic curves can be defined over any field although we generally see them, for applications in cryptography, used over finite fields $\mathbb{F}_q$, where $q = p^n$, p is a prime and $n \in \mathbb{N}$ a positive integers.

Ordinary binary elliptic curves In general, ordinary binary elliptic curves can be expressed by means of a short Weierstraß equation

$$y^2 + xy = x^3 + a_2x^2 + a_6 \tag{2.1}$$

where $a_2, a_6 \in \mathbb{F}_{2^n}$ with $a_6 \neq 0$. We write

$$E_{a_2,a_6}(\mathbb{F}_{2^n}) := \{(x, y) \in \mathbb{F}_{2^n}^2 : y^2 + xy = x^3 + a_2x^2 + a_6\} \cup \{\mathcal{O}\} \tag{2.2}$$

for the set of points on such a curve.

Other types of elliptic curves If $p \neq 2$ or 3 then any elliptic curve can be written as a plane algebraic curve defined by an equation of the form:

$$y^2 = x^3 + a_2x + a_6 \tag{2.3}$$

where $a_2, a_6 \in \mathbb{F}_{p^n}$. We write

$$E_{a_2,a_6}(\mathbb{F}_{p^n}) := \{(x, y) \in \mathbb{F}_{p^n}^2 : y^2 = x^3 + a_2x + a_6\} \cup \{\mathcal{O}\} \tag{2.4}$$

for the set of points on such a curve.

The point $\mathcal{O}$, used above in equations 2.2 and 2.4, is often referred as *point at infinity* and serves as neutral element in the set $E_{a_2,a_6}(\mathbb{F}_{p^n})$. These sets form a group with an efficiently computable group law, so that they are suited for implementing the cryptographic schemes, such as digital signature and key agreement protocols.

Unlike RSA, ECC are particularly appealing because they achieve the same level of security with much shorter key lengths. We can see the key size relationship between RSA and ECC provided by NIST (National Institute of Science and Technology) [44] below:

ECC Key Size	RSA Key Size	Key Size Ratio
163	1024	1:6
256	3072	1:12
384	7680	1:20
512	15360	1:30

Table 2.1: Key size relationship between ECC and RSA

2.3 DIGITAL SIGNATURE SCHEME

A digital signature is a method to sign a message electronically by a user which can be verified by anybody later. A digital signature protects data from being altered,

and enables the detection of modification. It has many applications in information security, including authentication, data integrity, and non-repudiation. We use digital signatures in a wide range of areas, such as certification of public keys in large networks, e-commerce, and software updates.

In 1976, Whitefield Diffie and Martin Hellman [22] first described the concept of a digital signature scheme. Two years later in 1978, Ronald Rivest, Adi Shamir and Len Adleman [52] designed the RSA algorithm, which could be used to produce primitive digital signatures. Here, we discuss the definition of signature scheme based on [36].

Definition 2.3.1. *A signature scheme* $\Sigma = (Gen, Sign, Verify)$ *is a triple of polynomial-time algorithms:*

1. *Gen: This is the probabilistic key-generation algorithm which takes a security parameter* 1^n *as an input and returns a pair of keys (**pub**, **priv**), i.e., public and private keys. Here, we assume **pub** and **priv** keys have length at least* n*, and that* n *can be determined from **pub**, **priv**.*

2. *Sign: This is the probabilistic signing algorithm which takes a private key **priv** and message* $m \in \{0,1\}^*$ *as an input and returns a signature* $\Sigma := Sign_{priv}(m)$ *as an output.*

3. *Verify: This is the deterministic verification algorithm which takes **pub**, message* m*, and a signature* Σ *as a input and returns* 1 *or* 0*, indicating if* Σ *is a valid signature for* m *under the public key **pub**.*

*For pairs (**pub**, **priv**) output by Gen, honestly generated signatures are always accepted, i.e.,* $Verify(Sign(m; priv); m; pub) = 1$ *for all messages* m

A digital signature scheme is said to be insecure if it is computationally feasible to obtain an *existential forgery*. Existential forgery is the creation (by an adversary) of at least one message-signature pair (m, Σ), where m has not been signed by the legitimate signer. The adversary need not have any control over m; m need not have any particular meaning. As long as the pair (m, Σ) is valid, the adversary has succeeded in constructing an *existential forgery*. Existential forgery is essentially the weakest adversarial goal, therefore the strongest schemes are those that are "existentially unforgeable".

2.4 PAIRING

A pairing is a bilinear map between elements of two groups (generally additive groups) to a third group (generally multiplicative). If the first two groups are identical, then the pairing is *symmetric* otherwise it is called *asymmetric*. Pairings can be used to reduce the hard problem from one group to a potentially easier problem in another group, such as the DLP from an elliptic curve group to a multiplicative group over finite field [40, 27]. In 2000, A. Joux [32] showed how to use a pairing to realize a three-round key agreement in one round. Recently many works and much interest have increased in cryptographic schemes based on pairings on elliptic curves [7, 12, 15, 17, 29, 48, 56]. In those schemes, pairing is one of the most essential operations. For implementation purposes, we can use a taxonomy by Freeman et al. [26], where a number of constructions to obtain pairing-friendly elliptic curves are discussed. The most common pairings in applications are the Weil and Tate pairings on an elliptic curve over a finite field; other pairings include the Eta pairing, the ate pairings, and their different forms.

In this section, we define a bilinear map and the hardness assumption of the

Strong Diffie-Hellman problem, which we use in Chapter 3 for the security proof of our signature scheme.

Definition 2.4.1 (Admissible bilinear map)**.** *Let* $(\mathbb{G}_1, +)$, $(\mathbb{G}_2, +)$, *and* $(\mathbb{G}_T, \cdot)$ *denote cyclic groups of prime order* $q \in [2^k, 2^{k+1}]$. *Then we refer to a map* $e : \mathbb{G}_1 \times \mathbb{G}_2 \longrightarrow \mathbb{G}_T$ *as* admissible bilinear map *if all of the following conditions are satisfied:*

Bilinearity: *For all* $(P_1, P_2) \in \mathbb{G}_1 \times \mathbb{G}_2$ *and* $a, b \in \mathbb{Z}$ *we have*

$$e(aP_1, bP_2) = e(P_1, P_2)^{ab}.$$

Non-degeneracy: *There exists* $(Q_1, Q_2) \in \mathbb{G}_1 \times \mathbb{G}_2$ *such that* $e(Q_1, Q_2) \neq 1$.

Efficiency: *The map* e *can be evaluated in polynomial time.*

The scheme described in Section 3.1 is formulated in the random oracle model, and the underlying hardness assumption will be the strong Diffie-Hellman assumption. Following [11] we capture this problem as follows:

Definition 2.4.2 (Strong Diffie-Hellman problem)**.** *Let* $(\mathbb{G}_1, +)$ *and* $(\mathbb{G}_2, +)$ *denote cyclic groups of prime order* q *such that there is an admissible bilinear pairing* $e : \mathbb{G}_1 \times \mathbb{G}_2 \longrightarrow \mathbb{G}_T$. *The* ℓ-Strong Diffie-Hellman (ℓ-SDH) problem *for* $(\mathbb{G}_1, \mathbb{G}_2)$ *is to find on input*

$$([r^i \cdot G_1]_{i=0}^{\ell}, [G_2, r \cdot G_2]) \in \mathbb{G}_1^{\ell+1} \times \mathbb{G}_2^2$$

with uniformly at random chosen generators $G_1 \in \mathbb{G}_1$, $G_2 \in \mathbb{G}_2$ *and uniformly at random chosen* $r \in \{0, \ldots, q-1\}$ *a pair* $(c, \frac{1}{r+c} \cdot G_1)$ *with* $c \in \{0, \ldots, q-1\} \setminus \{-r\}$.

In Section 3.1.2, we will show how one can derive an algorithm from an efficient algorithm $\mathcal{A}$, successfully attacking our protocol, to solve the ℓ-SDH problem in the underlying group pair efficiently.

2.5 IDENTITY BASED CRYPTOGRAPHY

In this section, we review the notion of identity based cryptography more precisely and give a formal definition of this notion. In traditional public key infrastructure, both sender and receiver must generate their encryption and signature key pairs. Before secure communication take place, they should get certificates, which should be signed from a certification authority. Certificate is used to authenticate a public key and is issued by some well-known trusted person or company, which is also called certification authority. This technical process could be time consuming and complicated to use. Problems with the traditional public key cryptosystems are the high cost of the infrastructure needed to manage and authenticate public keys, and the difficulty in managing multiple communities.

In 1984, Shamir [57] suggested an identity-based cryptosystem, where user's identities, such as her phone number, email address, zip code are used as her public key. Although it provides some advantages over traditional public key approaches, it has some drawbacks. In IBC, all users have to get their private key from a specific trusted authority. From the security point of view, this is less trustworthy since the complete security of the system depends on a single point. In this situation the required level of trust is stronger than the required level for public key certificates. In IBC, trusted authority publishes his public key and keeps his private key, which is also called the master key, to generate private keys for all system users.

In general, an identity based cryptographic protocol is a family of algorithms, usually consisting of four algorithms. Regardless of specific protocols, there are two fixed algorithms in identity based setting, i.e., **Setup** and **Extract**.

Setup : This algorithm takes a security parameter k as input and returns system parameters and a master-key. The system parameters include a description of

a finite message space and a description of a finite ciphertext space. Intuitively, the system parameters will be publicly known, while the master-key will be known only to the trusted authority.

Extract : This algorithm takes system parameter, master-key, and an arbitrary identity of the user as inputs. Here, an identity of the user is an arbitrary string that will be used as a public key. This algorithm extracts a private key for the user from his identity.

Now we discuss different algorithms for encryption and signature schemes in IBC.

2.5.1 Encryption scheme

In addition to the **Setup** and **Extract** algorithms, for an encryption scheme, two more algorithms, **Encrypt** and **Decrypt**, are used. Let us assume that Alice wants to send a message M to Bob. Before sending the message, Alice encrypts the message using the **Encrypt** algorithm and sends the resulting ciphertext to Bob. To decrypt the ciphertex, Bob needs his private key. This motivates Bob to contact the trusted authority to get his private key. Finally, Bob uses the **Decrypt** algorithm to decrypt the ciphertext.

Encrypt: This algorithm takes the identity of the receiver (in our case Bob's identity) and message as inputs, and it returns a corresponding ciphertext as an output.

Decrypt: This algorithm takes a ciphertext and private key of the receiver (in our case Bob's private key) as inputs, and it returns a plaintext message or dedicated error symbol as an output.

2.5.2 Signature scheme

In addition to **Setup** and **Extract** we need two additional algorithms: **Sign** and **Verify**. Let us assume that Alice wants to sign a message, then she needs her private key to sign it. She contacts the trusted authority and gets her private key. Now, she signs the message by using the **Sign** algorithm. Finally, anyone can, by using the **Verify** algorithm, verify that the message has been signed by Alice.

Sign: This algorithm takes the system parameters, signer's private key, and message as inputs, and it outputs a signature.

Verify: This algorithm takes the system parameters, message, signer's identity, and signature as inputs, and it returns valid or invalid as output.

2.6 TRAPDOOR FOR THE DISCRETE LOGARITHM PROBLEM

Pairing-friendly curves are versatile tools in the design of cryptographic protocols, especially for identity- based solutions. As documented in a taxonomy by Freeman et al. [26], a number of constructions to obtain pairing-friendly elliptic curves are available. Another cryptographically useful family of elliptic curves comes with a trapdoor for the discrete logarithm problem. At ASIACRYPT 2000, Paillier proposed several encryption schemes invoking such curves [46], and more recently Teske [60] suggested an elliptic curve cryptosystem with a trapdoor for the discrete logarithm problem. Interestingly, no constructions for elliptic curves in the intersection of these two families appear to be available in the literature. Differing from a setting considered by Dent and Galbraith [21], we conjecture that the efficient evaluation of the pairing ought to be possible without invoking secret trapdoor information. Our signature scheme in 3.1 depends on this conjecture.

