TÉCNICO LISBOA

On trapdoor Kolmogorov one-way functions and elliptic curves cryptography

Francisco Mantero Morais Pavão Martins

Thesis to obtain the Master of Science Degree in

Mathematics and Applications

Examination Committee

Chairperson:	Prof. Maria Cristina Sales Viana Serôdio Sernadas
Supervisor:	Prof. Paulo Alexandre Carreira Mateus
Co-supervisor:	Prof. André Nuno Carvalho Souto
Members of the Committee:	Prof. Carlos Manuel Costa Lourenço Caleiro

June 2014

Acknowledgments

I would like to express my grattitude to my supervisor, Paulo Mateus, for presenting me an interesting path to research and for guidance through the elaboration of this dissertation.
I also want to thank my co-supervisor, André Souto for his outstanding dedication and support when help was needed.
I want to thank Professor Klaus Altmann for guiding me in the first steps of this work and to motivate me to learn more about algebraic geometry.
I want to take this opportunity to thank all the people in the Department of Mathematics of IST, and in the Computer Science Section, for their support. In special, I wish to thank my classmates and former classmates that have always motivated and supported me through out this path.
I am also grateful to all my friends who provided me unconditional support, and for giving me a space to relax and maintain my health of mind and spirt.
Last but not least I wish to give my very special thanks to my family, for their interest and love, specially to my parents who have always supported me and provided me with the best resources and opportunities and to my grandfather who has always motivated me to achieve success in my life.

Resumo

O principal objectivo desta dissertação é estudar a segurança de sistemas criptográficos baseados em curvas elípticas usando complexidade de Kolmogorov e funções de sentido único como o principal veículo de análise da segurança de cada sistema.

A motivação por trás desta dissertação é estudar e perceber a análise de funções de sentido único usando complexidade de Kolmogorov. Compreender criptografia baseado em curvas elípticas e sabendo que funções de sentido único são suficientes para a construção de sistemas criptográficos de chave púbica foi outra grande motivação para a análise de esquemas criptográficos definidos por uma função Kolmogorov de sentido único com alçapão e construída sobre curvas elípticas.

Ao longo desta tese, iremos definir uma nova família de funções, que denotaremos por *funções Kolmogorov de sentido único com alçapão*. Iremos também provar que para cada função, o número de alçapões é sempre menor, (por uma fracção polinomial) do que o número de possíveis alçapões.

Iremos também apresentar um sistema criptográfico de chave publica baseado em curvas elípticas e denotaremos por f a função que emula este sistema. Serão obtidas conclusões sobre a segurança do sistema criptográfico, baseadas em observações feitas sobre o tamanho de cada chave privada visto que obteremos um resultado assimptótico que expressa um limite inferior para o tamanho de cada chave privada.

Assumindo que ECDLP não está em **P**, iremos provar que f é uma função Kolmogorov de sentido único e que pode ser vista, com recurso a uma função auxiliar, como uma função de uma família de funções Kolmogorov de sentido único com alçapão.

Palavras-chave: Complexidade de Kolmogorov. Função Kolmogorov de sentido único com alçapão. Criptografia baseada em curvas elípticas. Segurança criptográfica.

Abstract

The purpose of this dissertation is to study the security of cryptographic systems based on elliptic curves, using Kolmogorov complexity and one-way functions as the main tools to analyze the security of each scheme.

The main motivation for this thesis is understand the individual approach of analyzing one-way functions using Kolmogorov complexity. Knowing that trapdoor one-way functions are sufficient to the construction of public key encryption and signature schemes and understanding elliptic curves and cryptography was also a motivation to analyze a cryptographic scheme based on elliptic curves and defined through trapdoor Kolmogorov one-way functions.

We will define a new family of functions and will call them *trapdoor Kolmogorov one-way functions*. We will also prove that for each function, the number of trapdoors is always lower, (by a polynomial fraction), than the number of possible trapdoors.

We will present a public key cryptographic system based on elliptic curves and we will denote by f a function that emulates the system. We will draft conclusions on the security of the cryptographic scheme, based on observations made on the size of each private key as we arrive to an asymptotic result that yields a lower bound on the length of each private key.

