UNIVERSIDAD DE INVESTIGACIÓN DE TECNOLOGÍA EXPERIMENTAL YACHAY

Escuela de Ciencias Matemáticas y Computacionales

TÍTULO: On the Theory, Design, and Optimization of Classical and Quantum Algorithms for Isogeny-based Cryptography

Trabajo de integración curricular presentado como requisito para la obtención
del título de Matemático

Autor:

Caamaño Mayorga Renzo Sebastián

Tutor:

Dr. Anton Casto, Francesc, Ph.D.

Urcuquí, abril del 2021

Urcuquí, 30 de abril de 2021

SECRETARÍA GENERAL
(Vicerrectorado Académico/Cancillería)
ESCUELA DE CIENCIAS MATEMÁTICAS Y COMPUTACIONALES
CARRERA DE MATEMÁTICA
ACTA DE DEFENSA No. UITEY-ITE-2021-00005-AD

A los 30 días del mes de abril de 2021, a las 08:00 horas, de manera virtual mediante videoconferencia, y ante el Tribunal Calificador, integrado por los docentes:

Presidente Tribunal de Defensa	Dr. MAYORGA ZAMBRANO, JUAN RICARDO , Ph.D.
Miembro No Tutor	Dr. ARIZA GARCIA, EUSEBIO ALBERTO , Ph.D.
Tutor	Dr. ANTON CASTRO , FRANCESC , Ph.D.

El(la) señor(ita) estudiante **CAAMAÑO MAYORGA, RENZO SEBASTIAN**, con cédula de identidad No. **1726083718**, de la **ESCUELA DE CIENCIAS MATEMÁTICAS Y COMPUTACIONALES**, de la Carrera de **MATEMÁTICA**, aprobada por el Consejo de Educación Superior (CES), mediante Resolución **RPC-SO-15-No.174-2015**, realiza a través de videoconferencia, la sustentación de su trabajo de titulación denominado: **On the theory, design and optimization of Classical and Quantum algorithms for Isogenybased Cryptography.** , previa a la obtención del título de **MATEMÁTICO/A**.

El citado trabajo de titulación, fue debidamente aprobado por el(los) docente(s):

Tutor	Dr. ANTON CASTRO , FRANCESC , Ph.D.

Y recibió las observaciones de los otros miembros del Tribunal Calificador, las mismas que han sido incorporadas por el(la) estudiante.

Previamente cumplidos los requisitos legales y reglamentarios, el trabajo de titulación fue sustentado por el(la) estudiante y examinado por los miembros del Tribunal Calificador. Escuchada la sustentación del trabajo de titulación a través de videoconferencia, que integró la exposición de el(la) estudiante sobre el contenido de la misma y las preguntas formuladas por los miembros del Tribunal, se califica la sustentación del trabajo de titulación con las siguientes calificaciones:

Tipo	Docente	Calificación
Tutor	Dr. ANTON CASTRO , FRANCESC , Ph.D.	10,0
Miembro Tribunal De Defensa	Dr. ARIZA GARCIA, EUSEBIO ALBERTO , Ph.D.	10,0
Presidente Tribunal De Defensa	Dr. MAYORGA ZAMBRANO, JUAN RICARDO , Ph.D.	10,0

Lo que da un promedio de: **10 (Diez punto Cero)**, sobre 10 (diez), equivalente a: **APROBADO**

Para constancia de lo actuado, firman los miembros del Tribunal Calificador, el/la estudiante y el/la secretario ad-hoc.

Certifico que *en cumplimiento del Decreto Ejecutivo 1017 de 16 de marzo de 2020, la defensa de trabajo de titulación (o examen de grado modalidad teórico práctica) se realizó vía virtual, por lo que las firmas de los miembros del Tribunal de Defensa de Grado, constan en forma digital.*

Firmado electrónicamente por:
RENZO SEBASTIAN CAAMANO MAYORGA
Razón:
Localización: Quito-Ecuador

CAAMAÑO MAYORGA, RENZO SEBASTIAN
Estudiante

Firmado electrónicamente por:
JUAN RICARDO MAYORGA ZAMBRANO

Dr. MAYORGA ZAMBRANO, JUAN RICARDO , Ph.D.
Presidente Tribunal de Defensa

Dr. ANTON CASTRO , FRANCESC , Ph.D.
Tutor

FRANCESC ANTON CASTRO

Signé électroniquement par
FRANCESC ANTON CASTRO
cn=FRANCESC ANTON CASTRO, ou= ENTIDAD DE CERTIFICACION DE INFORMACION, o= SECURITY DATA S.A. 1, c= EC
Date: 2021.05.12 00:45:04 ECT

Dr. ARIZA GARCIA, EUSEBIO ALBERTO , Ph.D.
Miembro No Tutor

Firmado electrónicamente por:
EUSEBIO ALBERTO ARIZA GARCIA

TATIANA BEATRIZ TORRES MONTALVAN
Firmado digitalmente por TATIANA BEATRIZ TORRES MONTALVAN
Fecha: 2021.05.10 14:27:58 -05'00'

TORRES MONTALVÁN, TATIANA BEATRIZ
Secretario Ad-hoc

Autoría

Yo, **Renzo Sebastián Caamaño Mayorga**, con cédula de identidad **1726083718**, declaro que las ideas, juicios, valoraciones, interpretaciones, consultas bibliográficas, definiciones y conceptualizaciones expuestas en el presente trabajo; así cómo, los procedimientos y herramientas utilizadas en la investigación, son de absoluta responsabilidad de el autor del trabajo de integración curricular. Así mismo, me acojo a los reglamentos internos de la Universidad de Investigación de Tecnología Experimental Yachay.

Urcuquí, Febrero del 2021.

Firmado electrónicamente por:
RENZO SEBASTIAN CAAMANO MAYORGA

Renzo Sebastián Caamaño Mayorga
CI: 1726083718

Autorización de publicación

Yo, **Renzo Sebastián Caamaño Mayorga**, con cédula de identidad **1726083718**, cedo a la Universidad de Tecnología Experimental Yachay, los derechos de publicación de la presente obra, sin que deba haber un reconocimiento económico por este concepto. Declaro además que el texto del presente trabajo de titulación no podrá ser cedido a ninguna empresa editorial para su publicación u otros fines, sin contar previamente con la autorización escrita de la Universidad.

Asimismo, autorizo a la Universidad que realice la digitalización y publicación de este trabajo de integración curricular en el repositorio virtual, de conformidad a lo dispuesto en el Art. 144 de la Ley Orgánica de Educación Superior.

Urcuquí, Febrero del 2021.

Renzo Sebastián Caamaño Mayorga
CI: 1726083718

Dedicatoria

Para mi mejor amigo Sebastian Villareal. Algún día nos volveremos a encontrar para jugar a la play y la compu. Me contarás sobre la Historia y yo te hablaré de teoremas.

Para mi Papi Luchito, quien cada vez que me veía me preguntaba cientos de veces qué era lo que estudiaba y para qué servía. Desearía responderle cientos de veces más. Desearía poder demostrarle.

Renzo Sebastián Caamaño Mayorga

Agradecimientos

Tengo un cajón en mi corazón donde guardo a los que amo. Los conservo ahí porque les debo todo lo que he sido, soy y seré. Este trabajo, así como mi vida entera, se dio gracias a todos los seres que me han acompañado en algún momento de este camino. Cualquier logro o éxito que me depare el futuro, es por y para ellos.

Un buen matemático tiene que saber demostrar. A lo largo de la carrera lo que más hará será probar lemas, teoremas y corolarios. Creo tener cierto talento para demostrar y creo saber el porqué. No ha existido ningún solo momento de tiempo donde mis papitos no me hayan demostrado su amor. A pesar de cualquier discusión, situación económica, rupturas y desamparos; siempre he sabido que puedo contar con ellos. A mi papi Fausto le debo el gusto por la lectura, por lo social y por lo humano. Su bondad sin límites y su tenacidad para seguir luchando por lo que sueña es algo que a veces no entiendo, pero que siempre he admirado. Espero llenarlo de orgullo tanto como lo orgulloso que estoy de ser su hijo. A mi mami Adela, le debo la ternura, la paciencia y la responsabilidad. Su amor inquebrantable por los suyos y la angustia que siente por no poder dar lo mejor de si misma es mi mayor ejemplo a seguir. Jamás podré devolverle todos los mimos que me ha dado, pero siempre lo intentaré. A los dos les debo el baile, los chocolates y la certeza de que aún en los peores momentos, siempre se puede salir adelante.

Tengo tanta suerte que mis padres tienen sus correspondientes isomorfismos (objetos distintos pero que tienen la misma estructura), mi tío Luchito y mi tía Ivonne. No logro comprender como alguien que no te dio la vida te puede amar tanto como a mi me aman ellos. A ambos les debo el poder estudiar, el poder seguir mi sueños, e inclusive el poder comer. En la tristeza y en la alegría, siempre han estado ahí, y sé que siempre estarán. Su confianza y dedicación por mi, me hace querer comerme el mundo.

Junto a ellos, debo agradecer a mi familia que me abrigó y me dio un hogar. En especial a mi papi Luchito y mami Coqui, cuyos abrazos, aún cuando ya no están, me hacen seguir adelante. A mi tía Sarita, Mari, Vane, tío Rommel, Aldo, Dani, Romina, Eva y Daniela por todo su apoyo y la imagen que tienen de mi. Espero no defraudarlos.

Aquí no quiero ofender a nadie, los agradecimientos tratan de seguir un orden cronológico respecto a mi vida, y; aunque parezca raro, no puedo dejar de agradecer a quien desde el colegio me ha acompañado con su ronroneo y paz. Luna ha estado ahí para ver mis más grandes llantos, mis ataques de ansiedad y mis cientos de horas hablando solo. Lo que más me costó de haber estudiado lejos fue dejarla. Ella salvó mi vida cuando vino y ha sido mi más grande compañera este año que pasó. Escribir la tesis fue más llevadero porque al voltear a mi izquierda, la veía

descansar como si no existiera ningún mal.

Tuve la suerte de tener profesores y profesoras espectaculares. A cada uno de ellos les debo las herramientas que me han llevado hasta aquí. A mi teacher Rocío Larrea porque confiaba cuando ni yo lo hacía. A Yoli Zambrano que me enseñó a disfrutar más la vida. A Juan Mayorga, quien fue el primer matemático que me dio su aprobación y nunca dejo de insistir en que este era mi camino. Siento que nací para ser matemático, y no lo hubiera sabido sino fuera por él. A mis profes Hugo Leiva, Hugo Fernandes, Eusebio, Antonio, Saba, Zenaida, Rafael y Juan Carlos, quienes siempre me ayudaron con dudas que tenía y nunca les molestó mi presencia continua en sus oficinas. Aún cuando no me daban clases, siempre fueron mis maestros. Quiero destacar a mi profe Isidro Amaro a quien considero un gran amigo. A mi profe Eduardo Hidalgo quien casi me hace seguir química. Finalmente a mi profe Levis Zerpa por su ayuda en mi vida de investigador.

Quiero agradecer de todo corazón a mi tutor y amigo Francesc Castro. Nunca me he sentido tan retado como con sus exámenes y así mismo, nunca me he sentido tan apoyado y comprendido mientras hacía esta tesis. Admiro su pasión por la libertad y su lucha por lo que considera justo. Su pasión por la geometría y sus aplicaciones es lo que me hizo decantarme por el tema presente en este trabajo. Sus consejos, sugerencias y correcciones han hecho de este texto uno por el cual siento mucho orgullo. Que en el gran árbol de las matemáticas yo sea una hoja de su rama es algo que llevaré como emblema.

No se como, pero he logrado formar familias haya donde he estado. Mis amigos de toda la vida son mis hermanos y hermanas. Dani es simplemente mi Disney. Es aquella persona que es magia y te hace creer en un mundo mejor. Hemos vivido todos los grandes momentos juntos. Viviremos todos los que vengan igual. Mili es mi luz de libertad y lucha, es el reggae de mi reggaetón y la música de mi baile. Bachita, Verito, Raque, Joha y Majo son mis bebes. Con ellas las fiestas, los viajes, las caídas y las películas siempre son momentos inolvidables. Pablito, Sebas, Lobito, Raúl, Juanpa, Mauri y Khenany son los mejores tripulantes con los que uno puede cursar el gran viaje de la vida. Richie, Kevin y Pato son los tres hombres que más me hacen reir, y para alguien que pasa la mayor parte del tiempo triste, eso es un regalo.

Cuando llegué a Yachay me sentía perdido. Fue casi mortal para mi el haberme alejado de todo cuanto amaba. Por suerte, nuevos amores llegaron a mi vida y me hicieron sentir en casa. Rubén, Leandro, Mishu, Alex y Hazel fueron mis primeros grandes amigos en la universidad. Mi salud mental agradece que estos seres me hayan abierto las puertas. Casi al terminar mi carrera me volví a perder, cuando lo hice, fue Mari quien me rescató. Mari siempre será mi árbol de paz, mi esperanza de nuevos comienzos y mi cita para el baile.

Urcuquí e Ibarra siempre me parecieron aburridos y distantes, hasta que conocí a la familia Rodríguez Chauca. Rebe, Saúl, Arturo, Beni, Liss, Andre, Luciana, Luci, Mabelita, Talquinito, Sarahí, Stalin, Freddy, Otto y muchos más, hicieron que me sienta como en casa desde el primer momento. No solo me enseñaron sobre el amor, el trabajo y los sueños; me dieron un hogar. Gracias a ellos por dejarme ser parte de su familia por tanto tiempo. Gracias por *maltratarme mal*.

Tengo que mencionar a mi familia Mate 4G = {Nao, Katy, Joha, Marquito, Brandon, Dianita, Juanse, Ariel, Ray, Josue, Henry, Sammy y Roger}. Las noches de estudio comiendo pizza, las cenas en el Árbol, las tardes de calistenia, las borracheras en los bloques y sus hombros para llorar, me harán recordarlos siempre. No conozco matemáticos más talentosos. No puedo esperar por ver lo que el futuro les depara. Seguro será grandioso.

Por último, no se como empezar. ¿Cómo agradecer a una alma tan parecida a la mía? Lo que diga se quedará corto para explicar cuan agradecido estoy por tener a Nao en mi vida. Nunca me he sentido tan comprendido como en sus brazos. Nunca he vencido al miedo como al mirar sus ojos. Lo que más me dolió de la pandemia fue el no comer con ella, no verla leer comics, o escuchar sus preguntas infinitas. Me partió el corazón no poder estar con ella. Sin embargo, si lo estuve. Estos últimos dos años no podría haberme levantado de la cama sin sus palabras de aliento o sus gestos al escucharme. No recuerdo lo que es sentirse solo desde que la conozco. Gracias infinitas a quien, aún estando a 158 de kilómetros, supo hacerse sentir justo en el centro de mi corazón. Ahí se quedará.

Renzo Sebastián Caamaño Mayorga

Resumen

En este trabajo, presentamos la teoría detrás de la criptografía de curva elíptica (ECC) y la criptografía basada en isogenia. Comenzamos con una rápida introducción a la teoría de la complejidad y la criptografía. Damos una descripción detallada de las curvas elípticas y discutimos cómo se utilizan para crear protocolos seguros. Presentamos la fortaleza de este esquema y también sus debilidades con respecto a la computación cuántica. Para mostrar la necesidad de otros protocolos, revisamos algunos conceptos básicos de Computación Cuántica y presentamos cómo el algoritmo cuántico de Shor rompe ECC. A partir de entonces, examinamos la rica teoría de las isogenias entre las curvas elípticas ordinarias y supersingulares para terminar con protocolos criptográficos que utilizan estas isogenias para crear sistemas criptográficos post-cuánticos.

Palabras Clave: curvas elípticas, isogenias, criptosistemas de clave pública cuántico-resistentes, Algoritmo de Shor.

Abstract

In this work, we present the theory behind Elliptic Curve Cryptography (ECC) and Isogeny-based cryptography. We start with a quick introduction to Complexity Theory and Cryptography. We give a detailed description of elliptic curves and discuss how they are used to create secure protocols. We present the strength of this scheme and also its weaknesses regarding quantum computation. To show the necessity for other protocols, we review some basic concepts of Quantum Computing and present how Shor's Quantum Algorithm breaks ECC. Then, we survey the rich theory of isogenies between ordinary and supersingular elliptic curves to end with cryptographic protocols, that use these isogenies to create post-quantum cryptographic systems.

Keywords: elliptic curves, isogenies, quantum-resistant public-key cryptosystems, Shor's Algorithm.

Contents

List of Figures

Chapter 1

Introduction

If and when large quantum computers become practical, all currently widely deployed methods for public-key cryptography will break. So, it is of extreme importance to develop, test, and deploy new algorithms to have quantum-resistant encryption methods, [CCJ+16]. One class of quantum-resistant encryption methods are the isogeny-based encryption and key exchange methods, which are *ordinary isogeny Diffie-Hellman (OIDH), supersingular isogeny Diffie-Hellman (SIDH), and commutative SIDH (CSIDH)*, [PCZH19]. Most post-quantum encryption schemes require much longer keys to maintain current levels of protection. Isogeny-based encryption uses the shortest keys of any proposed post-quantum encryption methods. Isogeny-based encryption can be understood as an update from Elliptic curve cryptography (ECC). ECC is a public-key cryptosystem that relies on group properties of elliptic curves over finite fields and the difficulty of the discrete logarithm problem. However, quantum computers could solve the elliptic curve discrete logarithm problem efficiently, and so ECC is not quantum resistant. Isogeny methods are based on networks of isogenies between elliptic curves.

In Chapters 4 and 7, we give definitions for regular maps, Abelian varieties, elliptic curves and isogenies. For now let E_1 and E_2 be Abelian varieties of the same dimension over a field $\mathbb{K}$. An *isogeny* between E_1 and E_2 is a non-constant rational map defined everywhere, i.e., a morphism, that maps the identity point on E_1 to the identity point on E_2, [Mum08]. Elliptic curves are Abelian varieties, [Ols72], and isogenies between elliptic curves create equivalence classes of elliptic curves over the same field, [Coh06], which are the key behind all of the protocols that we will discuss.

Isogeny based cryptography is a demanding topic because of its cross-disciplinary nature and the abstract mathematical concepts that support it, [Cos19]. For these reasons, it is difficult for newcomers to obtain a broad overview of the most important techniques and results of the field. That is why our purpose in this thesis is to introduce the background material in complexity theory, quantum computing, cryptography, and algebraic geometry necessary to understand isogeny based cryptography. This is done at a level comprehensible to readers in senior years of Mathematics or Computer Science; a remarkable result since most of the works published in this area are in the master level and beyond.

The present work is structured in a linear story way. The idea is to present the concepts one by one and construct a plot for the necessity of isogeny based cryptography. In Chapter 2, we

make a summary of Computational Complexity; with special attention to complexity classes, and highlighting the importance of efficient algorithms to solve computational problems. In Chapter 3, we cover the basic aspects of Cryptography and we develop a more detailed description of Public Key Cryptography and the Discrete Logarithm Problem. Chapter 4 is about the mathematical background of Ellipictic curves. In Chapter 5, we focus on Elliptic Curve Cryptography (ECC), its advantages, and weaknesses. Chapter 6 is an introduction to Quantum Computing and, we explain how Shor's Algorithm breaks Elliptic Curve Cryptography. Finally, Chapter 7 is the development of Isogeny's theory and how it provides a framework to produce quantum-resistant encryption methods. In particular, we describe the SIDH/SIKE protocol, a candidate for the postquantum cryptography standardization process by NIST, which is based on the problem of finding isogenies between two elliptic curves.

Throughout this work we will use some concepts from Field and Galois Theory. Many of these are defined in the text itself or in footnotes. However, the ones that require further development are defined in the Appendix A. In order not to lose the continuity of the work, we will refer when necessary to this appendix for the interested reader.

Chapter 2

Computational Complexity: A tale of efficiency

Almost all concepts in modern cryptography have definitions around computational complexity. All the protocols, algorithms, and services that we use are carried by computers, thus we need to take into account the study of what kind of problems are solvable by computers, and if there is a way to find such solution in an *efficient* way. This chapter serves as a glossary of definitions that will appear in the next chapters. In Section 2.1, we explain what is a computational problem and we describe the two big families of computational problems: search and decision problems. We continue in Section 2.2, defining the concept of algorithms and what we mean when we say that an algorithm solves a problem or computes a function. Section 2.3 gives an overview of Turing machines, which are an abstract model of computation, and let us define complexity classes, which are covered in Section 2.4, and are the key concept behind many cryptographic definitions.

2.1 Computational Tasks

Computational complexity is the study of the time and space resources required to solve computational problems. A computational problem is a problem that a computer can solve. The two fundamental types of computational tasks are search problems and decision problems.

2.1.1 Search Problems

A search problem consists of finding an object (or a set of objects) that matches specific criteria in a given space (or set of spaces). For example, you are asked to search for the password of an encrypted system.

In the following definition, the potential solver is a function, and the set of possible solutions associated with each of the various instances (search criteria) are "packed" into a single binary relation.

Definition 2.1.1. (solving a search problem, [Gol03]): Let $R \subseteq \{0,1\}^* \times \{0,1\}^*$[1] and $R(x) := \{y : (x,y) \in R\}$ denote the set of solutions for the instance x. A function $f : \{0,1\}^* \to$

[1]The notation $\{0,1\}^*$ refers to the space of finite strings in the alphabet $\{0,1\}$, including the empty string.

$\{0,1\}^* \cup \{\perp\}$ solves the **search problem** of R if for every x, the following holds: if $R(x) \neq \varnothing$ then $f(x) \in R(x)$ and otherwise $f(x) = \perp$. [2]

In cryptography many times we only have one solution, so the search problems that have a special interest are the ones having a unique solution (for each possible instance). Finding the prime factorization of a composite number, used in RSA cryptography, is an example of this kind.

2.1.2 Decision Problems

A decision problem is a question of the existance of an effective computational procedure for deciding the truth of falsity of any instance of a parametric statement, [oM]. For example, the problem of determining the primality of a natural number. A case related with search problems above is the case of the set instances having a solution; that is, for any binary relation $R \subseteq \{0,1\}^* \times \{0,1\}^*$ we consider the set $\{x : R(x) \neq \emptyset\}$. Indeed, being able to determine whether or not a solution exists is a prerequisite to being able to solve the corresponding search problem. Intuitively, the problem consists of deciding if some object (or objects) match specific criteria.

Definition 2.1.2. (solving a decision problem, [Gol03]): Let $S \subseteq \{0,1\}^*$. A function $f : \{0,1\}^* \to \{0,1\}$ solves the **decision problem** of S (or decides membership in S) if for every x, it holds that $f(x) = 1$ if, and only if, $x \in S$.

2.2 Algorithms

To rigorously define computation, we need to specify some model of computation, which is a concrete definition of computing environments and a class of rules that can be applied to them. There are different models and all of them try to be an abstraction of a real computer. In this work, we are going to use two models of computation: *Turing machines* and *Quantum circuits*. Quantum circuits are discussed in Chapter 6. Turing machines are the classical framework where complexity theory is founded, thus specific algorithms are formalized by corresponding Turing machines, [Gol15].

When we say that a computer can solve a task, we mean that there is an algorithm that computes the function associated with the task. In other words, an algorithm A computes the function $f_A : \{0,1\}^* \to \{0,1\}^*$ defined by $f_A(x) = y$ if, when invoked on input x, algorithm A halts with output y. For decision and search problems we say that algorithm A solves the search problem of R or decides membership in S if f_A solves the search problem or decides membership. We will always associate an algorithm A with the function f_A computed by it; that is, we write $A(x)$ instead of $f_A(x)$, [Gol15].

[2] The symbol $\perp$ denotes an indication that something is wrong, in the case of a search problem, it says that there is no solution in the search space.

2.3 Turing Machines

This section provides a rough description of the model of Turing machines. This is done only to provide a concrete model that supports the study of algorithms and their complexity. A good introduction to Turing Machines and Computational Complexity Theory can be found in [Pap94].

The model of Turing machines tries to capture the limitations and abilities of a real-life computer. In essence, a computational task can be solved by a real-life computer if, and only if, it can be solved by a Turing machine. To describe Turing Machines as a valid model of computation we need to specify the set of possible environments, the set of computation rules, and the effect of applying such a rule on an environment, [Gol15].

- The main component in the *environment* of a Turing machine is an infinite sequence of cells, called the tape of the Turing machine, each capable of holding a single symbol member of some finite set Σ that contains the set of bits $\{0, 1\}$. This sequence starts at a leftmost cell and extends to infinity to the right. The environment also contains the current location of the machine on this sequence and the internal state of the machine (which is a member of a finite set Q).
- The main component in the *Turing machine* itself is a finite function called the transition function, which is defined over the set of all possible symbol-state pairs. This function is a mapping from $\Sigma \times Q$ to $\Sigma \times Q \times \{-1, 0, +1\}$, where $\{-1, +1, 0\}$ corresponds to a movement instruction ("left", "right" and "stay", respectively). In addition, the machine's description specifies an initial state and a halting state, and the computation of the machine halts when the machine enters its halting state.
- A single *computation step* of such a Turing machine depends on its current location on the tape, on the contents of the corresponding cell, and on the internal state of the machine. The transition function reads the current content and internal state and determines a new symbol-state pair as well as a movement instruction. The machine modifies the contents of the said cell and its internal state accordingly and moves as directed.

In other words, a Turing machine can be thought of as computing functions from the non-negative integers to the non-negative integers (using its bits representation); the initial configuration of the tape is used to represent the input of the function, and the final state of the tape is used to represent the output of the function, [MAN10].

We use the model of Turing machines for its simplicity. However, the Church-Turing Thesis asserts that nothing is lost by considering this model.

The Church-Turing Thesis. A function can be computed by some Turing machine if, and only if, it can be computed by some machine of any other "reasonable and general" model of computation. These other models can even be more 'capable' Turing machines; we can consider multiple tapes, two-way infinite tapes, or let them be non-deterministic.

To see the power of Turing machines, we can emulate a random-access machine (RAM) with one. This model very closely resembles the basic central processing unit (CPU) register

memory paradigm behind the design of modern computers. An abstract RAM consists of an infinite number of memory cells, each capable of holding an integer, a finite number of similar registers, one designated as the program counter, and a program consisting of instructions selected from a finite set. This set can contain, for instance, the read(j) instruction, which reads the contents from the jth address on the memory and places it in register zero. Other instructions include: reset(r_j), increment(r_j) and add(r_1, r_2). The actions of such instructions are clear by their name and observing that r_j is an index of a register. To emulate a RAM machine with a Turing machine we only need to hold the input, the contents of all registers, and the contents of the memory cells that were accessed during the computation. Thus, at each time, the Turing machine's tape contains a list of the RAM's memory cells that were accessed so far as well as their current contents. When we emulate a RAM instruction, we first check whether the relevant RAM cell appears on this list, and augment the list by a corresponding entry or modify this entry as needed, [Gol15].

To end this section, we present an advanced model of the basic Turing machine; the *Nondeterministic Turing machine*. This model is necessary to define the class **NP** later. Additionally, in cryptography and quantum computation we will mostly work with probabilistic algorithms so we need a nondeterministic model for those.

A *nondeterministic Turing machine* is similar to a standard Turing machine except for the fact that it does not have a single, uniquely defined next action, but a set of possible subsequent actions. So, the transition function is not a function anymore but a relation $\Delta \subset (\Sigma \times Q) \times [\Sigma \times Q \times \{-1, 0, +1\}]$. That is, for each symbol-state combination of the tape, there may be more than one appropriate next steps, [Pap94].

2.4 Time Complexity: Complexity classes

Having fixed the model of computation we turn to the study of the algorithms and computational problems. We focus on algorithms that halt on each input. To study these algorithms, we consider the number of steps taken by the algorithm on each possible input. This number of steps is a function called the time complexity of the algorithm: that is, $t_A : \{0,1\}^* \to \mathbb{N}$ is called the time complexity of algorithm A if, for every x, on input x algorithm A halts after exactly $t_A(x)$ steps. We are interested in the worst-case scenario, i.e., we consider $T_A : \mathbb{N} \to \mathbb{N}$ defined by $T_A := \max_{x \in \{0,1\}^n}\{t_A(x)\}$. Sometimes we refer to T_A as the time complexity of A, [Gol15].

Let us see how we can count the number of steps an algorithm needs to solve a problem using an example:

Example 2.1. *Bob has a a boolean function of n binary digits f ($f : \{0,1\}^n \to \{0,1\}$), that is either constant or balanced (a function that outputs 0 on exactly half inputs and 1 on the other half). Alice wants to know what kind of function Bob has.*
This is a computational problem since there is an algorithm that solves it. Alice can send $2^{n-1} + 1$ different strings to Bob. Then Bob will use its computer to compute the values of his function for all the strings and return the results to Alice. Alice then knows the outcome of half plus one of the possible strings, hence she knows exactly the nature of Bob's function, [MAN10].

We said that Alice needs to send $2^{n-1}+1$ strings to Bob to be sure that the function is constant or balanced. Of course, she can know this result before sending all the strings. In fact, suppose that Alice sends Bob the strings s_1 and s_2, if Bob returns $f(s_1)=1$ and $f(s_2)=0$, then Alice will know that the function is balanced and she is done. With only two strings sent, she has solved the problem. However, suppose $f(s_1)=f(s_2)=0$, then Alice can not know yet if the function is constant or not, so she needs to send s_3. If $f(s_3)=1$ then the function is not constant so it is balanced and we are done, but there is the chance that again $f(s_3)=0$ so Alice will need to send another string. In fact, since the function can be constant or balanced, Alice could have sent $2^n/2=2^{n-1}$ and have obtained 0 for all of them. So, she does not know yet the solution, but sending the string $s_{2^{n-1}+1}$ will give her the nature of the function; since if the output is 0, then the function must be constant because she is already in the other half of the strings. On the other hand, if the output is 1, then the function is balanced for the same reason, [MAN10]. In summary, if $t_A(n)$ represents the number of evaluations that Alice makes, it must lie between 2 and $2^{n-1}+1$, that is,

$$2 \leq t_A(n) \leq 2^{n-1}+1.$$

This last bound justifies the use of *asymtotic notation* to study the time complexity of algorithms and problems. Asymptotic notation can be used to summarize the *essential* behavior of a function. This notation can be used to summarize the essence of how many time steps it takes a given algorithm to run, without carrying too much about the exact time count, [MAN10].

Let f and g be functions from $\mathbb{N}$ to $\mathbb{N}$. We write '$f(n)$ is in the class of functions $\mathcal{O}(g(n))$', or just '$f(n)$ is $\mathcal{O}(g(n))$', if there are positive integers c and n_0 such that, for all $n \geq n_0$, $f(n) \leq cg(n)$. In a informal way this means that f grows as g or slower. If the opposite happens, that is, if $g(n) \in \mathcal{O}(f(n))$, we write '$f(n)$ is in the class of functions $\Omega(g(n))$'. Finally, $f(n)$ is said to be $\Theta(g(n))$ if f is $\mathcal{O}(g(n))$ and f is $\Omega(g(n))$. The Big Theta notation, Θ, means that f and g grow at the same rate, [Pap94]. In order to summarize the essential behavior of an algorithm A, we apply asymptotic notation to the function T_A defined before.

In Example 2.1, we saw that Alice determines the nature of Bob's function in time $\mathcal{O}(2^n)$ and also $\Omega(1)$, where 1 here is the constant function $f(n)=1$ for all $n \in \mathbb{N}$.

As stated at the beginning of this chapter, Complexity theory is not concerned with the time complexity of a specific algorithm. It is rather concerned with the time complexity of a problem, assuming that it is solvable at all (by some algorithm). The time complexity of a problem is the time complexity of the fastest algorithm that solves this problem. Moreover, analogous to asymptotic notation, we are interested in upper and lower bounds on the time complexity of algorithms that solve the problem. Formally, a certain problem Π has complexity T if it has time complexity at least T, in other words, if it is in $\Omega(T)$, [Gol15].

Assuming a given model of computation, if a computational problem can be solved in time $\mathcal{O}(p(n))$, where $p(n)$ is a polynomial, we shall say that the problem can be solved in polynomial rate of growth and this is equivalent to say that the problem has an acceptable time requirement to be solved. In other words, the problem can be solved *'efficiently'* if it has a polynomial rate of growth. In contrast, a problem is regarded as *hard, intractable* or *infeasible* if the best

possible algorithm requires exponential rates such as 2^n or $n!$, [Pap94].

The dichotomy between polynomial and nonpolynomial time bounds and its relation with efficient and non-efficient computations is rather vague. "There are efficient computations that are no polynomial, and polynomial computations that are not efficient in practice", [Pap94]. For example, an algorithm that solves a problem using $2^{n/100}$ operations is probably more useful than one which runs in n^{100} operations. However, the use of polynomial time-bound as a measure for efficiency is justified, among other reasons, by the **Cobham-Edmonds Thesis**. As stated before, the complexity of a problem depends on the specific model of computation in which algorithms that solve the problem are implemented. The Cobham-Edmonds Thesis asserts that the variation in time complexity in any two models of computation are polynomially related. That is, *a problem has time complexity t in some model of computation if, and only if, it has time complexity* poly(t) *in the model of (single-tape) Turing Machines.* For a discussion of this topic we recommend [GJG79].

To divide problems that have efficient algorithms from the ones that have not, we use the concept of *complexity classes*. Complexity classes are better understood with the formalism of formal languages. A language L over the alphabet Σ is a subset of the set Σ^* of all (finite) strings of symbols from Σ. In this framework of thought, the decision problems that we discussed earlier can be seen as questions about if a string of bits corresponds to a specific language. A language L is decided by an algorithm if the latter can decide whether an input x is a member of the language L or not. A problem is in $\mathbf{TIME}(T(n))$ if there exists a classical algorithm which decides whether a candidate x is in the language in time $\mathcal{O}(T(n))$, where n is the length of x. A problem is said to be solvable in polynomial time if it is in $\mathbf{TIME}(n^k)$ for some finite k. The collection of all languages which are in $\mathbf{TIME}(n^k)$, for some k, is denoted **P**. **P** is a complexity class, [MAN10].