2.7 HASH FUNCTIONS

A function that maps data of arbitrary length to data of a fixed length is called hash function, and its output is called hash value. A hash function can provide assurance of data integrity, and it is one of the widely used primitives in cryptographic protocols. For a particular message, the hash value can be seen as a fingerprint of a message, i.e., a unique representation for a message. In contrast to other cryptographic algorithms, hash functions do not have a key. In a signature scheme one commonly sign a message of arbitrary length by first converting it to a fixed length of a string by means of a hash function. So hash function is one of the essential parts of digital signature schemes. We also use hash functions in our signature scheme in Section 3.1. A cryptographic hash function should satisfies the following three (informal) properties:

1. pre-image resistance: This property is also called one-wayness as hash functions need to be one-way. Let us assume that our hash function is h, then for given hash output z, it must be computationally infeasible to find an input message m, such that $z = h(m)$.

2. second preimage resistance: This property is also called weak collision resistance. For given message m_1, it should be computationally infeasible to find m_2, such that $m_2 \neq m_1$ and $h(m_1) = h(m_2)$.

3. collision resistance: This is also called strong collision resistance. Here, an attacker is free to choose both m_1 and m_2. It should be computationally infeasible to find m_1 and m_2, such that $m_1 \neq m_2$ and $h(m_1) = h(m_2)$.

2.7.1 Random oracle model

The random oracle is simply a black box that takes a binary string of an arbitrary length as input and returns a uniformly at random chosen binary string of fixed length. If an input is repeated, then this box responds the same way every time that the input is submitted. Everyone can interact with the box privately, that means no one can have information about those interactions with the random oracle. The random oracle model has been used to design and prove the security of many cryptosystems because protocols designed in this model can be more efficient than the protocols designed in the standard model while retaining many advantages of provable security. So, first a scheme is designed and proven secure in random oracle model, and a cryptographic scheme is constructed and analyzed based on this assumption. However, to implement the scheme in the real world, a random oracle is not available. Therefore, instead of a random oracle, we use a hash function, such as SHA-1 or SHA-256.

2.8 SHOR'S ALGORITHM

Peter Shor presented two efficient quantum algorithms [58], one for factorization and another for finding discrete logarithms for which no polynomial time classical algorithms are known. Modern public-key cryptography and the security of digital signature methods have largely relied, so far, on the belief that no effective integer factorization or computation of discrete logarithm exists. Therefore, a design of a quantum computer capable of performing Shor's polynomial time quantum algorithms could have a great impact on modern cryptography. This section discusses Shor's polynomial time algorithms for the computation of discrete logarithms. Before that, we discuss some pertinent terminology.

Qubits: It is a quantum bit, the counterpart in quantum computing to the binary digit or bit of classical computing. Just as a bit is the basic unit of information in a classical computer, a qubit is the basic unit of information in a quantum computer. In a quantum system, binary bits 0 and 1 are represented by a prescribed pair of normalized and mutually orthogonal quantum states denoted as $\{|0\rangle, |1\rangle\}$. Any other qubits can be written as superposition $\alpha\,|0\rangle + \beta\,|1\rangle$ for some $\alpha \in \mathbb{C}$ and $\beta \in \mathbb{C}$, such that $|\alpha|^2 + |\beta|^2 = 1$.

Registers: A collection of n qubits is called quantum register of size n. Let $a = 2^0 a_0 + 2^1 a_1 + \cdots + 2^{n-1} a_{n-1}$ then a quantum register prepared with this value is represented by $|a\rangle = |a_{n-1}\rangle \otimes |a_{n-2}\rangle \otimes \cdots \otimes |a_0\rangle$ where $a_i \in \{0, 1\}$.

Quantum gate: A quantum logic gate is a device which performs a fixed unitary operation on selected qubits in a fixed period of time. The most common gates are the controlled NOT gate (CNOT), Toffoli gate, Hadamard gate, and T-gate.

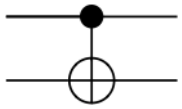


Figure 2.1: CNOT gate

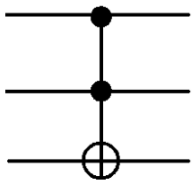


Figure 2.2: Toffoli gate

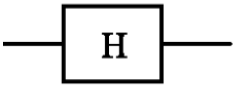


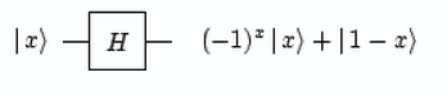
Figure 2.3: Hadamard gate

Figure 2.4: T-gate

Hadamard transform: A single qubit gate H performing the unitary transformation known as the Hadamard transform. It is defined as:

$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$

The matrix is written in the computational basis $\{|0\rangle, |1\rangle\}$. In general, a schematic representation of the gate H acting on a qubit in state $|x\rangle$, with $x = 0, 1$ is



Quantum Fourier transform (QFT): This is a linear transformation on quantum bits and it is a part of many quantum algorithms. It can be performed efficiently on a quantum computer, with a particular decomposition into a product of simpler unitary matrices.

The classical Fourier transform is a map:

$(x_0, x_1, \ldots, x_{N-1}) \to (y_0, y_1, \ldots, y_{N-1})$, with $y_k = \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} x_j \omega^{jk}$, where $\omega = e^{\frac{2\pi i}{N}}$ is a primitive N^{th} root of unity.

Similarly, the quantum Fourier transform is a map:

$|j\rangle \rightarrow \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} \omega^{jk} \, |k\rangle$

2.8.1 Quantum arithmetic

Basically, we are interested to reduce the number of $\mathbb{F}_{2^n}$ -operations to compute an addition between points on an elliptic curve over a binary field $\mathbb{F}_{2^n}$. The specific cost of each operation depends on the representation of $\mathbb{F}_{2^n}$. The following operations are of particular interest:

Addition Given $a, b \in \mathbb{F}_{2^n}$, compute their sum $a + b$.

Multiplication Given $a, b \in \mathbb{F}_{2^n}$, compute their product $a \cdot b$.

Multiplication by a constant For a fixed non-zero constant $\alpha \in \mathbb{F}_{2^n}^*$, on input $a \in \mathbb{F}_{2^n}$, compute $a \cdot \alpha$. The value α, for example, could be a coefficient in defining equation of an elliptic curve.

Squaring Given $a \in \mathbb{F}_{2^n}$, compute a^2.

Inversion Given $a \in \mathbb{F}_{2^n}^*$, compute a^{-1}.

2.8.2 Computing discrete logarithms with Shor's algorithm

Given a generator P and some scalar multiple $Q = r \cdot P$ on an elliptic curve over $\mathbb{F}_{2^n}$. Let us assume that the order of P i.e. $\mathrm{ord}(P)$ is known, then our goal is to find $r \in \{1, 2, 3, \ldots, \mathrm{ord}(P)\}$. With a reversible implementation for the basic elliptic curve group operations, it is possible to find r with a polynomial-depth quantum circuit. For this, we proceed as follows [55, Section 2.2]:

- Create two registers of length $n+1$ and initialize them with the zero state $|0\rangle$
- Apply a Hadamard transform H to each qubit. It results the state

  $\frac{1}{2^{n+1}} \sum_{x,y=0}^{2^{n+1}-1} |x, y\rangle$
- Implement the map

  $\frac{1}{2^{n+1}} \sum_{x,y=0}^{2^{n+1}-1} |x, y\rangle \rightarrow \frac{1}{2^{n+1}} \sum_{x,y=0}^{2^{n+1}-1} |x, y\rangle \, |xP + yQ\rangle$
- Then perform a two-dimensional quantum Fourier transform over the first two registers. It was shown in [51] that the creation of the above state can be reduced to adding a classically known point to a superposition of points. The reduction is made by using a double and add method analogous to the square and multiply method of modular exponentiation. Points of the form $2^k P$ and $2^k Q$ can be classically precomputed, and then, starting with the additive identity, group addition operations can be performed, controlled by the appropriate bits from $|x\rangle$ or $|y\rangle$. Note that all of the intermediate sums must be preserved until the computation is completed before they can be uncomputed.

Chapter 3

A "short" identity-based signature scheme

One of the main technical tools in identity-based cryptography are suitable bilinear maps (pairings), and a number of identity-based signature schemes have been proposed relying on the use of this tool. Despite progress in the efficient implementation of such pairings, their evaluation remains comparatively costly, and a low number of pairing computations for signing messages and verifying signatures is a desirable design goal. In [15], Choon and Cheon proposed an efficient construction in the random oracle model, where the verification of a signature involves two pairing computations (and a scalar multiplication). In combination with the efficient signing operation, dominated by two scalar multiplications, and the provable security guarantees this is quite attractive, but unlike for instance in Hess's scheme [29] the proposal in [15] does not offer savings in the verification cost when verifying multiple messages that are presumably signed by the same identity.

Building on Choon and Cheon's construction, we propose a new identity-based signature scheme in the random oracle model, which assumes the availability of a pairing-friendly group with discrete logarithm trapdoor. The security reduction for our scheme relies on the strong Diffie-Hellman assumption. The guarantees provided by our proposal are similar to those of the proposal in [15], but in the case of multiple signature verifications with the same identity, one of the two pairing operations can

be avoided. Moreover, if multiple messages are signed (batch signing), all but one signature can be reduced to a single group element. As detailed in Section 3.2, overall this results is an attractive performance compared to other proposed identity-based signature schemes with provable security guarantees.

A drawback is the limitation in the group choice that comes with the strong Diffie-Hellman assumption: results by Brown and Gallant [14], Cheon [16], and by Jao and Yoshida [31] indicate that for many groups of interest the strong Diffie-Hellman problem is easier to solve than the discrete logarithm problem.

3.1 DESCRIPTION AND ANALYSIS OF THE PROPOSED SCHEME

For $i = 1, 2$, denote by $\mathbb{G}_i$ an additively written cyclic group of prime order $q \in [2^k, 2^{k+1}]$ with uniformly at random chosen public generator G_i. We also assume that an admissible bilinar map $e : \mathbb{G}_1 \times \mathbb{G}_2 \longrightarrow \mathbb{G}_T$ is available for some group $\mathbb{G}_T$ of order q. As indicated already, the identity-based signature scheme we propose is formulated in the random oracle model: let $h : \{0,1\}^* \longrightarrow \{0, \ldots, q-1\}$ for a random oracle that maps a message into an integer modulo q and $H : \{0,1\}^* \longrightarrow \mathbb{G}_1 \times \mathbb{G}_2$ a random oracle that maps user identities to an element of $\mathbb{G}_1 \times \mathbb{G}_2$. To refer to the two components of a value $H(\mathrm{id})$ we will use the notation $H(\mathrm{id}) = (\underbrace{H_{\mathrm{id},1}}_{\in \mathbb{G}_1}, \underbrace{H_{\mathrm{id},2}}_{\in \mathbb{G}_2})$. We assume that the central authority has trapdoor information available to compute discrete logarithms with respect to the base point G_2.

3.1.1 An identity-based signature scheme

To specify an identity-based signature scheme we have to provide four (polynomial-time) algorithms:

Setup: This algorithm is run by the trusted authority to generate a secret master

key and public system parameters.

Extract: This algorithm is run by the trusted authority to extract a user-specific secret signing key from an identity.

Sign: This algorithm enables a user to create a signature for a message, using its secret user key (extracted by the trusted authority).

Verify: Given a message, a candidate signature, and the identity of the potential signer, this algorithm allows to decide if this signature is valid.

Figure 3.1 shows how each of these algorithms is realized in our proposal.

Setup: Selects a secret master key $s \in \{0, \ldots, q-1\}$ uniformly at random and outputs the public key $P_{\text{pub}} = s \cdot G_2 \in \mathbb{G}_2$.

Extract: On input an identity $\text{id} \in \{0,1\}^*$ and master key s, the values $r_{\text{id}} = \log_{G_2}(H_{\text{id},2}) \in \{0, \ldots, q-1\}$, and $S_{\text{id}} = s \cdot H_{\text{id},1} \in \mathbb{G}_1$ are returned.

Sign: To sign $m \in \{0,1\}^*$ with secret user key $(r_{\text{id}}, S_{\text{id}})$, compute

$$\sigma = \frac{1}{r_{\text{id}} + h(m)} \cdot S_{\text{id}},$$

and output the signature $\sigma \in \mathbb{G}_1$.

Verify: A signature $\sigma \in \mathbb{G}_1$ for $m \in \{0,1\}^*$ and identity id is accepted if and only if $e(\sigma, H_{\text{id},2} + h(m) \cdot G_2) = e(H_{\text{id},1}, P_{\text{pub}})$.