Assuming that ECDLP is not in **P**, we will prove that a function f is a Kolmogorov one-way function and, with the the help of auxiliar function, can be extended to an element of a family of trapdoor Kolmogorov one-way functions.

Keywords: Kolmogorov complexity. Trapdoor Kolmogorov one-way function. Elliptic curves cryptography. Cryptographic security.

Contents

List of Figures

Chapter 1

Introduction

This thesis consists on defining a new family of functions called *trapdoor Kolmogorov one-way functions* and exploring the application of this new class in public key encryption. We look in detail to the case where the cryptographic scheme is based on elliptic curves.

The main motivation for this thesis is to understand the individual approach of analyzing one-way functions using Kolmogorov complexity. Knowing that trapdoor one-way functions are sufficient to the construction of public key encryption and signature schemes and understanding elliptic curves and cryptography was also a motivation to analyze a cryptographic scheme based on elliptic curves and defined through trapdoor Kolmogorov one-way functions.

The Kolmogorov complexity, $K(x)$, (see Li and Vitányi [2009] and Lee [2006]) of an object x is the length of the shortest program producing x in a universal Turing machine. The time-bounded version of Kolmogorov complexity $K^t(x)$, is the length of the shortest program producing x within time $(|x|)$. We present some classic results on Kolmogorov complexity such as the Invariance Theorem 2.10, the Incompressibility Theorem 2.15 and the Symmetry of Information Theorem 2.20.

Intuitively, a one-way function is a function that is easy to compute but hard to invert. The existence of these functions is an open question which implies **P** $\neq$ **NP**. Given the importance of one-way functions and the impact of their applications, we analyze them at an individual level using Kolmogorov complexity. Classically there are several definitions of one-way function, namely: strong, weak and deterministic,(see Goldreich [2001]). An interesting fact about strong and weak one-way functions is that, although their definitions are not equivalent, weak one-way functions exist if and only if strong one-way functions exist, see Proposition 2.26.

We introduce a new family of functions, that we call Trapdoor Kolmogorov one-way function. These are Kolmogorov one-way functions f as in Antunes et al. [2013] with the extra property that there exists a polynomial time function h that for each function of the family, provides as input an extra information that one can use to invert the function f in polynomial time. Following the same rational we define what trapdoor strong one-way function, trapdoor weak one-way function and trapdoor deterministic one-way function are. We set an upper bound for the number of possible trapdoors that each trapdoor Kolmogorov one-way function has.

The subject of elliptic curves is one of the jewels of the nineteenth-century mathematics, originated by Abel, Gauss, Jacobi and Legendre. The purpose of the chapter on elliptic curves in this thesis is to introduce the reader to the elementary concepts of elliptic curves and to present the cryptographic scheme that we aim to analyze. Through the classical Weierstrass equation, we define an elliptic curve and present the set of rational points of an elliptic curve, (the set of points which are mapped to 0). It is known that the set of rational points of an elliptic curve is in fact an algebraic group with a specific arithmetic, these results are presented along with some examples. For more details see (Schmitt and Zimmer [2003] and Balasubramanian [2003]).

In order to use elliptic curves in computational problems, one has to look to the case where elliptic curves are defined over a finite field. We will introduce some results due to Hasse and Frobenius that will be helpful when working with elliptic curves, however we will not look in detail to these, the interested reader can see the details in Schmitt and Zimmer [2003]. One of many problems studied when working with elliptic curves over finite fields, is the *elliptic curves discrete logarithm problem* defined as follow: Given two rational points P and Q one desires to find an integer x such that $xP = Q$, see details in Definition 3.20. This problem yields an EL Gamal encryption scheme 3.5. See more details in Balasubramanian [2003], Blake et al. [1999] and Stinson [2002].

The main issue on each encryption scheme is the security of the scheme. One simple does not work with one scheme if this is not strong enough against different attacks. How secure an encryption scheme is and how one can evaluate this, are some of the questions that we attempt to answer through this work. We wish to build a cryptographic system corresponding to the ECDLP and does not lie in **P**, to ensure this we will only use private keys $l, m \in \mathcal{O}(\log n)$, see more details in Section 4.2 and Proposition 4.3.