A complexity class is specified by several parameters: the model of computation, the mode of computation, a resource to bound, and abound. We have already discussed the model and the mode of computation. The resource that we care to bound is the time complexity of a problem. A complexity class is a set of all languages decided by some algorithm M, and such that, for any input x, M expenses at most $T(|x|)$ units of time [Pap94]. Another complexity class of importance is the class **NP** which is the collection of problems that have efficiently verifiable solutions. Its formal definition uses nondeterministic Turing machines, namely, **NP** is the union of all problems in $\mathbf{NTIME}(n^k)$. Where $\mathbf{NTIME}(f(n))$ is the set of problems solved by a nondeterministic Turing Machine. A problem is said to be solved by a nondeterministic machine if there is some sequence of nondeterministic choices that solves the problem. Other choices may result in no solution, [Pap94].

Obviously $\mathbf{P} \subset \mathbf{NP}$, since deterministic machines form a subclass of the nondeterministic ones; but the problem of finding whether $\mathbf{P} = \mathbf{NP}$ is the biggest challenge in Computer Science, [For09]. Many problems in cryptography, rely on the assumption that $\mathbf{P} \neq \mathbf{NP}$.

When working with nondeterministic algorithms we would like to bound the error of obtaining a bad solution or no solution at all, hence we define the class **BPP** (bounded-error probabilistic time) that contains all the problems solvable by a probabilistic Turing Machine in

polynomial time with an error probability bounded away from 1/3 for all instances. **BPP** class is important for us because Quantum Computation has opened new models of computation and thus new complexity classes like **BQP** which is the quantum analog of **BPP**. Problems in **BQP** are problems that can be solved by quantum computers in bounded-error probabilistic polynomial time.

The last complexity class that we present is the set of **NP**-complete problems. These problems can be thought as the 'hardest' problems in **NP** in the sense that solving a **NP**-complete problem in time t allows any other problem in **NP** to be solved in time $\mathcal{O}(poly(t))$. This also means that if any **NP**-complete problem had a polynomial-time solution then, it would follow that **P=NP**.

Chapter 3

Cryptography: The art of making codes

Cryptography is a part of a bigger discipline called *Cryptology*, which is *an all-inclusive term that includes cryptography, cryptoanalysis, and the interaction between them*, [KS19],. Cryptoanalysis refers to the science behind breaking cryptosystems, i.e, to discover what is behind a codified information, [PP10], where Cryptography is the science of developing a system to hide information so that only the intended recipient of the information can understand, [KS19]. Cryptography can then be divided into three main branches, [PP10]:
Symmetric Algorithms occur when two parties have a secret key to encrypt and decrypt information.
Asymmetric (or Public-key) Algorithms differ from the former as here, the parties have two keys; one private for each of the parties and one public known by all.
Cryptographic Protocols deal with the application of symmetric and asymmetric algorithms.
An encryption algorithm, or cipher, is like a machine that transforms *plaintext* (ordinary readable text not hiding any information) into *ciphertext* (text that, apparently, does not mean anything). This machine requires a control given by a secret key. Mathematically, we can say that a cipher is a function $e : A \to B$, where A is the set of the possible plaintexts and B is the set consisting of the enciphered messages, that depends on the secret key k and maps a plaintext m in A to a ciphertext c in B, [Sma16], i.e,

$$c = e_k(m).$$

Of course, we need a way to revert this process and this is called decryption or decipherment, which mathematically means that there is a function $d : B \to A$ that converts the ciphertext c in plaintext m, [Sma16], i.e.,

$$m = d_k(c).$$

It is obvious that the functions c and d are public, so the secrecy of m given c is given totally by the secrecy of k, [Sma16]. In more general terms, the secret key k can belong to a set of keys K that can contain public and private keys. If a key is uniquely known by the two parties looking to exchange information, the encryption algorithm belongs to the family of Symmetric Algorithms. On the other hand, if the encryption function e depends on a public key u and a private key r_1, i.e., $e_{u,r_1}(m) = c$, and the decryption function d depends on the same public key u and in its own private key r_2, i.e., $d_{u,r_2}(c) = m$; we say that this algorithm belongs to the family of Asymmetric Algorithms.

As written in [BCC08], the most convenient alphabet used for plain text and ciphertext is the set of integer numbers, which are more suitable to the description of the transformations, and in our digital world, more easily treated by computers. Moreover, the functions e and d must be bijective and inverse of each other if we want to be able to find the original message given a codified one. Of course, this procedure depends on the used keys. In particular $d_k \circ e_k(m) = m$.

3.1 Symetric Cryptography

Let us suppose there are two users, Anna and Balto, who want to communicate over an insecure channel. There is also another user of the channel called Osiris, who wants to know the secrets between Anna and Balto's communication (Fig. 3.1).

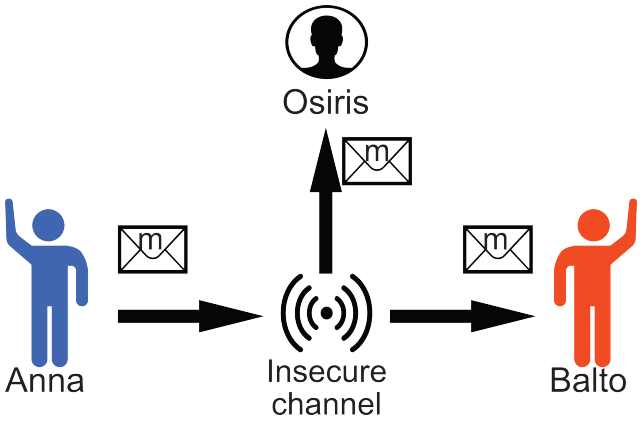


Figure 3.1: Communication over insecure channel

Since the channel is insecure, it is easy for Osiris to obtain the message m that Anna is sending to Balto. This is called *eavesdropping*. To keep Osiris outside the communication, Anna and Balto decide to use symmetric cryptography. Anna encrypts her plaintext m using a symmetric algorithm, yielding the ciphertext c. Balto receives this ciphertext and decrypts it (Fig. 3.2). Since the channel is insecure, Osiris is still receiving the message, but now it is ciphered into the ciphertext c, and a good encryption algorithm will make it almost impossible for him to decrypt it. However, in order for Balto to decrypt the message encrypted by Anna, he needs the key of encryption k, this key must be sent through a secure channel. Because, if not, Osiris will have access to the key and therefore decrypt the message by himself. This shows us, that the strength of a symmetric algorithm relies entirely upon how secure is our channel for key-transmission, and how good is our method to keep safe the common key, between Anna and Balto from Osiris.

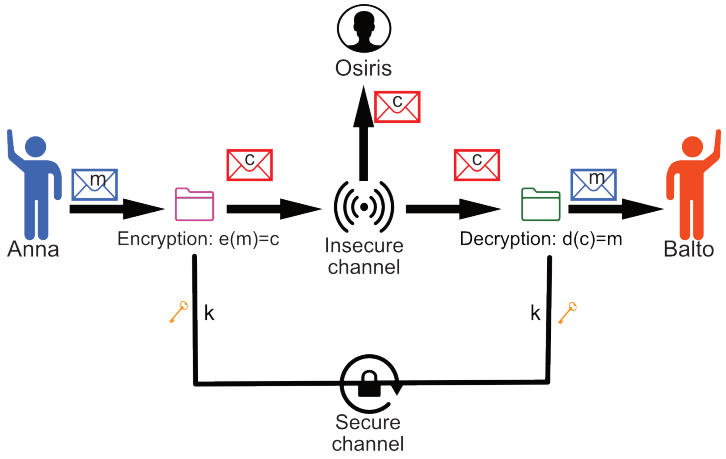


Figure 3.2: Symmetric-key cryptosystem

One could argue that a more secure approach will be having secret encryption and decryption algorithms, but these kinds of systems failed because they cannot be tested using cryptanalysis tools. Here, it is important to recall *Kerckhorffs' Principle*, postulated by Auguste Kerckhoffs in 1883, [PP10].

Definition 3.1.1. *Kerckhoffs' Principle*
A cryptosystem should be secure even if the attacker knows all details about the system, with the exception of the secret key. In particular, the system should be secure when the attacker knows the encryption and decryption algorithms.

In the system that we have discussed, the deciphering procedure is not difficult. If you know the enciphering method and the key used, you are done. In this case, we say that the deciphering function is *symmetric* to the enciphering function, i.e., is its inverse, [BCC08].
In our global and internet-connected world, it is difficult for users such as Anna and Balto to create a secure channel to send their secret keys. For example, Anna and Balto can live miles away and even don't know each other, so a better approach must be taken. This is where Asymmetric Cryptosystems appear and the family where Elliptic Curve and Isogeny-based Cryptography live.

3.2 Public Key Criptography

In Public Key Cryptography we use ciphers that allows both the encryption employed and the enciphering key to be made public without revealing the deciphering method. For the receiver of the encrypted message to be able to decipher in a *reasonably short* time the ciphertext, he or she would need to possesses more information, besides the public one. Without this private information, deciphering a message would require a time exceedingly long and so would it be unfeasible to do it.

The mathematical model that we use for this scenario is the notion of *one-way function.*

Definition 3.2.1. *(Informal definition [Rob11])* A **one-way function** is a function for which computation in one direction is straightforward, while computation in the reverse direction is

far more difficult. (See Fig. 3.3)
(*Formal definition* [Rob11]) A function $f : \{0,1\}^* \to \{0,1\}^*$ is called **(strongly) one-way** if the following conditions hold:

1. Easy to compute: There exists a (deterministic) polynomial-time algorithm A such that on input x algorithm A outputs $f(x)$ (i.e., $A(x) = f(x)$).

2. Hard to invert: For every probabilistic polynomial-time algorithm A', every positive polynomial $p(\cdot)$, and all sufficiently large n's,

$$\Pr[A'(f(U_n), 1^n) \in f^{-1}(f(U_n))] < \frac{1}{p(n)}.$$

   Where U_n denotes a random variable uniformly distributed over $\{0,1\}^n$. Hence, the probability in the second condition is taken over all possible values assigned to U_n and all possible internal coin tosses of A', with uniform probability distribution. Note that A' is not required to output a specific preimage of $f(x)$, [Gol03].

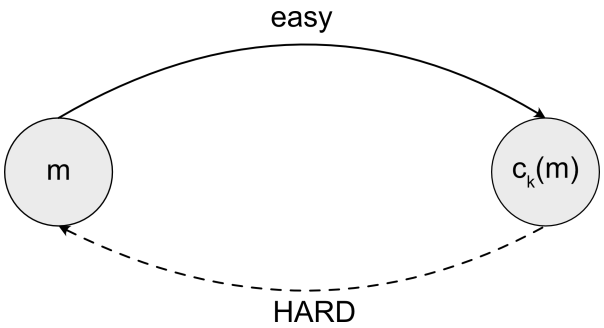


Figure 3.3: One-way functions: an illustration.

In the formal definition of *(strongly) one-way functions*, the input 1^n given to the inverting algorithm A' is the length, in binary notation, of the desired output. As written in [Gol03], we make this convention to rule out the possibility that a function will be considered one-way merely because it drastically shrinks its input, and so the inverting algorithm just does not have enough time to print the desired output.

The existence of these *(strongly) one-way functions* implies that there are efficient processes that are hard to reverse in complexity terms. Hence, the existence of security encryption schemes implies that there are tasks that can be performed by non-deterministic polynomial-time machines, yet cannot be performed by deterministic polynomial-time machines. Therefore, a necessary condition for the existence of one-way functions is that NP not be contained in BPP (and thus $P \neq NP$), [Gol03].

$P \neq NP$ is a necessary but not sufficient condition for the existence of one-way functions. Because, if we have that the breaking of some encryption scheme is NP-complete, this only implies that this encryption scheme is hard to break in the worst case, but not necessarily in most cases. Thus, security requires hardness in most cases, or at least "average-case hardness", [Gol03].

Finally, because we want someone to understand our secret message we need *"to generate hard instances together with auxiliary information that will enable us to solve these instances fast"*, [Gol03]. Because, if not, the intended recipients of the message will not have a computational advantage over the adversary. In summary, the existence of secure encryption schemes, and so the existence of one-way functions, implies the existence of an efficient way to generate instances with corresponding auxiliary input such that: 1) it is easy to solve these instances given the auxiliary input, but, 2) it is hard on the average, to solve these instances when not given the auxiliary input, [Gol03].

In the definition of one-way functions given earlier, we made a very strong requirement for them to be one-way. Namely, any efficient algorithm has negligible success in inverting them. We will now see that this is not necessary, since we can only require that all efficient inverting algorithms fail with some noticeable property, and then convert them to acquire the hardness alluded to in the earlier discussion, [Gol03]. Before giving this *Weak One-Way functions* definitions, few words concerning the notions of negligible and noticeable probability are in order.

We say that the success probability of an algorithm is negligible if, as a function of the input length, the success probability is bounded above by every polynomial fraction. Thus, repeating the algorithm polynomially (in the input length) will always yield a new algorithm that also has negligible success probability. On the other hand, we say that a function $\gamma : \mathbb{N} \to \mathbb{R}$ is *noticeable* if there exists a polynomial $p(\cdot)$ such that for all sufficiently large n's, it holds that $\mu(n) > \dfrac{1}{p(n)}$, [Gol03].

Definition 3.2.2. (Weak One-Way Functions [Rob11]): A function $f : \{0,1\}^* \to \{0,1\}^*$ is called **weakly one-way** if the following conditions hold:

1. Easy to compute: As in the definition of a strong one-way function.
2. Slightly hard to invert: There exists a polynomial $p(\cdot)$ such that for every probabilistic polynomial-time algorithm A' and all sufficiently large n's,

$$\Pr[A'(f(U_n), 1^n) \notin f^{-1}(f(U_n))] > \frac{1}{p(n)}.$$

As mentioned in [Gol03], we can always work with strong, weak one-way functions that are length-regular, i.e, do not shrink there inputs by more than a polynomial amount. So, we can give the inverting algorithms only $f(x)$ as input.

Extensive research and failed attempts to find efficient inverting algorithms have to lead to several candidates for one-way functions. These include Integer factorization, decoding of random linear codes, the subset-sum problem, and the discrete logarithm problem, [Rob11]. For this work, we focus on **the Discrete Logarithm Problem** which is the base for the security of Elliptic Curve Cryptography.

3.3 The Discrete Logarithm Problem

The material in this section is based on [WXWM15].

Definition 3.3.1. Let $m \leq 1$ and $a \in \mathbb{N}$, such that $gcd(a, m) = 1$ (gcd refers to greatest common divisor). The smallest positive integer d such that $a^d \equiv 1 \pmod{m}$ is called an **exponent of a modulo m** (it is also called an order, or a period), denoted by $\delta_m(a)$. If $\delta_m(a) = \varphi(m)$[1], then a is called a primitve root of 1 modulo m.

Definition 3.3.2. Let g be a primitve root in $\mathbb{Z}_p^*$. For any element $y \in \mathbb{Z}_p^*$, there is a unique x, with $1 \leq x < p-1$ such that $g^x \equiv y \pmod{p}$. We call x the **discrete logarithm modulo p of y with respect to the base g.**

Thus, given $y \in \mathbb{Z}_p^*$ and a base $g \in \mathbb{Z}_p^*$, the Discrete Logarithm Problem (DLP) is to find x. $1 \leq x < p-1$ such that $g^x \equiv y \pmod{p}$. Clearly, given $1 \leq x < p-1$, finding $y \in \mathbb{Z}_p^*$ such that $g^x \equiv y \pmod{p}$, is not difficult; however, so far there is no polynomial-time algorithm for its inverse problem. This is of course in the classical computational setting, we'll see that this is not true for quantum computing. The best algorithm known to solve the DLP is the NFS, [BCC08], (the number field sieve method); whose asymptotic time estimation is

$$e^{(1.923+O(1))(ln(p))^{1/3}(ln(ln(p)))^{2/3}}.$$

Other useful definitions, which are related to the DLP, in group signatures and traceable blind signatures, are given below.

Definition 3.3.3. Let $g, h \in \mathbb{Z}_p^*$ be unrelated primitive roots of p (i.e., the discrete logarithm of g with respect to h is unknown). For any a wih $gcd(a, p) = 1$, we represent a as $a \equiv g^\alpha h^\beta \pmod{(p)}$, then (α, β) is called a **representation of a modulo p with respect to the bases g, h.**

This definition extends to an arbitrary number of primitives.

Definition 3.3.4. Let $g_1, g_2, \cdots, g_s \in \mathbb{Z}_p^*$ be unrelated primitive roots of p. For any a wih $gcd(a, p) = 1$, we represent a as

$$a \equiv g_1^{\alpha_1} g_2^{\alpha_2} \cdots g_s^{\alpha_s} \pmod{p},$$

then $(\alpha_1, \alpha_2, \cdots \alpha_s)$ is called a **representation of a modulo p with recpect to the bases $g_1, g_2, \cdots, g_s$.**

To end this section, we introduce an assumption that is related to a discrete logarithm problem.

Definition 3.3.5. (Diffie-Hellman Problem) Let g be a generator of the cyclic group $\mathbb{Z}_p^*$. For any $a, b \in \mathbb{Z}_p^*$ such that

$$a \equiv g^x \pmod{p}, \quad b \equiv g^y \pmod{p},$$

with unknown x, y, the **Diffie-Hellman problem** is to find c such that $c \equiv g^{xy} \pmod{p}$.

The Diffie-Hellman problem (DH problem) is used to assess the security of Diffie-Hellman key exchange protocol. It is believed that the DH problem is hard and is related to the discrete logarithm problem as follows. If the discrete logarithm problem can be solved in polynomial time, then the DH problem can also be solved in polynomial time. We'll see that Elliptic Curve Cryptography relies on its own version of the discrete logarithm problem and thus in its version of DH protocol.

[1] $\varphi(m)$ is the Euler Function, i.e., the number of positive numbers that are smaller than m and coprime to m.

3.4 Digital signatures

At the beginning of the chapter, we said that cryptography is not a modern field, however, computers and the internet have had such an impact on communication, and securing that communication, that it seems like modern cryptography is a new thing every day. One of the revolutions in modern cryptography that came with computers was digital signatures. As written in [Gol03], the notion of *digital signature* did not exist in the pre-computerized world. The need for such credentials arose with the introduction of computer communication where parties need to commit themselves to proposals and/or declarations they make.

Encryption and signature methods were related because of the "digitalization" of both and the introduction of the computational-complexity approach to security. A *scheme for digital signatures* requires, [Gol03]:

- that each user be able to efficiently generate his or her signature on documents of his or her choice,
- that each user be able to efficiently verify whether or not a given string is a signature of another specific user on a specific document, and
- that no one can efficiently produce the signatures of other users to documents that those users did not sign.

A digital signature that obeys these requirements is called *unforgeable*. Schemes for unforgeable digital signatures can be constructed using the same computational assumptions as used in the construction of private-key encryption methods, [Gol03]. A concept related to digital signature is the task of message authentification.

Message Authentification

We have mentioned that, no matter what encryption scheme we are working on, we make the communication over a channel. Suppose this channel is insecure and an adversary is monitoring it and may alter the messages sent to the channel. The parties communicating through this insecure channel wish to authenticate the messages they send so that the intended recipient can discriminate between an original message from a modified one. A *scheme for message authentification* requires, [Gol03]:

- that each of the communicating parties be able to efficiently generate an authentification tag for any message of his or her choice,
- that each of the communicating parties be able to efficiently verify whether or not a given string is an authentication tag for a given message, and
- that no external adversary be able to efficiently produce authentication tags to messages not sent by the communicating parties.

From the requirements of both, digital signatures and message tags, it is clear that digital signatures provide a solution to the message-authentification problem. However, a message-authentification scheme does not necessarily constitute a digital-signature scheme.

3.5 Cryptographic Hash Functions

An essential building block for cryptographic applications is hash functions. Cryptographic hash functions map input strings of arbitrary length to short fixed-length output strings. They can be used in many applications like computing short unique identifiers of messages (e.g., for digital signatures), as one-way functions to hide messages, to commit a string in a protocol, and for key derivation. Some common hash functions are SHA-1, SHA-2, and SHA-3, [Til11],[EM17].

Chapter 4

Elliptic Curves

In this chapter, we introduce the theory of elliptic curves over arbitrary fields. We define the Weierstrass normal form, the addition law, and the multiplication formulas. In the last section, we define isogenies and endomorphisms.

4.1 Weierstrass normal form

Let us recall some basic facts from algebraic geometry. These facts are presented as in [SS03] and [Sil09].

Definition 4.1.1. a) Let $\mathbb{K}$ be a field and $\overline{\mathbb{K}}$ denote its closure (refer to Appendix A for an introduction to Field Theory). The *affine n-space* is $\mathbb{A}^n := \overline{\mathbb{K}}^n$. The affine n-space over $\mathbb{K}$ is $\mathbb{A}^n(\mathbb{K}) := \mathbb{K}^n$.

b) Two elements $(x_0, \ldots, x_n), (x'_0, \ldots, x'_n) \in \mathbb{A}^{n+1}(\mathbb{K})$ are equivalent if there exists a non-zero $\lambda \in \mathbb{K}$ with

$$x_i = \lambda x'_i \quad \text{for } i = 0, \ldots, n.$$

It's not difficult to see that this defines an equivalence class of $(x_0, \ldots, x_n)$. It is written as $[x_0 : \cdots : x_n]$.

c) The *projective* n-space is

$$\mathbb{P}^n := \{[x_0 : \cdots : x_n] \in \overline{\mathbb{K}}^{n+1} : \text{ not all } x_i = 0\},$$

the projective $n-$space over $\mathbb{K}$ is

$$\mathbb{P}^n(\mathbb{K}) := \{[x_0 : \cdots : x_n] \in \mathbb{K}^{n+1} : \text{ not all } x_i = 0\}.$$

d) A polynomial $F \in \mathbb{K}[X, Y, Z]$ is called *homogeneous of degree* d, with $d \in \mathbb{N}$, if

$$F(\lambda X, \lambda Y, \lambda Z) = \lambda^d F(X, Y, Z) \text{ for all } \lambda \in \mathbb{K}.$$

e) A polynomial $f(X, Y) \in \mathbb{K}[X, Y]$ of total degree d can be *homogenised* by defining

$$F(X, Y, Z) := Z^d f\left(\frac{X}{Z}, \frac{Y}{Z}\right) \in \mathbb{K}[X, Y, Z].$$

A homogeneous polynomial $F(X,Y,Z) \in \mathbb{K}[X,Y,Z]$ can be *dehomogenised* by defining

$$f(X,Y) := F(X,Y,1) \in \mathbb{K}[X,Y].$$

f) A *plane projective algebraic curve* over $\mathbb{K}$ is the set of roots in $\overline{\mathbb{K}}$ of a non-constant homogeneous polynomial $F(X,Y,Z) \in \mathbb{K}[X,Y,Z]$,

$$C := C(F) = \{[x:y:z] \in \mathbb{P}^2 : F(x,y,z) = 0\}.$$

We define

$$C(\mathbb{K}) := C(F)(\mathbb{K}) = \{[x:y:z] \in \mathbb{P}^2(\mathbb{K}) : F(x,y,z) = 0\}.$$

the set of $\mathbb{K}$-rational points of C. A *point at infinity* of this curve is a point $P = [x:y:z] \in C$ with $z = 0$.

g) A *plane affine algebraic curve* over $\mathbb{K}$ is the set of roots in $\overline{\mathbb{K}}$ of a non-constant polynomial $f(X,Y) \in \mathbb{K}[X,Y]$,

$$C := C(f) = \{(x,y) \in \mathbb{A}^2 : f(x,y) = 0\}.$$

We define

$$C(\mathbb{K}) := C(f)(\mathbb{K}) = \{(x,y) \in \mathbb{A}^2(\mathbb{K}) : f(x,y) = 0\},$$

the set of $\mathbb{K}$-rational points of C.

h) Let $C = C(f)$ be a plane affine algebraic curve over the field $\mathbb{K}$. The quotient field of $\mathbb{K}[X,Y]/(f)$ is called the function field of C, and it is denoted by $\mathbb{K}(C)$.

i) Let C and C' be two irreducible algebraic plane curves, and u, $v \in \mathbb{K}(C)$. The map $\varphi(P) = (u(P), v(P))$ is defined at all points P of C where both u and v are defined; it is called a *rational map* from C to C', if $\varphi(P) \in C'$, for every $P \in X$ at which φ is defined.

A rational map $\varphi : C \to C'$ is *birational*, or is a *birational equivalence* of C to C' if φ has a rational inverse. In this case we say that C and C' are *birational*, or *birationally equivalent*.

Let $C = C(F)$ be a plane projective algebraic curve defined by the homogeneous polynomial $F(X,Y,Z)$. Let f be the dehomogenised polynomial $f(X,Y) = F(X,Y,1)$. Then the plane affine algebraic curve $C(f)$ together with the points at infinity correspond to the projective curve $C(F)$. Using this correspondence we consider in this thesis the points at infinity as additional rational points on the affine curve.

Definition 4.1.2. An equation of the form

$$Y^2 + a_1XY + a_3Y = X^3 + a_2X^2 + a_4X + a_6$$

with $a_1, a_2, a_3, a_4, a_6 \in \mathbb{K}$ is a **long Weierstrass normal form**.

The projective long Weierstrass normal form is given by

$$Y^2Z + a_1XYZ + a_3YZ^2 = X^3 + a_2X^2Z + a_4XZ^2 + a_6Z^3.$$

Here we have one point at infinity: $\mathcal{O} = [0:1:0]$. In affine representation, this is the point $\mathcal{O} = (\infty, \infty)$.

Definition 4.1.3. We consider an equation in long Weierstrass normal form with coefficients $a_1, a_2, a_3, a_4, a_6 \in \mathbb{K}$. The **Tate values** are

$$\begin{aligned} b_2 &:= a_1^2 + 4a_2, \\ b_4 &:= 2a_4 + a_1a_3, \\ b_6 &:= a_3^2 + 4a_6, \\ b_8 &:= a_1^2a_6 + 4a_2a_6 - a_1a_3a_4 + a_2a_3^2 - a_4^2, \\ c_4 &:= b_2^2 - 24b_4, \\ c_6 &:= -b_2^3 + 36b_2b_4 - 216b_6. \end{aligned}$$

Furthermore, the discriminant is

$$\Delta := -b_2^2b_8 - 8b_4^3 - 27b_6^2 + 9b_2b_4b_6$$

and the *j-invariant*

$$j := \frac{c_4^3}{\Delta}.$$

These constants satisfy the relations

$$4b_8 = b_2b_6 - b_4^2 \quad \text{and} \quad 12^3\Delta = c_4^3 - c_6^2.$$

Using Tate values, if $\text{char}(\overline{\mathbb{K}}) \neq 2$ (see Appendix A), we can simplify a long Weierstrass normal form by completing the square. Thus the substitution

$$Y \mapsto \frac{1}{2}(Y - a_1X - a_3)$$

produces an equation of the form

$$Y^2 = 4X^3 + b_2X^2 + 2b_4X + b_6,$$

where

$$b_2 = a_1^2 + 4a_4, \quad b_4 = 2a_4 + a_1a_3.$$

If further $\text{char}(\overline{\mathbb{K}}) \neq 2, 3$, then the substitution

$$(X, Y) \mapsto \left(\frac{X - 3b_2}{36}, \frac{Y}{108}\right)$$

eliminates the X^2, yielding the simpler equation

$$Y^2 = X^3 - 27c_4X - 54c_6. \tag{4.1}$$

Definition 4.1.4. Let a plane algebraic curve C be defined by the polynomial equation $f(X, Y) = 0$. Then $P = (x_0, y_0) \in C$ is a **singular point** of C if, and only if,

$$\frac{\partial f}{\partial X}(x_0, y_0) = 0 \quad \text{and} \quad \frac{\partial f}{\partial Y}(x_0, y_0) = 0.$$

The singular point is a *double point*, if only the first partial differentials vanish. A double point is a *node*, if the point has two different tangents, a *cusp* if the two tangents coincide. A curve without singular points is called *nonsingular*.

Proposition 4.1. *Curves given by an equation in long Weierstrass normal form have the following classification:*

(i) The curve is nonsingular $\iff \Delta \neq 0$. *Otherwise the curve is singular with exactly one singular point.*

(ii) The curve has a node $\iff \Delta = 0$ *and* $c_4 \neq 0$.

(iii) The curve has a cusp $\iff \Delta = 0$ *and* $c_4 = 0$.

In cases (ii) and (iii), there is only the one singular point.

Proof. Let E be a curve given by the long Weierstrass normal form

$$E : f(x, y) = y^2 + a_1xy + a_3y - x^3 - a_2x^2 - a_4x - a_6 = 0.$$

First we show that the point at infinity is never singular. Thus we use the projective long Weierstrass normal form

$$\begin{aligned} F(X, Y, Z) &= Y^2Z + a_1XYZ + a_3YZ^2 - X^3 - a_2X^2Z - a_4XZ^2 - a_6Z^3 \\ &= 0 \end{aligned}$$

and we study the point $\mathcal{O} = [0, 1, 0]$. We have that

$$\frac{\delta F}{\delta Z}(\mathcal{O}) = 1 \neq 0,$$

so we see that $\mathcal{O}$ is a nonsingular point of E.

Next, suppose that E has a singular point at $P_0 = (x_0, y_0)$. The substitution

$$x = x' + x_0 \quad y = y' + y_0$$

leaves Δ and c_4 invariant, so without loss of generality we may assume that E is singular at $(0, 0)$. Then

$$a_6 = f(0, 0) = 0, \quad a_4 = \frac{\delta f}{\delta x}(0, 0) = 0, \quad a_3 = \frac{\delta d}{\delta y}(0, 0) = 0,$$

so the equation for E takes the form

$$E : f(x, y) = y^2 + a_1xy - a_2x^2 - x^3 = 0.$$

This equation has associated quantities

$$c_4 = (a_1^2 + 4a_2)^2 \quad \text{and} \quad \Delta = 0.$$

By definition, E has a node (respectively cusp) at $(0, 0)$ if the quadratic form $y^2 + a_1xy - a_2x^2$ has distinct (respectively equal) factors, which occurs if, and only, if the discriminant of this quadratic form satisfies

$$a_1^2 + 4a_2 \neq 0 \quad (\text{respectivley } a_1^2 + 4a_2 = 0).$$

This proves the "only if" part of (ii) and (iii).

To complete the proof of (i)-(iii), it remains to show that if E is nonsingular, then $\Delta \neq 0$. To simplify the computation, we assume that char$(K) \neq 2$ and consider a Weierstrass equation of the form

$$E : y^2 = 4x^3 + b_2x^2 + 2b_4x + b_6.$$

The curve E is singular if, and only if, there is a point $(x_0, y_0) \in E$ satisfying

$$2y_0 = 12x_0^2 + 2b_2x_0 + 2b_4 = 0.$$

Thus, the singular points are exactly the points of the form $(x_0, 0)$ such that x_0 is a double root of the cubic polynomial $4x^3 + b_2x^2 + 2b_4x + b_6$. This polynomial has a double root if, and only if, its discriminant 16Δ, vanishes. This completes the proof of (i)-(iii). Moreover, since a cubic polynomial cannot have two double roots, E has at most one singular point. □

An *elliptic curve* over $\mathbb{K}$ is a nonsingular curve of genus 1 over $\mathbb{K}$ together with one $\mathbb{K}$-rational point. The equation for such a curve can be transformed to a long Weierstrass normal form. The specified $\mathbb{K}$-rational point is then transformed to the point at infinity.

Definition 4.1.5. An ***ellipitic curve*** over $\mathbb{K}$, denoted $E|\mathbb{K}$, is a curve given in long Weierstrass normal form over $\mathbb{K}$ with non zero discriminant (and the specified point at infinity).

As an example consider the ellptic curve $F : U^3 + V^3 = 1$. This curve has a point at infinity $\mathcal{O}_F$, which can be seen by using the projective representation:

$$F : U^3 + V^3 = W^3, \quad \mathcal{O}_F = [1 : -1 : 0].$$

The transformation

$$U = \frac{6}{X} + \frac{Y}{6X}, \quad V = \frac{6}{X} - \frac{Y}{6X}$$

yields the Weierstrass equation

$$E_F : Y^2 = X^3 - 432,$$

where $\mathcal{O}_F$ transforms to $\mathcal{O} = [0 : -1 : 0]$.

When working with variable transformations between Elliptic Curves, we are interested in transformations which map a Weierstrass normal form into another one. The following is the unique variable transformation that does this,

$$X = u^2X' + r, \quad Y = u^3Y' + u^2sX' + t,$$

where $u, r, s, t \in \mathcal{K}$ with $u \neq 0$, [Sil09]. The inverse transformation is

$$X' = \frac{1}{u^2}(X - r), \quad Y' = \frac{1}{u^3}(Y - sX + sr - t).$$

These transformations are *birational*. We then have

$$\begin{aligned}
ua_1' &= a_1 + 2s,\\
u^2a_2' &= a_2 - sa_1 + 3r - s^2,\\
u^3a_3' &= a_3 + ra_1 + 2t,\\
u^4a_4' &= a_4 - sa_3 + 2ra_2 - (t + rs)a_1 + 3r^2 - 2st,\\
u^6a_6' &= a_6 + ra_4 + r^2a_2 + r^3 - ta_3 - t^2 - rta_1,\\
u^2b_2' &= b_2 + 12r,\\
u^4b_4' &= b_4 + rb_2 + 6r^2,\\
u^6b_6' &= b_6 + 2rb_4 + r^2b_2 + 4r^3,\\
u^8b_8' &= b_8 + 3rb_6 + 3r^2b_4 + r^3b_2 + 3r^4,\\
u^4c_4' &= c_4,\\
u^6c_6' &= c_6,\\
u^{12}\Delta' &= \Delta,\\
j' &= j.
\end{aligned} \tag{4.1}$$

Two equations in Weierstrass normal form are *isomorphic* if there is such a birational transformation between them.