Figure 3.1: an identity-based signature scheme

The signature computation fails if $r_{\text{id}} + h(m) = 0 \pmod{q}$, i. e., if the required inversion modulo q cannot be performed. As h is a random oracle, this happens with (negligible) probability $1/q$ only. Otherwise the correctness of our scheme follows

immediately from the equality

$$e(\sigma, r_{\mathrm{id}} \cdot G_2 + h(m) \cdot G_2) = e(S_{\mathrm{id}}, G_2) = e(H_{\mathrm{id},1}, P_{\mathrm{pub}}).$$

Section 3.2 offers a more detailed performance discussion, but one immediately observes the following:

- The right-hand side of the verification equation is message-independent and can be reused by the verifier.
- The signing algorithm is deterministic and can be implemented on devices which do not provide a (pseudo)random number generator.

3.1.2 Security analysis

To analyze the security of our scheme, we prove that from an algorithm $\mathcal{A}$ that produces an existential forgery, we can derive an algorithm $\mathcal{C}$ with comparable resource requirements that, for a suitable ℓ, solves the ℓ-SDH problem in the underlying group pair. The proof we give is an adaption of an analysis by Boneh and Boyen [11] and by Cha and Cheon [15] to our situation. The adversary is modeled as a probabilistic algorithm $\mathcal{A}$ which obtains the public parameters produced by **Setup** as input. Following [15], the algorithm $\mathcal{A}$ has access to the random oracles h and H and to two more oracles:

Key extraction oracle $\mathcal{E}$: On input an identity $\mathrm{id} \in \{0,1\}^*$, the corresponding secret key $(r_{\mathrm{id}}, S_{\mathrm{id}})$ is returned.

Signature oracle $\mathcal{S}$: On input a user identity $\mathrm{id} \in \{0,1\}^*$ and a message m, this oracle returns the output of **Sign** when being executed with input $(r_{\mathrm{id}}, S_{\mathrm{id}})$ and m.

The algorithm $\mathcal{A}$ outputs a user identity id_0, a message m_0, and a signature for m_0. If this signature is valid and neither id_0 has been queried to the key extraction oracle nor (id_0, m_0) has been queried to the signature oracle, then $\mathcal{A}$ succeeded in creating an existential forgery.

For the proof that an efficient algorithm to create existential forgeries in our scheme can be turned into an efficient algorithm to solve the ℓ-SDH problem for the group pair $(\mathbb{G}_1, \mathbb{G}_2)$ and a suitable ℓ, we make use of (the proof of) a lemma by Cha and Cheon [15, Lemma 1] and a lemma by Boneh and Boyen [11, Lemma 9].

Lemma 3.1.1. *Let $\mathcal{A}$ be an algorithm that in polynomial time with probability $\varepsilon_{\mathcal{A}}$ outputs an existential forgery for the scheme in Figure 3.1 for some identity id_0. Morever, let $\mathrm{id}_1 \in \{0,1\}^k$ be chosen uniformly at random and $q_H \geq 1$ an upper bound on the number of queries to H by $\mathcal{A}$. Then there is an algorithm $\mathcal{A}_1$ which outputs in polynomial time an existential forgery for id_1 with probability*

$$\epsilon_{\mathcal{A}_1} \geq \frac{1}{q_H + q_{\mathcal{E}} + q_{\mathcal{S}}} \cdot \left(1 - \frac{1}{q}\right) \cdot \left(1 - \frac{q_H}{2^k}\right) \cdot \varepsilon_{\mathcal{A}}.$$

The number of extraction and signature queries made by $\mathcal{A}_1$ is the same as for $\mathcal{A}$.

Proof. Without loss of generality we may assume that $\mathcal{A}$ does not repeatedly send the same query to H—the algorithm $\mathcal{A}$ can simplify maintain a list with already queried values and received responses. The algorithm $\mathcal{A}_1$ runs a simulation of $\mathcal{A}$, using the public key P_{pub} faced by $\mathcal{A}_1$ as input to $\mathcal{A}$ and simulating all oracles for the latter. Before starting the simulation, $\mathcal{A}_1$ chooses a value $t \in \{1, \ldots, q_H + q_{\mathcal{E}} + q_{\mathcal{S}}\}$ uniformly at random, where $q_\cdot$ denotes a polynomial upper bound on the number of queries that $\mathcal{A}$ submits to the respective oracle. The simulation of the individual oracles is almost completely faithful:

h: This is the trivial simulation—$\mathcal{A}_1$ forwards the query to its own h-oracle and returns the answer of that oracle.

H**:** For the t-th query, return $H(\mathrm{id}_1) = (H_{\mathrm{id}_1,1}, H_{\mathrm{id}_1,2})$, for all other queries simply forward the query to $\mathcal{A}$'s own H-oracle.

$\mathcal{E}$**:** If the queried identity id has not been queried to H yet, submit id to the simulation of H and then forward id to $\mathcal{A}$'s extraction oracle $\mathcal{E}$. Otherwise forward id to $\mathcal{E}$ directly. The answer of $\mathcal{E}$ is given to $\mathcal{A}$.

$\mathcal{S}$**:** If the identity id in a signature query (id, m) has not been submitted to H yet, submit id to the simulation of H and then forward id to $\mathcal{A}$'s signing oracle $\mathcal{S}$. Otherwise forward id to $\mathcal{S}$ directly. The answer of $\mathcal{S}$ is given to $\mathcal{A}$.

Let $(\mathrm{id}_0, m_0, \sigma_0)$ be the output of $\mathcal{A}$, interacting with the simulated oracles. If $H(\mathrm{id}_0) = H(\mathrm{id}_1)$ and the signature created by $\mathcal{A}$ is valid, then $\mathcal{A}_1$ outputs the valid signature $(\mathrm{id}_1, m_0, \sigma_0)$. In all other cases, $\mathcal{A}_1$'s strategy failed and it outputs a random guess $(\mathrm{id}_1, m_1, \sigma_1)$ with $m_1 = 0$ and $\sigma_1 \in \mathbb{G}_1$ chosen uniformly at random.

The simulation for $\mathcal{A}$ is perfect, unless id_1 has been queried to H (explicitly or implicitly through a signature or extraction query) before the t-th query. Since id_1 is chosen uniformly at random from $\{0,1\}^k$, the probability for a simulation failure can be bounded by

$$\Pr[\mathsf{SimulationFail}] \leq \frac{q_H}{2^k}.$$

So with the simulated oracles, $\mathcal{A}$ outputs a valid forgery with probability at least

$$\Pr[(\mathrm{id}_0, m_0, \sigma_0) \text{ is a valid signature}] \geq \left(1 - \frac{q_H}{2^k}\right) \cdot \varepsilon_{\mathcal{A}}. \tag{3.1}$$

Moreover, the probability that the output $(\mathrm{id}_0, m_0, \sigma_0)$ of $\mathcal{A}$ is a valid signature without id_0 having been queried to H is $\leq 1/q$, as then the right-hand side of the verification equation is a random element from $\mathbb{G}_T$. In other words, we have

$$\Pr[\mathrm{id}_0 \text{ was queried to } H | (id_0, m_0, \sigma_0) \text{ is a valid signature}] \geq 1 - \frac{1}{q}. \tag{3.2}$$

Since t is independently and randomly chosen, we also have

$$\Pr[\mathrm{id}_0 \text{ was the } t\text{-th query to } H | \mathrm{id}_0 \text{ was queried to } H \text{ and } (\mathrm{id}_0, m_0, \sigma_0) \text{ is a valid signature}] \geq \frac{1}{q_H + q_\mathcal{E} + q_\mathcal{S}}. \quad (3.3)$$

Combining inequalities (3.1)–(3.3), we obtain the desired lower bound for the success probability ε_{A_1} of $\mathcal{A}_1$:

$$\varepsilon_{\mathcal{A}_1} \geq \frac{1}{q_H + q_\mathcal{E} + q_\mathcal{S}} \cdot \left(1 - \frac{1}{q}\right) \cdot \left(1 - \frac{q_H}{2^k}\right) \cdot \varepsilon_\mathcal{A}$$

□

Building on the adversary constructed in the proof of Lemma 3.1.1, one can derive an algorithm $\mathcal{A}_2$ which solves the strong Diffie-Hellman problem in the group pair underlying the proposed signature scheme.

Lemma 3.1.2. *Let $\mathcal{A}_1$ be a polynomial time adversary against the scheme in Figure 3.1 and $\mathrm{id}_1 \in \{0,1\}^k$ chosen uniformly at random. Assume that $\mathcal{A}_1$ submits a total of at most $q_h \geq 1$ queries to h and that $\ell \geq q_h - 1$. If $\mathcal{A}_1$ succeeds with probability $\varepsilon_{\mathcal{A}_1}$ in creating a forgery for the identity id_1, then there is a polynomial time algorithm $\mathcal{C}$ which can solve the ℓ-SDH problem in $(\mathbb{G}_1, \mathbb{G}_2)$ with probability $\varepsilon_\mathcal{C} \geq \frac{1}{q_h} \cdot \left(1 - \frac{1}{q}\right)^4 \cdot \left(1 - \frac{1}{q_h}\right) \cdot \varepsilon_{\mathcal{A}_1}$.*

Proof. The algorithm $\mathcal{C}$ runs $\mathcal{A}_1$ as a subroutine, providing all inputs and simulating all oracles for $\mathcal{A}_1$. Let $([r^i \cdot G_1]_{i=0}^{\ell}, [G_2, r \cdot G_2]) \in \mathbb{G}_1^{\ell+1} \times \mathbb{G}_2^2$ be the ℓ-SDH challenge faced by $\mathcal{C}$. To answer queries to h, the algorithm $\mathcal{C}$ chooses $(h_1, \ldots, h_{q_h}) \in \{0, \ldots, q-1\}^{q_h}$ uniformly at random. In addition, $\mathcal{C}$ selects $t \in \{1, \ldots, q_h\}$ uniformly at random and defines the polynomial

$$f(y) = \prod_{\substack{i=1 \\ i \neq t}}^{q_h} (y + h_i) \in \mathbb{F}_q[y]$$

of degree $q_h - 1$. Note that by expanding this polynomial into standard distributive form and using the first component of the ℓ-SDH-challenge, $\mathcal{C}$ can compute $f(r) \cdot G_1$, provided that $\ell \geq q_h - 1$.

If $f(r) \cdot G_1$ happens to be $0 \in G_1$, the algorithm $\mathcal{C}$ can recover r as one of the h_i-values multiplied by $-1 \in \mathbb{F}_q$, making the ℓ-SDH problem trivial. So in the sequel we may assume that $f(r) \neq 0 \bmod q$. To create the public key, $\mathcal{C}$ chooses a master key $s \in \{0, \ldots, q-1\}$ uniformly at random and hands $P_{\text{pub}} = s \cdot G_2$ to the adversary $\mathcal{A}_1$. The individual oracles for $\mathcal{A}_1$ are simulated as follows:

h: In response to the i-th new query, the value h_i is returned. By keeping track of received queries, repeated queries are answered consistently.

H: In response to a new query $\text{id} \in \{0,1\}^* \setminus \{\text{id}_1\}$, choose $\mu_{\text{id}}, r_{\text{id}} \in \{0, \ldots, q-1\}$ uniformly at random and return

$$H(\text{id}) = \begin{cases} (\mu_{\text{id}} \cdot G_1, r \cdot G_2), & \text{, if id} \neq \text{id}_1 \\ (\mu_{\text{id}} \cdot f(r) \cdot G_1, r \cdot G_2) & \text{, if id} = \text{id}_1 \end{cases}.$$

(The value $r \cdot G_2$ is available as part of the ℓ-SDH-challenge.) By keeping track of received queries, repeated queries are answered consistently.

$\mathcal{E}$: the correct key extraction algorithm is executed, using the simulated h- and H-oracles.

$\mathcal{S}$: For identities other than id_1, the correct signing algorithm is executed, using the simulated h- and H-oracles. When being asked to sign a message m for id_1, then

$$s \cdot \underbrace{\mu_{\text{id}_1} \cdot \frac{f(r)}{r + h(m)} \cdot G_1}_{= \frac{1}{r+h(m)} \cdot H_{\text{id}_1, 1}}$$

is returned, using the simulated h- and H-oracles. Unless $h(m)$ ends up being the t-th new query to h, knowing the polynomials $f(y)$ and $y + h(m)$ as well as s and μ_{id}, the algorithm $\mathcal{C}$ can compute this signature by means of the values in the ℓ-SDH-challenge.