Using the cryptographic system based on the elliptic curves we prove that the function f that emulates the system is honest, (the object and the image are polynomial related), injective and computable in polynomially time. These results will help us prove that if ECDLP is not in **P** then f is a Kolmogorov one-way function, see more details in Theorem 4.6.

Finally using the results obtained and our initial definition of trapdoor Kolmogorov one-way function, we can build a polynomial time function that outputs a trapdoor for f. Using this we can extend the definition of f to a trapdoor Kolmogorov one-way function candidate, see more in 4.8. It is important to note that all these results are obtained under the assumption that the ECDLP is not in **P**.

Chapter 2

Kolmogorov complexity

2.1 Introduction

As part of this work we take a look at the notion of Kolmogorov complexity. We will start by defining it and prove some important results. After, we will analyze the notion of one-way functions and trapdoor one-way functions and see how these can be related with Kolmogorov complexity. We aim to use this new notion to be more precise when using the idea of trapdoor one-way functions, as they play a major role on public-key encryption and signature schemes.

We will start by presenting the notion of Kolmogorov complexity, with oracle access to f, in order to allow the verification of a pre-image quickly. We present Kolmogorov complexity this way in order to avoid technical issues of having to account the complexity of the function in use. We notice that we want precisely to account for the complexity of inverting images without worrying with the complexity of verifying the validity of the guessed objects.

2.2 Kolmogorov Complexity

We will work with strings usually denoted as x, y, z which are elements of $\Sigma^* = \{0,1\}^*$, we will denote by $\log$ the $\log_2$ and by $|x|$ the size of the string x. It is also assumed that any time-bound $t(n)$ is constructible and larger than n, to allow, at least, any program to be able to print the object.

Definition 2.1. Let T be a Turing machine and T_f, a Turing machine with oracle access to f. We denote by For any strings $x, y \in \Sigma^*$, the *Kolmogorov complexity of* x *given* y in T is:

$$C_f^T(x|y) = \min_p\{|p| : T_f(p, y) = x\}.$$

For any time constructible t, the *t-time-bounded Kolmogorov complexity, with oracle access to* f*, of* x *given* y is:

$$C_f^{T,t}(x|y) = \min_p\{|p| : T_f(p, y) = x \text{ in at most } t(|x|) \text{ steps}\}.$$

The default value for y is the empty string ε and for f is the null function. The following results are important in Kolmogorov complexity.

The following results lead to the Invariance Theorem. This is a cornerstone in Kolmogorov theory as we will see. To ease on notation, we will denote $c_f(x|\varepsilon) = c_f(x)$.

Definition 2.2. Let A be a subclass of the partial functions over $\mathbb{N}_0$. A function f is said to be *additively optimal for A* if $f \in A$ and for every function $g \in A$ there is a constant $c_{f,g}$ depending only on f and g, such that $C_f(x) \leq C_g(x) + c_{f,g}$ for all x. Replacing x by $\langle x, y\rangle$ with $\langle .\rangle$ the standard recursive bijective pairing function, yields the definition for two variable input.

We will now present the Invariance Theorem. This is a cornerstone for the subsequent development of the theory. We will start by proving an auxiliary lemma. For this we need to know that by *additively optimal universal partial recursive function* we understand a universal function that is optimal additively according to Definition 2.2 and is partial recursive. We will define by $T_1, T_2, ...$ a enumeration of all Turing machines, where T_i computes the partial recursive function ϕ_i.

Lemma 2.3. There is an additively optimal universal partial recursive function.

Proof. Let ϕ_0 be a function computed by a universal Turing machine U. Machine U expects inputs of the format:

$$\langle y, p\rangle = 1^{|y|}0yp$$

The interpretation is that the total program $\langle y, p\rangle$ is a two-part code of which the first part consists of a self-delimiting encoding of T_y and the second is the program p. This way U can simulate T_y and then run T_y with p as its input. That is,

$$\phi(\langle y, p\rangle) = \phi_y(p).$$

By convention we set U as T_0 and therefore $U\langle 0, p\rangle = U(p) = T_0(p)$. Since T_y computes the partial recursive function ϕ_y we have that:

$$C_{\phi_0}(x) \leq C_{\phi_y}(x) + c_{\phi_y},$$

Where c_{ϕ_y} can be set to $2|y| + 1$. □

For many applications we require a generalization to a conditional version as follows.