Theorem 4.1. *Let $E|\mathbb{K}$ be an elliptic curve, char$(\mathbb{K}) \neq 2, 3$. Then there exists a birational transformation $\psi : E \to E'$ to a curve $E'|\mathbb{K}$ of the form*

$$E' : Y^2 = X^3 + AX + B, \tag{4.2}$$

with $A, B \in \mathbb{K}$. We say that the curve has a representation in short Weierstrass normal form.

We have already give the proof of this fact in (4.1).

The discriminat and the j-invariant of an Elliptic curve in short Weierstrass normal form are

$$\Delta = -16(4A^3 + 27B^2), \quad j = \frac{-12^3(4A)^3}{\Delta}$$

The only birational transformations which leave short Weierstrass normal form invariant are of the form

$$X = u^2X', \quad Y = u^3Y',$$

for $u \in \mathbb{K}^*$. We then have

$$u^4A' = A, \quad u^6B' = B, \quad u^{12}\Delta' = \Delta.$$

When working with elliptic curves, and later with isogenies, we will see that the j-invariant is a powerful tool to characterize them, the reason is given in the following proposition.

Proposition 4.2. *Two elliptic curves in Weierstrass normal form are isomorphic over $\overline{\mathbb{K}}$ if, and only if, they have the same j-invariant.*

Proof. $\Rightarrow$ We have already proved this fact in (4.1).

$\Leftarrow$ First, assume that $\text{char}(\mathbb{K}) \neq 2, 3$. Let E and E' be two curves given in short Weierstrass normal form:

$$E : Y^2 = X^3 + AX + B, \quad E' : (Y'^2) = (X')^3 + A'X' + B'.$$

Assume that $j = j'$. We know that the only birational transformations that leave short Weirstrass normal form invariant are

$$X = u^2 X', \quad Y = u^3 Y'.$$

Thus, computing u will give us the desired isomorphism. We have

$$\begin{gathered} j = j', \\ (4A)^3(4(A')^3 + 27(B')^2) = (4A')^3(4A^3 + 27B^2), \\ A^3(B')^2 = (A')^3 B^2. \end{gathered}$$

If $A = 0$ then $B \neq 0$, hence $A' = 0$ and $B' \neq 0$. Thus we take $u = \left(\frac{B}{B'}\right)^{1/6}$.

If $B = 0$ then $A \neq 0$, hence $B' = 0$ and $A' \neq 0$. In this we can take $u = \left(\frac{A}{A'}\right)^{1/4}$.
If $AB \neq 0$ then $A'B' \neq 0$. Indeed, if $A'B' = 0$, then $A' = 0$ and $B' = 0$, hence $\Delta' = 0$, which is excluded. We have

$$(A')^3 B^2 = A^3 (B')^2 \leftrightarrow \frac{B^2}{(B')^2} = \frac{A^3}{(A')^3} \leftrightarrow \left(\frac{B}{B'}\right)^{1/6} = \left(\frac{A}{A'}\right)^{1/4}.$$

Thus, we can take $u = \left(\frac{A}{A'}\right)^{1/4} = \left(\frac{B}{B'}\right)^{1/6}$.

□

Moreover, we can always find an elliptic curve having a specific j-invariant $= j_0$, for any $j_0 \in \mathbb{K}$.

Proposition 4.3. *For each $j_0 \in \mathbb{K}$ there exists an elliptic curve defined over $\mathbb{K}$ with $j-$invariant j_0.*

Proof. Assume that $j_0 \notin \{0, 1728\}$ and consider the curve

$$E : y^2 + xy = x^3 - \frac{36}{j_0 - 1728}x - \frac{1}{j_0 - 1728}$$

A simple calculations yields

$$\Delta = \frac{j_0^2}{(j_0 - 1728)^3} \quad j = j_0.$$

This gives the desired elliptic curve (in any characteristic) provided that $j_0 \neq 0$, 1728. For $j_0 = 0$, 1728 we use the curves

$$\begin{aligned} &E : y^2 + y = x^3, \quad \Delta = -27, \quad j = 0, \\ &E : y^2 = x^3 + x, \quad \Delta = -64, \quad j = 1728. \end{aligned}$$

□

4.2 The addition law

Elliptic curve cryptography is based on the fact that the points on elliptic curves form an (additive) abelian group. In this section, we give the addition law of this group and its arithmetic and geometric interpretations.

Let $E|\mathbb{K}$ be an elliptic curve given in (long) Weierstrass normal form over any field $\mathbb{K}$. The set of rational points of E over $\mathbb{K}$ is

$$E(\mathbb{K}) = \{(x, y) \in E : x, y \in K\} \cup \{\mathcal{O}\},$$

where $\mathcal{O}$ is the point at infinity.
Let us enunciate some properties that allow us to define the addition law of points on elliptic curves.

Theorem 4.2. *a) A line intersects an elliptic curve in exactly 3 points (counting multiplicities).*

b) Let C and C' be two cubic curves over an infinite field $\mathbb{K}$ intersecting in exactly nine points in $\mathbb{P}(\overline{\mathbb{K}})$. If C'' is a plane cubic curve over $\mathbb{K}$ going through eight of the intersection points, then it goes through the ninth.

The proof of this is not difficult, but, though interesting, is tangential to the topic of this work, so we omit it and refer the reader to [Ful08].

We can define a binary operation, $+$, over an elliptic curve $E|\mathbb{K}$, we shall see that $(E(\mathbb{K}), +)$ forms an Abelian group; this structure is the base to construct elliptic curve cryptographic systems, [HMV04].

Definition 4.2.1 (Addition of points)**.** Let $P, Q \in E$, let L be the line through P and Q (if $P = Q$, let L be the tangent line to E at P), and let R be the third point of intersection of L with E. Let L' be the line through R and $\mathcal{O}$. Then L' intersects E at R, $\mathcal{O}$ and a third point. This third point is $P + Q$, [Sil09].

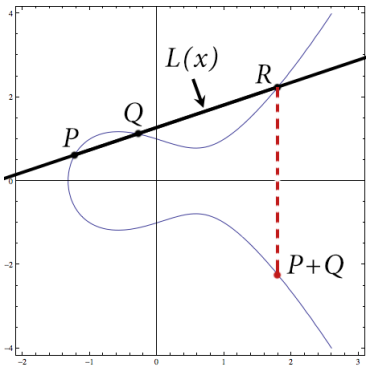


Figure 4.1: Geometric interpretation of the addition of two points in an elliptic curve

Theorem 4.3. *The set $(E(\mathbb{K}), +)$ is an additive Abelian group with the point at infinity $\mathcal{O}$ as the identity element.*
If $\mathbb{K}$ is a number field, the group $E(\mathbb{K})$ is called the Mordell-Weil group of E over $\mathbb{K}$.

Proof.

- (Binary operation and commutativity) It is clear that for $P_1, P_2 \in E(\mathbb{K})$ we have $P_1 + P_2 \in E(\mathbb{K})$. It is also obvious that $P_1 + P_2 = P_2 + P_1$
- (Identity element) Let $P \in E(\mathbb{K})$, then the third point intersecting the line through P and $\mathcal{O}$ is P again. Now, the line through P and $\mathcal{O}$ is the same as above thus, $P + \mathcal{O} = P$. Since this is valid for all $P \in E(\mathbb{K})$ it follows that $\mathcal{O}$ is the identity element.
- (Inverses) Let $P \in E(\mathbb{K})$. Let P' be the third point of intersection of the line through P and $\mathcal{O}$ with the curve. Then $P + P' = \mathcal{O}$, therefore $P' = -P$.
- (Associativity) Let $P_1, P_2, P_3 \in E(\mathbb{K})$. We have to show that

$$(P_1 + P_2) + P_3 = P_1 + (P_2 + P_3);$$

which is equivalent to prove that

$$-((P_1 + P_2) + P_3) = -(P_1 + (P_2 + P_3)).$$

We define the following lines (secants or tangents if two of the points coincide):
L_1: line through P_1 and P_2. This line intersects the curve in the third point $-(P_1 + P_2)$.
L_2: line through P_3 and $(P_1 + P_2)$. This line intersects the curve in the third point $-((P_1 + P_2) + P_3)$.
L_3: line through $(P_2 + P_3)$ and $\mathcal{O}$, This line intersects the curve in the third point $-(P_2 + P_3)$.
L_1': line through P_2 and P_3. This line intersects the curve in the third point $-(P_2 + P_3)$.
L_2': line through P_1 and $(P_2 + P_3)$. This line intersects the curve in the third point $-(P_1 + (P_2 + P_3))$.
L_3': line through $(P_1 + P_2)$ and $\mathcal{O}$, This line intersects the curve in the third point $-(P_1 + P_2)$.
Then we define the cubic curves

$$C := L_1 \cup L_2 \cup L_3, \quad C' := L_1' \cup L_2' \cup L_3'.$$

The curves C and E have no common components (because C is a union of 3 lines). From (4.2) we have that such curves have exactly 9 common points. For the curves C and E these are the points

$$\mathcal{O}, P_1, P_2, P_3, (P_1 + P_2), -(P_1 + P_2), (P_2 + P_3), -(P_2 + P_3), -((P_1 + P_2) + P_3).$$

The curve C' intersects at the first 8 of the common points of C and E. Therefore C' intersects also at the 9-th common point. On the other hand, C' has exactly 9 common points with E:

$$\mathcal{O}, P_1, P_2, P_3, (P_1 + P_2), -(P_1 + P_2), (P_2 + P_3), -(P_2 + P_3), -(P_1 + (P_2 + P_3)).$$

Hence

$$-((P_1 + P_2) + P_3) = -(P_1 + (P_2 + P_3)).$$

□

Computationally speaking we need to define the addition of points in coordinate terms.

Theorem 4.4. *(Addition theorem) Let $E|\mathbb{K}$ be an elliptic curve over the field $\mathbb{K}$ in long Weierstrass normal form and let $P_1 = (x_1, y_1), P_2 = (x_2, y_2) \in E(\mathbb{K})$. Then*

a) $-P_1 = (x_1, -y_1 - a_1x_1 - a_3)$.

b) If $x_1 = x_2$ and $y_2 + y_1 + a_1x_1 + a_3 = 0$, i.e. if $P_1 = -P_2$, then

$$P_1 + P_2 = \mathcal{O}.$$

c) Let $P_1 \neq P_2$. Define

$$\lambda = \frac{y_2 - y_1}{x_2 - x_1},$$
$$\upsilon = \frac{y_1x_2 - y_2x_1}{x_2 - x_1} = y_1 - \lambda x_1,$$

if $x_1 \neq x_2$, and

$$\lambda = \frac{3x_1^2 + 2a_2x_1 + a_4 - a_1y_1}{2y_1 + a_1x_1 + a_3},$$
$$\upsilon = \frac{-x_1^3 + a_4x_1 + 2a_6 - a_3y_1}{2y_1 + a_1x_1 + a_3} = y_1 - \lambda x_1,$$

if $x_1 = x_2$.

Then $P_1 + P_2 = P_3 = (x_3, y_3)$ with coordinates

$$x_3 = \lambda^2 + a_1\lambda - a_2 - x_1 - x_2,$$
$$y_3 = -(\lambda + a_1)x_3 - \upsilon - a_3 = \lambda(x_1 - x_3) - y_1 - a_1x_3 - a_3.$$

Proof. We write

$$E : f(X, Y) = Y^2 + a_1XY + a_3Y - X^3 - a_2X^2 - a_4X - a_6 = 0.$$

a) The line L through $P_1 = (x_1, y_1) \in E(\mathbb{K})$ and $\mathcal{O}$ is

$$L : X = x_1.$$

We compute the intersection point $P_1' = (x_1, y_1) \in E(\mathbb{K})$ and E:

$$\begin{aligned} f(x_1, Y) &= Y^2 + (a_1x_1 + a_3)Y - (x_1^3 + a_2x_1^2 + a_4x_1 + a_6) \\ &= (Y - y_1)(Y - y_1') \\ &= Y^2 + (-y_1 - y_1')Y + y_1y_1'. \end{aligned}$$

Comparing coefficients, we see that

$$y_1' = -y_1 - a_1x_1 - a_3.$$

The third intersection point of L with E is therefore

$$P_1' = (x_1, -y_1 - a_1x_1 - a_3).$$

With Theorem 4.3, this point P_1' is equal to $-P_1$.

b) follows from a).

c) We assume that $P_1 \neq -P_2$. First let $x_1 = x_2$, which means that $P_1 = P_2$. The tangent in P_1 at E is

$$L : f_X(x_1, y_1)(X - x_1) + f_Y(x_1, y_1)(Y - y_1) = 0$$

with the partial derivatives

$$f_X(x_1, y_1) = -(3x_1^2 + 2a_2x_1 + a_4 - a_1y_1), \quad f_Y(x_1, y_1) = 2y_1 + a_1x_1 + a_3.$$

The assumption of $P_1 \neq -P_2$ implies $f_Y(x_1, y_1) \neq 0$. Therefore we write

$$\begin{aligned} l : Y &= \frac{-f_X(x_1, y_1)}{f_Y(x_1, y_1)}(X - x_1) + y_1 \\ &= \frac{-f_X(x_1, y_1)}{f_Y(x_1, y_1)}X + \frac{x_1 f_X(x_1, y_1) + y_1 f_Y(x_1, y_1)}{f_Y(x_1, y_1)} \\ &= \lambda X + \upsilon \end{aligned}$$

with λ and υ as in the theorem.
In the other case, i.e. if $x_1 \neq x_2$ and thus also $P_1 \neq P_2$, the line through P_1 and P_2 is

$$L : \frac{Y - y_1}{X - x_1} = \frac{y_2 - y_1}{x_2 - x_1}.$$

Therefore we have

$$L : Y = \frac{y_2 - y_1}{x_2 - x_1}X + \frac{y_2 - y_1}{x_2 - x_1}(-x_1) + y_1 = \lambda X + \upsilon$$

with λ and υ as in the theorem.

In both cases the line through P_1 and P_2 is given by the equation

$$L : Y = \lambda X + \upsilon$$

. The third intersection point of the line L with E is a point $P_3' = (x_3', y_3')$ (by assumption $P_3' \neq \mathcal{O}$). We now compute this intersection point:

$$\begin{aligned} f(X, \lambda X + \upsilon) &= (\lambda X + \upsilon)^2 + a_1X(\lambda X + \upsilon) + a_3(\lambda X + \upsilon) - X^3 - a_2X^2 - a_4X - a_6 \\ &= -X^3 + (\lambda^2 + a_1\lambda - a_2)X^2 + (2\lambda\upsilon + a_1\upsilon + a_3\upsilon - a_4)X + (\upsilon^2 + a_3\upsilon - a_6) \\ &= -(X - x_1)(X - x_2)(X - x_3') \\ &= -X^3 + (x_1 + x_2 + x_3')X^2 + (-x_1x_2 - x_1x_3' - x_2x_3')X + x_1x_2x_3'. \end{aligned}$$

Comparing coefficients, we see that

$$x_3' = \lambda^2 + a_1\lambda - a_2 - x_1 - x_2.$$

Since P_3' is a point of L, one has

$$y_3' = \lambda x_3' + \upsilon.$$

Te point P_3' is a point of L, one has

$$y_3' = \lambda x_3' + \upsilon.$$

The point $P_3 = (x_3, y_3) = P_1 + P_2$ is $-P_3'$. According to a) this point has the coordinates

$$\begin{aligned} x_3 &= x_3' = \lambda^2 + a_1\lambda - a_2 - x_1 - x_2, \\ y_3 &= -y_3' - a_1 x_3' - a_3 = -(\lambda + a_1)x_3 - \upsilon - a_3. \end{aligned}$$

□

All elliptic curve cryptosystems use the elliptic curve operation on a point P to yield a new point Q that will serve as the ciphertext, [Rey17].

For $m \in \mathbb{Z}$ we write

$$mP \text{ for } \begin{cases} \sum_{j=1}^{m} P & \text{if } m > 0, \\ \mathcal{O} & \text{if } m = 0, \\ \sum_{j=1}^{-m}(-P) & \text{if } m < 0. \end{cases}$$

$$\pi_C(i) = \begin{cases} \pi(i) & \text{if } i \notin C, \\ \pi(i_{j+1}) & \text{if } i = i_j \in C. \end{cases}$$

This operation is called *Integer Multiplication of Elliptic Curves*. We can be sure that the point $mP = (x, y)$ is in the elliptic curve E and its coordinates x and y in the field F, thanks to the properties of groups and fields. Thus, if we use a finite field F, for example, $\mathbf{Z}_p = F$ (with p a prime number), we have finite points (a, b) that satisfy the equation corresponding to E, moreover, since F_p is a partition of $\mathbf{Z}$, we are working only with integer values, thus, a computer can make these operations in a predictable, feasible and precise way, three fundamental things in any cryptographic system.

As an example, consider the elliptic curve

$$E : Y^2 = X^3 + 1$$

over $\mathbb{Q}$. This curve has a $\mathbb{Q}$-rational point $P = (2, -3)$. Using the formulas above we get

$$2P = (0, -1), \quad 3P = (-1, 0), \quad 4P = (0, 1), \quad 5P = (2, 3), \quad 6P = \mathcal{O}.$$

Notice that $5P = -P$ and the point P has order 6.

In elliptic curve cryptography, we have to add points by the addition law. We have different methods for the representation of points on elliptic curves which lead to different addition formulas. We need to choose methods that have the smallest amount of computing time. We denote the computing times for taking the inverse in the field $\mathbb{K}$ by I, for multiplication in $\mathbb{K}$ by M, and for squaring in $\mathbb{K}$ by S. In the study of the complexity of these algorithms, we neglect

addition, subtraction, and multiplication by a small field constant. We denote $t(P_1 + P_2)$ to the computing time for an addition of two different points, the computing time for doubling by $t(2P)$.

We have already shown the addition formulas for the affine representation of the long Weierstrass normal form. In cryptography, we use fields with characteristic > 3 or 0, so elliptic curves can be transformed into short Weierstrass normal form

$$E : y^2 = x^3 + ax + b. \tag{4.3}$$

We consider the *affine representation on the short Weierstrass normal form.* Let

$$P_1 = (x_1, y_1), \quad P_2 = (x_2, y_2) \in E, \quad P_1 \neq -P_2.$$

Then, $P_1 + P_2 = (x_3, y_3)$ is

$$\begin{aligned} x_3 &= \lambda^2 - x_1 - x_2, \\ y_3 &= \lambda(x_1 - x_3) - y_1 \end{aligned}$$

with

$$\lambda = \begin{cases} \dfrac{y_2 - y_1}{x_2 - x_1} & \text{if } P_1 \neq P_2, \\ \dfrac{3x_1^2 + a}{2y_1} & \text{if } P_1 = P_2. \end{cases}$$

We have $t(P_1 + P_2) = I + 2M + S$, $t(2P_1) = I + 2M + 2S$. For the projective representation, if

$$E : Y^2Z = X^3 + a_4XZ^2 + a_6Z^3$$

is the projective elliptic curve and

$$P_1 = [x_1 : y_1 : z_1], P_2 = [x_2 : y_2 : z_2] \in E, \quad P_1 \neq P_2.$$

Then $P_1 + P_2 = [x_3 : y_3 : z_3]$ with

$$\begin{aligned} x_3 &= bc, \\ y_3 &= a(b^2x_1z_2 - c) - b^3y_1z_2, \\ z_3 &= b^3z_1z_2, \end{aligned}$$

where

$$\begin{aligned} a &:= y_2z_1 - y_1z_2, \\ b &:= x_2z_1 - x_1z_2, \\ c &:= a^2z_1z_2 - b^3 - 2b^2x_1z_2, \end{aligned}$$

if $P_1 \neq P_2$ and

$$\begin{aligned} x_3 &= 2ab, \\ y_3 &= a(4c - d) - 8y_1^2b^2, \\ z_3 &= 8b^3, \end{aligned}$$

where

$$\begin{aligned} a &:= a_4 z_1^2 + 3x_1^2, \\ b &:= y_1 z_1, \\ c &:= x_1 y_1 b, \\ d &:= a^2 - 8c, \end{aligned}$$

if $P_1 = P_2$. Here we obtain $t(P_1 + P_2) = 12M + 2S$, $t(2P_1) = 6M + 5S$.

For this work, we will only consider elliptic curves given in short Weierstrass normal form (4.3). Hence, a curve E over a field K is completely determined by the points $a, b \in K$.

4.3 Elliptic Curves over finite fields and Supersingular curves

When working with elliptic curve cryptosystems we work with curves over finite fields. For what follows, let p be a prime number, $k \in \mathbb{N}$, $q = p^k$, and $\mathbb{F}_q$ be the finite field with q elements.

The following definition introduces a central aspect behind isogeny based cryptography: *torsion groups*. We also give characterizations of such groups and classify the curves according to them.

Definition 4.3.1. Let $E|\mathbb{K}$ be an elliptic curve and $n \in \mathbb{N}$. The set

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

is called the set of **n-division points** of E, or the n-torsion group of E. The $\mathbb{K}$-rational n-division points of E are

$$E(\mathbb{K})[n] := \{P \in E(\mathbb{K}) : nP = \mathcal{O}\}.$$

Hence $E[n] = E(\bar{\mathbb{K}})[n]$ for the algebraic clousure $\bar{\mathbb{K}}$ of $\mathbb{K}$.

Proposition 4.4 ([Law08]). *Let $p = char(\mathbb{K})$ and consider m such that $(m, p) = 1$. Then:*

$$E[m] \cong \mathbb{Z}/m\mathbb{Z} \times \mathbb{Z}/m\mathbb{Z}. \tag{4.4}$$

For the remaining case of $m = p^e$, $e \in \mathbb{Z}^+$, two types of behaviour occur. Either

$$E[p^e] = \mathbb{Z}/p^e\mathbb{Z} \times \mathbb{Z}/p^e\mathbb{Z}, \quad e = 1, 2, 3, ... \tag{4.5}$$

in which case E is said to be ordinary, or

$$E[p^e] = \mathcal{O}, \quad e = 1, 2, 3, ... \tag{4.6}$$

in which case E is said to be supersingular.

Supersingular elliptic curves are essential to the best isogeny based cryptographic protocol. The former can be seen as a characterization. For its definition, we need to introduce the Frobenius endomorphism (refer to Appendix A for a definition of endomorphism).

Definition 4.3.2. Let $E|\mathbb{F}_q$ be an elliptic curve over the finite field $\mathbb{F}_q$. The **q-Frobenius endomorphism** $\varphi_q : E \to E$ is given by

$$\varphi_q(x, y) = (x^q, y^q), \quad \varphi_q(\mathcal{O}) = \mathcal{O}.$$

As $\varphi_q \in \text{End}(E)$ for elliptic curves $E|\mathbb{F}_q$, it follows that elliptic curves over finite fields always have complex multiplication.

Some properties of the *q-Frobenius endomorphism* are given in the next theorem.

Theorem 4.5. *Let $E|\mathbb{F}_q$ be an elliptic curve and φ_q the q-Frobenius endomorphism.*

a) Let $P \in E$. Then

$$P \in E(\mathbb{F}_q) \iff \varphi_q(P) = P.$$

b) The endomorphism φ_q is purely inseparable, i.e., the field extension $\mathbb{F}_q(E)/\mathbb{F}_q(E)^q$ is purely inseparable[1].

c) The degree of φ_q is deg(φ_q)= q.

d) There exists an integer $t = t_q$ such that

$$\varphi_q^2 - t\varphi_q + q = 0,$$

that is to say that, for all $P \in E$, we have the equation

$$\varphi_q^2(P) - t\varphi_q(P) + qP = \mathcal{O}.$$

This integer is called the trace of the q-Frobenius endomorphism.

e) The trace t of the q-Frobenius endomorphism is linked with the number of rational points by the relation:

$$\#E(\mathbb{F}_q) = q + 1 - t,$$

where $|t| \leq 2\sqrt{q}$.

Definition 4.3.3. Let $E|\mathbb{F}_q$ be an elliptic curve over a finite field of characteristic p with $|E(\mathbb{F}_q)| = q+1-t$. The curve is called **supersingular** if $p|t$. A curve which is not supersingular is called **ordinary**.

In cryptography, we do not use this definition of supersingular, rather we use the following characterizations.

Theorem 4.6. *Let $E|\mathbb{F}_q$ be an elliptic curve.*

a) The curve is supersingular if, and only if, End(E) is the maximal order in a quaternion algebra.

[1] A purely inseparable extension of fields is an extension $k \subseteq K$ of fields of characteristic $p > 0$ such that every element of K is a root of an equation of the form $x^q = a$, with q a power of p and a in k.

b) If char($\mathbb{F}_q$) = 2 or 3, the curve is supersingular if, and only if, $j(E) = 0$.

A proof of this Theorem can be found in [BBS06].

Another useful characterization for supersingular fields in prime fields is given by the following corollary of Theorem 4.5 c).

Corollary 4.1. *Let $E/\mathbb{F}_q$ be an elliptic curve. Then E is supersingular if, and only if, $\#E(\mathbb{F}_p) = p + 1$.*

Chapter 5

Elliptic Curve Cryptography

This chapter presents the most used algorithms, protocols, and problems associated with the use of elliptic curves in cryptography. We provide a description of how Elliptic Curve Cryptography (ECC) works, its current limitations, and future prospects.

5.1 Elliptic curve cryptography: History and its place in modern cryptography

Up until the 1970s, cryptography was all done symmetrically. When asymmetric methods were needed because of some of the reasons we discussed in previous chapters, Diffie, Hellman, and Merkle demonstrated the existence of such public-key cryptographic methods in 1976 [DH76]. They suggested a protocol now called Diffie-Hellman-Merkle Key Exchange, [Hel02], which is based on the Diffie-Hellman problem (3.3.5).

A year later, in 1977, Ron Rivest, Adi Shamir, and Leonard Adleman proposed another asymmetric encryption scheme, the famous RSA, [Adl78]. The security of RSA is based on the discrete factoring problem.

Definition 5.1.1. (Discrete factoring problem) Given a positive natural number N, a factor of two large primes p and q, find p and q.

The easy mathematics behind this problem allowed RSA to become the first public key algorithm to go into wide use, and is still adopted in many modern protocols. RSA improves DFM by providing signatures, and long term private keys. However, the factoring problem has sub-exponential algorithms, [Pom87], and thus, the increase in computational power has to lead to the growth of RSA-key sizes at a similar rate, which is inconvenient for example when working with embedded computer systems such as smartphones, [Sha04].

Victor Miller and Neal Koblitz introduced elliptic curve cryptography in 1985. Independently, they developed the idea of using elliptic curves as the basis of a group for the discrete logarithm problem, [Kob87] ,[Mil85]. Elliptic curve cryptography quickly gained interest because it provides more security with smaller key sizes. The NSA has Elliptic curve as its standard suite B algorithm for both encryption and signature, [Sha04].

5.2 Representing messages as points on Elliptic Curves

The procedure given in this section comes from [SS03].

Assume that a text unit consists of integers m_i such that $0 \leq m_i < M$, $M \in \mathbb{N}$. Let $k \in \mathbb{N}$, $p \in \mathbb{P}$, $p > 2$, $q = p^k$, and $q > Mk$. We choose an elliptic curve given in short Weierstrass form $E : Y^2 = X^3 + a_4X + a_6$ over $\mathbb{F}_q$ so that $j = 0$ if $p = 3$. There are two steps to represent the text units as points on $E(\mathbb{F}_q)$:

a) The elements in $\{1, ..., Mk\}$ are represented as elements of $\mathbb{F}_q$ using their p-adic representation:
Let $N \in \{1, ..., Mk\}$. Then there are integer numbers n_i such that:

$$N = \sum_{i=0}^{k-1} n_i p^i, \quad 0 \leq n_i < p.$$

On the other hand, using the euclidean division algorithm, take an integer m_i satisfying $0 \leq m_i < M$, such that

$$N = m_i k + j$$

with $1 \leq j < k$.
The field $\mathbb{F}_q$ is given as $\mathbb{F}_q \cong \mathbb{F}_p[x]/(f(x))$, where $f(x)$ is any polynomial irreducible over $\mathbb{F}_p$ and of degree k. We assign to $N = m_i k + j$ the element

$$x_{m_i,j} := \sum_{i=0}^{k-1} n_i X^i \pmod{f(X)}.$$

Conversely, from an element $x \in \mathbb{F}_q$ we can recover the parameters m_i, j which define a number $1 \leq N = m_i k + j \leq Mk$. Let

$$0 \neq x = \sum_{i=0}^{k-1} n_i X^i \pmod{f(X)}.$$

Since $N < p^k = q$, by construction we have

$$1 \leq N \leq Mk \iff 0 \leq N - 1 < Mk \iff 0 \leq \left\lfloor \frac{N-1}{k} \right\rfloor < M.$$

Hence we get the parameters

$$m_i := \left\lfloor \frac{N-1}{k} \right\rfloor, \quad j := N - mk.$$

It remains to show that $1 \leq j < k$:

$$\begin{aligned} 1 &= N - \frac{N-1}{k}k \\ &\leq N - \left\lfloor \frac{N-1}{k} \right\rfloor k \\ &= j \\ &= \frac{N-1}{k}k - \left\lfloor \frac{N-1}{k} \right\rfloor k + 1 \\ &< \left(\left\lfloor \frac{N-1}{k} \right\rfloor + 1 \right) k - \left\lfloor \frac{N-1}{k} \right\rfloor k + 1 \\ &= k + 1. \end{aligned}$$

b) Representation of the numbers $0 \leq m_i < M$ as points in $E(\mathbb{F}_q)$:
To represent a number m_i as a point in $E(\mathbb{F}_q)$, we first consider the number

$$N = m_i k + 1.$$

In Part a) we determined the element $x_{m_i,1} \in \mathbb{F}_q$ corresponding to N. Now we test if $x_{m_i,1}^3 + a_4 x_{m_i,1} + a_6$ is a square in $\mathbb{F}_q$. If this is the case, there exists an $y_{m_i,1} \in \mathbb{F}_q$ such that

$$y_{m_i,1}^2 = x_{m_i,1}^3 + a_4 x_{m_i,1} + a_6.$$

Then m_i is assigned the point $(x_{m_i,1}, y_{m_i,1})$. Otherwise consider the number $N = m_i k + 2$ and repeat the method.
In this way we consider successively all numbers $m_i k + j$ for $j = 1, 2, ...$ until we have found a point $(x_{m_i,j}, y_{m_i,j})$. If $j > k$ and no points is found, we have to pick another curve.
For any $x \in \mathbb{F}_q$, we have, in principle, q possibilities for the value

$$x^3 + a_4 x + a_6.$$

Only for $\frac{1}{2}|E(\mathbb{F}_q)|$ of these values we get a point on $E(\mathbb{F}_q)$. So the probability that x leads to a point is

$$\frac{|E(\mathbb{F}_q)|}{2q} \approx \frac{1}{2}.$$

By Hasse's estimate, we know that $(\sqrt{q}-1)^2 \leq |E(\mathbb{F}_q)| \leq (\sqrt{q}+1)^2$. Therefore, for a given m_i, the probability that $m_i k + 1, m_i k + 2, ..., m_i k + k$ lead to no point is about 2^{-k}. From a point $P = (x, y) \in E(\mathbb{F}_q)$, which represents the number m_i, we obtain m_i by using the method of Part a) for x.

5.3 Elliptic Curve Diffie-Hellman and elliptic curve discrete logarithm problem

Elliptic curve Diffie-Hellman is a key exchange protocol. Its security is based on the difficulty of the discrete logarithm problem for elliptic curves.

Definition 5.3.1. (**Discrete logarithm problem on elliptic curves** [SS03]) Let $E|\mathbb{F}_q$ be an elliptic curve over a finite field $\mathbb{F}_q$ and let $P \in E(\mathbb{F}_q)$ be a point of E over $\mathbb{F}_q$. The **discrete logarithm problem on E** is the question if, to a given point $Q \in E(\mathbb{F}_q)$, there exists an integer n with $Q = nP$ and if one can compute this n.

In practice, and recommended by security agencies, we usually work with finite fields of the form $\mathbb{Z}_p$, with p prime, or $\mathbb{Z}_{2^n}$, with n positive integer. The agencies also suggest "good" curves and "good" base points in a way we shall later explain. All of these suggestions are based on making it difficult to solve the discrete logarithm problem.

The scheme proposed for Diffie-Hellman applied to elliptic curves work as follows.

1. Alice and Bob would like to communicate over a secured channel using elliptic curves. First Alice and Bob agree on a set of domain parameters required to set up the Elliptic Curve protocol, namely:

   - The order q of the finite field $\mathbb{F}_q$
   - An elliptic curve $E : y^2 = x^3 + ax + b$; with $a, b \in \mathbb{F}_q$, defined over $\mathbb{F}_q$. This is, a point $(x, y) \in \mathbb{F}_q^2$ is in the curve E if, and only if, $y^2 \equiv x^3 + ax + b \bmod q$.[1]
   - A 'base' point $P \in E(\mathbb{F}_q)$ which generates a subgroup of E.
   - The order k of P.
   - The cofactor h of the group generated by P, i.e., the positive integer such that $hk = |E(\mathbb{F}_q)|$.