The above simulation is perfect, provided that $\mathcal{A}_1$ does not submit an extraction query for id_1 or the t-th new query to h results from a query to $\mathcal{S}$ with identity id_1. The former is not allowed for a successful forgery for the identity id_1, and so the success probability of $\mathcal{A}_1$ in creating a forgery for id_1 when interacting with the simulated oracles is at least

$$\left(1 - \frac{1}{q_h}\right) \cdot \varepsilon_{\mathcal{A}_1}.$$

Let the forgery output by $\mathcal{A}_1$ be for some message m^* with corresponding signature σ^*. If this forgery is not valid for id_1, or if m^* has not been the new query no. t to h, then $\mathcal{C}$'s strategy failed and $\mathcal{C}$ simply outputs a random guess (c, G_1) with $c \in \{0, \ldots, q-1\}$ chosen uniformly at random.

To resist some trivial attacks, we assume that the master key $s \neq 0 \mod q$ and $H_{\mathrm{id}_1,1} \neq 0$, which is true with the probability $(1 - \frac{1}{q})^2$.

If m^* has never been queried to h, then the left-hand side of the verification equation is a random element in G_T, and we see that

$$\Pr[m^* \text{ was queried to } h | (m^*, \sigma^*) \text{ is a valid forgery for } \mathrm{id}_1] \geq (1 - \frac{1}{q}).$$

Since t is independently and randomly chosen, if m^* has been queried to h, it was the new query no. t to h with probability

$$\Pr[h(m^*) = h_t | m^* \text{ was queried to } h \text{ and } (m^*, \sigma^*) \text{ is a valid forgery for } \mathrm{id}_1] \geq \frac{1}{q_h}.$$

Thus, algorithm $\mathcal{A}_1$ returns a valid forgery (m^*, σ^*) for id_1 such that $h(m^*) = h_t$ with

probability at least

$$\frac{1}{q_h} \cdot \left(1 - \frac{1}{q}\right)^3 \cdot \left(1 - \frac{1}{q_h}\right) \cdot \varepsilon_{\mathcal{A}_1}.$$

Since $\sigma^* \in \mathbb{G}_1$, we can write $\sigma^* = d \cdot G_1$ for some value $d \in \{0, \ldots, q-1\}$, and we claim that $d = \mu_{\mathrm{id}_1} \cdot s \cdot f(r)/(r + h_t) \bmod q$. From the verification condition we obtain

$$\begin{aligned} & e(d \cdot G_1, H_{\mathrm{id}_1,2} + h_t \cdot G_2) &=& \quad e(H_{\mathrm{id}_1,1}, P_{\mathrm{pub}}) \\ \iff \quad & e(G_1, G_2)^{d \cdot (r+h_t)} &=& \quad e(\mu_{\mathrm{id}_1} \cdot f(r) \cdot G_1, s \cdot G_2)\ , \\ \iff \quad & e(G_1, G_2)^{d \cdot (r+h_t)} &=& \quad e(G_1, G_2)^{\mu_{\mathrm{id}_1} \cdot s \cdot f(r)} \end{aligned}$$

and we may conclude that indeed

$$d = \frac{\mu_{\mathrm{id}_1} \cdot s \cdot f(r)}{r + h_t} \bmod q.$$

With probability $(1 - 1/q)$ the condition $\mu_{\mathrm{id}_1} \cdot s \neq 0 \mod q$ is satisfied and we can write

$$(\mu_{\mathrm{id}_1} \cdot s)^{-1} \sigma_* = \frac{f(r)}{r + h_t} \cdot G_1. \tag{3.4}$$

By construction $y + h_t$ does not divide $f(y)$, and so there exists a non-zero constant $\gamma_0 \in \mathbb{F}_q^*$ and a polynomial $\gamma(y) \in \mathbb{F}_q[y]$ of degree $\leq q_h - 1$ such that $f(y) = g(y) \cdot (y + h_t) + \gamma_0$. Consequently,

$$\frac{f(r)}{r + h_t} = \gamma(r) + \frac{\gamma_0}{(r + h_t)},$$

from which we obain the relation

$$\frac{f(r)}{r + h_t} \cdot G_1 - \gamma(r) \cdot G_1 = \frac{\gamma_0}{r + h_t} \cdot G_1.$$

By means of Equation (3.4) and using the values from the ℓ-SDH challenge, $\mathcal{C}$ can evaluate the left-hand side of this equation. A final division by γ_0 yields

$$\frac{1}{r + h_t} \cdot G_1 = \gamma_0^{-1} \cdot \left((\mu_{\mathrm{id}_1} \cdot s)^{-1} \sigma_* - \gamma(r) G_1\right),$$

i. e., a solution for the ℓ-SDH problem. □

From Lemma 3.1.1 and Lemma 3.1.2 we immediately obtain the desired security reduction.[1]

Theorem 3.1.1. *Assume there is a polynomial time algorithm $\mathcal{A}$ creating an existential forgery against the scheme in Figure 3.1 with non-negligible probability. If $\mathcal{A}$ queries h no more than $q_h \geq 1$ times and $\ell \geq q_h - 1$, then there is a polynomial time algorithm that solves the ℓ-SDH problem in the underlying group pair with non-negligible success probability.*

3.2 COMPARISON WITH OTHER IDENTITY-BASED SIGNATURE SCHEMES

In this section we compare our identity-based signature scheme with related schemes from a performance perspective. Table 3.2 compares the performance of selected schemes [7, 15, 17, 29, 48, 56] in terms of computations required for signature generation and verification. Computation complexity for signing and signature verification is given in number of pairing evaluations, scalar multiplications in $\mathbb{G}_i$, and exponentiations in $\mathbb{G}_T$ (denoted as P, M, and E, respectively). Since several schemes require one less pairing for subsequent verifications after the first, we distinguish first verifications for previously unknown identities (Vrfy once) from cases where the result of the constant pairing evaluation of that signer has already been stored (Vrfy subseq.). Operations such as point additions or multiplications in $\mathbb{G}_T$ are negligible and therefore do not appear in Table 3.2. This also applies to the inversion necessary during the signature generation in our scheme. This inversion is in $\mathbb{Z}_q^*$ and is hence also negligible compared to the other arithmetic operations.

The proposed scheme has several advantages if required type of elliptic curves is

[1] For the sake of readability we do not explicitly state the success probability and running times, which follow immediately from these lemmas and their proofs.

	Paterson [48]	Cha Cheon [15]	CYHC [17]	BLM [7]	Hess [29]	ours ours
Sign	3M	2M	1E+1M	1E+1M	1M+1E(+1P+1M)	1M
Vrfy once	2E+2P	1M+2P	1E+2P	1E+1M+1P	1E+2P	1M+2P
Vrfy subseq.	2E+1P	1M+2P	1E+1P	1E+1M+1P	1E+1P	1M+1P
Signature	$\mathbb{G} \times \mathbb{G}$	$\mathbb{G} \times \mathbb{G}$	$\mathbb{F}_q^* \times G$	$\mathbb{G}_1 \times \mathbb{G}_1$	$\mathbb{F}_q^* \times G$	$\mathbb{G}_1$

Table 3.1: Performance comparison of popular identity-based signature schemes. P denotes the number of pairing evaluations, M denotes scalar multiplications in $\mathbb{G}_1$ or $\mathbb{G}_2$, SM denotes scalar multiplications of the form $aP + bQ$, and E denotes exponentiations in $\mathbb{G}_T$

available:

- The proposed scheme is the first identity-based short signature scheme: Unlike the other schemes, the signature is a single group element $\sigma \in \mathbb{G}$.
- Our scheme does not require any pairing evaluation in the signing phase, meaning that no pairing implementation is required for signature generation. In fact, only a single scalar multiplication and the inversion in $\mathbb{G}_T$, giving it a more efficient signing operation than all of the compared schemes.

Unfortunately, our scheme relies on the ℓ-SDH problem, which narrows down the choices of suitable curves [14, 16, 31]. Suitable parameters for BN curves have been proposed in [47].

Compared to prior work, the proposed scheme is favorable: The only category where our scheme is outperformed is verification for an unknown identity, where [7] needs one less pairing. However, it needs an additional exponentiation $\mathbb{G}_T$ for any (i.e. also for subsequent) verification. In fact, only [15] and our scheme do not need to

compute exponentiation in $\mathbb{G}_T$ at all. But compared to [15], our verification requires one less pairing evaluation for known identities.

Compared to [17], all exponentiations in $\mathbb{G}_T$ are replaced by scalar multiplications in G_i, with $i \in \{1, 2\}$ in our scheme. Scalar multiplications for elliptic curves are usually more efficient than the exponentiation in $\mathbb{G}_T$. This is due to the fact that the extension field $\mathbb{G}_T$ has to be chosen significantly larger than the field the elliptic curve is defined on [26], especially for larger embedding degrees.

Chapter 4

Automatic synthesis of quantum circuits for point addition on ordinary binary elliptic curves

---

Ordinary binary elliptic curves are an algebraic structure of great cryptographic significance. All binary curves suggested in the Digital Signature Standard [45] fall in this class, and the cost of implementing Shor's quantum algorithm [58] in such groups has been explored by various authors. While optimizing the implementation of the Quantum Fourier Transform is a quite well understood task, minimizing the implementation cost of the scalar multiplication in Shor's algorithm remains a design challenge.

Approaches by Kaye and Zalka [34] and by Maslov et al. [38] rely on the use of projective coordinates and efficient circuits for adding fixed (classically precomputed) points in a right-to-left variant of the double-and-add algorithm. In fact, only a "generic" addition of a fixed point is implemented, avoiding a handling of special cases of the addition law (doubling a point, adding a point with its inverse, or with the identity element). As observed in [38], it is sufficient to represent the input and output points of such a point addition circuit with projective coordinates. Amento et al. [4] suggest to replace ordinary projective coordinates with a representation used by

Higuchi and Takagi [30], therewith reducing the number of T-gates[1] needed. Taking the number of T-gates as cost measure, this is so far the most efficient implementation proposed, but as noted in [55], alternative constructions can reduce the design complexity: If dedicated circuitry for doubling a point is available, scalar multiplication can be realized by invoking only two types of addition circuits—rather than several hundred different ones when dealing with cryptographically significant parameters. These doubling circuits do impact the gate count, however. Happily, with the software tool presented below, designing addition circuits can be automated, making the derivation of a few hundred addition circuits for different points a realistic option.

To optimize circuit depth, [55] suggest a tree-style organization of the scalar multiplication in Shor's algorithm. However, this method builds on general addition circuits for the elliptic curve, i. e., addition circuits which have two variable input points and handle all cases of the addition law. In [55] complete binary Edwards curves [10] are used for this purpose. While the resulting circuit depth is compelling, the number of T-gates and number of qubits is much worse than with a right-to-left double and add procedure. When aiming at a small T-gate count, optimizing quantum circuits for the "generic" addition of a fixed point appears to be the more preferrable research direction. In this thesis, the central optimization criteria is the number of T-gates, and as secondary criteria we take the T-depth and the number of qubits into account.

Building on an addition formula by Al-Daoud et al. [2] we show that the number of T-gates in the best available circuit to add a fixed point [3] can be reduced by more than 60% without affecting the T-depth negatively. At the same time the number of qubits can be reduced at the cost of a depth increase of about $4n$ when working

[1] As is common, we do not distinguish between T- and $T^\dagger$-gates in statements on the number of T-gates or the T-depth.

with curves over $\mathbb{F}_{2^n}$. The circuit descriptions are derived automatically, and by means of edge colorings of certain bipartite graphs it is ensured that the involved subcircuits for $\mathbb{F}_2$-linear operations—such as multiplication by a constant, squaring and computing a square root—are optimized. For parameters of interest the latter allow substantial savings in the number of CNOT gates compared to the bounds used in [3]. Building on an available polynomial-basis arithmetic for the underlying binary field, the `Python` [25] software we introduce synthesizes for a given curve and curve point an optimized addition circuit and outputs this circuit as a `.qc` file. This file can then be processed with `QCViewer` [24], for instance, or more generally serve as input for automated or manual post-processing.