Definition 2.4. Let $x, y, p \in \mathbb{N}$. Any partial recursive function ϕ such that $\phi(\langle y, p\rangle) = x$, is a description of x. The complexity C_ϕ of x conditional to y is defined by:

$$C_\phi(x|y) = min\{|p| : \phi(\langle y, p\rangle) = x\},$$

and $C_\phi(x|y) = \infty$ if there are no such p. We call p a program to compute x by ϕ, given y.

Theorem 2.5 (Invariance Theorem)**.**
There is an additively optimal universal partial recursive function ϕ_0 for the class of partial recursive

functions to compute x given y. Therefore, $C_{\phi_0}(x|y) \leq C_\phi(x|y) + c_\phi$ for all partial recursive functions ϕ and all x and y, where c_ϕ is a constant depending on ϕ but not on x or y.

Proof. Let ϕ_0 be the function computed by universal Turing machine U such that U started on input $\langle y, \langle n, p\rangle\rangle$ simulates T_n on input $\langle y, p\rangle$. That is, if T_n computes the partial recursive function ϕ_n, then

$$\phi_0(\langle y, \langle n, p\rangle\rangle) = \phi_n(\langle y, p\rangle).$$

Hence, for all n,

$$C_{\phi_0}(x|y) \leq C_{\phi_n}(x|y) + c_{\phi_n},$$

where $c_{\phi_n} = 2|n| + 1$. □

One of the reasons to work with Kolmogorov complexity is to assign to each string a probability. Let us consider the set $A^{\leq n}$ defined as follow:

$$A^{\leq n} = \{x \in \Sigma^* : |x| \leq n\}.$$

We know that for all $n > 1$ we have that:

$$\sum_{x \in A^{\leq n}} 2^{-|x|} > 1.$$

Since we want to assign a probability to each string, then from previous result, one can easily conclude that this is not the rule we should take in order to assign a probability, since it does not follow the basic laws of probability. Our next try is to consider only the size of the smallest program p, in the lexicographic order, that characterizes x, in other words $C(x)$. Still we get the following:

$$\sum_{x \in \Sigma^*} 2^{-|C(x)|} \geq \sum_{x \in \mathbb{N}} 2^{-|C(x)|} \geq \sum_{x \in \mathbb{N}} 2^{-\log n}$$

Since $\sum_{n \in \mathbb{N}} \frac{1}{n} \to \infty$ and $2^{-\log(n)} = \frac{1}{n}$, then $\sum_{x \in \Sigma^*} 2^{-|C(x)|} > 1$.

We have to restrict even further the domain of our functions in order to respect the laws of probability. One possible solution is to consider a new set of strings and we will redesign our notion of Kolmogorov complexity.

With resource to the Invariance Theorem 2.5, we are able to restrict our domain in order to answer the question raised before.

Definition 2.6. Let $x, y \in \Sigma^*$. We call x a *prefix* of y if there is a $z \in \Sigma^*$ such that $y = xz$.
A set $A \subseteq \Sigma^*$ is *prefix-free* if no element in A is the prefix of another element in A.
A function $D : \Sigma^* \to \Sigma^*$ defines a *prefix-code* if its domain is prefix-free.

Example 2.7. *A simple prefix-code is obtained by reserving* 0 *as a stop sign of the length of the string and encoding* x *as* $1^{|x|}0x$*. This way we get the following encoding:*

$$E_i(x) = \begin{cases} 1^{|x|}0 & \text{for } i = 0 \\ E_{i-1}(x)x & \text{for } i > 0 \end{cases}$$

Thus $E_1(x) = 1^{|x|}0x$ *and has length* $|E_1(x)| = 2|x| + 1$. ▷

This encoding is so important that we will present a simpler notation, that will be used later in this work:

- $\bar{x} = 1^{|x|}0x$.
- $|\bar{x}| = 2|x| + 1$.