   The procedure to generate these parameters is computationally expensive; so Alice and Bob will (and should) use pre-computed parameters suggested by security agencies.

2. Having agreed on a set of parameters, Alice and Bob now have access to two types of keys:

   - A **private key** $n \in \mathbb{N}$.
   - The corresponding **public key** $Q = nP$.

   Both Alice and Bob generate a key-pair $(n, Q = nP)$. Let (n_A, Q_A) and (n_B, Q_B) be their pairs respectively.

3. Alice and Bob exchange their public key over a potentially insecure channel. An eavesdropper may learn Q_A or Q_B but won't be able to compute n_A or n_B, thanks to the difficulty of the discrete logarithm problem.

4. Alice computes $n_A \cdot Q_B$, and Bob computes $n_B \cdot Q_A$. Alice and Bob now share the point $n_A n_B P$. They can use the x or y coordinate of the shared point to establish a secured symmetric channel for communication.

[1] $E(\mathbb{F}_q)$ is an abelian group where the addition law and the formulas given in last chapter still apply, though one needs to do all the computation in mod q.

5.4 ElGamal Method for elliptic curves

The ElGamal encryption system is an asymmetric key encryption algorithm for public-key cryptography which is based on the Diffie Hellman key exchange. It can be defined over any cyclic group G, [PP10].

The ElGamal method for elliptic curves works under the assumption that the discrete logarithm problem on elliptic curves is difficult to solve, [SS03].

In the ElGamal method, we represent the text units $0 \leq m < M$ as points P_m of $E(\mathbb{F}_q)$. Suppose Alice wants to send a secret message to Bob. Both decide to use a basis point $P \in E(\mathbb{F}_q)$ as the "channel of communication", [SS03]. In practice, this basis point is public and it has been tested to be a good point in a way that we explain later. By his side, Bob chooses a secret integer n, which is his private key and makes the point nP public, this point is his public key. On the other hand, Alice chooses a secret integer k, her private key, and maps the text unit m to the pair

$$(kP, P_m + k(nP)).$$

Here, kP is Alice's public key and $P_m + k(nP)$ is Alice and Bob's shared message or the ciphertext. (The term knP is the shared point that we got on the Diffie Hellman key exchange protocol.) Moreover, notice that $P_m + k(nP)$ is a point in $E(\mathbb{F}_q)$ by group properties. Bob can read the plain text m from this pair because he knows the number n, that is, he can compute

$$P_m = (P_m + k(nP)) - n(kP).$$

Notice, that in this process, neither Alice nor Bob needs to know the private key of each other. Moreover, a spy Eve knows neither n nor k, so normally she cannot compute the point P_m and so decipher the message, [SS03].

Of course, this method relies upon the assumption that Eve cannot know the private keys of Alice or Bob. That is, she cannot compute the integers n or k such that $Q_A = kP$, $Q_B = nP$.

5.5 Key pair generation using ECC

In both, the DH protocol and the ElGamal method, we need to generate the private-public pair of keys. To obtain them we use the following algorithm provided we have a set of parameters defined as above.

Algorithm 1 Key pair generation [HMV04]

INPUT: Domain Parameters $D = (q, E, P, k, h)$
OUTPUT: Public key Q, private key d.
PROCEDURE:

1: Select $d \in [1, k-1]$.
2: Compute $Q = dP$.
3: Return (Q, d).

We don't use h since we will work only with parameters already established by some security party. Notice that for an adversary would obtain the private key d from the public key Q, it must solve the ECDLP. The choice of d must be arbitrary to avoid attacks. The integer multiplication in the state 2 is what "dominate the execution time of elliptic curve cryptographic schemes", [HMV04]. Many algorithms compute the integer multiplication in a much more efficient way than the one that we present, see [HMV04] Chapter 3. However, as an educational approach, we consider this as the best way to understand the procedure. It is based on the basic repeated-square-and-multiply methods for exponentiation, [HMV04].

Algorithm 2 Integer Multiplication of Elliptic Curves

INPUT: $(q, E, P \in E(F_q), d)$
OUTPUT: $Q = mP$
PROCEDURE:

1: $N \leftarrow P$
2: $Q \leftarrow \mathcal{O}$
3: $b \leftarrow d_2 = (k_{t-1}, ..., k_1, k_0)_2$
4: **for** i from 0 to $t-1$ **do**
5: **if** $k_i = 1$ **then**
6: $Q \leftarrow Q + P$
7: **end if**
8: $P \leftarrow 2P$
9: **end for**
10: Return Q

This algorithm is more efficient than the direct approach of adding P to itself d times. For example, if d were a 256-bit number, then this algorithm would require a maximum of 512 elliptic-curve operations, whereas the direct method would require (at least) $2^{255} - 1$ operations, [Ent14].

5.6 Elliptic curve digital signature algorithm (ECDSA)

We can use elliptic curves to create a scheme for message authentication that satisfies the requirements given in 3.4. The ECDSA protocol works as follows, [Woh]:

1. Alice wants to send Bob a message m. She takes a hash of m and truncates it so that it has bit-length k, where k is the order of the base point P. Let z be the resulting truncated hash of m.

2. Alice has its own random key-pair (n_A, Q_A). She chooses a cryptographically secture randome integer n from $[1, k-1]$, the choice of this n should be random to prevent attacks, as we will in Section 5.8. Alice computes $Q = nP$. Let x_A be the x-coordinate of the point Q, modulo k. If $x_A = 0$, then select another n.

3. Alice then calculates $s = n^{-1}(z + x_A n_A) \bmod k$, where n^{-1} is the multiplicative inverse of n mod k. If $s = 0$, go back to previous step. The pair (x_A, s) forms the signature.

To verify if the message m was sent by Alice, Bob proceeds as follows:

1. Bob computes $u_1 = s^{-1}z \bmod k$.
2. Bob computes $u_2 = s^{-1}x_A \bmod k$.
3. Finally, Bob computes the point $Q' = u_1P + u_2Q_A$ and verifies $x_A = x_{Q'} \bmod n$.

To see that this is a correct scheme consider the following proof of correctness.

Proof. We have that

$$\begin{aligned} Q' &= u_1P + u_2Q_A \\ &= u_1P + u_2n_AP \\ &= (u_1 + u_2n_A)P \\ &= (s^{-1}z + s^{-1}x_An_A)P \pmod{k} \\ &= s^{-1}(z + x_An_A)P \pmod{k} \\ &= n(z + x_An_A)^{-1}(z + x_An_A)P \pmod{k} \\ &= nP. \end{aligned}$$

Notice that this is only possible if the message m was signed by Alice's private key. Notice also that Bob does not need to know Alice's private key. □

5.7 ECC versus RSA

To see the power of the ECC, let us compare it to the RSA protocol, which is the most used cryptographic system. For a review on RSA cryptography, see [NF17].

The principal criteria that need to be satisfied by a technique of PKC are functionality, security, and performance, [HMV04]. Both RSA and ECC provide such features and are the most efficient algorithms in PKC, [Kum06]. However, they differ in how much security we can obtain regarding the size of the keys generated. For example, a "key size of 160-bit ECC has an equivalent security level with a key of 1024-bit RSA", [BYMH13]. This is not an isolated case, Torri and Nokoyama, [TY00] concluded that RSA security required 1024 bits for corporate use and 2048 bits for extremely valuable keys. Hence, it is clear that ECC is more efficient than RSA when we use the pair security/key-size as the parameter of comparison. To see this in more detail, consider the table given below.

Table 5.1: Key Size Ratio for RSA and ECC with equivalent security level [Age09]

Key size		**Key size ratio**	**Security Level (bits)**	**Ratio of Cost**
RSA	**ECC**			
1024	160	7:1	80	3:1
2048	224	10:1	112	6:1
3072	256	12:1	128	10:1
7680	384	20:1	192	32:1
15360	521	30:1	256	64:1

Table 5.7 shows the key sizes used in RSA and ECC, the level of security (in bits) that each key provides, the ratio between the sizes of the keys, and the ratio of cost to obtain such keys. It is clear that ECC is a better option when we are working with embedded systems like mobile devices, electronic cards, etc. This is because in such systems the restrictions of memory are noticeable; so smaller keys are more effective since it requires fewer hardware resources. In fact, the transmission of smaller keys requires; less memory for storage, low cost of arithmetic computation, and low bandwidth, [Lau04].

Nevertheless, there are certain difficulties for ECC implementation in hardware and software, [BYMH13]. For example, the implementation of ECC in software requires higher power consumption than in hardware, [PKS09]. And of course, it is the fact the ECDLP could have a feasible solution (unlikely) and this will destroy this public-key scheme. However, efficient EC arithmetic algorithms need to improve in order to maximize the application of ECC in the smaller embedded systems. Moreover, the promise of quantum computation puts elliptic curve cryptography into a problem, and all other cryptosystems based on the discrete logarithm problem, since it has been shown that there are quantum algorithms that solve the DLP in polynomial time, [Sho99]. Even more, it turns out that quantum computers are better at breaking elliptic curve cryptography than RSA. Pross and Zalka, [PZ03] found that a 160-bit elliptic curve cryptographic key could be broken on a quantum computer using around 1000 qubits while factoring the security-wise equivalent 1024 bit RSA modulus would require about 2000 qubits.

Before getting into quantum computing and quantum attacks on ECC, let us discuss some classical attacks that threaten ECC.

5.8 Classical attacks on ECC

Recall from Section 5.4, that the difficulty of the ECDLP and thus, the security of ECC protocols depends on the finite field $\mathbb{F}_q$, the elliptic curve E and the basis point P. Let us describe what classes of elliptic curves should *not* be used for cryptosystems and a brief overview of some of the classical algorithms that try to break ECC.

For arbitrary finite Abelian groups, Pohlig and Hellmann showed that the DLP on finite abelian groups can be reduced to the DLP on finite abelian groups of prime power order. Thus, we can solve the DLP for the subgroups of primer order first, then apply this result to solve the DLP for the subgroups of primer power order to finally solve the DLP for the whole group using the Chinese Remainder Theorem, [PH06]. Hence, we require that the group of points to use should have at least one subgroup of large prime order. If the group order is not too large we can solve the DLP associated with it, using methods like the bay step-giant step or λ-method suggested by Pollard, [SS03] &[Pol00].

For the specific case of DLP on elliptic curves over finite fields, there are methods such as MOV-attack, [LMS+04], and the method of Frey and Ruck, [Gal01]. In particular, MOV-attack reduces the DLP on elliptic curves to that on finite fields, if the smallest value of l such that $q^l \equiv 1 \pmod{m}$ is not too large. This result eliminates the use of supersingular curves since, if a curve is supersingular, it follows that

$$m = |E(\mathbb{F}_q)| = q + 1 - t \quad \text{with } t^2 = 0, q, 2q, 3q, \text{ or } 4q.$$

In the cases $t^2 = 0, q, 2q, 3q$ there exists a j, $1 \leq j \leq 6$, with $q^j \equiv 1 \pmod{m}$.

Other curves that we need to avoid are anomalous elliptic curves. An elliptic curve $E|\mathbb{F}_q$ is called *anomalous* if the trace of the Frobenius endomorphism is 1. Smart, Satoh, and Araki, [KM00] proposed the use of p-adic elliptic logarithm to directly solve the DLP for anomalous elliptic curves.

In summary, let $E|\mathbb{F}_q$ be an elliptic curve with $m = q + 1 - t = |E(\mathbb{F}_q)|$. For E to be used in elliptic curve cryptography, the following properties are required:

- The group $E(\mathbb{F}_q)$ should have a subgroup of large prime order.
- The curve E should not be anomalous (i.e. $q \neq m$).
- The smallest value of l such that $q^l \equiv 1 \pmod{m}$ should be large. (This condition removes curves with $t = 0$, $t = 2$ (for $q \in \mathbb{P}$), and supersingular curves.)

For specific ECC protocols, there are also other security issues that we need to take care of. In 5.6, we mentioned that the number $n \in [1, k-1]$, that Alice uses for signing her message, needed to be random. The reason for this, is given by the following proposition.

Proposition 5.1. *Let n be a number in $[1, k-1]$ for the signature algorithm ECDSA. If n is fixed, then we can recover the private key n_A from Alice's signature.*

Proof. Let (r_1, s_1) and (r_2, s_2) be two signatures Alice made on two different messages, using the same "random" number n. Since $r = x_Q$, where $Q = nP$, then since n is fixed, we have $r_1 = r_2$. We also know that $s_1 - s_2 = n^{-1}(z_1 - z_2) \bmod n$, where z_1 and z_2 are the hashed truncated versions of the respective messages. Thus we obtain $n = (z_1 - z_2)(s_1 - s_2)^{-1} \bmod n$. The private key can thus be recovered:

$$n_A \equiv r_1(s_1 n - z_1) \bmod n.$$

□

n does not need to be fixed for ECDSA to be insecure. In 2011, Sony suffered an attack on its ECDSA for using a vulnerable pseudo-random number generator for the seed n in the ECDSA signature scheme, which led to a similar attack to what is described above, [Woh].

Another possible security issue is that the elliptic curve in our protocol may have been built with a backdoor[2]. However, it is possible to check that a curve was generated randomly, which suggests that it would have been difficult to make it weak on purpose, [Woh].

Proposition 5.2. *Curve parameters generated by a hash function are verifiably random.*

Proof. Given a random seed S, it is possible to generate parameters a and b (describing the elliptic curve E in short Weierstrass form) using a hash of s, $H(s)$. Then, an arbitrary function can generate a and b from $H(s)$. The idea is that given the parameters and the seed, one can check that the parameters came from that seed. However, specifying a certain seed for desired parameters requires to inverse the hash function, which is computationally very hard. □

[2] A "backdoor" in computing is a method of bypassing the normal method of authentication. In cryptography specifically, a backdoor would allow an intruder to access the encrypted information without having the correct credentials. The backdoor would either a) allow the intruder to guess the access key based on the context of the message or b) allow the intruder to present a skeleton key that will always grant him access [Won16].

To finish this section, let us describe the best known classical algorithm for breaking elliptic curve cryptosystems: *Pollard's* ρ algorithm.

Suppose $Q = nP$. Let k be the order of the subgroup generated by the point P. *Pollard's* ρ algorithm can find n from Q and P in time roughly $\mathcal{O}(\sqrt{k})$, which is exponential in the number of bits of k.

The first step in Pollard's ρ is to find integers a, b, c, d such that $aP + bQ = cP + dQ$, where $(a, b) \neq (c, d)$. We could try to run over all possible randomly selected pairs and find the integers in $\mathcal{O}(k^2)$ running time. A more efficient method, Floyd's cycle, lives at the heart of Pollard's ρ and is the key to solve ECDLP.

Algorithm 3 Floyd's cycle finding algorithm

INPUT: P, Q
OUTPUT: (a, b, c, d)
RUNTIME: $\mathcal{O}(\sqrt{k})$
PROCEDURE:

1: Choose a random sequence S of (a, b) pairs.
2: Use pointers $p_1 = 0$ and $p_2 = 0$ to walk S.
3: **while** $aP + bQ \neq cP + dQ$ **do**
4: $\quad p_1 = p_1 + 1$
5: $\quad p_2 = p_2 + 2$
6: $\quad (a, b) = S[p_1]$
7: $\quad (c, f) = S[p_2]$
8: **end while**
9: **return** (a, b, c, d)

Algorithm 4 Pollard's ρ

INPUT: P, Q
OUTPUT: n
RUNTIME: $\mathcal{O}(\sqrt{k})$
PROCEDURE:

1: Use Floyd's cycle finding algorithm to find two pairs (a, b) and (c, d) such that $aP + bQ = cP + dQ$.
2: Then $n = (a - c)(d - b)^{-1} \bmod k$, where k is the order of the subgroup generated by P.
3: **return** n.

Let us prove that this algorithm gives n.

Proof. Given two pairs (a, b) and (c, d) distinct such that $aP + bQ = cP + dQ$ as follows:

$$\begin{aligned} aP + bQ &= cP + dQ \\ aP + bnP &= cP + dnP \\ (a-c)P &= (d-b)nP \\ (a-c)P &\equiv (d-b)nP \\ (a-c) &\equiv (d-b)n \\ n &\equiv (a-c)(d-b)^{-1}. \end{aligned}$$

□

We have shown that Pollar's ρ breaks standard elliptic curve protocols, however, it takes months for doing so. This is the reason to believe that elliptic curves will remain strong before going into quantum computation. Because, as we have observed many times, quantum computers can break the discrete logarithm problem of elliptic curves, and to see how they can achieve such work, we turn to the study of quantum computation.

Chapter 6

Quantum Computation

Unless stated otherwise, background material in the following chapter is collected from [MAN10].

One of the most promising advances in science and technology is the quantum model of computation. Quantum computers could spur the development of breakthroughs in science, mathematics, and cryptography. Until now, we have been reasoning in terms of classical computing. This simple model, which we studied in Chapter 2 using Turing Machines, has already a lot of benefits. However, as we saw with NP problems, there are challenges that today's systems will never be able to solve, or at least to solve efficiently. This is a good thing in certain cases, for example, cryptography. We have mentioned that the security of elliptic curve cryptography relies upon the fact that classical computers cannot solve the discrete logarithm problem efficiently. Nevertheless, universal quantum computers based on the quantum mechanical phenomena of superposition and entanglement could be powerful enough to tackle problems that no classical computers can solve. In fact, in 1994, Peter Shor [Sho99] described a quantum algorithm for efficiently solving discrete logarithms over finite groups. Thus, if one day we can construct a large-scale quantum computer that can execute such an algorithm then it is clear that the landscape of cryptography will be forever changed. In this section, we give an introduction to quantum computing, how to develop quantum algorithms and how Shor's algorithm works. With respect to the complexity of quantum algorithms we use assymtotic notations $\mathcal{O}', \Omega', \Theta'$ as defined in the classical setting but with respect to to the quantum setting. For example, a problem is in $\mathcal{O}'(p)$ where p is a polynomial, if there is a quantum algorithm that solves the problem in polynomial time.

6.1 Overview of Quantum Computing

Quantum computation is the study of tasks that can be accomplished using quantum mechanical systems. Quantum mechanics was first discovered in 1920 as a way to explain physical phenomena that classical physics failed to do. Since its discovery, quantum mechanics has been an indispensable part of science and technology. Quantum mechanics is a mathematical framework for the construction of physical theories. Quantum computation was born as a way to satisfy the desire of physicists to better understand quantum mechanics. Many results in quantum mechanics are not intuitive, so one of the goals of quantum computation is to develop tools that sharpen our intuition about quantum mechanics. Physicists developed the idea of quantum computation to try to control quantum systems instead of just understand them. Conversely,

the ability to control quantum systems leads to a better application of quantum computation, [MAN10]. Today we have small quantum computers in some tech companies like Amazon or Google, [TQ19], however, these systems are only capable of doing dozens of operations on a few quantum bits (the unit of information for quantum computation). We also have programming languages and libraries to develop real quantum algorithms, but their use is focused on doing theoretical research, [WVMN19]. Nevertheless, one field where quantum computing has ready to go applications is cryptography. For example, we mentioned in Chapter 3 that Symmetric cryptosystems rely on the secure transmission of keys, quantum mechanics can be used to do key distribution in such a way that Alice and Bob's security cannot be compromised. To do this we exploit the quantum mechanical principle that observation, in general, disturbs the system being observed. To understand how quantum computing works, we need to define its unit of information, the qubit.

6.2 Quantum Bits

In classic computation, the basic unit of information is the *bit*. A bit is an element that can be in two states: 0 and 1. We can observe the state in which the bit is. In quantum computation, the basic unit of information is the *qubit*. A qubit is a mathematical object with certain specific properties that try to simulate some properties of quantum systems. A qubit can be in states $|0\rangle$ and $|1\rangle$ or in a combination of them, called **superposition**. The symbol $|\cdot\rangle$ corresponds to **the braket notation**, also called *Dirac notation*, which is used in quantum mechanics to denote vectors in a complex vector space. Thus, the mathematical model of a qubit is the following:

$$|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \quad |1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix},$$

For a qubit in superposition, we mean:

$$|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle\,,$$

with $\alpha, \beta \in \mathbb{C}$, such that $|\alpha|^2 + |\beta|^2 = 1$. α and β are called amplitudes. It is clear that the set $\{|0\rangle, |1\rangle\}$ is a basis of $\mathbb{C}^2$. When we observe the state of a qubit it will always be in state $|0\rangle$ with probability $|\alpha|^2$ or in state $|1\rangle$ with probability $|\beta|^2$. Nevertheless, even if we cannot know the exact state of a qubit; qubit states can be manipulated and transformed in ways that lead to observation outcomes that depend distinctively on the different properties of the state. Another way to express the observation of a qubit is to say that we measure the qubit. We have defined the qubit as an abstract mathematical object, however, qubits are real. Similar to bits, they can be realized with many physical phenomena; for example, the two different polarization of a photon, the alignment of a nuclear spin in a uniform magnetic field, and as two states of an electron orbiting a single atom. We are not going to discuss the physical interpretation of qubit nor the physical implementation of quantum computers.

Another useful way to represent a qubit is through the **Bloch sphere**. As explained in [Gle05], we can express a quantum state in the following way:

$$|\psi\rangle = \cos\left(\frac{\theta}{2}\right)|0\rangle + e^{i\phi}\sin\left(\frac{\theta}{2}\right)|1\rangle\,.$$

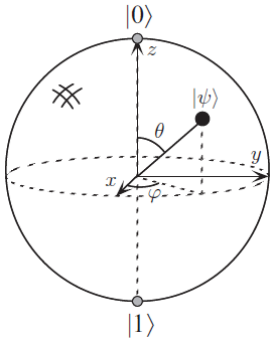


Figure 6.1: Bloch sphere representation of a qubit

The real numbers θ and ϕ define a point on the unit three-dimensional sphere. We'll see that many operations on qubits can be understood as rotations of this sphere. Of course, a qubit is not enough to do useful computation so we need to introduce systems of multiple qubits. With n qubits, there are 2^n computatioanl basis states denoted $|0\cdots0\rangle, |0\cdots1\rangle, \cdots, |1\cdots1\rangle$. Each computational basis state corresponds to a basis vector of $\mathbb{C}^{2^n}$ given by:

$$\left|a_{(2),n}\right\rangle = \begin{pmatrix} 0 \\ \vdots \\ 1 \\ \vdots \\ 0 \end{pmatrix},$$

where $a_{(2),n}$ is the n-bit representation of the integer $0 \leq a \leq 2^n - 1$ in binary base; and $\left|a_{(2),n}\right\rangle$ is a complex vector in $\mathbb{C}^2$ with all entries 0, except at position a where there is 1 (counting from 0). Morover, a n-qubit system can also be in superposition specified by 2^n amplitudes,

$$\alpha_0 |0\cdots0\rangle + \cdots \alpha_{2^n-1} |1\cdots1\rangle,$$

where $\alpha_i \in \mathbb{C}$ for all $0 \leq i \leq 2^n - 1$, such that $|\alpha_0|^2 + \cdots + |\alpha_{2^n-1}|^2 = 1$.

The mathematical expression for n-qubit systems is to express basis states as a tensor product of basic states of the components. The tensor product of two one qubit states is given by the Kronecker product. We say that a set of qubits is in *product state* if its state can be expressed as the tensor product of the states of its components. For example, the state $|01\rangle = (0, 1, 0, 0)$ is the Kronecker product of $|0\rangle |1\rangle$. If a set of qubits is not in product state, it is in *entangled state*.

6.3 Quantum gates

Changes occurring to a quantum state can be described using the language of quantum computation. A quantum computer is built from a *quantum circuit* containing wires and elementary *quantum gates* to carry around and manipulate quantum information. Quantum gates need to act linearly and preserve the norm. Thus quantum gates can be described using unitary

matrices.

This result implies that every quantum gate is reversible, which does not hold for classical gates. For example, analogous for the classical NOT gate. It takes the state $|0\rangle$ to the state $|1\rangle$, and vice versa. On a state in superposition, it acts *linearly*.

$$X \equiv \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix},$$
$$X \begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \begin{bmatrix} \beta \\ \alpha \end{bmatrix}.$$

The NOT gates belong to the family of Pauli gates:

$$\sigma_1 = \sigma_x = X \equiv \mathbf{P}_1 \equiv \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix},$$
$$\sigma_2 = \sigma_y = Y \equiv \mathbf{P}_2 \equiv \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix},$$
$$\sigma_3 = \sigma_z = Z \equiv \mathbf{P}_3 \equiv \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}.$$

Three other quantum gates that will play a large part in what follows, the Hadamard gate H, phase gate S, and $n/8$ gate T.

$$H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}, \quad S = \begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix}, \quad T = \begin{bmatrix} 1 & 0 \\ 0 & e^{\frac{i\pi}{4}} \end{bmatrix}.$$

The Pauli matrices give rise to three useful classes of unitary matrices when they are exponentiated, the rotation operators about the 'x', 'y', and 'z' axes.

$$R_x(\theta) \equiv e^{-\theta X/2} = \cos\big(\theta/2\big) I - i \sin\big(\theta/2\big) X = \begin{bmatrix} \cos\big(\theta/2\big) & -i\sin\big(\theta/2\big) \\ -i\sin\big(\theta/2\big) & \cos\big(\theta/2\big) \end{bmatrix}, \tag{6.1}$$
$$R_y(\theta) \equiv e^{-\theta Y/2} = \cos\big(\theta/2\big) I - i \sin\big(\theta/2\big) Y = \begin{bmatrix} \cos\big(\theta/2\big) & -\sin\big(\theta/2\big) \\ \sin\big(\theta/2\big) & \cos\big(\theta/2\big) \end{bmatrix}, \tag{6.2}$$
$$R_z(\theta) \equiv e^{-\theta Z/2} = \cos\big(\theta/2\big) I - i \sin\big(\theta/2\big) Z = \begin{bmatrix} e^{-i\theta/2} & 0 \\ 0 & e^{i\theta/2} \end{bmatrix}. \tag{6.3}$$

These rotation matrices allow us to construct any unitary operator.

Theorem 6.1 (Z-Y decomposition for a single qubit, [MAN10])**.** *Suppose U is a unitary operation on a single qubit. Then, there exists real numbers α, β, γ and δ such that*

$$U = e^{i\alpha} R_z(\beta) R_y(\gamma) R_z(\delta).$$

In fact, this results works for any non-parallel real unit vectors in three dimensions $\hat{m}$ and $\hat{n}$:

$$U = e^{i\alpha} R_{\hat{n}}(\beta) R_{\hat{m}}(\gamma) R_{\hat{n}}(\delta).$$

In the last equation, we have that if $\hat{n} = (n_x, n_y, n_z)$ then

$$R_{\hat{n}}(\theta) \equiv e^{-i\theta \hat{n}\cdot\vec{\sigma}/2} = \cos\big(\theta/2\big) I - i \sin\big(\theta/2\big)(n_x X + n_y Y + n_z Z),$$

where $\vec{\sigma}$ denotes the three component vector (X, Y, Z) of Pauli Matrices.

The utility of the last theorem lies in the following corollary, which is the key to the construction of controlled multi-qubit unitary operations.

Corollary 6.1. *Suppose U is a unitary gate on a single qubit. Then, there exists unitary operators A, B, C on a single qubit such that $ABC = I$ and $U = e^{i\alpha} AXBXC$, where α is some overall phase factor.*

The prototypical controlled operation is the controlled-NOT. the CNOT gate has two input qubits, known as the *control* qubit and the *target* qubit, respectively. In terms of the computational basis, the action of the CNOT is given by $|c\rangle\,|t\rangle \to |c\rangle\,|t \oplus c\rangle$; that is, if the control qubit is set to $|1\rangle$ then the target qubit is flipped, otherwise the target qubit is left alone. Thus, in the computational basis $|\text{control, target}\rangle$ the matrix representation of the CNOT is

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

For controlled arbitrary gates, suppose U is any unitary matrix acting on some number n of qubits, so U can be regarded as a quantum gate on those qubits. Then we can define a **controlled-U gate**. Such a gate has a single control qubit, indicated by the line with the black dot, and n target qubits, indicated by the boxed U. If the control qubit is set to 0 then nothing happens to the target qubits. If the control qubit is set to 1 then the gate U is applied to the target qubits. If there is just one target qubit t, the operation of this gate is $|c\rangle\,|t\rangle \to |c\rangle\,U^c\,|t\rangle$.

6.4 Quantum Algorithms

The power of quantum computing comes from the power of quantum algorithms. First of all, we have that any classical circuit can be replaced by an equivalent circuit containing only reversible elements. Any classical reversible circuit has an equivalent quantum circuit, thus quantum computers are capable of, at least, do the same things that classical computers do. However quantum computers can solve efficiently, at least in theory, problems that classical computers cannot solve in a polynomial way. This is because of *quantum parallelism.*

Quantum parallelism is a fundamental feature of the most powerful quantum algorithms. It allows quantum computers to evaluate a function f for many different values of x simultaneously. For example, let $f : \{0,1\} \to \{0,1\}$ be a function with a one-bit domain and

range. Consider a two-qubits quantum gate which starts in-state $|x, y\rangle$ and transforms it into $|x, y \oplus f(x)\rangle$. The qubits are called registers: the first register is called the 'data' register, and the second the 'target' register. The mapping $U_f : |x, y\rangle \to |x, y \oplus f(x)\rangle$ is unitary and thus the quantum gate is well defined. The computation of f is given by a black box, but it can be shown that given a classical circuit that for computing f there is a quantum circuit of comparable efficiency which computes the transformation U_f on a quantum computer.

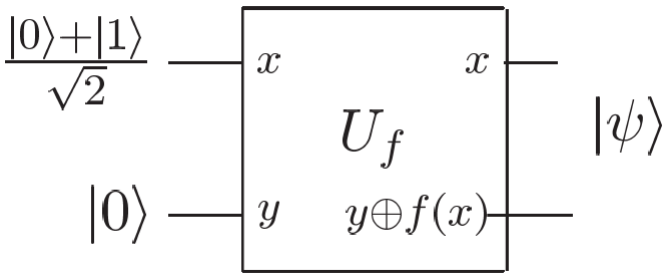


Figure 6.2: Quantum circuit for evaluating $f(0)$ and $f(1)$ simultaneously.

Figure 6.2 shows the circuit that applies U_f to an input not in the computational basis. The data register is prepared in the $|+\rangle$ state. Applying U_f gives the state:

$$\frac{|0, f(0)\rangle + |1, f(1)\rangle}{\sqrt{2}}.$$

This state contains information about both $f(0)$ and $f(1)$; in a way we have evaluated $f(x)$ for two values of x simultaneously, this feature is what we call 'quantum parallelism'. This is not the same as parallel computing, where multiple circuits each built to compute $f(x)$ are executed simultaneously, here a single $f(x)$ circuit is employed to evaluate the function for multiple values of x simultaneously, by using the ability to be in a superposition of different states that quantum computers have. We can generalize this result to function on an arbitrary number of bits, by using the *Hadamard transform*. This transformation is just n Hadamard gates acting in parallel on n qubits. In summary, quantum parallelism enables all possible values of the function f to be evaluated simultaneously. Nevertheless, this parallelism is not immediately useful because when we measure the state $\sum_x |x, f(x)\rangle$ we will only obtain only $f(x)$ for a single value of x. Thus, quantum algorithms require the ability to extract information about more than one value of $f(x)$ from superposition states. To see how this may be done, let us describe a quantum algorithm to solve the problem given in Example 2.1.

The Deutsch-Jozsa algorithm

Let us recall the statement of the problem we want to solve: *Bob has a boolean function of n binary digits f ($f : \{0,1\}^n \to \{0,1\}$), that is either constant or balanced (a function that outputs 0 on exactly half inputs and 1 on the other half). Alice wants to know what kind of function Bob has.* We saw in chapter 2 that Alice may only send Bob one value of x in each letter. We showed that Alice will need at worst $2^n/2 + 1$ queries to Bob. If Bob and Alice were able to exchange qubits, and Bob computes f using a unitary transformation U_f, then Alice

could achieve her goal in just one sending, using the following algorithm.

Algorithm 5 Deutsch-Jozsa

INPUT: A black box U_f performs the transformation $|x\rangle|y\rangle \rightarrow |x\rangle|y \oplus f(x)\rangle$, for $x \in \{0, ..., 2^n - 1\}$ and $f(x) \in \{0, 1\}$. f is either constant or balanced.
OUTPUT: 0 if, and only if, f is constant.
RUNTIME: One evaluation of U_f. Always succeeds.
PROCEDURE:

1: $|0\rangle^{\otimes n}|1\rangle$ ▷ initialize state
2: $\rightarrow \frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} |x\rangle \left[\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right]$ ▷ create superposition using Hadamard gates
3: $\rightarrow \sum_x (-1)^{f(x)} |x\rangle \left[\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right]$ ▷ calculate function f using U_f
4: $\rightarrow \sum_z \sum_x \frac{(-1)^{x\cdot z+f(x)}|z\rangle}{\sqrt{2^n}} \left[\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right]$ ▷ perform Hadamard transform
5: $\rightarrow z$ ▷ measure to obtain final output z

Alice has n register qubits to store her query in, and a single qubit register which she will give to Bob, to store the answer in. She first prepares her query and answer register in a superposition state. She sends these states to Bob and he will evaluate $f(x)$ using quantum parallelism and save the result in the answer register. Alice then interferes with states in the superposition using a Hadamard transform on the query register and makes a suitable measurement to determine the nature of f. The quantum circuit that performs this algorithm is given in Figure 6.3.