4.1 QUANTUM CIRCUITS FOR $\mathbb{F}_{2^n}$-ARITHMETIC

A binary field $\mathbb{F}_{2^n}$ can be represented in various different ways, resulting in potentially very different quantum circuits to realize the arithmetic. The use of a normal basis has been considered [4], but for elliptic curve addition with a small T-gate complexity, a polynomial basis representation seems the preferrable choice (cf. [3, Section 2]). In a polynomial basis representation, $\mathbb{F}_{2^n}$ is expressed as a quotient

$$\mathbb{F}_{2^n} = \mathbb{F}_2[x]/(p)$$

of the univariate polynomial ring $\mathbb{F}_2[x]$ with binary coefficients, where $p \in \mathbb{F}_2[x]$ is an irreducible polynomial of degree n. Having fixed p, each element $a \in \mathbb{F}_{2^n}$ is uniquely represented by a bit vector $(a_0, a_1, \ldots, a_{n-1}) \in \mathbb{F}_2^n$ such that $a = a_0 + a_1 x + \cdots + a_{n-1}x^{n-1} \pmod{p}$. This bit vector is naturally represented with n qubits $|a_0\rangle \ldots |a_{n-1}\rangle$. To implement point addition with a projective representation on a binary elliptic curve, we rely on addition, multiplication, multiplication with a non-zero constant and squaring in the underlying finite field. Quantum circuits for these

tasks are available (cf. [8, 34, 38, 3]):

Addition To add two field elements $a, b \in \mathbb{F}_{2^n}$, we can simply use n CNOT gates that operate in parallel:

$$|a, b\rangle \longmapsto |a, a \oplus b\rangle.$$

Alternatively, if the operands are to remain unchanged, we can implement

$$|a, b\rangle\ |0^n\rangle \longmapsto |a, b\rangle\ |a \oplus b\rangle$$

in the obvious way with $2n$ CNOT gates in depth 2.

Multiplication Optimizing the field multiplier is outside the scope of this thesis, and subsequently we will use a linear-depth construction by Maslov et al. [38]. With this method one can multiply two elements $a, b \in \mathbb{F}_{2^n}$ with no more than n^2 Toffoli gates and $n^2 - 1$ CNOT gates. For certain choices of p, including trinomials, this bound can be improved further. We note that the point addition circuit developed in Section 4.2 treats the underlying $\mathbb{F}_{2^n}$-multiplier as a black box. If more efficient field multipliers become available, integrating these into our synthesis tool should be straightforward.

Multiplication with a non-zero constant and squaring Both of these operations are linear, and [3] argue that an LUP decompositon yields a circuit of depth $\leq 2n$ that can be realized with $n^2 + n$ CNOT gates. Although no T-gates are needed for these operations, optimizing this step further is worthwhile: for the binary elliptic curves in the Digital Signature Standard [45], we have $n \geq 163$ and accordingly the complete scalar multiplicaton in Shor's algorithm involves several hundred addition circuits.

4.1.1 Optimizing $\mathbb{F}_2$-linear operations and minimal edge colorings

Multiplication by a constant and squaring are special cases of finding a quantum circuit implementing a map

$$|a\rangle\ |0^n\rangle \longmapsto |a\rangle\ |b_0 \dots b_{n-1}\rangle$$

where $a = \sum_{i=0}^{n-1} a_i x^i + (p)$ is an arbitrary input from $\mathbb{F}_{2^n}$ and $(b_0, \dots, b_{n-1}) = (a_0, \dots, a_{n-1}) \cdot M$ for some non-singular matrix $M \in \mathrm{GL}_n(\mathbb{F}_2)$. Obviously such a vector-by-matrix multiplication can be implemented with one CNOT gate for each non-zero entry of M. This can be done without ancillae qubits using a total of weight(M) CNOT gates. To minimize the circuit depth we interpret $M = (m_{i,j})_{0 \le i < n}$ as biadjacency matrix of a bipartite graph. Namely, the graph associated with M has $2n$ vertices with the vertex set splitting into the "control part" $\{a_0, \dots, a_{n-1}\}$ and the "target part" $\{b_0, \dots, b_{n-1}\}$. Each CNOT corresponds to exactly one edge: there is an edge between a_i and b_j if and only if $m_{i,j} = 1$. An edge coloring of this graph with d colors immediately yields a quantum circuit to multiply by M in depth d—all CNOT gates corresponding to an edge of the same color operate on disjoint qubits and therewith can be executed in parallel.

The minimal possible value of d is known as *chromatic index* of the graph. For a bipartite graph the chromatic index is equal to the maximum degree of a vertex, i. e., equal to the maximal Hamming weight of the rows and columns of M. Efficient classical algorithms for finding such a minimal edge coloring are known (see, e. g., [20]). For our software implementation we use a solution by Pointdexter [50] to find the required edge colorings.

Proposition 4.1.1. *Multiplication by a matrix $M \in \mathrm{GL}_n(\mathbb{F}_2)$, i. e., the map*

$$|u\rangle\ |v\rangle \longrightarrow |u\rangle\ |v + M \cdot u\rangle$$

with arbitrary input vectors $u, v \in \mathbb{F}_2^n$*, can be implemented with* $\mathrm{weight}(M)$ *CNOT gates. For this, an ancillae-free circuit of depth equal to the maximal Hamming weight of the rows and colums of* M *is sufficient.*

As worst-case bounds this implies the following.

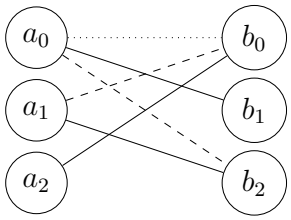
Corollary 4.1.1. *Multiplication by an arbitrary matrix* $M \in \mathrm{GL}_n(\mathbb{F}_2)$ *can be implemented with at most* $n^2 - n + 1$ *CNOT gates, using an ancillae-free circuit of depth at most* n*.*

Proof. Because of Proposition 4.1.1 it suffices to show that $\mathrm{weight}(M) \leq n^2 - n + 1$. Suppose this is not true, i. e., $\mathrm{weight}(M) \geq n^2 - n + 2$. Then M must contain at least two rows with all entries being equal to 1, as having $n - 1$ rows each of weight $\leq n - 1$ results in a matrix of weight $\leq n + (n-1) \cdot (n-1) = n^2 - n + 1$. To bring the weight to $n^2 - n + 2$ at least one more row must be completed to an all-one row. Thus M has two identical rows, which contradicts $M \in \mathrm{GL}_n(\mathbb{F}_2)$. □

Example 4.1.1 (Constant multiplication in $\mathbb{F}_8$)**.** *Consider* $n = 3$ *and* $p = 1 + x + x^3$*, i. e.,* $\mathbb{F}_{2^3} = \mathbb{F}_2[x]/(1 + x + x^3)$*. Then multiplying an arbitrary polynomial* $a = a_0 + a_1 x + a_2 x^2 + (p)$ *with* $1 + x + x^2 + (p)$ *can be interpreted as multiplying the coefficient vector* (a_0, a_1, a_2) *with the following matrix of weight* 6*:*

$$\begin{pmatrix} 1 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix}$$

From this matrix we obtain the subsequent graph with six vertices and six edges.



Consequently, we need a total of six CNOT gates, and in accordance with the matrix containing a row (and a column) of weight three, the graph has chromatic index 3*, yielding the quantum circuit shown in Figure 4.1. The first three CNOT gates can be executed simultaneously (solid edges), similarly the next two CNOT gates can be applied at the same time (dashed edges), and finally the last CNOT gate can be applied (dotted edge).*

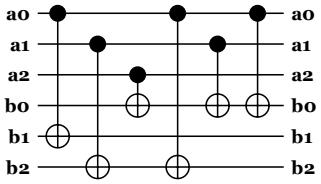


Figure 4.1: A circuit for ancillae-free multiplication of $a \in \mathbb{F}_2[x]/(1 + x + x^3)$ with $1 + x + x^2 + (1 + x + x^3)$.

Example 4.1.2 (Squaring in $\mathbb{F}_{128}$)**.** *Now let* $n = 7$ *and choose* $p = 1 + x + x^7$*. Squaring* $a_0 + a_1x + \cdots + a_6x^6 + (p) \in \mathbb{F}_2[x]/(p)$ *can be expressed as multiplying the coefficient vector* $(a_0, \ldots, a_6) \in \mathbb{F}_2^7$ *by the matrix*

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{pmatrix}.$$

This matrix has 10 *non-zero entries, and a maximal row or column weight of* 2*. We obtain the depth* 2 *circuit shown in Figure 4.2 which corresponds to the following bipartite graph with chromatic index* 2*:*

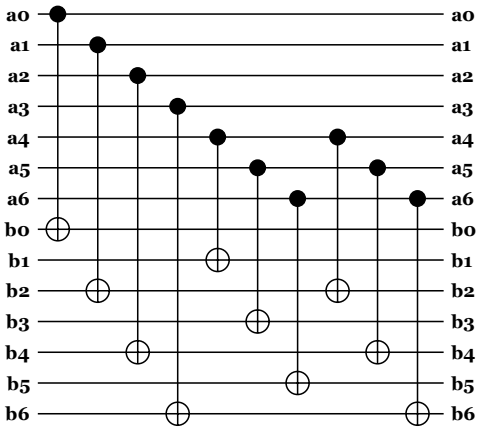
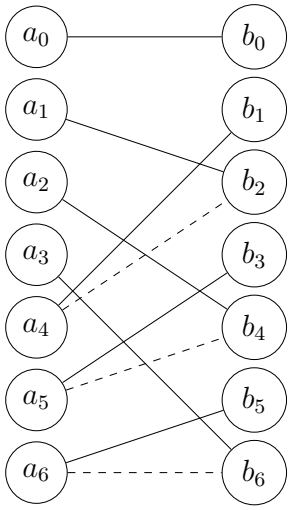


Figure 4.2: Ancillae-free squaring of $a \in \mathbb{F}_2[x]/(1 + x + x^7)$ in depth 2.



Example 4.1.3 (ECDSA: squaring)**.** *The Digital Signature Standard [45] specifies five different fields for use in connection with binary elliptic curves along with a polynomial-basis representation for each of these fields. We used our software to find the depth and number of CNOT gates needed for an ancillae-free squaring operation with each of these representations. The corresponding values are listed in Table 4.1.*

The last two examples suggest that trinomials are an attractive choice for deriving compact ancillae-free squaring circuits, and this is indeed the case. The same holds

irreducible polynomial	depth	CNOT gates
$1 + x^3 + x^6 + x^7 + x^{163}$	8	415
$1 + x^{74} + x^{233}$	3	386
$1 + x^5 + x^7 + x^{12} + x^{283}$	7	722
$1 + x^{87} + x^{409}$	3	656
$1 + x^2 + x^5 + x^{10} + x^{571}$	7	1438

Table 4.1: Resource count of an ancillae-free squaring operation for binary fields in [45].

true for computing the unique square root of an element in $\mathbb{F}_{2^n}$; the latter will be helpful for us, as the circuit used to establish Proposition 4.1.2 involves squarings as well as a square root computation for "uncomputing". To quantify the benefit of a "trinomial basis representation", first we can exploit that the irreducibility of $1 + x^m + x^n \in \mathbb{F}_2[x]$ (with $m < n$) implies the irreducibility of $1 + x^{n-m} + x^n \in \mathbb{F}_2[x]$ [41, Fact 4.75]. So we may choose the middle-term to be of degree $\leq \lfloor n/2 \rfloor$. From the explicit formulae for a classical implementation by Rodríguez-Henríquez et al. [53] we obtain the following.

Proposition 4.1.2. *Let $\mathbb{F}_{2^n} = \mathbb{F}_2[x]/(1 + x^m + x^n)$ with $m \leq \lfloor n/2 \rfloor$. Then the map*

$$|a\rangle\ |c\rangle \longmapsto |a\rangle\ |c + a^2\rangle$$

(with variable input $c \in \mathbb{F}_{2^n}$) can be implemented with an ancillae-free quantum circuit of depth $\leq m + 1$ using no more than $3n$ CNOT gates.

Moreover, the map $|a\rangle\ |c\rangle \longmapsto |a\rangle\ |c + \sqrt{a}\rangle$ can be implemented with an ancillae-free quantum circuit using no more than $5n$ CNOT gates.

Proof. Let $A := a_0 + a_1 x + \cdots + a_{n-1} x^{n-1}$ be a representative of an $\mathbb{F}_{2^n}$-element a. In [53] explicit expressions for computing the representations of a^2 and $\sqrt{a}$ from

$a_0, \ldots, a_{n-1}$ are given. Each coefficient of a^2 can be obtained as a sum of at most three a_is. Similarly, each coefficient of $\sqrt{a}$ can be obtained as a sum of no more than five a_is.