However a relatively minor improvement yields an asymptotically optimal code-word alphabet. The next example states this improvement.

Example 2.8. *We will encode* x *by* $E(x) = |\bar{x}|x$*, that is, by encoding first the length of* x *in prefix-free form, followed by the literal representation of* x*. We will encode the length of* x *using the following sequence:*

$$(\epsilon, |x| = 0), (0, |x| = 1), (1, |x| = 2), (00, |x| = 3), (01, |x| = 4), (10, |x| = 5)...$$

Code E *is a prefix-free, since if we know the length of* x *as well as the start of literal representation, then we also know where it ends. The length set of this code is given by:*

$$|E(x)| = |x| + 2|(|x|)| + 1.$$

For instance take $x = 01011$*. From the previous sequence, the length of* x *is given by* 10*, then using Example 2.7, one easily get that* $E(x) = 1101001011$*. This is the main code-word used in Kolmogorov complexity and the interested reader can find out more about it in Li and Vitányi [2009].*

Based on this idea, we can present what a partial recursive prefix-free function is:

Definition 2.9. A partial recursive function:

$$\phi : \Sigma^* \to \mathbb{N}$$

is said to be a *partial recursive prefix-free function* if for any $x, y \in \Sigma^*$ if $\phi(x) < \infty$ and $\phi(y) < \infty$ then x is not a prefix of y.

Based on the enumeration of Turing machines on Li and Vitányi [2009] construct an enumeration of all Turing machines that compute partial recursive prefix-free functions. We will denote this enumeration by $T'_1, T'_2, ...$ These Turing machines, will be referred as *prefix-free Turing machines.* One can prove that there are Universal prefix-free Turing machines, i.e., a machine U that given as input $\langle M, x\rangle$ computes

$M(x)$. The existence of such machine leads to the version of the Invariance Theorem in the case of prefix-free.

Theorem 2.10 (Invariance Theorem for prefix-free functions)**.**
There exists an additively optimal universal partial recursive prefix-free function ψ_0 for the class of partial recursive prefix-free functions to compute x given y. Therefore for every partial recursive prefix-free function ψ there is a constant c_ψ and for all $x, y \in \mathbb{N}$ we have:

$$C^U_{\psi_0}(x|y) \leq C^U_\psi(x|y) + c_\psi.$$

Proof. Let ψ_0 be the function computed by a Universal Turing machine U such that U started on input $\langle \bar{n}, \langle y, p\rangle\rangle$ simulates a prefix Turing machine T'_n in input $\langle y, p\rangle$. T'_n computes the partial recursive prefix function ψ_n.
Then $\psi_0(\langle \bar{n}, \langle y, p\rangle\rangle) = \psi_n(\langle y, p\rangle)$. Hence, for all n:

$$C^U_{\psi_0}(x|y) \leq C^U_{\psi_n}(\langle y, p\rangle) + c_{\psi_n}.$$

Where $c_{\psi_n} = 2|n| + 1$. Notice that the constant does not depend on the input x but only on the machines. In fact the constant is the cost of translatability between machines. □

This result allows us to fix once and for all an universal machine as model to develop the theory of Kolmogorov complexity without any loss of generality. In particular, with simple adaptations in the argument, one can extend the result to work with the case where the machines can access to oracles or even with computation resource bounds. In the following definition we condense this idea.

Definition 2.11. Set ψ_0 as the reference recursive prefix-free function and U as the reference prefix-free Turing machine corresponding to ψ_0. Then by *prefix-free conditional complexity* of x conditional by y we understand:

$$K(x|y) = C^U_{\psi_0}(x|y).$$

From now on, we will use this as our definition of Kolmogorov complexity. We will mainly work with t-time bounded Kolmogorov complexity with access to an oracle f and we will denote it by K^t_f, the definition of this is straightforward to present.