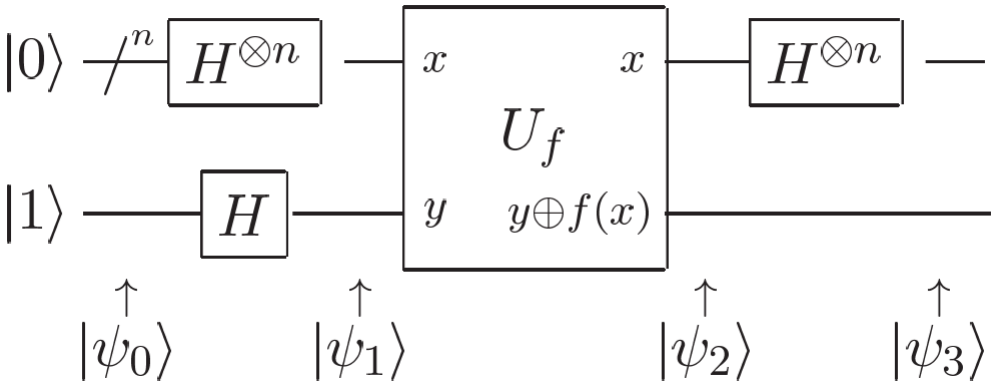


Figure 6.3: Quantum circuit implementing Deutsch-Jozsa algorithm.

Let us explain how the states change in every step of this circuit. We start with the input state

$$|\psi_0\rangle = |0\rangle^{\otimes n}|1\rangle \tag{6.4}$$

The system of n qubits $|0\rangle^{\otimes n}$ is the query register and the state $|1\rangle$ is the answer register. We apply a Hadamard transform to the query register and a Hadamard gate to the answer register,

obtaining

$$|\psi_1\rangle = \sum_{x\{0,1\}^n} \frac{|x\rangle}{\sqrt{2^n}} \left[\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right]. \tag{6.5}$$

The query register is now a superposition of all values in the domain of f, and the answer register is in an evenly weighted superposition of 0 and 1. Now, Bob, uses its black box to evaluate the function f, giving

$$|\psi_2\rangle = \sum_{x} \frac{(-1)^{f(x)}\,|x\rangle}{\sqrt{2^n}} \left[\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right]. \tag{6.6}$$

Alice now has a set of qubits in which the result of Bob's function evaluation is stored in the amplitude of the qubit superposition state. She then interferes terms in the superposition using a Hadamard transform on the query register to obtain,

$$|\psi_3\rangle = \sum_{z}\sum_{x} \frac{(-1)^{x\cdot z+f(x)}\,|z\rangle}{\sqrt{2^n}} \left[\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right]. \tag{6.7}$$

Alice now observes the query register. The amplitude for the state $|0\rangle^{\otimes n}$ is $\sum_x (-1)^{f(x)}/2^n$. This amplitude changes depending on whether f is constant or balanced. If f is constant the amplitude for $|0\rangle^{\otimes n}$ is +1 or -1, depending on the constant value $f(x)$ takes. Since the state $|\psi_3\rangle$ is unitary all other amplitudes must be zero, and a measurement of this state will yield 0s for all qubits in the query register. On the other case, if f is balanced then the positive and negative contributions to the amplitude for $|0\rangle^{\otimes n}$ cancel, leaving an amplitude of zero, and measurements must yield a 1 on at least one qubit in the query register.
This problem shows the power of quantum algorithms. In fact, we have shown that we can solve the problem with one evaluation of the function f compared to the classical requirement for $2^{n-1}+1$ evaluations. The most important thing to remember here is how quantum algorithms work. Since we cannot access directly the states in superposition we need to try to pass the information in the states to its amplitudes; because amplitudes can be observed and studied. The most famous quantum algorithms that use this method are based upon quantum versions of the Fourier transform; in fact, the quantum algorithm to solve the Elliptic Curve Discrete Logarithm Problem belongs to this family.

6.5 The quantum Fourier transform and Shor's Algorithm

In this section, we develop the *quantum Fourier transform*, which is the key ingredient for the quantum discrete logarithm algorithm. The quantum Fourier transform is an efficient quantum algorithm for performing a Fourier transform of quantum mechanical amplitudes. The most important feature of the Quantum Fourier transform is that, it enables *phase estimation*, which is the approximation of the eigenvalues of a unitary operator. A generalization of phase estimation called *hidden subgroup problem*, is the base to an efficient quantum algorithm for the discrete logarithm problem.

The quantum Fourier transform

The quantum Fourier transform on an orthonormal basis $|0\rangle, ..., |N-1\rangle$ is defined to be a linear operator with the following action on the basis states,

$$|j\rangle \to \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} e^{2\pi ijk/N} |k\rangle . \tag{6.8}$$

The action on a state in superposition is

$$\sum_{j=0}^{N-1} x_j |j\rangle \to \sum_{k=0}^{N-1} y_k |k\rangle , \tag{6.9}$$

where the amplitudes y_k are the discrete Fourier transform of the amplitudes x_j. To implement this transformation on quantum circuits, we need to show that this transformation is unitary, this is direct by letting j and m be quantum states from an orthonormal basis, and computing the inner product of its transformations:

$$\begin{aligned}
\langle m|j\rangle \to & \left(\frac{1}{\sqrt{N}} \sum_{n=0}^{N-1} e^{2\pi imn/N} \langle n| \right) \left(\frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} e^{2\pi ijk/N} |k\rangle \right) \\
&= \frac{1}{N} \sum_{n,k=0}^{N-1} e^{-2\pi imn/N} e^{2\pi ijk/N} \langle n|k\rangle \\
&= \frac{1}{N} \sum_{n,k=0}^{N-1} e^{2\pi i(jk-mn)/N} \delta_{n,k} \\
&= \frac{1}{N} \sum_{k=0}^{N-1} e^{2\pi i(jk-mk)/N} \\
&= \frac{1}{N} \sum_{k=0}^{N-1} e^{2\pi i(j-m)k/N} \\
&= \delta_{j,m}.
\end{aligned}$$

Thus we can implement the Quantum Fourier Transform as a valid quantum gate. Before doing that, let us introduce some notation. For the remainder of this chapter, let $N = 2^n$, where n is some integer, and the basis $|0\rangle, ..., |2^n - 1\rangle$ is the computational basis for an n qubit quantum computer. The state $|j\rangle$ has a binary representation $j = j_1 j_2 ... j_n$ and we adopt the notation $0, j_l j_{l+1} ... |jm\rangle$ to represent the *binary fraction* $j_l/2 + j_{l+1}/4 + ... + j_m/2^{m-l+1}$. This notation let us give a product representation to the quantum transform:

$$|j_1, ..., j_n\rangle \to \frac{(|0\rangle + e^{2\pi i 0, j_n} |1\rangle)(|0\rangle + e^{2\pi i 0, j_{n-1} j_n} |1\rangle) \cdots (|0\rangle + e^{2\pi i 0, j_1 j_2 \cdots j_n} |1\rangle)}{2^{n/2}}. \tag{6.10}$$

Notice that, for a quantum circuit to apply the quantum transform we need to apply controlled notations over each factor state. Let us denote R_k to the unitary transformation given by

$$R_k \equiv \begin{pmatrix} 1 & 0 \\ 0 & e^{2\pi i/2^k} \end{pmatrix}. \tag{6.11}$$

Using controlled-R_i and Hadamard gates we can create aN efficient circuit for the quantum Fourier Transform (Figure 6.4)

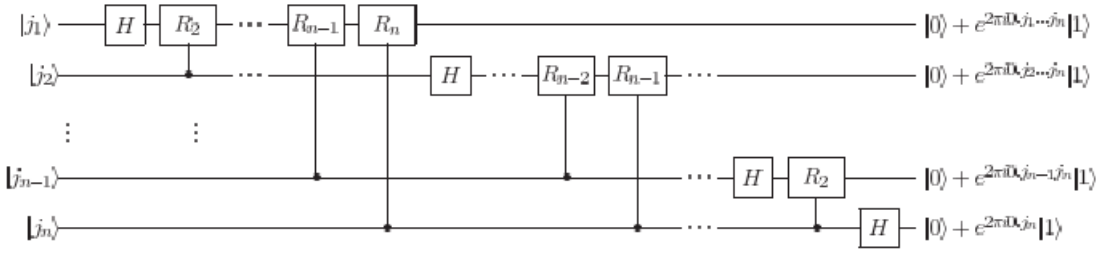


Figure 6.4: Quantum circuit that applies the quantum Fourier transform over the state $|j\rangle$. Notice that at the end the output of each bit component of j is reversed, so we need to apply swap gates at the end to reverse the order of the qubits.

Some top classical algorithm for computing the discrete Fourier transform, such as the *Fast Fourier Transform (FFT)*, need to use $\Theta(n2^n)$ gates for 2^n element. In contrast, this quantum algorithm needs only $\Theta'(n^2)$ gates for performing the quantum Fourier transform on the same number of elements. In summary, there is an exponential saving when working with quantum algorithms rather than classical ones. Unfortunately, we cannot use the quantum Fourier transform to speed up the computation of the classic Fourier transform. This is again because measurement operation destroys quantum information; there is no way of determining the Fourier transformed amplitudes of the original state. Also, a general method to efficiently prepare the original state to be Fourier transformed is not known. The algorithms that use this computation of Quantum Fourier Transform make a more subtle application of it.

Phase estimation

A usual computational problem is determining the eigenvalue of a unitary operator U. The eigenvalues of unitary operators have the form $e^{2\pi i\phi}$, where $\phi \in \mathbf{R}$. The eigenvector associated with this value is denoted $|u\rangle$. Hence, phase estimation consists in finding ϕ given U and $|u\rangle$. In practice, we do not need the exact value of ϕ, and our goal is just to estimate it with arbitrary precision. To perform the estimation we must assume that we have available black boxes (oracles) capable of preparing the state $|u\rangle$ and performing the controlled U^{2^j} operation, for suitable natural numbers j.
The quantum phase estimation procedure uses two registers. The first register contains t qubits in the state $|0\rangle$. The election on t depends on:

1. how good our estimation must be;

2. the probability of success for the phase estimation procedure.

The second register begins in the state $|u\rangle$ and contains as many qubits as necessary to store $|u\rangle$. Phase estimation is performed in three stages:

1. We apply the following circuit

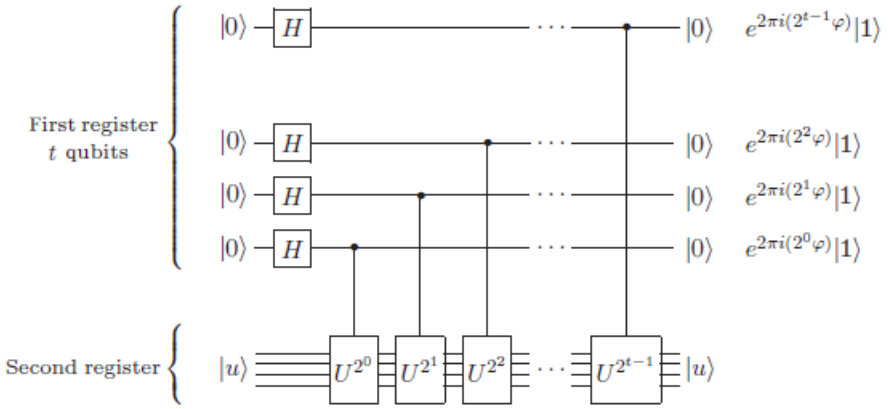


Figure 6.5: First stage of the phase estimation procedure. We omit normalization factors of $1/\sqrt{2}$.

This circuit[1] begins by applying a Hadamard transform to the first register, followed by the application of controlled-U operations on the second register, with U raised to successive powers of two.
The final state of the first register is

$$\frac{1}{2^{t/2}} \sum_{k=0}^{2^t-1} e^{2\pi i \phi k} |k\rangle .$$

The effect of the sequence of controlled-U operations is to take the state $|j\rangle |u\rangle$ to $|j\rangle U^j |u\rangle$.

2. The second stage of phase estimation is to apply the inverse quantum Fourier transform on the first register.

3. The third stage of phase estimation is to read out the state of the first register by doing measurements on the computational basis.

The output of the measurement is an approxiomation to ϕ accurate to $t - \left\lceil \log(2 + 1/2\epsilon) \right\rceil$ bits, with probability $1 - \epsilon$.

To see that this algorithm works, suppose ϕ may be expressed exactly in t bits, as $\phi = 0.\phi_1 ... \phi_t$. Then the state resulting from the first stage of phase estimation may be rewritten

$$\frac{1}{2^{t/2}}(|0\rangle + e^{2\pi i 0.\phi_t} |1\rangle)(|0\rangle + e^{2\pi i 0.\phi_{t-1}\phi_t} |1\rangle) \cdots (|0\rangle + e^{2\pi i 0.\phi_1\phi_2\cdots\phi_t} |1\rangle).$$

Notice that this is similar to what we obtain when applying QFT to a quantum state, thus the second stage of phase estimation is to apply the $IQFT$. But comparing the last state with the product form for the Fourier transform, we see that the output state form the second stage

[1] Recall that the sum of all amplitudes in a quantum state must be equal to 1, thus, the powers of $1/\sqrt{2}$ are the normalization constantn to make sure the state/eigenvector is a unit vector.

is the product state $|\phi_1 \cdots \phi_t\rangle$. A measurement in the computational basis, therefore, gives us ϕ exactly.

In summary, the phase estimation procedure let us estimate the phase ϕ of an eigenvalue of a unitary operator U, given the corresponding eigenvector $|u\rangle$. The key feature of this algorithm is the ability of the *inverse Fourier transform* to perform the transformation

$$\frac{1}{2^{t/2}} \sum_{j=0}^{2^t-1} e^{2\pi i \phi j} |j\rangle |u\rangle \rightarrow \left|\tilde{\phi}\right\rangle |u\rangle ,$$

where $\left|\tilde{\phi}\right\rangle$ denotes a state which is a good estimator for ϕ when measured.

Algorithm 6 Quantum phase estimation

INPUT: A black box which performs a controlled-U^j operation for integers j, (2) an eigenstate $|u\rangle$ of U with eigenvalue $e^{2\pi i \phi_u}$, and (3) $t = n + \lceil log(2 + 1/2\epsilon) \rceil$ qubits initialized to $|0\rangle$. ($\epsilon > 0$ is arbitrary and is the tolerance of the algorithm.)
OUTPUT: An n-bit approximation $\tilde{\phi}_u$ to ϕ_u.
RUNTIME: $\mathcal{O}'(t^2)$ operations and one call to controlled-U^j black box. Succeds with probabily at least 1ϵ.
PROCEDURE:

1: $|0\rangle |u\rangle$ ▷ initial state
2: $\rightarrow \frac{1}{\sqrt{2^t}} \sum_{j=0}^{2^t-1} |j\rangle |u\rangle$ ▷ create superposition
3: $\rightarrow \frac{1}{\sqrt{2^t}} \sum_{j=0}^{2^t-1} |j\rangle U^j |u\rangle = \frac{1}{\sqrt{2^t}} \sum_{j=0}^{2^t-1} e^{2\pi i j \phi_u} |j\rangle |u\rangle$ ▷ apply black box
4: $\rightarrow \left|\tilde{\phi}_u\right\rangle |u\rangle$ ▷ apply inverse Fourier transform
5: $\rightarrow \tilde{\phi}_u$ ▷ measure first register

Order finding

An application of the phase estimation procedure is the *Order finding algorithm*. This is the base case for a big family of problems where the Discrete Logarithm problem lives. It is also worth noticing that order finding implies the factoring problem and so efficient algorithms for order finding and factoring are a threat to RSA public-key cryptosystems since they can be used to break those.

For positive integers x and N, $x < N$, such that $gcd(x, N) = 1$, the order of x modulo N is defined to be the least positive integer r such that $x^r = 1 \pmod N$. The order finding problem is to determine the order for some specified x and N. In classical computation, this problem is believed to be hard. In the quantum world, the algorithm form order-finding is just the phase estimation algorithm applied to the unitary operator

$$U |y\rangle \equiv \left|xy \pmod N\right\rangle , \quad L = \lceil \log N \rceil ;$$

with $y \in \{0,1\}^L$. The eigenstates for this operator have the form:

$$|u_s\rangle \equiv \frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} \exp\left[\frac{-2\pi i s k}{r}\right] \left|x^k \mod N\right\rangle,$$

for integers $0 \leq s \leq r-1$. The phase of the eigenstate associated to $|u_s\rangle$ is s/r Thus, phase estimation procedure allows us to obtain, with high accuracy, the term s/r and from this we can obtain r using classical algorithms such as *continued fraction expansion.*

As we wrote earlier in order to apply phase estimation we need to be able to compute the operator U and to prepare one of its eigenstates. In the case of order finding, the former issue can be solved using modular exponentiation (a classical algorithm), with which we can implement the entire sequence of controlled-U^{2^j} operations applied by the phase estimation procedure using $\mathcal{O}(L^2)$ gates. For the latter issue, instead, we rely on the fact that

$$\frac{1}{\sqrt{r}} \sum_{s=0}^{r-1} |u_s\rangle = |1\rangle.$$

Hence, if we use $t = 2L + 1 + \left\lceil 2 + \frac{1}{2\epsilon} \right\rceil$ qubits in the first register, and prepare the second register on the state $|1\rangle$, it follows that for each s in the range 0 through $r-1$, we will obtain an estimate of the phase $\phi \approx s/r$ accurate to $2L+1$ bits, with probability at least $(1-\epsilon)/r$.

About the cost of this algorithm, we can say that the Hadamard transform requires $\mathcal{O}'(L)$ gates, and the inverse Fourier transform requires $\mathcal{O}'(L^2)$ gates. The major cost in the proper quantum circuit actually comes from the modular exponentiation which uses $\mathcal{O}'(L^3)$ gates. The continued fraction algorithm adds $\mathcal{O}'(L^3)$ gates to obtain r. Thus, the total cost of this algorithm is $\mathcal{O}'(L^3)$.

Algorithm 7 Quantum order-finding

INPUT: A black box $U_{x,N}$ which performs the transformation $|j\rangle|k\rangle \to |j\rangle\left|x^j k \mod N\right\rangle$, for x co-prime to the $L-$bit number n, (2) $t = 2L + 1 + \left\lceil \log\left(2 + 1/2\epsilon\right) \right\rceil$ qubits initialized to $|0\rangle$, and (3) L qubits initializaed to the state $|1\rangle$.
OUTPUT: The order r, of x modulo N.
RUNTIME: $\mathcal{O}'(L^3)$ operations. Succeds with probabily $\mathcal{O}(1)$.
PROCEDURE:

1: $|0\rangle|1\rangle$ ▷ initial state
2: $\to \frac{1}{\sqrt{2^t}} \sum_{j=0}^{2^t-1} |j\rangle|1\rangle$ ▷ create superposition
3: $\to \frac{1}{\sqrt{2^t}} \sum_{j=0}^{2^t-1} |j\rangle\left|x^j \mod N\right\rangle \approx \frac{1}{\sqrt{r2^t}} \sum_{s=0}^{r-1}\sum_{j=0}^{2^t-1} e^{2\pi i s j/r}|j\rangle|u_s\rangle$ ▷ apply $U_{x,N}$
4: $\to \frac{1}{\sqrt{r}} \sum_{s=0}^{r-1} \left|\tilde{s/r}\right\rangle |u_s\rangle$ ▷ apply inverse Fourier transform to first register
5: $\to \tilde{s/r}$ ▷ measure first register
6: $\to r$ ▷ apply continued fraction algorithm

Period-finding

In the next section, we will see that the Discrete Logarithm problem is just the problem of period finding but in two variables, hence it is a good thing to study this problem and how it is related to the order finding problem. Suppose f is a periodic function with range $\{0,1\}$ and unknown period $0 < r < 2^L$, where $x, r \in \{0,1,2,...\}$. Assume you have a quantum black box U which performs the unitary transform $U|x\rangle|y\rangle \to |x\rangle|y \oplus f(x)\rangle$ (where $\oplus$ denotes addition modulo 2), then there is a quantum algorithm which solved the problem of finding r using one query, and $\mathcal{O}(L^2)$ other operations.

Algorithm 8 Quantum period-finding

INPUT: A black box U which performs the transformation $U|x\rangle|y\rangle = |x\rangle|y \oplus f(x)\rangle$, (2) a state to store the function evaluation, initialized to $|0\rangle$, and (3) $t = \mathcal{O}(L + \log(1/\epsilon))$ qubits initialized to $|0\rangle$.
OUTPUT: The period r of f.
RUNTIME: One use of U, and $\mathcal{O}'(L^2)$ operations. Succeds with probabily $\mathcal{O}(1)$.
PROCEDURE:

1: $|0\rangle|0\rangle$ ▷ initial state

2: $\to \frac{1}{\sqrt{2^t}} \sum_{x=0}^{2^t-1} |x\rangle|0\rangle$ ▷ create superposition

3: $\to \frac{1}{\sqrt{2^t}} \sum_{x=0}^{2^t-1} |x\rangle|f(x)\rangle \approx \frac{1}{\sqrt{r2^t}} \sum_{l=0}^{r-1} \sum_{x=0}^{2^t-1} e^{2\pi i l x/r} |x\rangle \left|\hat{f}(l)\right\rangle$ ▷ apply U

4: $\to \frac{1}{\sqrt{r}} \sum_{l=0}^{r-1} \left|\tilde{l/r}\right\rangle \left|\hat{f}(l)\right\rangle$ ▷ apply inverse Fourier transform to first register

5: $\to \tilde{l/r}$ ▷ measure first register

6: $\to r$ ▷ apply continued fraction algorithm

Notice that this algorithm is almost the same as the quantum algorithm for order-finding, the key step is step 3, in which we introduce the Fourier transform of $f(x)$.

$$\left|\hat{f}(l)\right\rangle \equiv \frac{1}{\sqrt{r}} \sum_{x=0}^{r-1} e^{-2\pi i l x/r} |f(x)\rangle \tag{6.12}$$

In step 3, we use the approximation sign because 2^t may not be an integer multiple of r in general. This is not a problem since this issue is taken into account by the phase estimation bounds. To the state in step 3, we apply the inverse Fourier transform and we obtain an estimate of the phase l/r, where l is chosen randomly, in step 4. Finally, r can be obtained using a continued fraction algorithm.
This algorithm works because we exploit a property of the quantum Fourier transform called *shift-invariance*. Namely, suppose we are in N dimensions and given a state of the form

$$|\phi\rangle = \sum_{n=0}^{N/r-1} c\,|l + nr\rangle\,,$$

where $|c| = \sqrt{r/N}$. We say that is state is a *periodic* state with *period* r and *offset* l. Applying the Fourier transform to this period state, we obtain

$$\left|\tilde{\phi}\right\rangle = \sum_{m=0}^{r-1} \alpha_m \left|mN/r\right\rangle,$$

where $|\alpha_m| = \sqrt{1/r}$ for all m. This state is also periodic, in the following sense: the amplitudes α_m can have nontrivial phases, but they are all of equal weight, and the offset is zero. Thus, if we measure this state on the standard computational basis, we are guaranteed to get a multiple of N/r.

Shor's quantum algorithm for solving the ECDLP

The principal references for this section are [Sho99], [PZ03] and [RNSL17]. First, let us justify some notations and assumptions. We are going to denote quantum states as $|n\rangle$ where n is any positive number or 0, computationally speaking this should be understood in its bit representation, for example, $|4\rangle = |100\rangle$, this notation is for educational purposes and will be useful to understand the algorithm in a better way. We assume that the order of the base point used in the ECDLP is prime and we know it. This is true for the cases standardized for cryptographic use. Finally, we assume that we can implement the quantum Fourier transform of arbitrary order $\gamma \in \mathbb{N}$, this is, over an arbitrary number of qubits. Later we will give a real circuit with a QFT of order 2^n because is easier to implement and under certain circumstances, the result of the algorithm is going to be good.

We are given an instance of the ECDLP as described in Section 5.3.1. Let $P, Q \in E(\mathbb{F}_q)$ such that Q is an element of the subgroup generated by P; our goal is to find the integer $m \in \{0, ..., r-1\}$, where r is the order of P, such that $Q = mP$. Shor's algorithm proceeds as follows.

1. Two registers initialized in the $|0\rangle$ state. A third register called the "accumulator" register is initialized with the neutral element $\mathcal{O}$.

2. A Hadamard transform is applied to each qubit in the first two registers resulting in the sate
$$\frac{1}{r}\sum_{k,l=0}^{r-1} |k,l\rangle .$$

3. Conditioned on the content of the register holding the label k or l, we add the corresponding multiple of P and Q, respectively:
$$\frac{1}{r}\sum_{k,l=0}^{r-1} |k,l\rangle \mapsto \frac{1}{r}\sum_{k,l=0}^{r-1} |k,l\rangle \, |kP + lQ\rangle .$$

   Imagine that we measure the last register (which is not necessary). Then we will obtain a random element αP of the group generated by P, where α is between 0 and r-1. We will then find the first two registers in a superposition of all k, l with
$$kP + lQ = kP + l(mP) = \alpha P.$$

Because the order of P is r, this is equivalent to

$$k + lm \equiv \alpha \pmod{r}.$$

Thus for each l there is exactly one solution, and so the state of the first two registers is:

$$\frac{1}{\sqrt{r}} \sum_{l=0}^{r-1} |\alpha - ml, l\rangle$$

4. The third register is discarded and a quantum Fourier transform QFT_r is computed on each of the two registers. Hence, the state $|k\rangle$ is send to $\left|\tilde{k}\right\rangle$ and the state $|l\rangle$ is send to $\left|\tilde{l}\right\rangle$ with probability amplitude $\frac{1}{r}\exp\Big(2\pi i(k\tilde{k} + l\tilde{l})/r\Big)$. This is, we take the state $|k, l\rangle$ to the state

$$\frac{1}{\sqrt{r}r} \sum_{\tilde{k},\tilde{l}=0}^{r-1} \sum_{l=0}^{r-1} \exp\Big(2\pi i((\alpha - ml)\tilde{k} + l\tilde{l})/r\Big) \left|\tilde{k}, \tilde{l}\right\rangle.$$

The sum over l is easy to calculate. It gives $r\exp\Big(2\pi i\alpha\tilde{k}/r\Big)$ if $\tilde{l} \equiv m\tilde{k} \pmod{r}$ and vanishes otherwise. Thus we get:

$$\frac{1}{\sqrt{r}} \sum_{\tilde{k}=0}^{r-1} \exp\Big(2\pi i\alpha\tilde{k}/r\Big) \left|\tilde{k}, \tilde{l} = m\tilde{k} \bmod r\right\rangle.$$

5. We measure the state of the first two registers. We see now that the probability of measuring a basis state is independent of α, thus it does not matter which α we measured above. By measuring we obtain a pair $\tilde{k}, \tilde{l}$ from which we can calculate $m = \tilde{l}(\tilde{k})^{-1} \mod r$ as long as $\tilde{k} \neq 0$. (Here is why we assume that r is prime, if not we would require $\gcd(\tilde{k}, r) = 1$.)

Algorithm 9 Shor's algorithm for DLPEC

INPUT: A point $Q = mP \in E(\mathbb{F}_q)$.
OUTPUT: The natural integer m.
PROCEDURE:

1: $|0\rangle\,|0\rangle\,|\mathcal{O}\rangle$ ▷ initial state

2: $\rightarrow \dfrac{1}{r}\displaystyle\sum_{k,l=0}^{r-1} |k, l\rangle\,|\mathcal{O}\rangle$

$= \dfrac{1}{\sqrt{r}}\displaystyle\sum_{l=0}^{r-1} |\alpha - ml, l\rangle\left|(\alpha - ml)P + lQ\right\rangle$ ▷ create superposition in the first two registers

3: $\rightarrow \dfrac{1}{\sqrt{r}r}\displaystyle\sum_{\tilde{k},\tilde{l}=0}^{r-1}\sum_{l=0}^{r-1} \exp\Big(2\pi i((\alpha - ml)\tilde{k} + l\tilde{l})/r\Big)\left|\tilde{k}, \tilde{l}\right\rangle$

$= \dfrac{1}{\sqrt{r}}\displaystyle\sum_{\tilde{k}=0}^{r-1} \exp\Big(2\pi i\alpha\tilde{k}/r\Big)\left|\tilde{k}, \tilde{l} = m\tilde{k} \bmod r\right\rangle$ ▷ QFT computed on each of the two registers

4: $\rightarrow \tilde{k}, \tilde{l} = m\tilde{k} \bmod r$ ▷ measure the first two registers

5: $\rightarrow m$ ▷ Compute m using k^{-1}

As stated before, in practice we replace each of the two QFT_r's with a quantum Fourier transform of order 2^n, because this is easy to implement. Proos and Zalka [PZ03] showed in detail how to implement this algorithm. They found out that a smaller quantum computer can solve the problem of the discrete logarithm of elliptic curves, more easily than the problem of integer factorization. Roettler, Naehrig, Svore, and Lauter [RNSL17] gave precise quantum resource estimates for this algorithm over prime fields. The quantum circuit that they implemented is shown in Figure 6.6. They also supported Proos and Zalka results and concluded that ECC seems like an easier target than RSA.

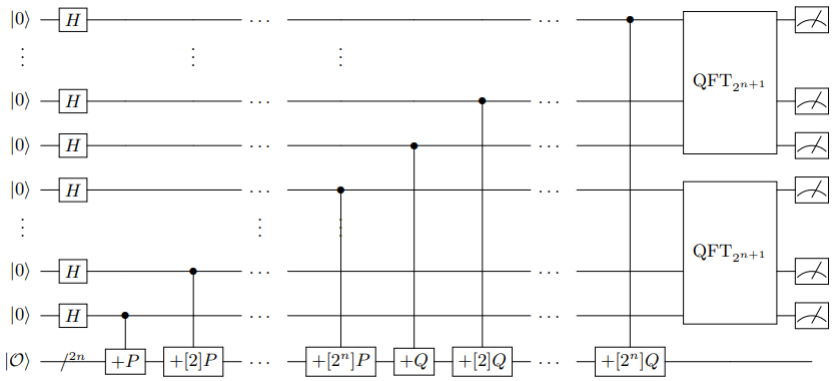


Figure 6.6: Shor's algorithm to solve the discrete logarithm problem on elliptic curves. The circuit decomposes into three parts: (i) the Hadamard transform on the left, (ii) a double scalar multiplication (implemented as a cascade of conditional point additions), and (iii) the quantum Fourier transform QFT and subsequent measurement in the computational basis at the end.

In summary, Shor's algorithm poses a serious threat to elliptic curve cryptography. While quantum computers remain mostly theoretical, if one day we can build quantum computers with a practical number of qubits, then Elliptic Curve Cryptography would become obsolete. Hopefully, elliptic curves can still be useful in other protocols that seem to be quantum-resistant.

Chapter 7

Isogeny based cryptography

In 2006, Rostovtsev and Stolbunov, [RS06], suggested a Diffie-Hellamn-Merkle key exchange protocol using elliptic curves isogenies. This protocol was later discarded as a subexponential quantum algorithm to break it was found in 2014. Nevertheless, De Feo, Jao, and Plut found in 2014, [DFJP14], that isogenies of supersingular elliptic curves can be used to create quantum-resistant cryptosystems. In this chapter, we develop the background theory of isogenies and we explain how to use isogenies of supersingular curves to create quantum-resistant cryptosystems. In this final chapter, we develop the mathematical theory behind isogenies, we show the relation with complex numbers and lattices to finish with some cryptographic protocols that are based on isogenies between elliptic curves.

7.1 Background on Isogenies

Unless stated otherwise, background material in the following section is collected from [SS03].

Definition 7.1.1. Let E_1, E_2 be elliptic curves over a field $\mathbb{K}$.

a) A **morphism** from E_1 to E_2 is a rational map which is defined at every point of E_1.
b) An **isogeny** from E_1 to E_2 is a non-constant morphism which maps $\mathcal{O}$ on E_1 to $\mathcal{O}$ on E_2.
c) A isogeny $\phi : E_1 \to E_2$ leads to an injection of function fields

$$\phi^* : \overline{\mathbb{K}}(E_2) \to \overline{\mathbb{K}}(E_1), \quad f \mapsto f \circ \phi.$$

The isogeny ϕ is called **separable** (**purely inseparable**) if, and only if, the finite extension $\mathbb{K}(E_1)|\phi^*(\mathbb{K}(E_2))$ is separable (purely inseparable).
d) For a non-constant isogeny $\phi : E_1 \to E_2$ define the **degree** of ϕ as

$$\deg(\phi) := [\mathbb{K}(E_1) : \phi^*(\mathbb{K}(E_2))],$$

the *separable degree* of ϕ as

$$\deg_s(\phi) := [\mathbb{K}(E_1) : \phi^*(\mathbb{K}(E_2))]_s,$$

and the *inseparable degree* of ϕ as

$$\deg_i(\phi) := [\mathbb{K}(E_1) : \phi^*(\mathbb{K}(E_2))]_i.$$

For the zero isogeny 0 we define $\deg(0) := 0$.

We have defined the degree of an isogeny as its degree as a rational map. Thus, for separable isogenies, to have degree l means to have a kernel of size l. Moreover, every separable isogeny of degree greater than 1 can be factored into a composition of isogenies of prime degree over $\overline{\mathbb{F}}_q$, [Cou06].

Definition 7.1.2. Let $E|\mathbb{K}$ be an elliptic curve over a field $\mathbb{K}$. The set of all isogenies from E to E forms the **ring of endomorphisms** $\mathrm{End}(E)$ of E.