To justify the depth bound m for a squaring operation, let

$$B := a_0 + a_1 x^2 + a_2 x^4 + \cdots + a_{n-1} x^{2n-2}.$$

Then B is a representative of a^2, and the degree of B is $\leq 2n-2$. To find the coefficients of a^2, we have to find $B \bmod x^n + x^m + 1$, i. e., a representative of degree less than n. With

$$\eta := n + (n \bmod 2)$$

being the smallest even number greater or equal to n, we can write

$$\begin{aligned} B &= \underbrace{a_0 + a_1 x^2 + \cdots + a_{(\eta/2)-1} x^{\eta-2}}_{=:B_0} + \\ &\quad \underbrace{a_{\eta/2} x^{\eta} + \cdots + a_{n-1} x^{2n-2}}_{=:B_1}. \end{aligned}$$

No reduction is needed for B_0, and we have

$$\begin{aligned} B_1 &= x^n \cdot \left(a_{\eta/2} x^{\eta-n} + \cdots + a_{n-1} x^{n-2} \right) \\ &= (1 + x^m) \cdot \left(a_{\eta/2} x^{\eta-n} + \cdots + a_{n-1} x^{n-2} \right) \\ &= \underbrace{a_{\eta/2} x^{\eta-n} + \cdots + a_{n-1} x^{n-2}}_{=:B_{10}} + \\ &\quad \underbrace{a_{\eta/2} x^{\eta+m-n} + \cdots + a_{n-1} x^{m+n-2}}_{=:B_{11}} \end{aligned} .$$

No reduction is needed for B_{10}, and we can compute $B_0 + B_{10}$ in depth $2 - (n \bmod 2)$.We can reduce B_{11}, a polynomial of degree $\leq m + n - 2$, in the same way as we just did with B_1, and after at most $m-1$ reduction steps we obtain a representative of degree less than n. This increases the circuit depth at most by $m-1$, resulting in a total depth of at most $(m-1) + 2 - (n \bmod 2) \leq m + 1$. □

4.2 ADDING A FIXED POINT WITH REDUCED T-GATE COMPLEXITY

With the affine representation of an ordinary binary elliptic curve from Section 2.2, the group law is summarized in the following Algorithm 1, taken from [59]. At this $P_1 = (x_1, y_1) \in E_{a_2,a_6}(\mathbb{F}_{2^n})$ and $P_2 = (x_2, y_2) \in E_{a_2,a_6}(\mathbb{F}_{2^n})$.[2]

Data: Points $P_1 = (x_1, y_1)$ and $P_2 = (x_2, y_2)$ on $E_{a_2,a_6}(\mathbb{F}_{2^n})$.

Result: Point $P_3 = (x_3, y_3)$ with $P_3 = P_1 + P_2$.

if $P_1 = \mathcal{O}$ **then**
  return P_2

if $P_2 = \mathcal{O}$ **then**
  return P_1

if $x_1 = x_2$ **then**

  if $y_1 + y_2 = x_2$ **then** /* $P_1 = -P_2$ */
    return $\mathcal{O}$

  else /* $P_1 = P_2$ */

    $m = x_2 + y_2/x_2;$

    $x_3 = m^2 + m + a_2;$

    $y_3 = x_2^2 + (m + 1) \cdot x_3$

else /* $P_1 \neq \pm P_2$ */

  $m = (y_1 + y_2)/(x_1 + x_2);$

  $x_3 = m^2 + m + x_1 + x_2 + a_2;$

  $y_3 = (x_2 + x_3) \cdot m + x_3 + y_2$

return (x_3, y_3)

Algorithm 1: Adding two points on an ordinary binary elliptic curve using affine coordinates.

Kaye and Zalka [34] argue that to implement Shor's algorithm it is sufficient

[2] As $(0,0) \notin E_{a_2,a_6}(\mathbb{F}_{2^n})$, the neutral element $\mathcal{O}$ can be represented as $(0,0)$.

to provide a quantum circuit that implements the "generic branch" $P_1 \neq \pm P_2$ of Algorithm 1 for a fixed point P_2, and we restrict to this situation. To avoid the (costly) inversion operation, one usually implements this point addition in a projective representation. The standard projective representation $(X, Y, Z) \in \mathbb{F}_{2^n} \setminus \{(0, 0, 0)\}$ of an affine point (x, y) satisfies $x = X/Z$ and $y = Y/Z$. Here we follow a different convention, introduced by López and Dahab [37], that has also been used for the addition circuit in [3]: the affine point (x, y) is represented projectively by (X, Y, Z) with $x = X/Z$ and $y = Y/Z^2$. Accordingly, the curve given by Equation (2.1) would be expressed as

$$Y^2 + XYZ = X^3Z + a_2X^2Z^2 + a_6Z^4, \tag{4.1}$$

the identity element $\mathcal{O}$ being represented by $(X, 0, 0) \in \mathbb{F}_{2^n}^3 \setminus \{(0, 0, 0)\}$. Based on an addition formula by Higuchi and Takagi [30] for this type of projective representation, in [3] the following result is given, where

- $G_M(n)$ and $D_M(n)$ denote the number of gates and depth needed to implement an $\mathbb{F}_{2^n}$-multiplier, respectively;
- $G_M^T(n)$ and $D_M^T(n)$ denote the number of T-gates and T-depth needed to implement an $\mathbb{F}_{2^n}$-multiplier, respectively.

Proposition 4.2.1 ([3, Proposition 3.2])**.** *Let P_2 be a fixed point on $E_{a_2,a_6}(\mathbb{F}_{2^n})$. With the above-mentioned variant of projective coordinates, the addition*

$$|X_1\rangle\ |Y_1\rangle\ |Z_1\rangle\ |0\rangle\ |0\rangle\ |0\rangle \longmapsto |X_1\rangle\ |Y_1\rangle\ |Z_1\rangle\ |X_3\rangle\ |Y_3\rangle\ |Z_3\rangle$$

can be carried out with a quantum circuit $\mathcal{C}$ satisfying all of the following:

- *The total number of T-gates in $\mathcal{C}$ is $13 \cdot G_M^T(n)$.*
- *The total number of gates in $\mathcal{C}$ is at most $13 \cdot G_M(n)$ plus $12n^2 + \mathrm{O}(n)$ (the latter being CNOT gates).*

- *The T-depth of $\mathcal{C}$ is $4 \cdot D_M^T(n)$.*
- *The overall depth of $\mathcal{C}$ is $4 \cdot D_M(n)$ plus $4n + \mathrm{O}(1)$ (the latter being CNOT gates).*

This includes the cost of cleaning up ancillae. If (X_1, Y_1, Z_1) is not the identity or equal to $\pm P_2$, then (X_3, Y_3, Z_3) is a representation of the sum of (X_1, Y_1, Z_1) and the fixed point P_2 in the above-mentioned variant of projective coordinates.

To the best of our knowledge, in terms of T-gate complexity this is currently the most efficient quantum circuit that has been published for the "generic addition" of a fixed point on an ordinary binary elliptic curve.

4.2.1 An addition circuit based on a formula by Al-Daoud et al.

Invoking López-Dahab coordinates as described above, in [2] Al-Daoud et al. present a point addition formula which seems well suited for a quantum circuit that aims at adding a fixed point. Besides requiring only four general multiplications, in two cases a constant multiplication and a squaring operation can naturally be combined into a single matrix-vector multiplication. More specifically, let $P_2 = (x_2, y_2, 1)$ be a fixed point on the curve given by Equation (4.1), and let $P_1 = (X_1, Y_1, Z_1)$ be an arbitrary point on this curve (which will be given as input to our quantum circuit). We assume that $\mathcal{O} \neq P_1 \neq \pm P_2$. Then a representation (X_3, Y_3, Z_3) of the sum $P_1 + P_2$ can be

computed as follows.

$$\begin{aligned} A &= Y_1 + y_2 Z_1^2, \\ B &= X_1 + x_2 Z_1, \\ C &= B \cdot Z_1, \\ Z_3 &= C^2, \\ D &= x_2 Z_3, \\ X_3 &= A^2 + C \cdot (A + B^2 + a_2 C), \\ Y_3 &= (D + X_3) \cdot (A \cdot C + Z_3) + \\ &\quad (y_2 + x_2) Z_3^2 \end{aligned}$$

The above formulation is taken from the *explicit-formulas database* [9, *madd-2005-dl*] (see also [19, Chapter 13.3.1.d]). To characterize the complexity of our addition circuit, it is appropriate to distinguish between the resources for general multiplication, squaring, and other matrix-vector multiplications. As manifested in Proposition 4.1.2 and Table 4.1, for certain field representations the resource count of a squaring operation is remarkably modest, even for cryptographically significant field sizes. So in the sequel we write $G_S(n) \leq n^2 - n + 1$ for the number of (CNOT) gates needed to implement a squaring operation with the underlying representation of $\mathbb{F}_{2^n}$ and analogously $D_S(n) \leq n$ for the depth of such a circuit. The number of qubits needed in our construction will depend on the details of the underlying $\mathbb{F}_{2^n}$-multiplier. So to quantify the number of qubits, we assume that the multiplication of any $a, b \in \mathbb{F}_{2^n}$—i. e., the function $|a\rangle\ |b\rangle\ |c\rangle \mapsto |a\rangle\ |b\rangle\ |c + a \cdot b\rangle$ with $c \in \mathbb{F}_{2^n}$ arbitrary—is realized with

$$\underbrace{n + n}_{\text{input}} + \underbrace{n}_{\text{output}} + \underbrace{A_M(n)}_{\text{ancillae}}$$

qubits. With this notation we obtain the following.

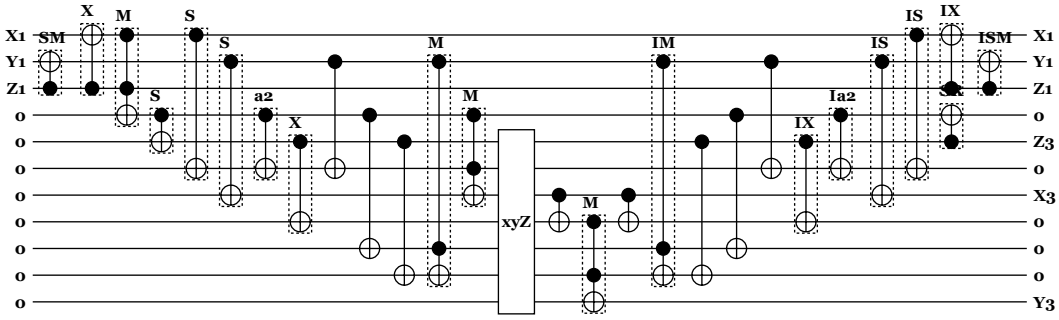


Figure 4.3: A complete circuit adding a point in López-Dahab coordinates on $E_{1,1}(\mathbb{F}_2)$ with the fixed affine point $(1,1)$. Parsing the circuit from left to right, the initial gates labeled **SM**, **X**, and **M** correspond to the operations in Steps 1–3 of the proof of Theorem 4.2.1. The subsequent three gates (labeled **S**) implement the parallel squarings in Step 4. This is followed by the parallel scalar multiplications with a_2 (**a2**) and x_2 (**X**) from Step 5 and three CNOT gates operating on disjoint wires to implement Step 6. The two multipliers in Step 7 are realized by two Toffoli gates (marked **M**), and for the sake of completeness we include a box labelled **xyZ** which is actually the identity as for our specific example the value of $x_2 + y_2$ in Step 7 is 0. Step 8 corresponds to a single CNOT gate, and the subsequent Toffoli gate labelled **M** implements Step 9. Starting the clean-up part of the circuit, the CNOT from Step 10 is used, followed by the reversal of a multiplier in Step 11 (**IM**). Step 12 results in three CNOT gates. This is followed by the reversal of the scalar multiplications in Step 13 (**IX** and **Ia2**). Reversing of the squaring operations from Step 14 is implemented by the two gates marked **IS**. They can be executed in parallel with the gate marked **SR**, realizing the square root computation in Step 14. Eventually, Step 15 corresponds to the gate labelled **IX**, and Step 16 is realized by a singe CNOT gate (**ISM**).