Theorem 2.12 (Kraft Inequality)**.**
Let $l_1, l_2, ...$ be a finite or infinite sequence of natural numbers. There is a prefix-code with this sequence as lengths of its binary code words iff:

$$\sum_n 2^{-l_n} \leq 1.$$

Proof. We will prove each implication separately. We will start to proof the only if implication. A prefix-code corresponds to a set of disjoint intervals in $[0, 1)$ such that each interval i has length 2^{-l_i}, then the desired inequality holds.
Let us now prove the other implication. Let $l_1, l_2, ...$ be such that the inequality holds. We can assume

that the sequence is non-decreasing. Choose disjoint sets $I_1, I_2, ...$ of length $2^{-l_1}, 2^{-l_2}...$ from the interval $[0, 1)$. For each $n \geq 1$ the right end of I_n is given by:

$$\sum_{i=1}^{n} 2^{-l_i}.$$

(Note that the right end of I_n is the left end of I_{n+1}). Since the sequence l_n is non decreasing, we can take each interval I_n correspond to an interval $[0.x, 0.x + 2^{-l_n}[$, where x is a binary string of length $|x| = 2^{-l_n}$. If we take the binary string x corresponding to I_n as the nth code word we arrive to the desired result. □

Since the domain of the function K is a prefix-free set it follows from Kraft inequality that:

$$\sum_{x} 2^{-K(x)} \leq 1.$$

Therefore, considering prefix-free sets one can assign probabilities to strings.
With this observation at hand we can now assign probabilities to each string based on prefix-free Kolmogorov complexity. This leads to the following definition.

Definition 2.13. For Σ^* we set the following probability distribution,

$$\begin{aligned} m : \Sigma^* &\rightarrow [0, 1] \\ x &\mapsto 2^{-K(x)} \end{aligned}$$

We call m *universal semimeasure*.

The universal semimeasure m is said to be universal in the sense that for any other distribution n there is a constant c such that:

$$cm(x) \geq n(x).$$

The proof of this stament, the proof of existence of m as well as other important properties, will not be the focus of our work. However, the interested reader can find more about it in Li and Vitányi [2009].

Theorem 2.14. For every length n and any string $z \in \{0, 1\}^*$ there exists $x \in \{0, 1\}^n$ with $x \neq z$ such that $K(x|z) \geq n$.

Proof. The proof is by simple counting. There are 2^n strings of length n and only $2^n - 1$ many potential programs of length less than n. Thus at least one string $x \in \{0, 1\}^n$ must require a program of length at least n, even given z. □

The following result is one of the main results of Kolmogorov complexity as it proves the existence of incompressible strings, *i.e.*, objects for which their smallest representation is as short as the object itself.

Theorem 2.15 (Incompressibility Theorem)**.**

1. For each $n \in \mathbb{N}$ we have that

$$\max\{K(x) : |x| = n\} = n + K(|x|) + \mathcal{O}(1).$$

2. For each constant r, the number of x of length n with

$$K(x) \leq n + K(n) - r$$

does not exceed $2^{n-r+\mathcal{O}(1)}$.

Proof.

1. We will prove each inequality separately. Let us start by proving $\leq$. We consider U to be a universal Turing machine and T a prefix machine, that on input qx, where $U(q) = |x|$ computes $T(qx) = x$. Let $T_1, T_2, ...$ be a enumeration of prefix Turing machines, then for some $m \in \mathbb{N}$ $T = T_m$ and we have that:

$$U(\bar{m}qx) = T_m(qx) = x.$$

Hence $K(x) \leq |x| + K(|x|) + |\bar{m}|$, where m is independent from x.
We will prove the second point of the theorem before proving the other inequality.

2. Take x of length n and consider the following equality:

$$K(x) + K(n|x, K(x)) = K(n) + K(x|n, K(x)) + \mathcal{O}(1).$$

(This equality is based on Theorem 2.20, this relation can be found Li and Vitányi [2009]. We did not find a satisfactory way to avoid dependence on later material). Since $|x| = n$ we have that $K(n|x, K(x)) = \mathcal{O}(1)$. Thus we can arrive to the following equality:

$$K(x) = K(n) + K(x|n, K(x)) + \mathcal{O}(1).$$

Using the result obtained on the first point of the proof, we have that:

$$K(x|n, K(x)) \leq n - r + \mathcal{O}(1).$$

Therefore there are less than $2^{n-r+\mathcal{O}(1)}$ strings x satisfying this equality.