Proposition 7.1. *Let $\phi : E_1 \to E_2$ be an isogeny of degree d. There exists a unique isogeny $\hat{\phi} : E_2 \to E_1$ with*

$$\hat{\phi} \circ \phi = d \quad (= \textit{multiplication with } d \textit{ on } E_1),$$
$$\phi \circ \hat{\phi} = d \quad (= \textit{multiplication with } d \textit{ on } E_2).$$

The isogeny $\hat{\phi}$ is called he dual isogeny to ϕ. One has

$$deg(\hat{\phi}) = deg(\phi).$$

Proposition 7.2. *Let $\phi : E_1 \to E_2$ be a non-constant isogeny. Then, for every $\mathcal{Q} \in E_2$,*

$$|\phi^{-1}(\mathcal{Q})| = deg_s(\phi),$$

where $|\cdot|$ denotes the size of the set. In particular

$$|ker(\phi)| = deg_s(\phi).$$

If ϕ is separable then

$$|ker(\phi)| = deg(\phi).$$

An important family of endomorphisms is the set of multiplication maps.

Proposition 7.3. *Let $E|\mathbb{K}$ be an elliptic curve and $m \in \mathbb{Z}$. The multiplication map corresponding to m is defined as*

$$[m] : E \to E, \quad P \mapsto mP.$$

a) The map m is an endomorphism on E. Therefore the ring of integers $\mathbb{Z}$ is a subring of the endomorphism ring End(E).

b) If gcd(m, char($\mathbb{K}$)) = 1 or if char($\mathbb{K}$) = 0 then, the endomorphism m is separable.

c) For the degree one has

$$deg(m) = m^2.$$

d) The endomorphism m is its own dual isogeny:

$$\hat{m} = m.$$

Theorem 7.1. *Let $E|\mathbb{K}$ be an elliptic curve over a field $\mathbb{K}$. Then the endomorphism ring of E is one of the following rings:*

- *End(E) = $\mathbb{Z}$,*

- *End(E) is isomorphic to an order in an imaginary quadratic field,*
- *End(E) is isomorphic to an order in a quaternion algebra.*

Recall from Theorem 4.5. that if $\mathrm{End}(E)$ is isomorphic to an order in a quaternion algebra, then E is supersingular. On the other hand, if $\mathrm{End}(E)$ is isomorphic to an order in an imaginary quadractic field, E is ordinary.

Since we work with Weierstrass equations, an isogeny between two elliptic curves is fully characterized by its action on the affine models for E_1 and E_2. If E_1/K and E_2/K are two elliptic curves in Weierstrass short form, and $\phi : E_1(\overline{K}) \to E_2(\overline{K})$ a isogeny, then, ϕ can be represented as:

$$\phi(x,y) = \left(\frac{f_1(x,y)}{g_1(x,y)}, \frac{f_2(x,y)}{g_2(x,y)}\right) : f_1, f_2, g_1, g_2 \in \overline{K}[x,y]. \tag{7.1}$$

Lemma 7.1. *For any isogeny $\phi : E_1 \to E_2$, $ker(\phi)$ is finite.*

Proof. Let us assume that E_1 and E_2 are in Weierstrass forms. From any of its affine representations like (7.1), we see that its kernel must be a subset of the intersection of the zero sets of $g_1(x,y)$ and $g_2(x,y)$ on $E_1(\overline{K})$. This has to be finite since $E_1(\overline{K})$ is an irreducible subset of $\mathbf{A}^2(\overline{K})$. □

Example 7.1. *Pointed isomorphisms are the simplest examples of isogenies. Let $E : y^2 = x^3 + ax + b$, $a, b \in K$ and $E^{(d)} : y^2 = x^3 + d^2ax + d^3b$ for $d \in \overline{K}^*$. Then E and $E^{(d)}$ are isomorphic via the pointed isomorphism*

$$\phi : E \to E^{(d)}$$
$$\phi(x,y) = (dx, d^{\frac{3}{2}}x).$$

We use pointed isomorphisms to define an equivalence relation over isogenies.
The family of m-multiplication maps defined in 7.3 has an affine representation as :

$$[m](x,y) := \left(\frac{\omega_m(x,y)}{\psi_m^2(x,y)}, \frac{\Omega_m(x,y)}{\psi_m^2(x,y)}\right), \tag{7.2}$$

where $\omega_m, \Omega_m, \psi_m$ are distinguished polynomials that can be computed recursively in m. Especial attention deserve the family ψ_m; its roots on the locus of E are the m-torsion points $E[m]$, the kernel of the map $[m]$.

Definition 7.1.3. Isogenies $\phi_1 : E \to E_1$ and $\phi_2 : E \to E_2$ are **isomorphic** if there is a pointed isomorphism $\mu : E_1 \to E_2$ such that $\phi_2 = \mu \circ \phi_1$.

An important family of isogenies is the set of Frobenius morphisms. Consider $E/\mathbb{F}_q$, where $\mathrm{char}(\mathbb{F}_q) = p$, given in short Weierstrass form as $y^2 = x^3 + Ax + B$. Then $E^p : y^2 = x^3 + A^px + B^p$ is non-degenerate and the map

$$\pi(x,y) := (x^p, y^p) \tag{7.3}$$

is a bijective isogeny from E to E^p. This map is known as the Frobenius morphism. Its inverse exists but is not rational, hence this map is not an isomorphism. However, if $q = p^n$, then the

n-th power of the Frobenius map $\pi_q := \pi^n$ fixes $\mathbb{F}_q$ and thus $E^{(q)} = E$, and π_q is an endomorphism known as the *Frobenius endomorphism.* (This is the q-Frobenius endomorphism defined in 4.3.2).

The Frobenius map gives place to separable isogenies. These isogenies defined above are important since certain results and algorithms only hold for them (and/or normalized isogenies which are defined later).

Proposition 7.4. *Let E_1, E_2 be elliptic curves. Then any inseparable isogeny $\psi : E_1 \to E_2$ factors as the composition:*

$$\psi : \psi_{sep} \circ \pi^r, \tag{7.4}$$

where r is a positive integer and ψ_{sep} is a separable isogeny such that $deg_s(\psi) = deg(\psi)$.

To define normalized isogenies, consider again $\phi : E_1/K \to E_2/K$; this rational map induces, functorially, a map $\psi^* : \overline{K}(E_2) \to \overline{K}(E_1)$, and a map $\psi^* : \Omega_{E_2} \to \Omega_{E_1}$, where Ω_E is the 1-dimensional $\overline{K}(E)$-vector space of rational differential forms. Of these forms, an important one is the *invariant differential:*

$$\omega = \frac{dy}{3x^2 + a} = \frac{dx}{2y}.$$

Proposition 7.5. *Let $\phi : E_1/K \to E_2/K$ be an isogeny, and ω_1, ω_2 be the respective invariant differentials. Then*

$$\phi^*(\omega_2) = c\omega_1, \tag{7.5}$$

for some $c \in \overline{K}(E_1)$.

Definition 7.1.4. If $c = 1$ in 7.5, then ϕ is said to be a **normalized isogeny**.

We can normalize any isogeny using pointed isomoprhisms.

Proposition 7.6. *Every isogeny $\phi : E_1 \to E_2$ can be postcomposed with a pointed isomorphism τ such that the composition $\psi = \tau \circ \phi$ is normalized.*

Proof. Let us assume that E_2 is given in short Weierstrass form. Let $\phi^*(\omega_2) = c\omega_1$. Postcomposing with an isomorphism with scaling factor μ (such that the curves are isomorphic) changes the invariant differential $\omega_2' = \frac{1}{\mu}\omega_2$. Therefore, choose $\mu = c$ and apply the corresponding isomorphism to E_2. □

Using kernels to characterize Isogenies

When working over finite subgroups of Elliptic Curves we can characterize an isogeny from its kernel.

Proposition 7.7. *Let E be an elliptic curve and let Φ be a finite subgroup of E. There is a unique elliptic curve E' and a separable isogeny*

$$\phi : E \to E' \quad \textit{satisfying} \quad \ker(\phi) = \Phi.$$

A proof of this proposition can be found in [Sil09]. The elliptic curve E' in the proposition above is often denoted by the quotient E/Φ. This notation refers to the group structure of the quotient group of a variety. In general, the quotient of any variety by a finite group of automorphisms is again a variety, [Sha13]. In 1971, Jacques Vélu gave explicit formulas to compute E/Φ and its associated isogeny ϕ, [Vé71].
Let

$$E : y^2 = f(x) = x^3 + ax + b$$

be an elliptic curve. Compute the following values:

$$\nu = \sum_{P \in \Phi - \{\mathcal{O}\}} f'(P)$$
$$\omega = \sum_{P \in \Phi - \{\mathcal{O}\}} x(P) f'(P).$$

Then E/Φ is given by

$$E/\Phi : y^2 = x^3 + Ax + B,$$

where

$$A = a - 5\nu,$$
$$B = b - 7\omega.$$

Finally, the isogeny $\phi : E \to E/\Phi$ is given by [1]

$$\phi := \left(x(P) + \sum_{Q \in \Phi - \{\mathcal{O}\}} [x(P+Q) - x(Q)], y(P) + \sum_{Q \in \Phi - \{\mathcal{O}\}} [y(P+Q) - y(Q)] \right).$$

The curve obtained is a non-degenerate elliptic curve and ϕ is a normalized and separable isogeny with kernel Φ, [Law08]. Moreover, using these formulas, if $\#\Phi = \mathfrak{l}$, where $\mathfrak{l}$ is a positive integer, then the algebraic complexity of computing the codomain E/Φ and evaluating the isogeny ϕ on a point of $E(K)$ is $\mathcal{O}(\mathfrak{l})$ operations in $\overline{K}$ and $\mathcal{O}(\mathfrak{l}M(d))$ operations in K, where d is the degree of the minial extension F/K that contains Φ, and $M : \mathbb{N} \to \mathbb{N}$ a function such that multiplying polynomials of degree n costs $M(n)$ base field operations, [Shu09].
An important result of ϕ is that it is unique up to isomorphism.

Theorem 7.2. *Given a separable isogeny $\chi : E_1 \to E_2$ with kernel Φ, there is a pointed isomorphism $\varphi : E_1/\Phi \to E_2$ such that $\chi = \varphi \circ \phi$, [Sut15].*

This theorem asserts that ϕ is the unique separable isogeny with kernel Φ up to isomorphism. Moreover, for any inseparable isogeny $\chi : E_1 \to E_2$ with kernel Φ, ϕ is isomorphic to χ_{sep}.

In summary, we can represent isogenies using finite subgroups of elliptic curves that serve as the kernels. However, given Φ, where is the isogeny ϕ defined? To answer this, we need to consider the Galois action on the points Φ. (Refer to Appendix A for an introduction to Galois

[1] $X(P)$ denotes the x-coordinate of the point P. $Y(P)$ denotes the y-coordinate of the point P.

Theory)

Given E/K, the Galois group (refer to the Appendix) $\mathrm{Gal}(\overline{K} : K)$ acts on points of E by acting on their coordinates. Namely,

$$\forall \sigma \in \mathrm{Gal}(\overline{K} : F), \forall P \in K : P \in E \Rightarrow \sigma(P) \in E.$$

We are interested in where the points of Φ are taken under these Galois actions for any subgroup Φ of E, [San15].

Proposition 7.8. *Consider a subgroup $\Phi \subseteq E(\overline{K})$. Then Φ is said to be F-Galois stable iff for all $P \in \Phi$, and all $\sigma \in Gal(\overline{K} : F)$, it holds that $\sigma(P) \in \Phi$. The quotient isogeny ϕ determined by Velu's formula is defined over F if, and only if, Φ is F-Galois stable, [BBS06].*

As a corollary of this proposition we have that the Frobenius morphism π_q coincides with the Galois action whenever $K = \mathbb{F}_q$, since $\mathrm{Gal}(\overline{K} : K)$ is generated by the Frobenius map $x \to x^q$, [San15].

Using isogenies to classify elliptic curves. Tate's Theorem

Since every isogey has its dual we can define an equivalence relation of 'being isogenous' partioning elliptic curves into isogeny classes. E_1 and E_2 are F-isogenous if there is an isogeny $\psi : E_1 \to E_2$ defined over F. The prefix F is important since changing the field F potentially changes the classes, [San15].

A necessary and sufficient condition for two curves E, E' to be isogenous over a finite field $\mathbb{F}_q$ is given by Tate's Theorem.

Theorem 7.3 (Tate's Theorem.)**.** *Let $E/\mathbb{F}_q, E'/\mathbb{F}_q$ be elliptic curves. Then E and E' are isogenous over $\mathbb{F}_q$ if, and only if, $\#E(\mathbb{F}_q) = \#E'(\mathbb{F}_q)$, [Tat66].*

7.2 Elliptic Curves over $\mathbb{C}$

The study of elliptic curves was born in the field of complex analysis. Elliptic curves over complex numbers can be parametrized by a function called Weierstrass $\mathfrak{P}$-function and its derivative to define an analytic group homomorphic to the quotient of the additive group of complex numbers $\mathbb{C}$ by a lattice. With this characterization, isogeny theory simplifies, [SS03].

Definition 7.2.1. a) A **lattice** L (in $\mathbb{C}$) is the additive subgroup

$$L := \omega_1\mathbb{Z} + \omega_2\mathbb{Z} = \langle \omega_1, \omega_2 \rangle_{\mathbb{Z}}$$

in $\mathbb{C}$, generated by ω_1, $\omega_2 \in \mathbb{C}$ which are supposed to be linearly independent over $\mathbb{R}$. The generators ω_1, ω_2 are called *fundamental periods* of the lattice. A *period parallelogram* associated to L is given by

$$\{m_1\omega_1 + m_2\omega_2 : m_1, m_2 \in \mathbb{R}, 0 \leq m_1, m_2 < 1\}.$$

b) Let L, L' be two lattices in $\mathbb{C}$ and $\lambda \in \mathbb{C}$ with $\lambda L \subset L'$. Then λ induces a *homomorphism*

$$z \bmod L \mapsto \lambda z \bmod L',$$

and denoted by λ :

$$\lambda : \mathbb{C}/L \to \mathbb{C}/L'.$$

Either $\lambda = 0$ or $\lambda : \mathbb{C}/L \to \mathbb{C}/L'$ is surjective with kernel isomorphic to $L'/\lambda L$. Such a lattice homomorphism is called an *isogeny*.

c) Two lattices L and L' in $\mathbb{C}$ are **linearly equivalent**, if there exists a non-zero complex number λ such that $\lambda L = L'$. This complex number induces an *isomorphism* $\lambda : \mathbb{C}/L \to \mathbb{C}/L'$.

d) For two lattices L and L' in $\mathbb{C}$ with $L' \subset L$, the **index** of L' in L is defined as

$$[L : L'] := \frac{a(L')}{a(L)},$$

where $a(L)$ is the area of a period parallelogram associated to L.

The quotient structure $\mathbb{C}/L$, which is a torus, has a induced group law as well as an induced topology from the canonical projection map $p : \mathbb{C} \to \mathbb{C}/L$. From this torus we define a special bijection Φ_L to a complex elliptic curve E_L, [SS03]. Define

$$E_L : y^2 = x^3 + Ax + B,$$

where A and B are the convergent quantities

$$A = 15 \sum_{\omega \in L - \{0\}} \frac{1}{\omega^4}, \tag{7.6}$$

$$B = 35 \sum_{\omega \in L - \{0\}} \frac{1}{\omega^6}. \tag{7.7}$$

The bijection Φ_L uses the Weierstrass $\mathfrak{P}$-function.

Definition 7.2.2. Let L be a lattice in $\mathbb{C}$. The **(classical) Weierstrass $\mathfrak{P}$-function** associated to L is the function

$$\mathfrak{P} : \mathbb{C} \to \mathbb{C} \cup \{\infty\}$$

with[2]

$$\mathfrak{P}(L; z) := \mathfrak{P}(z) := \frac{1}{z^2} + \sum_{\omega \in L - \{0\}} \left(\frac{1}{(z-\omega)^2} - \frac{1}{\omega^2} \right).$$

This function gives a bijection $\Phi_L : \mathbb{C}/L \to E_L$ defined by

$$\Phi_L(z) = \left(\mathfrak{P}(z; L), \frac{\mathfrak{z}; \mathfrak{L}'}{2} \right); \tag{7.8}$$

from the torus $\mathbb{C}/L$ to the complex elliptic curve E_L.

It is because of the $\mathfrak{P}$-Weierstrass function that, Φ_L is not only a bijection but a group isomorphism.

[2]The notation $\mathfrak{P}(L; z)$ denotes that L is not a parameter of the function, since the domain is $\mathbb{C}$, but is used in the formula as an auxiliar parameter.

Proposition 7.9 ([San15]). *Φ_l is a group isomorphism $\mathbb{C}/L \to E_L(\mathbb{C})$ as well as an isomorphism of complex manifolds.*

Moreover, we can relate any complex elliptic curve, different from the family E_L, to complex tori of the form $\mathbb{C}/L$.

Theorem 7.4 (Uniformization Theorem [Sut15]). *For every complex elliptic curve E in short Weierstrass form, there is a lattice L such that $\mathbb{C}/L$ is isomorphic to E via Φ_L.*

Thus, we can fully identify complex elliptic curves with complex tori. From now on, let $\Phi = \{\Phi_L\}$ refer to this overall correspondence.

In this context, isogenies between elliptic curves are the same as isogenies between lattice quotiented fields. Let $E_1 = E_{L_1}$ and $E_2 = E_{L_2}$ be two complex elliptic curves. Then, an isogeny $\phi : E_{L_1} \to E_{L_2}$ induces a corresponding 'isogeny' $\phi^* : \mathbb{C}/L_1 \to \mathbb{C}/L_2$ of tori. This is done by $\phi^* := \Phi_{L_2}^{-1}\phi\Phi_{L_1}$. The following commutative diagram shows this procedure:

$$\begin{array}{ccc} \mathbb{C}/L_1 & \xrightarrow{\phi^*} & \mathbb{C}/L_2 \\ {\scriptstyle \Phi_{L_1}}\downarrow & & \downarrow{\scriptstyle \Phi_{L_2}} \\ E_{L_1} & \xrightarrow{\phi} & E_{L_2} \end{array}$$

Moreover, we have that ϕ^* is simply a 'multiplication by λ' for some fixed constant $\lambda \in \mathbb{C}$; which is known as *dilatation*.

Theorem 7.5 ([SS03]). *Let E_1, E_2 be elliptic curves over $\mathbb{C}$ corresponding to the lattices L_1, L_2. Then, there is a bijection*

$$\{\phi : E_1 \to E_2, \phi \text{ is an isogeny } \} \to \{\lambda : \mathbb{C}/L_1 \to \mathbb{C}/L_2 : \lambda L_1 \subseteq L_2\}.$$

This theorem asserts that ϕ^* can de described as a 'multiplication by λ' map and that the converse is also true, i.e., every $\lambda \in \mathbb{C}$ such that $\lambda L_1 \subseteq L_2$ induces a map $\lambda : \mathbb{C}/L_1 \to \mathbb{C}/L_2$ that comes from an isogeny ϕ in 7.2. Thus we can use the notion of isogenies defined in terms of tori (7.2.1) interchangeably with the notion of isogenies between elliptic curves. These isogenies define an equivalence relation on the lattice quotient fields called *homothety*, i.e., the complex tori $\mathbb{C}/L_1$ and $\mathbb{C}/L_2$ are isomorphic when one lattice is a scalar of the other. Again, this equivalence relation has its correspondence for elliptic curves.

Corollary 7.1. *Φ establishes a bijective correspondence between isomorphism classes of complex elliptic curves and complex tori, modulo the relation of homothety.*

Next results will use 'complex elliptic curve' interchangeably to denote a complex elliptic curve E or its torus $\mathbb{C}/L$.

Definition 7.2.3. A curve $\mathbb{C}/L$ has **complex multiplication** if $\mathbb{Z} \subset \mathrm{End}(E)$.

Proposition 7.10. *If a curve $\mathbb{C}/L$ has complex multiplication, where $L = L_{\omega_1,\omega_2}$, then $\tau := \frac{\omega_2}{\omega_1}$ is an imaginary quadratic number and End(E) is an order in $K = \mathbb{Q}(\tau)$.*

For applications in cryptography, we are interested, for an order O in an imaginary quadratic field K, in the set of isomorphism classes of complex elliptic curves with endomorphism ring O, denoted $\mathrm{ELL}_O(\mathbb{C})$. We can classify these classes once we recognized O and its ideals as lattices. Sets that are particularly relevant are:

$$I_O = \{\text{invertible ideals of } O\}, \tag{7.9}$$
$$\mathbb{C}/I_O := \{\mathbb{C}/\mathfrak{a} | \mathfrak{a} \in I_O\}. \tag{7.10}$$

Lemma 7.2. *([Sut15]) For any $E \in \mathbb{C}/I_O$, End(E) = O. Conversely every complex elliptic curve with endomorphism ring O is isomorphic (trough homothety in their lattices) to a member of $\mathbb{C}/I_O$.*

This lemma let us use $\mathbb{C}/I_O$ as a full set of representatives for $\mathrm{Ell}_O(\mathbb{C})$. Nevertheless, these representatives are not in general distinct isomorphism classes of curves. In fact,

$$\mathbb{C}/\mathfrak{a} \equiv \mathbb{C}/\mathfrak{a}' \iff \mathfrak{a}\ \mathfrak{a}' \iff [\mathfrak{a}'] \text{ in Cl(O)}.$$

This relation let us establish the following corollary.

Corollary 7.2. *Cl(O) is bijective with $Ell_O(\mathbb{C})$, by:*

$$[\mathfrak{a}] \to \mathbb{C}/\mathfrak{a}, \tag{7.11}$$

where $\mathfrak{a}$ is a representative for an ideal class such that $\mathfrak{a} \in I_O$. In particular, $Ell_O(\mathbb{C})$ is finite and its size is $h(O)$, the class number of the order.

Moreover, $\mathrm{Cl}(O)$ induces, canonically, a free and transitive action on $\mathrm{Ell}_O(\mathbb{C})$.

$$\begin{aligned} *:&\mathrm{Cl}(O) \times \mathrm{Ell}_O(\mathbb{C}) \to \mathrm{Ell}_O(\mathbb{C}) \\ &[\mathfrak{a}] * j(\mathbb{C}/\mathfrak{b}) = j(\mathbb{C}/\mathfrak{a}^{-1}\mathfrak{b}). \end{aligned}$$

This action comes from a 'parent' action, [Sut15], that works with isogenies:

$$\begin{aligned} \star:&I_O \times \mathbb{C}/I_O \to \mathbb{C}/I_O \\ &\mathfrak{a} \star \mathbb{C}/\mathfrak{b} = \mathbb{C}/\mathfrak{a}^{-1}\mathfrak{b}. \end{aligned}$$

Because $\mathfrak{b} \subseteq \mathfrak{a}^{-1}\mathfrak{b}$, this action derives a quotient isogeny

$$\begin{aligned} \phi_\mathfrak{a}:&\mathbb{C}/\mathfrak{b} \to \mathbb{C}/\mathfrak{a}^{-1}b \\ &z \bmod \mathfrak{b} \mapsto z \bmod \mathfrak{a}^{-1}\mathfrak{b}. \end{aligned}$$

The suffix $\mathfrak{a}$ in the action denotes the fact that the kernel of this map is the set of points annihilated by $\mathfrak{a}$. Using the isomorphism φ, this maps transforms into an isogeny:

$$\phi_\mathfrak{a} : E_\mathfrak{b} \to E_{\mathfrak{a}^{-1}\mathfrak{b}}$$

with the $\mathfrak{a}$-torsion $E_{\mathfrak{b}[\mathfrak{a}]}$ as its kernel. This kernel has degree $N(\mathfrak{a})$.

Since this action can be considered modulo relations of isomorphism, both form the acting group and the acting space, its induced action on $\mathrm{Ell}_O(\mathbb{C})$ is a *principal homogenenous space* for $\mathrm{Cl}(O)$, i.e., $\mathrm{Cl}(O)-$*torsor*, [Sil94].

This result is valid for any field, luckily for finite fields used in cryptography, the group action torsor reduces in the following sense.

The j invariants of $\mathrm{Ell}_O(\mathbb{C})$ all lie in the Hilbert class field H_O, with extension degrees the class number $h(O)$. H_O is a Galois extension of K. Therefore, we restrict the study of this action to curves over H_O and consider $\mathrm{Ell}_O(\mathbb{C}) = \mathrm{Ell}_O(H_O)$. H_O is a number field and so, the theorems of During, [Koh96], come into place.

Theorem 7.6 ([Koh96])**.** *Let $\tilde{E}/\tilde{\mathbb{Q}}$ be an elliptic curve with endomorphism ring $End(\tilde{E}) = O$, where O is an order in an imaginary quadratic extension K of $\mathbb{Q}$. Let $\mathfrak{P}$ be a prime ideal of $\mathbb{Q}$, over a prime p, at which $\tilde{E}$ has nondegenerate reduction E. The curve E is supersingular if, and only if, p has only one prime of K above it. It p splits in K, then let m be the conductor of O, so that $O = \mathbb{Z} + mO_K$. Write $m = p^r m_0$, where p^r is the largest power of p dividing m. Then the endomorphism ring of E is as follows:*

1. *$End(E) = \mathbb{Z} + m_0 O_K$ is the order of K with conductor m_0.*
2. *If $(p, m) = 1$ the map $\varphi \to \tilde{\varphi}$ is an isomorphism of $End(\tilde{E})$ onto $End(E)$.*

Theorem 7.7 ([Koh96])**.** *Let E be an elliptic curve over a finite field k of characteristic p and let φ be an endomorphism of E. Then, there exists an elliptic curve $\tilde{E}$ defined over a number field H, and endomorphism $\tilde{\varphi}$ of $\tilde{E}$, and a prime $\mathfrak{P}$ over p in H such that E is isomorphic to the reduction of $\tilde{E}$ at $\mathfrak{P}$, and φ corresponds to a reduction of $\tilde{\varphi}$ under this isomorphism.*

Consider a prime p that will give such an isomorphism of endomorphism rings between the complex curves and reduced ordinary curves, i.e.,

1. p splits in the endomorphism algebra K.
2. The prime $\mathfrak{P} \in H_O$ over p induces a non-degenerate reduction of Weierstrass forms.
3. $(p, m) = 1$, where m is the conductor of the order O.

Now, let $\mathbb{F}_q$ be a field with $q = N(\mathfrak{P}) = p^f$, let $\mathrm{Ell}_O(\mathbb{F}_q)$ be the isomorphism classes of *ordinary* curves defined over $\mathbb{F}_q$. Then the reduction map $O_{H_O} \to O_{H_O}/\mathfrak{P}$ extends to a reduction

$$g : \mathrm{Ell}_O(H_O) \to \mathrm{Ell}_O(\mathbb{F}_q) \tag{7.12}$$

that is a bijection for most cases, [BCL08]. This reduction preserves the group actions, to see this consider the set

$$E_O(\mathbb{F}_q) := \{E/\mathbb{F}_q | \ \mathrm{End}(E) \in O\}. \tag{7.13}$$

Then we have the reduction of the 'parent' action:

$$\begin{aligned} \star : I_O \times E_O(\mathbb{F}_q) &\to E_O(\mathbb{F}_q) \\ \mathfrak{a} \star E &= E/E[\mathfrak{a}] \end{aligned}$$

and, modulo isomorphims, the reduction of the induced action known as the *complex multiplication operator*:

$$\begin{aligned} * : \mathrm{Cl}(O) \times \mathrm{Ell}_O(\mathbb{F}_q) &\to \mathrm{Ell}_O(\mathbb{F}_q) \\ [\mathfrak{a}] * j(E) &= j(E/E[\mathfrak{a}]) \end{aligned}$$

that makes $\mathrm{Ell}_O(\mathbb{F}_q)$ a $\mathrm{Cl}(O)$-torsor. This fact is what motivated isogeny based cryptography. As in the general case, the action $\star$ derives an isogeny: $\phi_{\mathfrak{a}} : E \to E/E[\mathfrak{a}]$ of degree $N(\mathfrak{a})$. In fact, we can use $\star$ as a valid representation for this isogeny. An important fact of this isogeny is that it is $\mathbb{F}_q$-rational, [San15].

Lemma 7.3. *The isogeny $\phi_{\mathfrak{a}}$ induced by an ideal $\mathfrak{a} \in I_O$ is $\mathbb{F}_q$-rational.*

Proof. For any $P \in E[\mathfrak{a}]$, and for any $a \in \mathfrak{a}$, we have that $a(\pi_E(P)) = \pi_E(a(P)) = \mathcal{O}$ since a, π_E are both in O. By the next corollary of Propositon 7.8, it follows that $\pi_E(P) \in E[\mathfrak{a}]$ □

Corollary 7.3 (Corollary of Proposition 7.8)**.** *Let $E/\mathbb{F}_q$ be an elliptic curve and $\Phi \subseteq E(\overline{\mathbb{F}}_q)$ be a finite subgroup. Then the quotient isogeny ϕ determined by Velu's formula is defined over $\mathbb{F}_{q^r}$ if it is the case that for all $P \in \Phi$, $\pi_q^r(P) \in \Phi$.*

Regarding the computational cost of computing the action $\star$, we have different methods. First of all, since isogenies can be understood as lattices, we don't need to evaluate any isogeny but just lift E to a complex curve and compute the action there, to finally reduce it. However, this method is not efficient where the class number $h(O)$ is very large, which it usually is, [San15].

A better approach is to decompose the isogeny in its prime components. Consider the case where $\mathfrak{a}$ decomposes into split or ramified prime ideals $\mathfrak{p}_i$ of small norm[3] and with small exponents e_i, i.e.,

$$\mathfrak{a} = \mathfrak{p}_1^{e_1} \cdots \mathfrak{p}_n^{e_n}. \tag{7.14}$$

Using this 'factorization', we can evaluate $\phi_{\mathfrak{a}}$ as the evaluation of the composition of the prime degree isogenies $\phi_{\mathfrak{p}_i}$ induced by the corresponding prime ideals.

When $\mathfrak{a}$ is not smooth, the best algorithms compute an isogeny induced by a member of $[\mathfrak{a}]$ that is smooth. Let $\mathfrak{a}' \in [\mathfrak{a}]$ be such ideal, it differs from $\mathfrak{a}$ by a principal ideal, i.e. $\mathfrak{a} = (\alpha)\mathfrak{a}'$, and we have efficient algorithms to compute the principal ideal and evaluate its corresponding isogeny. Nevertheless, this approach is optimized for evaluating the star operator $\star$, and thus if computing it is the only task we want to achieve we can skip the last step, [San15].

The basic step of this procedure is to evaluate an isogeny $\phi_{\mathfrak{l}}$ induced by a split prime ideal $\mathfrak{l}$ of small prime norm l. $\mathfrak{l}$ is generated by $(l, c + d\pi_e)$, where $c, d \in \mathbb{N}$.

A first approach to evaluate the isogeny is to compute all $l+1$ cyclic subgroups of $E[\mathfrak{l}]$l and then compute the action of $\mathfrak{l}$ on these subgroups. The subgroup annihilated by $\mathfrak{l}$ is the kernel

[3]The norm of an ideal is a generalization of a norm of an element in the field extension. This is, if $\mathfrak{a} \in I_O$ is an ideal, then its norm is the size of the finite quotient field $I_O/\mathfrak{a}$, [Jan96].

of $\phi_\mathfrak{l}$. Using Velu's formulas we evaluate $\phi_\mathfrak{l}$ as a rational map.

However, since we know that the isogenies we work on are $\mathbb{F}_q$-rational (7.3) we can improve our approach ignoring isogenies that cannot be defined over $\mathbb{F}_q$.

7.3 Cryptography based on Ordinary Curves

Couveignes [Cou06] in 1997 and Stolbunov [GS13] in 2009 used ordinary curves, specifically the $\mathrm{Cl}(O)$-torsor $\mathrm{Ell}_O(\mathbb{F}_q)$, to create cryptographic schemes. They noticed that a set with group action that satisfies certain assumptions of 'efficient' and 'hard' problems in the homogenous space generalizes the classical dichotomy of exponentiation vs. discrete logarithm in finite fields, and thus leads to cryptographic primitives emulating classical schemes such as the Diffie-Hellman key exchange and Elgamal encryption.

Using Hard Homogenous Spaces for Public-key Protocols

Let us start by describing how a general G-torsor X, for a commutative group G, can lead to creating asymmetric algorithms for cryptography.

Definition 7.3.1. A **G-torsor** X, for a commutative group G, based on the group action

$$* : G \times X \to X$$

is a ***hard homogenous space*** (HHS) if it meets the following general requirements. The following computations should be computationally efficient:

1. (Group computation) Compute for any g, g^{-1}, or for any two elements g_1, g_2, the product $g_1 g_2$.
2. (Random element) Find a random element $g \in G$.
3. (Membership) Test whether a given element x is contained in X.
4. (Action) Given $g \in G$ and $x \in X$, compute $g * x$.

The following problems should be computationally hard:

1. (Vectorization) Given $x, y \in X$, find the g such that $g * x = y$.
2. (Parallelization) Given x and $y = g * x$, respond to a challenge $z \in X$, with $g * z$.
3. (Parallel testing) Given a challenge set $x, y, z, z' \in X$, such that $y = g * x$, decide if $z' = g * z$.

The quality of being 'hard' depends on the context where the torsor is defined. For example, since the vectorization problem can generally be solved in $\mathcal{O}(|X|^{1/2})$ computations of the group action, we must require for $|X| = |G|$ to have exponential size.

To see how these hard homogeneous spaces can be used in cryptography, we give three examples of public-key primitives based on the classical Diffie-Hellman, Elgamal scheme, and

a zero-knowledge identification protocol.

Key Exchange

This protocol generalizes the Diffie-Hellman protocol. Its security relies on the parallelization problem associated with the space, [San15].

1. Public parameter: a basepoint $x \in X$.
2. Alice chooses a random element $a \in G$, and computes $k_a = a * x$, sending this value to Bob.
3. Bob chooses a random element $b \in G$, and computes $k_b = b * x$, sending this value to Alice.
4. Alice computes $k_{AB} = a * k_b$, while Bob computes the same key $k_{AB} = b * k_A$.