Theorem 4.2.1. *Let P_2 be a fixed point on the curve $E_{a_2,a_6}(\mathbb{F}_{2^n})$. Using López-Dahab coordinates, the addition $|X_1\rangle\ |Y_1\rangle\ |Z_1\rangle\ |0\rangle\ |0\rangle\ |0\rangle \longmapsto |X_1\rangle\ |Y_1\rangle\ |Z_1\rangle\ |X_3\rangle\ |Y_3\rangle\ |Z_3\rangle$ of this point can be carried out with a quantum circuit $\mathcal{C}$ satisfying all of the following:*

- *The total number of T-gates in $\mathcal{C}$ is $5G_M^T(n)$.*
- *The total number of gates in $\mathcal{C}$ is at most $5G_M(n)$ plus $5G_S(n) + 10n^2 - 2n + 10$ (the latter being CNOT gates).*
- *The T-depth of $\mathcal{C}$ is $4D_M^T(n)$.*

- *The overall depth of* $\mathcal{C}$ *is* $3D_M(n) + \max(D_M(n), n)$ *plus* $D_S(n) + 7n + 4$ *(the latter being CNOT gates).*

- *The total number of qubits, including the* $3n$ *qubits for storing the input, is* $11n + 4A_M(n)$*.*

This includes the cost to clean up ancillae. If (X_1, Y_1, Z_1) *is not the identity or equal to* $\pm P_2$*, then* (X_3, Y_3, Z_3) *is a representation in López-Dahab coordinates of the sum of* (X_1, Y_1, Z_1) *with* P_2*.*

Proof. To find X_3, Y_3 and Z_3 we proceed as follows.

1. Using $\leq (n^2 - n + 1)$-CNOT gates and depth $\leq n$, we can compute $A = Y_1 + y_2 \cdot Z_1^2$, i. e., the wires originally storing Y_1 now store A.

2. Similarly, we now compute $B = X_1 + x_2 \cdot Z_1$ using $\leq n^2 - n + 1$ CNOT gates in depth $\leq n$, storing B in the wires originally holding X_1.

3. Multiplying B and Z_1 we store $C = B \cdot Z_1$ into a new set of n wires (initialized to $|0\rangle$). This increases the depth by $D_M(n)$ and uses $G_M(n)$ gates. Similarly the T-depth and number of T-gates are increased by $D_M^T(n)$ and $G_M^T(n)$, respectively.

4. Square C, A and B to obtain Z_3, A^2 and B^2 in parallel in depth $D_S(n)$ by using $3G_S(n)$-CNOT gates. To store the results we add $3n$ additional ($|0\rangle$-initialized) qubits. The wires holding A^2 will be used to store X_3.

5. Now compute $a_2 \cdot C$ in depth $\leq n$ using $\leq n^2 - n + 1$-CNOT gates. The result of this operation is added directly to B^2. At the same time, compute $D = x_2 \cdot Z_3$ using another $\leq n^2 - n + 1$ CNOT gates. The latter result is stored in n new ($|0\rangle$-initialized) qubits.

6. Add A to the qubits holding $B^2 + a_2 \cdot C$. For this, n CNOT gates and depth 1 suffice. Simultaneously we can apply $n + n$ more CNOT gates to create a copy C' of C and a copy Z_3' of Z_3 in a set of $n+n$ new ($|0\rangle$-initialized) qubits. Having C' available enables us to perform the next two multiplications in parallel.

7. With $\leq 2G_M(n)+n^2-n+1$ gates and increasing the depth by $\leq \max(D_M(n), n)$, we can now compute in parallel

   - $C \cdot (A + B^2 + a_2 \cdot C)$ and add the result onto the wires for the value X_3;
   - $A \cdot C'$ and add the result onto the wires holding Z_3';
   - $(x_2 + y_2) \cdot Z_3^2$ and store the result in the ($|0\rangle$-initialized) wires for the value Y_3.

   This step increases the T-depth by $D_M^T(n)$ and the number of T-gates by $2G_M^T(n)$.

8. With n CNOT gates in depth 1 we can add X_3 to the wires storing D.

9. Find $(D + X_3) \cdot (A \cdot C' + Z_3')$ using $G_M(n)$ gates which increases the depth by $D_M(n)$ units. The result is added to the wires which are to hold Y_3. This step increases the number of T-gates by $G_M(n)$ and the T-depth by D_M^T.

   At this point we have computed all of X_3, Y_3 and Z_3, and we are left with cleaning up ancillae and restoring the input values.

10. Add X_3 to the wires holding $D + X_3$ for which we need n CNOT gates and which increases the circuit depth by 1.

11. Reversing the multiplication $A \cdot C'$ takes depth $D_M(n)$ and requires $G_M(n)$ gates. This also increases the T-depth by $D_M^T(n)$ and the number of T-gates accordingly by $G_M^T(n)$.

12. To reverse the CNOT operations from Step 6 we execute them again in depth 1, using $3n$ CNOT gates.

13. Next, the two linear operations from Step 5 can be run backwards simultaneously, increasing the depth by $\leq n$ and adding $\leq 2 \cdot (n^2 - n + 1)$ CNOT gates to the total gate count.

14. The squarings of A and B can be run backwards simultaneously using $2G_S(n)$ gates. Simultaneously we can apply a square root computation to $Z_3 = C^2$ to cancel the output C of the multiplier in Step 3. The square root computation can be done with $\leq n^2 - n + 1$ CNOT gates, and with $D_S(n) \leq n$ we see that the overall depth of this step is $\leq n$.

15. Reversing the computation of B takes $\leq n^2 - n + 1$ CNOT gates and can be completed in depth $\leq n$.

16. Finally, reversing the computation of A increases the gate count by $\leq n^2 - n + 1$ CNOT gates and the depth by n.

Table 4.2 summarizes the resource count for each of these steps. In the column for the number of qubits we count all qubits that are used on top of the $3n$ qubits necessary to represent the input (X_1, Y_1, Z_1); this includes the $3n$ bits needed to store the result (X_3, Y_3, Z_3). Exploiting that the multipliers are the only parts of the circuit involving T-gates, from this table we immediately obtain the bounds claimed.

□

step	no. gates	depth	no. qubits
1	$n^2 - n + 1$	n	0
2	$n^2 - n + 1$	n	0
3	$G_M(n)$	$D_M(n)$	$n + A_M(n)$
4	$3G_S(n)$	$D_S(n)$	$3n$
5	$2 \cdot (n^2 - n + 1)$	n	n
6	$3n$	1	$2n$
7	$2G_M(n) + n^2 - n + 1$	$\max(D_M(n), n)$	$n + 2A_M(n)$
8	n	1	0
9	$G_M(n)$	$D_M(n)$	$A_M(n)$
10	n	1	0
11	$G_M(n)$	$D_M(n)$	0
12	$3n$	1	0
13	$2 \cdot (n^2 - n + 1)$	n	0
14	$2 \cdot G_S(n) + n^2 - n + 1$	n	0
15	$n^2 - n + 1$	n	0
16	$n^2 - n + 1$	n	0

Table 4.2: Resource bounds for each step of the circuit in the proof of Theorem 4.2.1.

Remark 4.2.1. *Proposition 4.2.1 does not give an explicit count for the number of qubits, but the proof of [3, Proposition 3.2] emphasizes parallelization. Step 1–3 of the latter already add* $6n$ *new wires to the* $3n$ *qubits for the input. Step 4 then executes* 4 *field multiplications in parallel (invoking* $4A_M(n)$ *ancillae), and Step 6 runs three more multipliers in parallel, storing the result in* $3n$ *new wires, so it is fair to conclude that the total number of qubits is larger than the bound* $11n + 4A_M(n)$ *established in Theorem 4.2.1.*

Also, it is worth noting that the resource bounds in Theorem 4.2.1 are indeed worst-case bounds. In cryptographic applications it is common to choose $a_2 \in \{0, 1\}$*, thereby eliminating the need to implement the (*T*-gate free) computation of* $a_2 \cdot C$*.*

The proof of Theorem 4.2.1 is constructive, and we implemented a software tool which for a given irreducible polynomial $p \in \mathbb{F}_2[x]$, a curve point $Q \in E_{a_2,a_6}(\mathbb{F}_2[x]/(p))$ and $a_2 \in \mathbb{F}_2[x]/(p)$ generates the corresponding quantum circuit for the "generic addition" of Q. As programming language we chose `Python`, and the resulting quantum circuits are stored in a text file using the `.qc` format. This format supports the grouping of gates into subcircuits, allowing a user to hide the details of, e. g., a field multiplier, when viewing the circuit in `QCViewer`. Figure 4.3 gives an example of a complete addition circuit using the curve $E_{1,1}(\mathbb{F}_2)$ and $P = (1, 1)$ as fixed point to be added. The $\mathbb{F}_2$-multiplier is realized as a Toffoli gate, requiring $A_M(1) = 0$ ancillae. As detailed by Amy et al. in [5], a Toffoli gate can be decomposed into a circuit involving a total of $G_M(1) = 15$ gates, $G_M^T(1) = 7$ of which are T-gates and the remaining ones being CNOT and Hadamard gates. This can be done with a T-depth of $D_M^T(1) = 4$ and an overall depth of $D_M(1) = 8$.

By means of our software, we also experimented with larger curves. For larger curves, the detailed T-gate complexity of the circuit depends very much on the complexity of the underlying $\mathbb{F}_{2^n}$-multiplier. For our experiments we built on an existing

`Python` code by Brittanney Amento to produce a `.qc` description of an $\mathbb{F}_{2^n}$-multiplier. Our software treats the multiplier basically as a black box, however. So if improved quantum circuits for $\mathbb{F}_{2^n}$-multiplication become available, integrating them with the existing code should not be a problem.

As final example, we take a look at the square root computation (see Step 14 in the proof of Theorem 4.2.1) for binary fields in the Digital Signature Standard:

Example 4.2.1 (ECDSA: square root computation)**.** *Table 4.3 lists depth and gate counts for the ancillae-free square root computation for the binary fields in [45]. As can be seen, for the case of a "trinomial basis" this operation can be implemented quite efficiently.*

irreducible polynomial	depth	CNOT gates
$1 + x^3 + x^6 + x^7 + x^{163}$	104	7399
$1 + x^{74} + x^{233}$	6	591
$1 + x^5 + x^7 + x^{12} + x^{283}$	94	11657
$1 + x^{87} + x^{409}$	2	613
$1 + x^2 + x^5 + x^{10} + x^{571}$	273	76172

Table 4.3: Resource count of an ancillae-free square root computation for binary fields in [45].

Chapter 5

Conclusion

Assuming the availability of pairing-friendly groups with a discrete logarithm trapdoor, we show how a very efficient identity-based signature scheme with short signatures can be obtained from such curves. This scheme is proved secure in the random oracle model based on the Strong Diffie-Hellman problem in the domain of used pairing.

The presented quantum circuit for point addition reduces an important cost parameter over the best previous solution—the number of T-gates can be reduced by more than 60% without increasing T-depth. At the same time, the number of qubits can be reduced. The overall depth increases linearly, but in view of the savings achieved the depth increase looks acceptable. Aiming at the implementation of elliptic curve arithmetic for cryptanalytic applications, the ability to synthesize (optimized) point addition circuits automatically seems very helpful. We also hope that the concrete complexity bounds provided along with the capability to derive actual circuits in an established format simplifies quantitative comparisons and stimulates follow-up research on more efficient implementations.

BIBLIOGRAPHY

[1] The case for elliptic curve cryptography. Available at http://www.nsa.gov/business/programs/elliptic_curve.shtml, Jan 2009.

[2] Essame Al-Daoud, Ramlan Mahmod, Mohammad Rushdan, and Adem Kilicman. A New Addition Formula for Elliptic Curves over GF(2^n). *IEEE Transactions on Computers*, 51(8):972–975, August 2002.

[3] Brittanney Amento, Martin Rötteler, and Rainer Steinwandt. Efficient quantum circuits for binary elliptic curve arithmetic: reducing T-gate complexity. *Quantum Information & Computation*, 13:631–644, July 2013.

[4] Brittanney Amento, Martin Rötteler, and Rainer Steinwandt. Quantum binary field inversion: improved circuit depth via choice of basis representation. *Quantum Information & Computation*, 13:116–134, January 2013.

[5] Matthew Amy, Dmitri Maslov, Michele Mosca, and Martin Roetteler. A Meet-in-the-Middle Algorithm for Fast Synthesis of Depth-Optimal Quantum Circuits. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 32(6):818–830, June 2013. For a preprint version see [6].

[6] Matthew Amy, Dmitri Maslov, Michele Mosca, and Martin Roetteler. A meet-in-the-middle algorithm for fast synthesis of depth-optimal quantum circuits. arXiv:quant-ph/1206.0758v3, January 2013. Available at http://arxiv.org/abs/1206.0758v3.