3. Let us now consider again the second inequality on the first point of the theorem. Since there are 2^n strings of length n and using the result we have just obtained we arrive to the desired inequality.

□

Looking back at Theorem 2.14 and generalizing the argument, one can obtain the result obtained in Theorem 2.15.

Corollary 2.16. *For $x, y \in \{0,1\}^*$, one has that:*

$$K(x|y) \leq K(x) + \mathcal{O}(1).$$

Corollary 2.16 is an immediate result of Theorem 2.15.

Definition 2.17. Let x be a string of length n. If $K(x) \geq n$, then x is said to be *incompressible*.

We will now present a very important result in Kolmogorov complexity, *Symmetry of Information Theorem*. For this we will present first some new notations.

We recall the *halting problem*. This problem consists in, given a Turing machine T find whether the program halts or runs forever. It is known that this problem is undecidable.

Another important notion we need to know is the notion of *dovetail* that we present in the next definition.

Definition 2.18 (Dovetail)**.**
For T a Turing machine, by *dovetailing* we understand the following procedure for finding a halting stage:

1. In stage 1, we run the empty program on T for one time step.
2. In stage k we run all programs of length i for j time steps, where $i + j = k$ with $i, j \in \mathbb{N}$.

Notice that if T has a halting stage, through this procedure we will find it.

We will now present the Language Compression Theorem which takes a main role in the proof of the Symmetry of Information Theorem.

Theorem 2.19 (Language Compression Theorem)**.**
For any recursively enumerable set A, $K(x) \leq \log |A^{\leq n}| + \mathcal{O}(\log n)$ for all $x \in A$ of length n.

Proof. Fix $n \in \mathbb{N}$. Let M be a machine which enumerates A. We dovetail the running of M over all strings $x \in \{0,1\}^n$. For each string $x \in A$, the machine M will eventually halt and say *accept*. Furthermore, these computations will halt in an exact order. We therefore can describe each $x \in A$ by an index $i \in \mathbb{N}$ saying that x_i is the $i-th$ string of length n that M will accept when run in a dovetail procedure on strings of length n. The description requires $\log |A|$ bits to specify the index, and $\mathcal{O}(\log n)$ bits to describe n and the machine M. □

We are finally able to present and prove the Symmetry of Information Theorem.

Theorem 2.20 (Symmetry of Information)**.**
For all $x, y \in \{0,1\}^n$, the following equality holds:

$$K(x, y) = K(x) + K(y|x) + \mathcal{O}(\log n).$$

Proof. We will prove each inequality separately. We will start by proving that $K(x,y) \leq K(x) + K(y|x) + \mathcal{O}(\log n)$. Given a program for x and a program for y given x, we start by running the program for x to produce x and then run the program for y given x to produce y. The final output is the pair (x,y).
Let us look now at the other direction. Fix $x^*, y^* \in \{0,1\}^n$, and say that $K(x^*,y^*) = m$. Consider the set of strings,

$$A = \{(x,y) : K(x,y) \leq m\}.$$

This set is recursively enumerable given m. We can now dovetail the running of all programs of length less or equal than m; if the pair (x,y) is in A, then eventually one of these programs will halt having printed (x,y). Notice that the cardinality of the set is less than 2^{m+1} by counting the number of short programs. Consider the set $A_{x^*} = \{y : K(x^*,y) \leq m\}$. Notice that y^* is an element of this set and that A_{x^*} is recursively enumerable given x^* and m. Therefore by Theorem 2.19 we have that,

$$K(y^*|x^*) \leq \log |A_{x^*}| + \mathcal{O}(\log n).$$

Let k be such that $2^k \leq |A_x^*| \leq 2^{k+1}$. Consider the set

$$B_k = \{x : \text{there are at least } 2^k \text{ elements } y : K(x,y) \leq m\}.$$