Public Key Encryption On Stolbunov's thesis [Sto12], there is a generalization of Elgamal scheme for public key encryption. Its security relies on the parallel testing problem associated to the space.

1. Public parameters: Hash function $H = \{H_k\} : X \to \{0,1\}^w$, public base point $x \in X$.
2. Key Generation: secret key $sk \in G$ randomly chosen and public key $pk = sk * x$, $k \in K$.
3. Encryption: Choose $a \in G$. Then $E(m) = (c, z) := (H_k(a * pk) \oplus m, a * x)$.
4. Decryption: $m = D(c, z) = H_k(sk * z) \oplus c$.

Identification protocol

Alice proves herself (via her identity $g_a \in G$) to Bob as follows. This protocol is computationally zero-knowledge where its security relies on the parallel testing problem of the associated space, [Cou06].

1. Public parameters: public basepoint $x \in X$, public-key $p_k = g_a * x$.
2. Alice chooses a random $r \in G$ and computes and transmits $y = r * pk$,
3. Bob chooses a random challenge bit $b \in \{0, 1\}$.
4. If $b = 0$, Alice reveals r and Bob checks that $r * pk = y$. Else Alice reveals $g = r * g_a$ and Bob checks that $g * x = y$.

Now that we know how to create cryptographic schemes using Hard Homogeneous Spaces, let us discuss how the $\mathrm{Cl}(O)$-torsor $\mathrm{Ell}_O(\mathbb{F}_q)$ can be used as an HHS.

The Cl(O)-torsor Ell$_O(\mathbb{F}_q)$ as a HHS

$$\begin{array}{ccc} j(E) & \longrightarrow & [\mathfrak{a}] * j(E) \\ \downarrow & & \downarrow \\ [\mathfrak{b}] * j(E) & \longrightarrow & [\mathfrak{a}][\mathfrak{b}] * j(E) \end{array}$$

Figure 7.1: DH-type key exchange using ordinary curves

The commutative diagram in Fig. 7.1 is the basic scheme for DH-type key exhange using ordinary curves: A base ordinary elliptic curve given by $j(E)$. Alice's private key is $[\mathfrak{a}]$ and Bob's private key is $[\mathfrak{b}]$. $[\mathfrak{a}] * j(E)$ and $[\mathfrak{b}] * j(E)$ are Alice's and Bob's public key respectively. $[\mathfrak{a}][\mathfrak{b}] * j(E)$ is then their shared secret key.

There are certain conditions that must be achieved in order for Ell$_O(\mathbb{F}_q)$ be an HHS. First, $|\text{Ell}_O(\mathbb{F}_q)| = h(O)$ needs to be exponentially large.
Now, before using Ell$_O(\mathbb{F}_q)$ to construct cryptographic protocols, we need to take into account that the encryption process is still hard. The encryption part in all protocols requires the ability to compute the complex multiplication operator, which is the group action of the space. However, the best algorithms to compute it are subexponential in $\log\big(|\Delta|\big)$ (and polynomial in the degree of the acting ideal), [San15]. More work is required to improve performance of these computations.

Algorithms to compute the group action

Suppose you are given a $j(E) \in \text{Ell}_O(\mathbb{F}_q)$, and $[\mathfrak{a}] \in \text{Cl}(O)$, you need to compute $[\mathfrak{a}] * j(E)$. As mentioned above, the best strategy involves factoring the ideal into its prime components

$$[\mathfrak{a}] = [\mathfrak{p}_1]^{e_1}[\mathfrak{p}_2]^{e_2} \cdots [\mathfrak{p}_n]^{e_n} \tag{7.15}$$

where the $\mathfrak{p}_i$ are split prime ideals of small norm and e_i are small exponents. Then, you compute sequentially the action of a family of isogenies in each of these ideals. This procedure is summarized in the following algorithms.

Algorithm 10 Compute $\phi_{\mathfrak{l}}$ for a split prime ideal $\mathfrak{l}$

INPUT: An elliptic curve $E/\mathbb{F}_q$ in Weiersrtrass form, a split prime ideal in I_O $\mathfrak{l} = (l, c + d\pi_E)$, where $O \cong \text{End}(E)$ an order in K, and $l \nmid [O_K : \mathbb{Z}[\pi_E]]$
OUTPUT: An evaluation of $\phi_{\mathfrak{l}} : E \to E'$
PROCEDURE:

1: Construct the modular polynomial $\Phi_{\mathfrak{l}}(X, Y) \in \mathbb{F}_q$ and find the two roots j_1 and j_2 of $\Phi_{\mathfrak{l}}(j(E), Y)$ over $\mathbb{F}_q$.
2: For each root, find target curves $E_1/\mathbb{F}_q$ and $E_2/\mathbb{F}_q$ and a prime degree isogeny algorithm to find points P_1 and P_2 such that $\langle P_1 \rangle$ and $\langle P_2 \rangle$ are the kernels of the isogenies to E_1 and E_2 respectively.
3: Find which points satisfies $[c] + [d]\pi_q(P_i) = \mathcal{O}$; this point P_i is the kernel of $\phi_{\mathfrak{l}}$.
4: Use Velu's formula for $\langle P_i \rangle$ to evaluate $\phi_{\mathfrak{l}}$.

Algorithm 11 Computing the complex multiplication operator

INPUT: $\Delta, q, [\mathfrak{a}], j(E)$

OUTPUT: The element $j(E') \in \mathrm{Ell}_{\Delta}(\mathbb{F}_q)$ such that $[\mathfrak{a}] * j(E) = j(E')$.

PROCEDURE:

1: Compute a factor base $\mathfrak{F} = \{\mathfrak{p}_1, \cdots, \mathfrak{p}_n\}$ of split prime ideals of norm $\leq B$ for some specified B.
2: Compute a vector $e = (e_1, \cdots, e_n)$ of small L^1 norm such that $[\mathfrak{a}] = [\mathfrak{p}_1]^{e_1}[\mathfrak{p}_2]^{e_2} \cdots [\mathfrak{p}_n]^{e_n}$
3: Compute a sequence of $(\phi_1, \cdots, \phi_n)$ of isogenies such that the composition $\phi_c : E \to E_c$ of the sequence has kernel $E[\mathfrak{p}_1^{e_1} \cdots \mathfrak{p}_n^{e_n}]$ using Algorithm 10.
4: Return $j(E)$.

Some comments regarding the enlisted algorithms are important. In Algorithm 10 we mentioned in the first step the construction of a modular polynomial. A modular polynomial of degree N is a symmetric polynomial $\Phi_N(X, Y)$ of degree $2N + 1$ in $\mathbb{Z}[X, Y]$ that parameterizes pairs of j invariants over $\mathbb{C}$ that are related by a cyclic N-isogeny (for any representative curves). From this definition we get that given a curve E and a prime L, the roots of $\phi_l(j(E), X)$ give the j-invariants of complex curves l-isogenous to E. In the second step of this algorithm , we use 'Atkins-Elkies' formulas [Sch95] for each root j' to construct a curve E' and a normalized l-isogeny $E \to E'$. The curve E' has j-invariant j' and establishes the existence of a normalized and separable l-isogeny $\phi : E \to E'$.[Sch95]. Knowing both curves E, E'. Bostan, Morain, Salvy and Schost [BMSS08] implemented an algorithm to compute the isogeny ϕ in $\mathcal{O}(M(l)\log(l))$ operations. The output of this algorithm is a *kernel polynomial*, which is a polynomial in x whose roots are exactly the x-coordinates of the points in $\ker(\phi)$.

For the Algorithm 11, the important step is step 3, which is computed in time proportional to $\Pi|e_i|N(\mathfrak{p}_i)^2$, thus depends on the quality of the factorization in steps 1 and 2.

In the factor base $\mathfrak{F}$ computed in step 1 all class group elements can decompose, i.e.,

$$\mathrm{Cl}(O) = \bigotimes_{\mathfrak{F}} \mathfrak{p}_i. \tag{7.16}$$

The existence of $\mathfrak{F}$ with bounded norm is justified by Minkowski's bound [San15]. Moreover, we can improve this bound under the Generalized Riemman Hypothesis (GRH) [Bac90].

Of course, this does not tell us how to find decomposition (7.16) for a given ideal class $[\mathfrak{a}]$. Thus, the search for a factor base with small norms requires a tradeoff: the more split prime ideals we include, the more relations we can find efficiently, and the easier it is to factor with small exponents [San15]. The first to suggest this strategy were Broker, Charles, and Lauter on a paper from 2008 [BCL08]. Since then, many authors have presented different algorithms for factoring, whose performance is improved for specific parameter choices and heuristics.

The search for better factoring lets to find weaknesses in the schemes based on ordinary curves. In particular, in 2010 Jao and Soukharev [JS10], showed a way to evaluate large degree isogenies between elliptic curves in subexponential time under reasonable heuristics. Their approach was based on factoring the ideal corresponding to the kernel of the isogeny, modulo

principal ideals, into a product of smaller primer ideals for which the isogenies can be computed directly.

Eight years later, Jao and Sokharev together with Childs, [CJS14], gave a quantum algorithm for constructing elliptic curves in subexponential time. Assuming only the GRH, they found out that the problem of finding an isogeny between ordinary curves can be reduced to a hidden shift problem, which is known to be solved by Kuperberg's quantum algorithm.[Kup05]. The Hidden Subgroup Problem is one of the problems that have quantum algorithms which achieve super polynomial speedup in the quantum setting in contrast to the classical one. For example, Shor's algorithm discussed in Section 6.5 solves the hidden subgroup problem over the finite group generated by a point P of an elliptic curve.

Definition 7.3.2 (**The Hidden Subgroup Problem**)**.** For a finite set S, let $|S\rangle$ denote the state

$$|S\rangle = \frac{1}{\sqrt{|S|}} \sum_{s \in S} |s\rangle \, .$$

In the hidden subgroup problem, we have a known finite group G, a finite set S, and (black-box access to) a function $f : G \to S$ which is constant on all (right) cosets of a hidden subgroup $H \leq G$.[4] The goal is to discover H, say as a set of its generators.

The result from Childs et al, suggests that isogeny-based cryptosystems may be uncompetitive with more powerful quantum-resistant cryptosystems. Thus, justifying the turn to the supersingular case.

7.4 Crytography based on Supersingular Curves

De Feo, Jao, and Plut decided to see if supersingular curves were better at resistant quantum attacks. In 2014, they published a paper called "Towards quantum-resistant Cryptosystems From Supersingular Elliptic Curve Isogenies" [DFJP14] were inspired by the protocols in the ordinary case they designed supersingular cryptographic protocols. This is the birth of Supersingular Isogeny Diffie–Hellman/Supersingular Isogeny Key Encapsulation SIDH/SIKE protocol which is a candidate for the postquantum cryptography standardization process by NIST. The security of all these schemes relies on the conjectured difficulty of finding isogenies between supersingular elliptic curves. As in the ordinary case, the idea is to transmit the images of torsion bases under the isogeny in order to allow parties to construct a shared commutative square despite the noncommutativity of the endomorphism ring. Contrary to the ordinary case, the fastest known quantum attack to the protocols that will be discussed here remains exponential, since the noncommutativity of the endomorphism ring means that the approach used in the ordinary case does not apply.

We say that we used a shared commutative diagram as the principle behind cryptosystems from supersingular elliptic curves isogenies, however, the endomorphism ring in the supersingular case is not commutative. How is this possible? By allowing auxiliary information that is

[4] If a subset H of a group G is itself a group under the binary operation of G, then H is a subgroup of G, written as $H \leq G$ or $G \geq H$.

conjectured not to compromise security.

7.4.1 Parameters

The protocols for supersingular settings require supersingular curves of smooth order. Notice that these curves are avoided in the classical elliptic curve cryptography since they have easy discrete logarithms. In this setting, we do not rely upon discrete logarithms and so these issues do not affect us. Even more, we should have smooth order curves since they will have a large number of isogenies that are fast to compute. Specifically, the parameters needed for our setting are the following:

1. The characteristic of the prime subfield where the curves live of the form $p = l_A^{e_a} l_B^{e_b} f \pm 1$, where f is a cofactor such that p is prime. These primes are dense [LO77], so for any choice of l_A, l_B, e_A, e_B, a simple trial and error algorithm finds a prime of this form.

2. The base point $E/\mathbb{F}_{p^2}$ is a supersingular curve over $\mathbb{F}_{p^2}$ of rational cardinality $(l_A^{e_a} l_B^{e_b} f)^2$. This curve can be computed efficiently by Broker's algorithm [BCL08].

3. Torsion bases P_A, Q_A of $E[l_A^{e_A}]$ and P_B, Q_B of $E[l_B^{e_B}]$. These can be computed efficiently using a simple randomized algorithm that scales random points of E, and tests linear independence via the Weil pairing [Sil09].

From this list, public parameters include: $p, E, l_A, e_A, e_B, P_A, P_B, Q_A$ and Q_B. The private keys are cyclic subgroups $\langle S \rangle$ and $\langle R \rangle$ of the $l_A^{e_A - 1}(l_A + 1)$ full cyclic subgroups possible in $E[l_A^{e_A}]$ and the $l_B^{e_B - 1}(l_B + 1)$ full cyclic subgroups possible in $E[l_B^{e_B}]$, respectively.

All of these protocols revolve around the following commutative diagram,

$$\begin{array}{ccc} E & \xrightarrow{\phi} & E/\langle S \rangle \\ {\scriptstyle\psi}\downarrow & & \downarrow \\ E/\langle R \rangle & \longrightarrow & E/\langle S, R \rangle \end{array}$$

Figure 7.2: Principle behind cryptosystems from supersingular elliptic curves isogenies

where ϕ and ψ are random walks in the graphs of isogenies of degrees l_A and l_B respectively. Their security relies on the difficulty of finding a path connecting two given vertices in a graph of supersingular isogenies, [DFJP14].

Since we use graphs of isogenies, few definitions regarding Graph Theory are necessary to understand the protocols.

7.4.2 Isogeny graphs

Let $G = (\mathcal{V}, \mathcal{E})$ be a finite graph on h vertices $\mathcal{V}$ with undirected edges $\mathcal{E}$. Suppose G is a regular graph of degree k, i.e., exactly k edges meet at each vertex. Given a labeling of the vertices $\mathcal{V} = \{v_1, \cdots, v_h\}$, the adjacency matrix of G is the symmetric $h \times h$ matrix A whose $ij-$th entry $A_{i,j} = 1$ if an edge exists between v_i and v_j and 0 otherwise.
Using this labeling, we can identify functions on $\mathcal{V}$ with vectors in $\mathbb{R}^h$, and thus A can be seen as a self-adjoint operator on $L^2(\mathcal{V})$. By construction, all of the eigenvalues of A satisfy the bound $|\lambda| \leq k$. The trivial eigenvalue $\lambda_{triv} = k$ has constant vectors as associated eigenvectors. Consider a family of such graphs G with $h \to \infty$. If all non-trivial eigenvalues of their adjacency matrices are bounded away from the trivial one by a fixed amount, the family is said to be a sequence of *expander graphs*. A **Ramanujan graph** is an expander graph which has $|\lambda| \leq 2\sqrt{k-1}$ for any nontrivial eigenvalue which is not equal to $-k$, [PS88].

We use expander graphs to prove the rapid mixing of the random walk on $\mathcal{V}$ along the edges $\mathcal{E}$.

Proposition 7.11. *Let G be a regular graph of degree k on h vertices. Suppose that the eigenvalue λ of any constant eigenvector satisfies the bound $|\lambda| \leq c$ for some $c < k$. Let S be any subset of the vertices of G, and x be any vertex in G. Then, a random walk of length at least $\frac{\log\left(2h/|S|^{1/2}\right)}{\log\left(k/c\right)}$ starting from x will land in S with probability at least $\frac{|S|}{2h} = \frac{|S|}{2|G|}$.*

An **isogeny graph** is a graph whose nodes consist of all elliptic curves in $\mathbb{F}_q$ belonging to a fixed isogeny class, up to $\overline{F}_q-$isomorphism (so that two elliptic curves which are isomorphic over $\overline{\mathbb{F}}_q$ represent the same node in the graph). Since isomorphic curves share j-invariant, in practice the nodes are represented using them. Isogeny graphs have the Ramanujan property, [CFL+19].

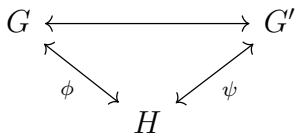
Since every supersingular elliptic curve in characteristic p is defined over either $\mathbb{F}_p$ or $\mathbb{F}_{p^2}$, [Sil94], we fix $\mathbb{F}_q = \mathbb{F}_{p^2}$ as the field where we work on. In this context, in any given isogeny class there are approximate $g+1 = \frac{p}{12}+1$ isomorphism classes of supersingular curves, [Mes86]. Even more, all supersingular curves defined over $\mathbb{F}_{p^2}$ belong to the same isogeny class. Thus, we define supersingular graphs in terms of isomorphisms between isogenies. For a fixed prime value of $l \neq p$, the **supersingular isogeny graph** $\mathcal{G}$ is a graph whose vertices are the g isomorphism classes of curves, with edges given by isomorphism classes of degree-l isogenies. It turns out that $\mathcal{G}$ is a connected $k = l+1$-regular multigraph satisfying the Ramanujan bound of $|\lambda| \leq 2\sqrt{k-1}$ for the nontrivial eigenvalues of its adjacency matrix, [CFL+19].

With this notation in mind, we are ready to present the cryptographic protocols.

7.4.3 Zero-knowledge proof of identity

Suppose Alice wants to identify herself to Bob. She decides to use isogeny based cryptography to do this. Alice and Bob select curves E and $E/\langle S\rangle$ as their road to transmit the information. These curves are publicly known. Alice knows in secret a cyclic degree $l_A^{e_A}$ isogeny $\phi : E \to E/\langle S\rangle$. Alice will prove to Bob that she knows a generator for $\langle S\rangle$, without revealing it.

Alice will use a protocol similar to the zero-knowledge proof of membership for Graph Isomorphism, [GMW91]. In this protocol, Alice shows that she knows a graph isomorphism $G \equiv G'$ by first publishing a random H such that the following diagram commutes.



and then revealing only one among ψ and ϕ. This protocol is *perfectly* zero knowledge because the random permutation of G or G' that Alice reveals could be easily computed by anyone without her help.
Analogously, in the isogeny setting, Alice publishes the vertices of the diagram 7.2, and then reveals some of its arrows. In contrast to the Graph Isomorphism protocol, this one is not *perfectly* zero-knowledge since Alice needs to use her secret knowledge to compute the diagram. However, under certain conditions, it is *computationally* zero-knowledge. [DFJP14].
The diagram used in this protocol is the following:

$$\begin{array}{ccc} E & \xrightarrow{\phi} & E/\langle S\rangle \\ {\scriptstyle\psi}\downarrow & & \downarrow{\scriptstyle\psi'} \\ E/\langle R\rangle & \xrightarrow[\phi']{} & E/\langle S,R\rangle \end{array} \tag{7.17}$$

In this diagram, $\langle S\rangle$ is the kernel of the secret isogeny ϕ of degree $l_A^{e_A}$, while $\langle R\rangle$ is a cyclic group of order $l_B^{e_B}$. Then, to compute the diagram Alice needs to:

- Use Velu's formulas to compute the isogeny $\psi : E \to E/\langle R\rangle$;
- Compute the image $R' = \phi(R)$ and the isogeny $\psi' : E/\langle S\rangle \to E/\langle S, R\rangle$;
- Compute the image $S' = \psi(R)$ and the isogeny $\phi' : E/\langle R\rangle \to E/\langle S, R\rangle$;

The knowledge of any pair (ψ, ϕ') or (ψ', ϕ') will allow anyone to compute the kernel of ϕ. However, there is no obvious way to compute ϕ from the sole knowledge of ϕ' and one of ψ or ψ'. On the other hand, the revealing of R and $\phi(R)$ uncovers some information on the action of ϕ on $E[l_B^{e_B}]$ and could potentially allow anyone to evaluate ϕ on the whole $E[l_B^{e_B}]$. Nevertheless, it is conjectured that this does not compromise Alice's secret. The whole protocol is summarized below.
Secret paramenters: A supersingular curve E defined over F_q and a primitive $l_A^{e_A}$-torsion point S defining an isogeny $\phi : E \to E/\langle S\rangle$.
Public parameters: The curves E and $E/\langle S\rangle$. Generators P, Q of $E[l_B^{e_B}]$ and their images $\phi(P)$, $\phi(Q)$.
Identification: Repeat m times:

1. Alice chooses a random primitive $l_B^{e_B}$-torsion point R and computes diagram 7.17.
2. Alice sends the curves $E_1 = E/\langle R\rangle$ and $E_2 = E/\langle S, R\rangle$ to Bob.
3. Bob selects a random bit b and sends it to Alice.

4. If $b = 0$, Alice reveals the points R and $\phi(R')$. Bob accepts if they have order l_B^{eB} and generates the kernels of isogenies $E \to E_1$ and $E/\langle S\rangle \to E_2$, respectively.

5. If $b = 1$, Alice reveals the point $\psi(S)$. Bob accepts if it has order l_A^{eA} and generates the kernel of an isogeny $E_1 \to E_2$.

Key exchange

As above, this protocol is a variation of the Diffie-Hellman protocol using diagram (7.2). The main idea is to let Alice choose ϕ and Bob ψ. The method is similar to the ordinary case, but here ideal classes no longer commute thus making the diagram no commutative and so we need to give extra information in order to ensure that both parties arrive at the same common value.
Public parameters: A supersingular curve E_0 defined over $\mathbb{F}_{p^2}$. Bases $\{P_A, Q_A\}$ and $\{P_B, Q_B\}$ for $E_0[l_A^{e_A}]$ and $E_0[l_B^{e_B}]$.
Key generation: Alice chooses two random elements $m_A, n_A \in \mathbb{Z}$ not both divisible by l_A, and computes an isogeny $\phi_A : E_0 \to E_A$ with kernel $K_A := \langle [m_A]P_A + [n_A]Q_A\rangle$. Alice also computes the image $\{\phi_A(P_B), \phi_A(Q_B)\} \subset E_A$ and sends these to Bob together with E_A. Similarly, Bob selects two random elements $m_B, n_B \in_R \mathbb{Z}/l_B^{e_B}$ and computes an isogeny $\phi_B : E_0 \to E_B$ with kernel $K_B := \langle [m_B]P_B + [n_B]Q_B\rangle$, along with the points $\{\phi_B(P_A), \phi_B(Q_A)\}$ which are transfered to Alice together with E_B. Both Alice and Bob compute isogenies $\phi'_A : E_B \to E_{AB}$ and $\phi'_B : E_A \to E_{BA}$, respectively. The kernel of Alice's isogeny is $\langle [m_A]\phi_B(P_A) + [n_A]\phi_B(Q_A)\rangle$. Bob isogeny's kernel is $\langle [m_B]\phi_A(P_B) + [n_B]\phi_A(Q_B)\rangle$. Thus, Alice and Bob can use the $j-$invariant of

$$E_{AB} = \phi'_B(\phi_A(E_0)) = \phi'_A(\phi_B(E_0)) = E_0/\langle [m_A]P_A + [n_A]Q_A, [m_B]P_B + [n_B]Q_B\rangle,$$

to form a secret shared key.

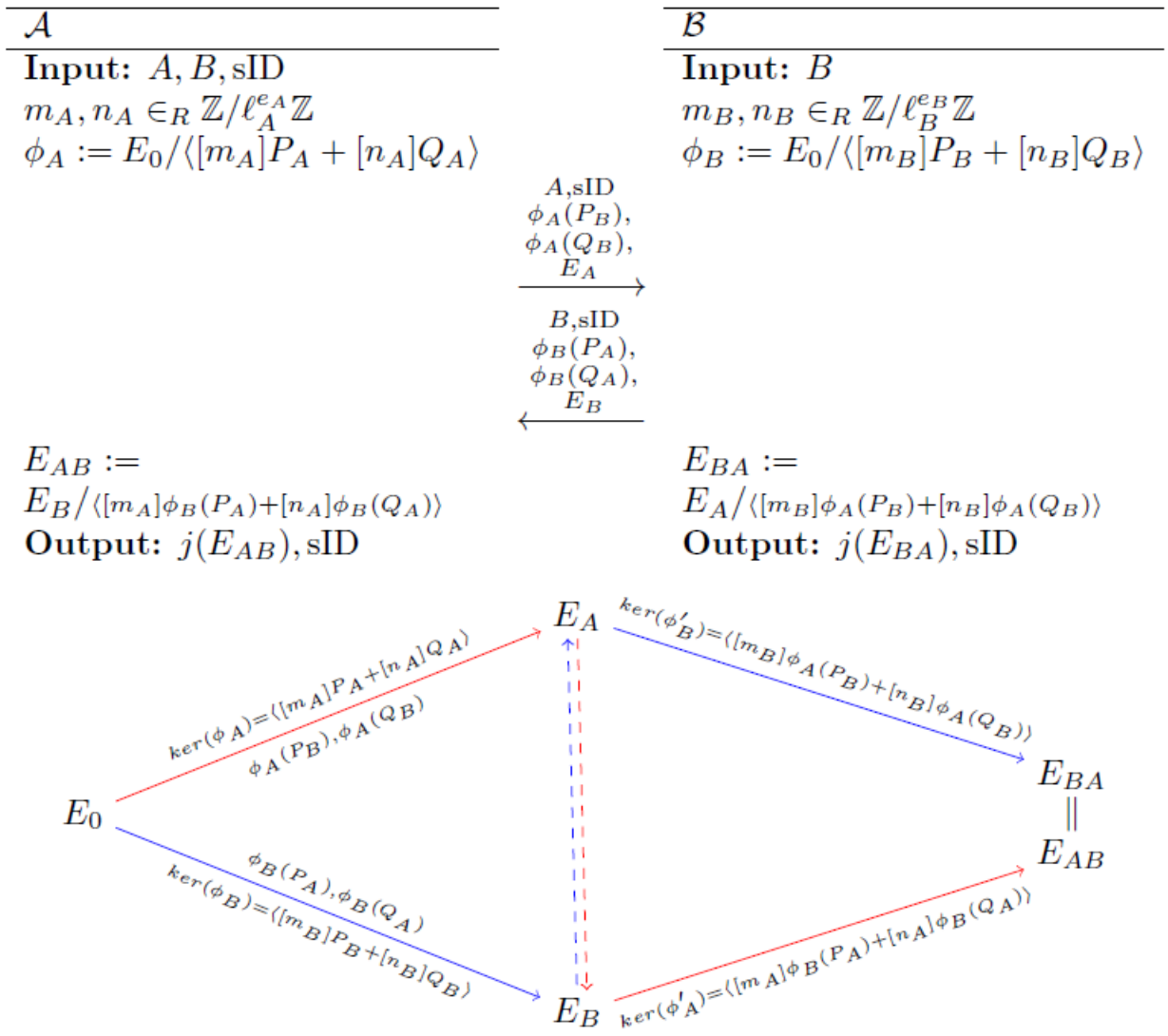


Figure 7.3: Key-exchange protocol using isogenies on supersingular curves [DFJP14].

The full protocol is given in Figure 7.3, where sID denotes the unique session identifier.

7.4.4 Public-key encryption

Analogous to ElGamal encryption following from Diffie-Hellman, this pubick-key cryptosystem is an adapation of the key-exhange protocol given in Fig.(7.1).
Setup: Choose $p = l_A^{e_A} l_B^{e_B} \cdot f \pm 1$, E_0, $\{P_A, Q_A\}$, $\{P_B, Q_B\}$. Let $\mathcal{H} = \{H_k : k \in K\}$ be a hash function family indexed by a finite set K, where each H_k is a function from $\mathbb{F}_{p^2}$ to the message space $\{0,1\}^\omega$.
Key generation: Choose two random elements $m_A, n_A \in_R \mathbb{Z}/l_A^{e_A}\mathbb{Z}$, not both divisible by l_A. Compute $E_A, \phi_A(P_B), \phi_A(Q_B)$ and choose a random element $k \in_R K$. The public key is the tuple $(E_A, \phi_A(P_B), \phi_A(Q_B), k)$ and the private key is (m_A, n_A, k).
Encryption: If you want to send a message $m \in \{0,1\}^\omega$ using a public key $(E_A, \phi_A(P_B), \phi_A(Q_B), k)$,

choose two random elements $m_B, n_B \in_R \mathbb{Z}/l_B^{e_B}\mathbb{Z}$, not both divisible by l_B, and compute

$$h = H_k(j(E_{AB})),$$
$$c = h \oplus m.$$

The ciphertext is $(E_B, \phi_B(P_A), \phi_B(Q_A), c)$.
Decryption: If you want to decipher the ciphertex $(E_B, \phi_B(P_A), \phi_B(Q_A), c)$ using your private key (m_A, n_A, k), compute the $j-$invariant $j(E_{AB})$ and set

$$h = H_k(j(E_{AB})),$$
$$m = h \oplus c.$$

Thus obtaining the plaingtext m.

7.4.5 Algorithmic aspects

In [DFJP14], the authors give a detailed description of the algorithms used to implement these last protocols. However, because many of these algorithms are based on the theory that we cannot cover, we are going to summarize the work done in this field and suggest specific bibliography for the interested reader.

The generation of the parameters follows from classical results. Given numbers $l_A^{e_A}$ and $l_B^{e_B}$, it is easy to test random values of f such that $p = l_A^{e_A} l_B^{e_B} \cdot f \pm 1$ is prime. Lagarias and Odlyzko [LO77] provide a sufficient lower bound for the density of these primes.
Broker [BCL08] gives a detailed way to find a supersingular curve E over $\mathbb{F}_{p^2}$ with $|E| = (p \pm 1)^2$. From E, using random walks on the isogeny graph we can go to a random supersingular curve E_0, or just take $E_0 = E$. In either case, E_0 has group structure $(\mathbb{Z}/(p \pm 1)\mathbb{Z})^2$. A basis $\{P_A, Q_A\}$ for $E_0[l_A^{e_A}]$ can be obtained choosing a random point $P \in_R E_0(\mathbb{F}_{p^2})$ and multiplying it by $(l_B^{e_B} \cdot f)^2$ to get a point P'. We repeat this proccess until we get a point $P_A = P'$ of order $l_A^{e_A}$. The same holds for getting a point Q_A. Finally you need to use the Weil pairing to see if P_A and Q_A are independent, if not, choose another Q_A [Aft11].
For the protocols requiring the computation of a kernel $\langle [m]P + [n]Q \rangle$ we need any generator of it. Without loss of generality, we can assume that m is invertible and use the generator $R' = P + [m^{-1}n]Q$. To compute R', we use a three-point ladder method based on Montgomery ladders [Mon87].

The idea behind algorithm (12) is that at each iteration, the registers A, B and C contain respectively the values $[x]Q, [x+1]Q$ and $P + [x]Q$ for x equal to the leftmost bits of $m^{-1}n$. The function $\text{dadd}(A, B, C)$ is a differential addition: it computes the sum $A + B$ knowing $C = A - B$.

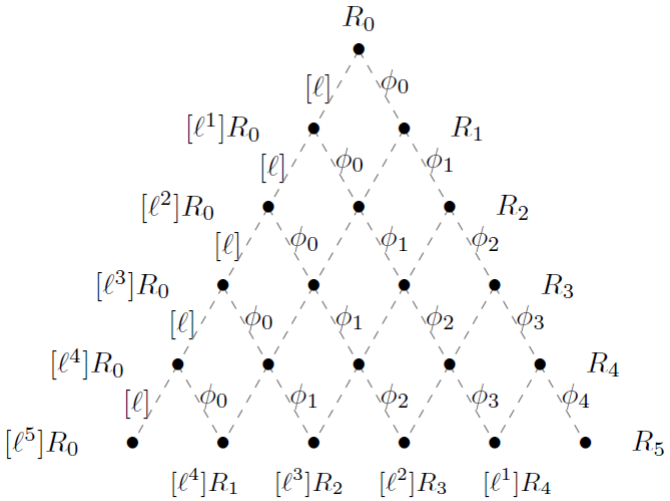


Figure 7.4: Computational structure of the construction of $\phi = \phi_{e-1} \circ \cdots \circ \phi_0$ [DFJP14].

Algorithm 12 Three-point ladder to compute $P + [t]Q$

INPUT: t, P, Q
OUTPUT: $C = P + [t]Q$
PROCEDURE:

Set $A = 0$, $B = Q$, $C = P$.
for i decreasing **from** $|t|$ **to** 1 **do**
  Let t_i be the i-th bit of t;
  if $t_i = 0$ **then**
    $A = 2A$, $B = \text{dadd}(A, B, Q)$, $C = \text{dadd}(A, C, P)$;
  else
    $A = \text{dadd}(A, B, Q)$, $B = 2B$, $C = \text{dadd}(B, C, Q, -P)$;
  end if
end for

The most difficult task around these protocols is the computation and evaluation of isogenies. Namely, given an elliptic curve E and a point R of order l^e, we need to compute the image curve $E/\langle R\rangle$ and to evaluate the isogeny $\phi : E \to E/\langle R\rangle$ at some points of E. In the supersingular case, the degree of ϕ is smooth, so we decompose it as a chain of $l-$isogenies. Set $E_0 = E$, $R_0 = R$ and, for $0 \leq i < e$, let

$$E_{i+1} = E_i/(l^{e-i-1}R_i), \quad \phi_i : E_i \to E_{i+1}, \quad R_{i+1} = \phi_i(R_i).$$

Then $E/\langle R\rangle = E_e$ and $\phi = \phi_{e-1} \circ \cdots \circ \phi_0$.