[7] Paulo S.L.M. Barreto, Benoît Libert, Noel McCullagh, and Jean-Jacques Quisquater. Efficient and Provably-Secure Identity-Based Signatures and Signcryption from Bilinear Maps. In Bimal Roy, editor, *Advances in Cryptology – ASIACRYPT 2005*, Lecture Notes in Computer Science, pages 515–532. Springer, 2005.

[8] Stéphane Beauregard, Gilles Brassard, and José M. Fernandez. Quantum Arithmetic on Galois Fields. arXiv:quant-ph/0301163v1, January 2003. Available at http://arxiv.org/abs/quant-ph/0301163v1.

[9] Daniel J. Bernstein and Tanja Lange. Explicit-formulas database. http://www.hyperelliptic.org/EFD/index.html.

[10] Daniel J. Bernstein, Tanja Lange, and Reza Rezaeian Farashahi. Binary Edwards Curves. In Elisabeth Oswald and Pankaj Rohatgi, editors, *Cryptographic Hardware and Embedded Systems – CHES 2008*, volume 5154 of *Lecture Notes in Computer Science*, pages 244–265. International Association for Cryptologic Research, Springer, 2008.

[11] Dan Boneh and Xavier Boyen. Short Signatures Without Random Oracles and the SDH Assumption in Bilinear Groups. *Journal of Cryptology*, 21:149–177, 2008.

[12] Dan Boneh and Matt Franklin. Identity-based encryption from the weil pairing. In Joe Kilian, editor, *Advances in Cryptology CRYPTO 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 213–229. Springer Berlin Heidelberg, 2001.

[13] Dan Boneh, Ben Lynn, and Hovav Shacham. Short Signatures from the Weil Pairing. In Colin Boyd, editor, *Advances in Cryptology – ASIACRYPT 2001*, volume 2248 of *Lecture Notes in Computer Science*, pages 514–532. Springer-Verlag, 2001.

[14] Daniel R. L. Brown and Robert P. Gallant. The Static Diffie-Hellman Problem. Cryptology ePrint Archive: Report 2004/306, June 2005. http://eprint.iacr.org/2004/306.

[15] Jae Choon Cha and Jung Hee Cheon. An Identity-Based Signature from Gap Diffie-Hellman Groups. In Yvo G. Desmedt, editor, *Public Key Cryptography – PKC 2003*, volume 2567 of *Lecture Notes in Computer Science*, pages 18–30. Springer, 2003.

[16] Jung Hee Cheon. Security Analysis of the Strong Diffie-Hellman Problem. In Serge Vaudenay, editor, *Advances in Cryptology – EUROCRYPT 2006*, volume 4004 of *Lecture Notes in Computer Science*. Springer, 2006.

[17] Sherman S.M. Chow, Siu-Ming Yiu, Lucas C.K. Hui, and K.P. Chow. Efficient Forward and Provably Secure ID-Based Signcryption Scheme with Public Verifiability and Public Ciphertext Authenticity. In Jong-In Lim and Dong-Hoon Lee, editors, *Information Security and Cryptology – ICISC 2003*, volume 2971 of *Lecture Notes in Computer Science*, pages 352–369. Springer, 2004.

[18] Clifford Cocks. An identity based encryption scheme based on quadratic residues. In Bahram Honary, editor, *Cryptography and Coding*, volume 2260 of *Lecture Notes in Computer Science*, pages 360–363. Springer Berlin Heidelberg, 2001.

[19] Henri Cohen and Gerhard Frey, editors. *Handbook of Elliptic and Hyperelliptic Curve Cryptography*. Discrete mathematics and its applications. Chapman & Hall/CRC, 2006.

[20] Richard Cole, Kirstin Ost, and Stefan Schirra. Edge-coloring bipartite multigraphs in $O(E \log D)$ time. *Combinatorica*, 21(1):5–12, 2001.

[21] Alexander W. Dent and Steven D. Galbraith. Hidden Pairings and Trapdoor DDH Groups. In Florian Hess, Sebastian Pauli, and Michael E. Pohst, editors, *Algorithmic Number Theory, 7th International Symposium, ANTS-VII*, volume 4076 of *Lecture Notes in Computer Science*, pages 436–451. Springer-Verlag, 2006.

[22] W. Diffie and M. Hellman. New directions in cryptography. *IEEE Trans. Inf. Theor.*, 22(6):644–654, September 2006.

[23] Andreas Enge. *Elliptic Curves and Their Applications to Cryptography: An Introduction*. Kluwer Academic Publishers, Norwell, MA, USA, 1999.

[24] Institute for Quantum Computing. QCViewer. http://qcirc.iqc.uwaterloo.ca/index.php?n=Projects.QCViewer, 2013.

[25] Python Software Foundation. Python Programming Language – Official Website. http://www.python.org, 2013.

[26] David Freeman, Michael Scott, and Edlyn Teske. A Taxonomy of Pairing-Friendly Elliptic Curves. *Journal of Cryptology*, 23(2):224–280, 2010.

[27] Gerhard Frey and Hans-Georg Rück. A remark concerning m-divisibility and the discrete logarithm in the divisor class group of curves. *Math. Comput.*, 62(206):865–874, April 1994.

[28] Jozef Gruska. *Quantum Computing*. McGraw Hill, 1999.

[29] Florian Hess. Efficient Identity Based Signature Schemes Based on Pairings. In Kaisa Nyberg and Howard Heys, editors, *Selected Areas in Cryptography – SAC 2002*, volume 2595 of *Lecture Notes in Computer Science*, pages 310–324. Springer, 2003.

[30] Akira Higuchi and Naofumi Takagi. A fast addition algorithm for elliptic curve arithmetic using projective coordinates. *Information Processing Letters*, 76:101–103, 2000.

[31] David Jao and Kayo Yoshida. Boneh-Boyen Signatures and the Strong Diffie-Hellman Problem. In Hovav Shacham and Brent Waters, editors, *Paring-Based Cryptography – Pairing 2009*, volume 5671 of *Lecture Notes in Computer Science*, pages 1–16. Springer, 2009.

[32] Antoine Joux. A One Round Protocol for Tripartite Diffie-Hellman. *Journal of Cryptology*, 17(4):263–276, 2004.

[33] Mark Joye and Gregory Neven, editors. *Identity-Based Cryptography*. Cryptology and Information Security. IOS Press, 2008.

[34] Phillip Kaye and Christof Zalka. Optimized quantum implementation of elliptic curve arithmetic over binary fields. arXiv:quant-ph/0407095v1, July 2004. Available at http://arxiv.org/abs/quant-ph/0407095v1.

[35] Neal Koblitz. Elliptic curve cryptosystems. *Mathematics of computation*, 48(177):203–209, 1987.

[36] Jonathan Katz & Yehuda Lindell. *Introduction to Modern Cryptography*. Cryptography & network security. Chapman & Hall/CRC, 2007.

[37] Julio López and Ricardo Dahab. Improved Algorithms for Elliptic Curve Arithmetic in $GF(2^n)$. In Stafford Tavares and Henk Meijer, editors, *Selected Areas in Cryptography – SAC'98*, volume 1556 of *Lecture Notes in Computer Science*, pages 201–212. Springer, 1999.

[38] Dmitri Maslov, Jimson Mathew, Donny Cheung, and Dhiraj K. Pradhan. An $O(m^2)$-depth quantum algorithm for the elliptic curve discrete logarithm problem over GF(2^m). *Quantum Information & Computation*, 9(7):610–621, 2009. For a preprint version see [39].

[39] Dmitri Maslov, Jimson Mathew, Donny Cheung, and Dhiraj K. Pradhan. On the Design and Optimization of a Quantum Polynomial-Time Attack on Elliptic Curve Cryptography. arXiv:0710.1093v2, February 2009. Available at http://arxiv.org/abs/0710.1093v2.

[40] Alfred J. Menezes, Paul C. van Oorschot, and Scott A. Vanstone. Reducing elliptic curve logarithms to logarithms in a finite field. *Information Theory, IEEE Transactions on*, 39(5):1639–1646, Sep 1993.

[41] Alfred J. Menezes, Paul C. van Oorschot, and Scott A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, August 2001. Sample chapters available at http://cacr.uwaterloo.ca/hac/.

[42] Victor S. Miller. Use of elliptic curves in cryptography. In HughC. Williams, editor, *Advances in Cryptology CRYPTO 85 Proceedings*, volume 218 of *Lecture Notes in Computer Science*, pages 417–426. Springer Berlin Heidelberg, 1986.

[43] National Institute of Standards and Technology, Gaithersburg, MD 20899-8900. *Recommendation for Key Management*, July 2012. Available at http://csrc.nist.gov/publications/nistpubs/800-57/sp800-57_part1_rev3_general.pdf.

[44] National Institute of Standards and Technology, Gaithersburg, MD 20899-8930. *Recommendation for Key Management: Part 1: General (Revision 3)*, July 2012. Available at http://csrc.nist.gov/publications/nistpubs/800-57/sp800-57_part1_rev3_general.pdf.

[45] National Institute of Standards and Technology, Gaithersburg, MD 20899-8900. *FIPS PUB 186-4. Federal Information Processing Standard Publication. Digital Signature Standard (DSS)*, July 2013. Available at http://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.186-4.pdf.

[46] Pascal Paillier. Trapdooring Discrete Logarithms on Elliptic Curves over Rings. In Tatsuaki Okamoto, editor, *Advances in Cryptology – ASIACRYPT 2000*, volume 1976 of *Lecture Notes in Computer Science*, pages 573–584. Springer-Verlag, 2000.

[47] Cheol-Min Park and Hyang-Sook Lee. Pairing-friendly curves with minimal security loss by cheon's algorithm. *ETRI Journal*, 33(4):656 – 659, August 2011.

[48] Kenneth G. Paterson. ID-based signatures from pairings on elliptic curves. *Electronics Letters*, 38(18):1025–1026, 2002.

[49] Christof Paar & Jan Pelzl. *Understanding Cryptography*. Springer, 2010.

[50] Alain Pointdexter. edge-coloring of a bipartite graph (Python recipe). Available at http://code.activestate.com/recipes/498092-edge-coloring-of-a-bipartite-graph/, September 2013.

[51] John Proos and Christof Zalka. Shor's discrete logarithm quantum algorithm for elliptic curves, 2003.

[52] Ron Rivest, Adi Shamir, and Leonard Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM*, 21(2):120–126, February 1978.

[53] Francisco Rodríguez-Henríquez, Guillermo Morales-Luna, and Julio López. Low-Complexity Bit-Parallel Square Root Computation over $GF(2^m)$ for all Trinomials. *IEEE Transactions on Computers*, 57(4):472–480, April 2008. For a preprint version see [54].

[54] Francisco Rodríguez-Henríquez, Guillermo Morales-Luna, and Julio López-Hernández. Low Complexity Bit-Parallel Square Root Computation over $GF(2^m)$ for all Trinomials. Cryptology ePrint Archive: Report 2006/133, April 2006. Available at http://eprint.iacr.org/2006/133.

[55] Martin Rötteler and Rainer Steinwandt. A quantum circuit to find discrete logarithms on ordinary binary elliptic curves in depth $O(\log^2 n)$. *Quantum Information & Computation*, 14(9 & 10):888–900, 2014.

[56] Ryuichi Sakai, Kiyoshi Ohgishi, and Masao Kasahara. Cryptosystems Based on Pairing. In *The 2000 Symposium on Cryptography and Information Security (SCIS)*, volume C20, 2000.

[57] Adi Shamir. Identity-based crypto systems and signature schemes. In G. R. Blakley and David Chaum, editors, *Advances in Cryptology – CRYPTO'84*, volume 196 of *Lecture Notes in Computer Science*, pages 47–53. Springer-Verlag, 1985.

[58] Peter W. Shor. Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer. *SIAM Journal on Computing*, 26(5):1484–1509, 1997.

[59] Jerome A. Solinas. An Improved Algorithm for Arithmetic on a Family of Elliptic Curves. In Burton S. Kaliski Jr., editor, *Advances in Cryptology – CRYPTO '97*, volume 1294 of *Lecture Notes in Computer Science*, pages 357–371. Springer, 1997.

[60] Edlyn Teske. An Elliptic Curve Trapdoor System. *Journal of Cryptology*, 19(1):115–133, 2006.