Both, the curve E_{i+1} and the isogeny ϕ_i can be computed using Velu's formulas once the l-torsion subgroup $\langle R_i\rangle$ of E_i is known. The best strategy to do this is given by Figure (7.4) for the case $e = 6$. Here, bullets represent points, with points on the same horizontal level having the same order, and points on the same left diagonal belonging to the same curve.

Dashed edges are directed top-bottom; leftward edges represent multiplication by l, and rightward edges represent an evaluation of an l-isogeny. The final goal is to compute all the points in the bottom line. The best strategy to do this depends on the combinatorial structure of the problem rather than its number-theoretic nature. We are not going to discuss more of this, but it is important to mention that given the point $[l^{e-i-1}]R_i$, we can compute the kernel of ϕ_i using $\mathcal{O}(l)$ point additions. And then, continue using Velu's formulas for the next steps.

Since many aspects of these protocols are computationally hard, it is important to use models for elliptic curves that offer the fastest formulas for doubling, addition, isogeny computation, and evaluation, etc. The curves that we use in these cryptosystems are isomorphic to twisted Edward and Montgomery curves, [BBJ+08]. Twisted Edwards curves, $E_{a,d} : ax^2 + y^2 = 1 + dx^2y^2$, have very efficient addition and doubling formulas, [BBJ+08]. Montgomery curves, $M_{B,A} : By^2 = x^3 + Ax^2 + x$, are also good for doubling points, [Mon87], and there are efficient formulas for isogeny evaluation of degree 2, 3 and 4 over them, [DFJP14].

7.4.6 Complexity assumptions

As before, the security of the protocols relies on the difficulty of the following problems, [DFJP14]:

Decision Supersingular Isogeny (DSSI) problem. Let E_A be another supersingular curve defined over $\mathbb{F}_{p^2}$. Decide whether E_A is $l_A^{e_A}$-isogenous to E_0.

Computational Supersingular Isogeny (CSSI) problem. Given E_A and the images $\phi_A(P_B), \phi_A(Q_B)$, find a generator R_A for the kernel of ϕ_A.

Supersingular Computational Diffie-Hellman (SSCDH) problem. Given the curves E_A, E_B and the points $\phi_A(P_B), \phi_A(Q_B), \phi_B(P_A), \phi_B(P_A)$, find the j-invariant of $E_0/\langle [m_A]P_A + [n_A]Q_A, [m_B]P_B + [n_B]Q_B\rangle$.

Supersingular Decision Diffie-Hellman (SSDDH) problem. Given a tuple sampled with probability 1/2 from one of the following two distributions:

- $(E_A, E_B, \phi_A(P_B), \phi_A(Q_B), \phi_B(P_A), \phi_B(Q_A), E_{AB})$, where

$$E_{AB} \equiv E_0/\langle [m_A]P_A + [n_A]Q_A, [m_B]P_B + [n_B]Q_B\rangle,$$

- $(E_A, E_B, \phi_A(P_B), \phi_A(Q_B), \phi_B(P_A), \phi_B(Q_A), E_C)$, where

$$E_C \equiv E_0/\langle [m'_A]P_A + [n'_A]Q_A, [m'_B]P_B + [n'_B]Q_B\rangle,$$

  where m'_A, n'_A, m'_B, n'_B are chosen following the same conditions of the original ones,

determine from which distribution the tuple is sampled.

All of these problems are conjectured to be computationally infeasible. It is assumed that DSSI and CSSI are equivalent to SSDDH. DSSP can be reduced to DSSI and it is assumed

that is easier to solve, [DFJP14].

Most of the difficulty comes from the problem of computing an isogeny between two supersingular curves. The fastest known algorithm for this takes $\mathcal{O}(\sqrt{p}\log^2(p))$ time, [LG09]. However, the curves used in the protocols have a smooth known degree and there is a lack of literature addressing the security of the isogenies of the special form that protocols use. For example, Sankar [San15] gives a quantum algorithm that breaks $SSDDH$ but has exponential-complexity in general and subexponential complexity in a particular case that can be easily avoided. However, this is not a big threat to the schemes and even on a quantum computer, there are no efficient algorithms to solve these problems.

Chapter 8

Conclusions and Recommendations

8.1 Conclusions

We have done an extensive review of cryptography based on elliptic curves and isogenies. We have presented all the basic tools for understanding a subject that requires a great deal of background and advanced jargon not seen in regular undergraduate courses. In a non-traditional way, we have included small introductions to complexity, cryptography, and quantum computing for a better understanding of the protocols and their attacks. We have shown that cryptography based on elliptic curves is useful in our classical system and in particular, it works well for embedded systems. However, if quantum computing is achieved on a large scale, the family of protocols based on elliptic curves will be rendered useless. Hence, the need for isogenies-based cryptography. Isogenies between ordinary curves are vulnerable to quantum attacks, but those based on supersingular curves offer a new family of conjecturally quantum-resistant public-key cryptographic protocols.

8.2 Recommendations

Most of the members of this last family rely on the factorization of isogenies and the use of isogeny graphs. This way to do things avoids the number theoretic aspects of isogenies and elliptic curves to give attention to the structural properties of their isogeny graphs. This is understandable since most of the current work in this area is done by people with computational sciences and discrete mathematics as their background, where graphs are very important. However, we think that we can explore the algebraic and geometric aspects of these protocols to achieve interesting results. Since elliptic curves are both algebraic curves and abelian groups; and isogenies are just morphisms between them, we believe that there are algorithms coming from Computational Algebra that could result useful. As an example, we will like to consider Ritt-Wu's Decomposition Algorithm to decompose the kernel characterizing an isogeny into its prime components. Ritt-Wu's algorithm consists of the decomposition of an algebraic set into an union of irreducible varieties and the transformation of its ideal into an ascending chain of polynomials corresponding to elimination ideals, [XCG09]. For example, suppose that we are given elliptic curves E and E/G, where G is a finite ideal. To compute the isogeny $\phi : E \to E/G$, we can decompose G into its prime components $G_1, G_2, \ldots$ and then use Velu's

formulas in each of the quotiented rings $E/G_1, E/G_2, \ldots$ to obtain their respective isogenies and see, if the union of their images, is the same as the image of $\phi(E)$. If thise procedure results in a feasible approach to obtain an isogeny, then this could give more insight into elliptic curves, isogenies, and the protocols presented here.

Nevertheless, we consider that it is worthwhile to continue working on cryptographic schemes based on isogenies between supersingular elliptic curves since they provide protocols that improve upon many other quantum-resistant schemes. However, because of the advanced amount of mathematics needed to understand these protocols, we believe it is important to make them more accessible, such that, more mathematicians can work in this area.

Appendix A

Fields and Galois Theory

This appendix presents concepts from Field Theory and Galois Theory used throughout the text. It is based on Milne's notes, [Mil20]. This appendix is not intended to replace a textbook and should be accompanied by historical sources such as [Gal10], [MM96], and [Edw97]. We do not give proofs for theorems or statements given here, for the interested reader, we refer to Milne's notes.

A.1 Basic Definitions

A.1.1 Rings

A **ring** is a set R with two binary operations $+$ and $\cdot$ such that

(a) $(R, +)$ is a commutative group;

(b) $\cdot$ is associative, and there exists an identity element 1_R such that $a \cdot 1_R = a = 1_R \cdot a$ for all $a \in R$;

(c) the distributative law holds: $\forall a, b, c \in R$,

$$(a + b) \cdot c = a \cdot c + b \cdot c,$$
$$a \cdot (b + c) = a \cdot b + a \cdot c.$$

We usually omit "$\cdot$" and write 1 for 1_R when this causes no confusion.

A **subring** of a ring R is a subset S that is also a ring.

A **homorphism of rings** $\alpha : R \to R'$ is a map such that

$$\alpha(a + b) = \alpha(a) + \alpha(b), \quad \alpha(ab) = \alpha(a)\alpha(b), \quad \alpha(1_R) = 1'_R$$

for all $a, b \in R$. A ring R is said to be **commutative** if multiplication is commutative. A commutative ring is said to be an **integral domain** if $1_R \neq 0$ and the cancellation law holds for multiplication,

$$ab = ac, a \neq 0, \text{ implies } b = c.$$

An **ideal** I in a commutative ring R is a subgroup of $(R,+)$ that is closed under multiplication by elements of R,

$$r \in R, a \in I, \to ra \in I.$$

The ideal generated by elements $a_1, \ldots, a_n$ is denoted by $(a_1, \ldots, a_n)$.

A.1.2 Fields

Definition A.1.1. A **field** is a set F with two composition laws $+$ and $\cdot$ such that

(a) $(F,+)$ is a commutative group;

(b) $(F-\{0\},\cdot)$ is a commutative group;

(c) the distributive law holds.

Thus a field is a nonzero commutative ring such that every nonzero element has an inverse. A **subfield** S of a field F is a subset of F which is also a field.

Example A.1. *The following are fields:* $\mathbb{Q}, \mathbb{R}, \mathbb{C}, \mathbb{F}_p = \mathbb{Z}/p\mathbb{Z}$ *(p prime).*

A **homomorphism of fields** is a homomorphism of rings. Such a homomorphism is always inyective.

A.1.3 The characteristic of a field

It is easy to see that the map

$$\mathbb{Z} \to F, \quad n \mapsto n \cdot 1_F := 1_F + 1_F + \cdots + 1_F \quad (n \text{ copies of } 1_F),$$

is a homomorphism of rings. For example, $m \cdot 1_F + n \cdot 1_F = (m+n) \cdot 1_F$ because of the associativity of addition. Therefore its kernel is an ideal in $\mathbb{Z}$.

Case 1: The kernel of the map is (0), so that

$$n \cdot 1_F = 0 \in F \Rightarrow n = 0 \in \mathbb{Z}.$$

Nonzero integers map to invertible elements of F under $n \mapsto n \cdot 1 - F : \mathbb{Z} \to F$, and so this map extends to a homomorphism

$$\frac{m}{n} \mapsto (m \cdot 1_F)(n \cdot 1 - F)^{-1} : \mathbb{Q} \hookrightarrow F.$$

In this case, F contains a copy of $\mathbb{Q}$, and we say that it has **characterisitc zero**, $\text{char}(F) = 0$.

Case 2: The kernel of the map is $\neq (0)$, so that $n \cdot 1_F = 0$ for some $n \neq 0$. The smallest positive such n will be a prime p, and p generates the kernel. Thus, the map $n \mapsto n \cdot 1_F : \mathbb{Z} \to F$ defines an isomorphism from $\mathbb{Z}/p\mathbb{Z}$ onto the subring

$$\{m \cdot 1_F | m \in \mathbb{Z}\}$$

of F. In this case, F contains a copy of $\mathbb{F}_p$, and we say that it has **characteristic** p, $\text{char}(F) = p$.

A field isomorphic to one of the fields $\mathbb{F}_2, \mathbb{F}_3, \mathbb{F}_5, \ldots, \mathbb{Q}$ is called a **prime field**. Every field contains exactly one prime field (as a subfield).

More generally, a commutative ring R is said to have **characteristic** p (resp. 0) if it contains a prime field (as a subring) of characteristic p (resp. 0). Then the prime field is unique and, by definition, contains 1_R. When R has characteristic p, it is true that

$$(a+b)^p = a^p + b^p \text{ for all } a, b \in R,$$

and so the map $a \mapsto a^p : R \to R$ is a homomorphism of rings. It is called the **Frobenius endomorphism** of R. The map $a \mapsto a^{p^n} : R \to R$, $n \geq 1$, is the composite of n copies of the Frobenius endomorphism, and so it also is a homomorphism. Therefore,

$$(a_1 + \cdots + a_m)^{p^n} = a_1^{p^n} + \cdots a_m^{p^n}$$

for all $a_i \in R$.

When F is a field, the Frobenius endomorphism is injective, and hence an automorphism if F is finite.

The **characteristic exponent** of a field F is 1 if F has characteristic 0, and p if F has characteristic $p \neq 0$. Thus, if q is the characteristic exponent of F and $n \geq 1$, then $x \mapsto x^{q^n}$ is an isomorphism of F onto a subfield of F (denotes F^{q^n}).

A.1.4 Polynomial rings

Let F be a field.

The ring $F[X]$ of polynomials in the symbol (or "indeterminate" or "variable") X with coefficients in F is an F-vector space with basis $1, X, \ldots, X^n, \ldots$, and with the multiplication

$$\left(\sum_i a_i X i i\right)\left(\sum_j b_j X^j\right) = \sum_k \left(\sum_{i+j=k} a_i b_j\right) X^k.$$

Let I be a nonzero ideal in $F[X]$, and let f be a nonzero polynomial of least degree in I; then $I = (f)$. When we choose f to be **monic**, i.e., to have leading coefficient one, it is uniquely determined by I. Thus, there is a one-to-one correspondence between the nonzero ideals of $F[X]$ and the monic polynomials in $F[X]$. The prime ideals correspond to the irreducible monic polynomials.

As $F[X]$ is an integral domain, we can form its field of fractions $F(X)$. Its elements are quotients f/g, f and g polynomials, $g \neq 0$.

A.1.5 Extensions

Let F be a field. A field containing F is called an **extension** of F. An extension E of F is, in particular, an F-vector space, whose dimension is called the **degree** of E over F. It is denoted

by $[E : F]$. An extension is said to be **finite** if its degree is finite, and quadratic, cubic, etc., if it is of degree 2,3, etc.

When E and E' are extensions of F, an F-**homomorphism** $E \to E'$ is a homomorphism $\varphi : E \to E'$ such that $\varphi(c) = c$ for all $c \in F$.

Example A.2. *(a) The field of complex numbers $\mathbb{C}$ has degree 2 over $\mathbb{R}$ (basis $\{0, i\}$). (b) The field of real numbers $\mathbb{R}$ has infinite degree over $\mathbb{Q}$. (c) The field $F(X)$ has infinite degree over F; in fact, even its subspace $F[X]$ has infinite dimension over F (basis $1, X, X^2, \ldots$).*

A.1.6 The subring generated by a subset

An intersection of subrings of a ring is again a ring. Let F be a subfield of a field E, and let S be a subset of E. The intersection of all the subrings of E containing F and S is the smallest subring of E containing both F and S. We call it the subring of E **generated by** F **and** S, and we denote it by $F[S]$. When $S = \{\alpha_1, \ldots, \alpha_n\}$, we write $F[\alpha_1, \ldots, \alpha_n]$ for $F[S]$. For example, $\mathbb{C} = \mathbb{R}[\sqrt{-1}]$.

A.1.7 The subfield generated by a subset

An intersection of subfields of a field is again a field. Let F be a subfield of a field E, and let S be a subset of E. The intersection of all the subfields of E containing F and S is the smallest subfield of E containing both F and S. We call it the subfield of E **generated by** F **and** S, and we denote it $F(S)$. It is the field of fractions of $F[S]$ in E because this is a subfield of E containing F and S and contained in every other such field. When $S = \{\alpha_1, \ldots, \alpha_n\}$, we write $F(\alpha_1, \ldots, \alpha_n)$ for $F(S)$. Thus, $F[\alpha_1, \ldots, \alpha_n]$ consists of all elements of E that can be expressed as polynomials in the α_i with coefficients in F, and $F(\alpha_1, \ldots, \alpha_n)$ consists of all elements of E that can be expressed as a quotient of two such polynomials.

A.2 Galois Theory

A.2.1 Groups of automorphisms of fields

Let F be a field, and let E and E' be fields containing F. Recall that n F-homomorphism is a homomorphism $\varphi : E \to E'$ such that $\varphi(a) = a$ for all $a \in F$. Thus an F-homomorphism φ maps a polynomial

$$\sum a_{i_1 \cdots i_m} \alpha_1^{i_1} \cdots \alpha_m^{i_m}, \quad a_{i_1 \cdots i_m} \in F, \quad \alpha_i \in E,$$

to

$$\sum a_{i_1 \cdots i_m} \varphi(\alpha_1)^{i_1} \cdots \varphi(\alpha_m)^{i_m}.$$

An F-**isomorphism** is a bijective F-homomorphism. An F-isomorphism $E \to E$ is called an F-**automorphism** of E. The F-automorphism of E form a group, which we denote $\mathrm{Aut}(E/F)$.

Example A.3. *(a) There are two obvious automorphisms of $\mathbb{C}$, namely the identity map and complex conjugation. (b) Let $E = \mathbb{C}(X)$. A $\mathbb{C}$-automorphism of E sends X to another generator*

of E over $\mathbb{C}$.
$Aut(E/\mathbb{C})$ consists of the maps $f(X) \mapsto f\left(\frac{aX+b}{cX+d}\right)$, $ad-bc \neq 0$, and so

$$Aut(E/\mathbb{C}) \simeq PGL_2(\mathbb{C}),$$

the group of invertible 2×2 matrices with complex coefficients module its centre.

A.2.2 Splitting fields

Let f be a polynomial with coefficients in F. A field E containing F is said to **split** f if f splits in $E[X]$, i.e.,

$$f(X) = a\Pi_{i=1}^{m}(X - \alpha_i) \text{ with all } \alpha_i \in E.$$

If E splits f and is generated by the roots of f,

$$E = F[\alpha_1, \ldots, \alpha_m],$$

then it is called a **splitting** or **root field** for f.

A.2.3 Algebraic extension and Algebraic closure

An element x of a field extension E/F is algebraic over F if it is a root of a nonzero polynomial with coefficients in F. For example, $\sqrt{2}$ is algebraic over the rational numbers, because it is a root of $x^2 - 2$. If an element x of E is algebraic over F, the monic polynomial of lowest degree that has x as a root is called the minimal polynomial of x. This minimal polynomial is irreducible over F.

An algebraic extension E/F is an extension such that every element of E is algebraic over F. For example, $\mathbb{Q}(\sqrt{2}, \sqrt{3})$ is an algebraic extension of $\mathbb{Q}$.

Definition A.2.1. An **algebraic closure** of a field F, denoted $\overline{F}$, is an algebraic extension of F that is algebraically closed, i.e., every non-constant polynomial in $F[X]$ has a root in $\overline{F}$.

A.2.4 Separable, normal, and Galois extensions

Definition A.2.2. An algebraic extension E/F is **separable** if the minimal polynomial of every element of E is separable; otherwise, it is **inseparable**.

For example, the extension $\mathbb{F}_p(T)$ of $\mathbb{F}_p(T^p)$ is inseparable extension because T has minimal polynomial $X^p - T^p$.

Definition A.2.3. An extension E/F is **normal** if it is algebraic and the minimal polynomial of every element of E splits in $E[X]$.

In other words, an algebraic extension E/F is normal if, and only if, every irreducible polynomial $f \in F[X]$ having at least one root in E splits in $E[X]$.

E/F is separable and normal if, and only if, the minimal polynomial of every element α of E has $[F[\alpha] : F]$ distinct roots in E.

Theorem A.1. *For an extension E/F, the following statements are equivalent:*

(a) E is the splitting field of a separable polynomial $f \in F[X]$;

(b) E is finite over F and $F = E^{Aut(E/F)}$;

(c) $F = E^G$ for some finite group G of automorphisms of E;

(d) E is normal, separable, and finite over F.

Definition A.2.4. An extension E/F of fields is **Galois** if it satisfies the equivalent conditions of the above Theorem. When E/F is Galois, $\mathrm{Aut}(E/F)$ is called the **Galois group** of E over F, and it is denoted by $\mathrm{Gal}(E/F)$.

Bibliography

[Adl78] R.L. Rivest; A. Shamir; L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Cpommun. ACM*, 21:120–126, 1978.

[Aft11] Alex Edward Aftuck. The weil pairing on elliptic curves and its cryptographic applications. 2011.

[Age09]] National Security Agency. The case for elliptic curve cryptography. *online*, 2009.

[Bac90] Eric Bach. Explicit bounds for primality testing and related problems. *Mathematics of Computation*, 55(191):355–380, 1990.

[BBJ+08] Daniel J Bernstein, Peter Birkner, Marc Joye, Tanja Lange, and Christiane Peters. Twisted edwards curves. In *International Conference on Cryptology in Africa*, pages 389–405. Springer, 2008.

[BBS06] I. Blake, Ian F. Blake, and G. Seroussi. *Elliptic Curves in Cryptography*. Cambridge University Press, 2006.

[BCC08] Maria Wlleda Baldoni, Ciro Ciliberto, and Giulia Maria Pacentini Cattaneo. *Elementary Number Theory, Cryptography and Codes*. Springer, 2008.

[BCL08] Reinier Bröker, Denis Charles, and Kristin Lauter. Evaluating large degree isogenies and applications to pairing based cryptography. In *International Conference on Pairing-Based Cryptography*, pages 100–112. Springer, 2008.

[BMSS08] Alin Bostan, François Morain, Bruno Salvy, and Éric Schost. Fast algorithms for computing isogenies between elliptic curves. *Mathematics of Computation*, 77(263):1755–1778, 2008.

[BYMH13] Mohsen Bafandehkar, Sharifah Md Yasin, Ramlan Mahmod, and Zurina Mohd Hanapi. Comparison of ecc and rsa algorithm in resource constrained devices. *IEEE*, 13, 2013.

[CCJ+16] Lily Chen, Lily Chen, Stephen Jordan, Yi-Kai Liu, Dustin Moody, Rene Peralta, Ray Perlner, and Daniel Smith-Tone. *Report on post-quantum cryptography*, volume 12. US Department of Commerce, National Institute of Standards and Technology, 2016.

[CFL+19] Anamaria Costache, Brooke Feigon, Kristin Lauter, Maike Massierer, and Anna Puskás. Ramanujan graphs in cryptography. In *Research Directions in Number Theory*, pages 1–40. Springer, 2019.

[CJS14] Andrew Childs, David Jao, and Vladimir Soukharev. Constructing elliptic curve isogenies in quantum subexponential time. *Journal of Mathematical Cryptology*, 8(1):1–29, 2014.

[Coh06] Henri Cohen. *Handbook of elliptic and hyperelliptic curve cryptography*. Chapman & Hall/CRC, Boca Raton, 2006.

[Cos19] Craig Costello. Supersingular isogeny key exchange for beginners. In *International Conference on Selected Areas in Cryptography*, pages 21–50. Springer, 2019.

[Cou06] Jean Marc Couveignes. Hard homogeneous spaces. 2006.

[DFJP14] Luca De Feo, David Jao, and Jérôme Plût. Towards quantum-resistant cryptosystems from supersingular elliptic curve isogenies. *Journal of Mathematical Cryptology*, 8(3):209–247, 2014.

[DH76] Whitfield Diffie and Martin Hellman. New directions in cryptography. *IEEE transactions on Information Theory*, 22(6):644–654, 1976.

[Edw97] H.M. Edwards. *Galois Theory*. Graduate Texts in Mathematics. Springer New York, 1997.

[EM17] Thomas W. Edgar and David O. Manz. Chapter 2 - science and cyber security. In Thomas W. Edgar and David O. Manz, editors, *Research Methods for Cyber Security*, pages 33 – 62. Syngress, 2017.

[Ent14] Entrust. Zero to ecc in 30 minutes. *online*, 2014.

[For09] Lance Fortnow. The status of the p versus np problem. *Communications of the ACM*, 52(9):78–86, 2009.

[Ful08] W. Fulton. *Algebraic Curves: An Introduction to Algebraic Geometry*. 2008.

[Gal01] Steven D Galbraith. Supersingular curves in cryptography. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 495–513. Springer, 2001.

[Gal10] J. Gallian. *Contemporary Abstract Algebra*. Cencage Learning, USA, 2010.

[GJG79] M. R. Garey, D. S. Johnson, and Michael R. Garey. *Computers and Intractability: A Guide to the Theory of Np-Completeness*. W H FREEMAN WORTH PUB 3PL, 1979.

[Gle05] Ian Glendinning. The bloch sphere. In *QIA Meeting TechGate*, 2005.

[GMW91] Oded Goldreich, Silvio Micali, and Avi Wigderson. Proofs that yield nothing but their validity or all languages in np have zero-knowledge proof systems. *Journal of the ACM (JACM)*, 38(3):690–728, 1991.

[Gol03] Oded Goldreich. *Foundations of cryptography*. Cambridge University Press, Cambridge, U.K. New York, NY, 2003.

[Gol15] Oded Goldreich. *Computational Complexity*. Cambridge University Press, 2015.

[GS13] Steven Galbraith and Anton Stolbunov. Improved algorithm for the isogeny problem for ordinary elliptic curves. *Applicable Algebra in Engineering, Communication and Computing*, 24(2):107–131, mar 2013.

[Hel02] Martin E Hellman. An overview of public key cryptography. *IEEE Communications Magazine*, 40(5):42–49, 2002.

[HMV04] Darrel Hankerson, Alfred J. Menezes, and Scott Vanstone. *Guide to elliptic curve cryptography*. Springer, 1st edition, 2004.

[Jan96] Gerald Janusz. *Algebraic number fields*. American Mathematical Society, Providence, R.I, 1996.

[JS10] David Jao and Vladimir Soukharev. A subexponential algorithm for evaluating large degree isogenies. In *Lecture Notes in Computer Science*, pages 219–233. Springer Berlin Heidelberg, 2010.

[KM00] Noboru Kunihiro and Kenji Members. Two discrete log algorithms for superanomalous elliptic curves and their applications. *IEICE Transactions on Fundamentals of Electronics Communications and Computer Sciences*, E83A, 04 2000.

[Kob87] Neal Koblitz. Elliptic curve cryptosystems. *Mathematics of computation*, 48(177):203–209, 1987.

[Koh96] David Russell Kohel. *Endomorphism rings of elliptic curves over finite fields*. PhD thesis, University of California, Berkeley, 1996.

[KS19] Richard Klima and Neil Sigmon. *Cryptology, Classical and modern*. CRC Press, 2 edition, 2019.

[Kum06] S. Kumar. Elliptic curve cryptography for constrained devices. *Dissertation, Rurh-University Bochum*, 2006.

[Kup05] Greg Kuperberg. A subexponential-time quantum algorithm for the dihedral hidden subgroup problem. *SIAM Journal on Computing*, 35(1):170–188, 2005.

[Lau04] K Lauter. The advantages of elliptic curve cryptography for wireless security. *IEEE Wireless Comunications*, pages 62–67, 2004.

[Law08] C Washington Lawrence. *Elliptic Curves: number theory and cryptography*. CRC press, 2008.

[LG09] Denis Xavier Charles; Kritin E. Lauter and Eyal Z. Goren. Cryptographic has functions from expender graphs. *Journal of Cryptography*, 22:93–113, 2009.

[LMS+04] Florian Luca, David Jose Mireles, Igor E Shparlinski, et al. Mov attack in various subgroups on elliptic curves. *Illinois Journal of Mathematics*, 48(3):1041–1052, 2004.

[LO77] Jerey C. Lagarias and Andrew M. Odlyzko. *Effective versions of the Chebotarev density theorem.* Academic Press, London, 1977.

[MAN10] Isaac L. Chuang Michael A. Nielsen. *Quantum Computation and Quantum Information.* Cambridge University Pr., 2010.

[Mes86] Jean-Francois Mestre. La methode des graphes. examples et applications. In *Proceedings of the international conference on class numbers and fundamental unis of algebraic number fields.* Katata, 1986.

[Mil85] Victor S. Miller. Use of elliptic curves in cryptography. *Advances in Cryptology CRYPTO85 Proceedings*, pages 417–426, 1985.

[Mil20] James S. Milne. Fields and galois theory (v4.61), 2020. Available at www.jmilne.org/math/.

[MM96] P. Morandi and P.J. Morandi. *Field and Galois Theory.* Graduate Texts in Mathematics. Springer, 1996.

[Mon87] Peter L. Montgomery. Speeding the pollard and elliptic curve methods for factorization. *Mathematics of Computation*, 48(177):243–264, 1987.

[Mum08] David Mumford. *Abelian varieties.* Published for the Tata Institute of Fundamental Research By Hindustan Book Agency International distribution by American Mathematical Society, Mumbai New Delhi, 2008.

[NF17] Shireen Nisha and Mohammed Farik. Rsa public key cryptography algorithm – a review. *International Journal of Scientific and Technology Research*, 6:187–191, 07 2017.

[Ols72] Loren D Olson. An elementary proof that elliptic curves are abelian varieties. *Preprint series: Pure mathematics http://urn. nb. no/URN: NBN: no-8076*, 1972.

[oM] Encyclopedia of Mathematics. Decision problem.

[Pap94] Christos Papadimitriou. *Computational complexity.* Addison-Wesley, Reading, Mass, 1994.

[PCZH19] Cong Peng, Jianhua Chen, Sherali Zeadally, and Debiao He. Isogeny-based cryptography: A promising post-quantum technique. *IT Professional*, 21(6):27–32, nov 2019.

[PH06] S. Pohlig and M. Hellman. An improved algorithm for computing logarithms over gf(p) and its cryptographic significance (corresp.). *IEEE Trans. Inf. Theor.*, 24(1):106–110, September 2006.

[PKS09] M. Paryasto, S. Kuspriyanto, Sutikno, and A Sasongko. Issues in elliptic curve cryptography implementation. *Internetworking Indonesia Journal*, 1(1):29–33, 2009.

[Pol00] J. M. Pollard. Kangaroos, monopoly and discrete logarithms. *Journal of Cryptology*, 13(4):437–447, aug 2000.

[Pom87] Carl Pomerance. Fast, rigorous and discrete logarithm algorithms. *Discrete algorithms and complexity*, pages 119–143, 1987.

[PP10] Christof Paar and Jan Pelzl. *Understanding Cryptography.* Springer, 2010.

[PS88] Alexander Lubotzky; Ralph Phillips and Peter Sarnak. Ramanujan graphs. *Combinatorics*, 8(3)(8):261–277, 1988.

[PZ03] John Proos and Christof Zalka. Shor's discrete logarithm quantum algorithm for elliptic curves. *Quantum Information and Computation*, 3, 02 2003.

[Rey17] Omar Reyad. Text message encoding based on elliptic curve cryptography and a mapping methodology. *Information Sciences Letters*, 7(1):7–11, 2017.

[RNSL17] Martin Roetteler, Michael Naehrig, Krysta M. Svore, and Kristin Lauter. Quantum resource estimates for computing elliptic curve discrete logarithms. In *Advances in Cryptology – ASIACRYPT 2017*, pages 241–270. Springer International Publishing, 2017.

[Rob11] Matthew J. B. Robshaw. *One-Way Function*, pages 887–888. Springer US, Boston, MA, 2011.

[RS06] A. Rostovtsev and Anton Stolbunov. Public-key cryptosystem based on isogenies. *IACR Cryptol. ePrint Arch.*, 2006:145, 2006.

[San15] Anirudh Sankar. Classical and quantum algorithms for isogeny-based cryptography. Master's thesis, University of Waterloo, 2015.

[Sch95] René Schoof. Counting points on elliptic curves over finite fields. *Journal de théorie des nombres de Bordeaux*, 7(1):219–254, 1995.

[Sha04] Nils Gura; Arun Patel; Arvinderpal Wander; Hans Eberle; Scheueling Chang Shantz. Comparing elliptic curve cryptography and rsa on 8-bits cpus. *Cryptographic hardware and embedded systems-CHES 2004*, pages 119–132, 2004.

[Sha13] I. R. Shafarevich. *Basic algebraic geometry.* Springer, Berlin, 2013.

[Sho99] Peter W Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM review*, 41(2):303–332, 1999.

[Shu09] Daniel Shumow. Isogenies of elliptic curves: a computational approach. *arXiv preprint arXiv:0910.5370*, 2009.

[Sil94] Joseph Silverman. *Advanced topics in the arithmetic of elliptic curves.* Springer-Verlag, New York, 1994.

[Sil09] J.H. Silverman. *The arithmetic of elliptic curves*, volume 106. Springer, 2009.

[Sma16] Nigel Smart. *Cryptography Made Simple.* Springer, 2016.

[SS03] Horst G. Zimmer Susanne Schmitt. *Elliptic Curves: A Computational Approach.* De Gruyter, 2003.

[Sto12] Anton Stolbunov. Cryptographic schemes based on isogenies. 2012.

[Sut15] Andrew Sutheland. Lecture notes for elliptic curves 18.783. 2015.

[Tat66] John Tate. Endomorphisms of abelian varieties over finite fields. *Inventiones mathematicae*, 2(2):134–144, 1966.

[Til11] Henk Tilborg. *Encyclopedia of cryptography and security.* Springer, New York, 2011.

[TQ19] Swamit S Tannu and Moinuddin K Qureshi. Not all qubits are created equal: a case for variability-aware policies for nisq-era quantum computers. In *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 987–999, 2019.

[TY00] N. Torri and K. Yokoyama. Elliptic curve cryptosystem. *FUJITSU. Sci. Tech. J.*, 36(2):140–146, 2000.

[Vé71] Jacques Vélu. Isogénies entre courbes elliptiques. *CR Acad. Sci. Paris Sér. AB*, 273:A238–A241, 1971.

[Woh] Jeremy Wohlwend. Elliptic curve cryptography: Pre and post quantum. Technical report.

[Won16] David Wong. How to backdoor diffie-hellman. *IACR Cryptol. ePrint Arch.*, 2016:644, 2016.

[WVMN19] Robert Wille, Rod Van Meter, and Yehuda Naveh. Ibm's qiskit tool chain: Working with and developing for real quantum computers. In *2019 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 1234–1240. IEEE, 2019.

[WXWM15] Xiaoyun Wang, Guangwu Xu, Mingqiang Wang, and Xianmeng Meng. *Mathematical Foundations of Public Key Cryptography.* Taylor and Francis Ltd., 2015.

[XCG09] Gao Xiaoshan, Yuan Chunming, and Zhang Guilin. Ritt-wu's characteristic set method for ordinary difference polynomial systems with arbitrary ordering. *Acta Mathematica Scientia*, 29(4):1063–1080, jul 2009.