Notice that $x^* \in B_k$ and that this set is recursively enumerable given m and k, and that the size of this set is less than 2^{m-k+1} since the size of A is less than 2^{m+1}, thus applying Theorem 2.19 again we have that,

$$K(x^*) \leq m - k + 1 + \mathcal{O}(\log n).$$

Putting the two together, we have that,

$$K(x^*) + K(y^*|x^*) \leq m + \mathcal{O}(\log n) \leq K(x^*,y^*) + \mathcal{O}(\log n).$$

□

Theorem 2.20 is only valid for exponential time, and since there is not any result on symmetry of information for lower times, it is unknown if Symmetry of Information is true for polynomial time. This open question is related to the existence of one-way functions and with the famous problem of **P** = **NP** and although this is out of the scope of this work, the interested reader can learn more about this in Longpre and Mocas [1993] and Longpré et al. [1992].

2.3 One-way functions

In this section we take a closer look at some preliminaries on one-way functions. Heuristically speaking a function is said to be *one-way* if it easy to compute but hard to invert. We will define four different types of one-way functions and we will prove some results about them.

Definition 2.21. A function f is said to be *honest* if $|f(x)|$ and $|x|$ are polynomially related, i.e. for some $k > 0$ and for every $x \in \Sigma^*$ we have:

$$(|f(x)| \leq |x|^k + k) \wedge (|x| \leq |f(x)|^k + k).$$

If we take f to be a non honest function, then one could consider an object x and the respective image $f(x)$ not polynomially related. This would lead to the existence of one-way functions as we can see in the next example.

Example 2.22. *Consider the following function:*

$$\begin{aligned} f : \Sigma^{2^n} &\rightarrow \Sigma^n \\ (x_1, ..., x_{2^n}) &\mapsto (x_1, ..., x_n). \end{aligned}$$

The function f maps the first n elements of the given tuple. Consider now the function f^{-1} that inverts the function f. The function f^{-1} is given by $f^{-1}(x_1, ..., x_n) = (x_1, ..., x_{2^n})$. One can easily see that f^{-1} can not return the output in polynomial time therefore f is a one-way function.

In order to avoid this problem, from now on, we will consider f to be an honest function.

Definition 2.23 (Deterministic one-way function)**.**
A function $f : \Sigma^* \rightarrow \Sigma^*$ is said to be a *deterministic one-way function* if the following two conditions hold:

1. There is a deterministic polynomial time algorithm A such that on every input x we have that $A(x) = f(x)$.
2. For any deterministic polynomial time algorithm B, for some polynomial $q(.)$ and for every sufficiently large n,
$$pr_{x \in \Sigma^n}[f(B(f(x), n)) \neq f(x)] > \frac{1}{q(n)}.$$

Definition 2.24 (Weak one-way function)**.**
A function $f : \Sigma^* \rightarrow \Sigma^*$ is said to be a *weak one-way function* if the following two conditions hold:

1. There is a deterministic polynomial time algorithm A such that on every input x we have that $A(x) = f(x)$.
2. For any polynomial $t(.)$, there is a polynomial $q(.)$ such that for every probabilistic t-time bounded algorithm B and for every sufficiently large n,
$$pr_{x \in \Sigma^n}[f(B(f(x), r, n)) \neq f(x)] > \frac{1}{q(n)}.$$

Definition 2.25 (Strong one-way function)**.**
A function $f : \Sigma^* \rightarrow \Sigma^*$ is said to be a *strong one-way function* if the following two conditions hold:

1. There is a deterministic polynomial time algorithm A such that on every input x we have that $A(x) = f(x)$.

2. For any polynomial $t(.)$, for every positive polynomial $q(.)$, for every probabilistic t-time bounded algorithm B and for every sufficiently large n,

$$pr_{x\in\Sigma^n}[f(B(f(x),r,n))=f(x)]<\frac{1}{q(n)}.$$

In the previous definitions r denotes the randomness used by the algorithm B that tries to invert f. As we will see in the next proposition, it is very easy to relate these three definitions.

Proposition 2.26. Take $f:\Sigma^*\to\Sigma^*$.

1. If f is a strong one-way function, then f is a weak one way function.
2. If f is a weak one-way then f is a deterministic one-way function.

Proof.

1. If f is a strong one-way function, then for any polynomial $t(.)$, for every positive polynomial $q(.)$, for every probabilistic t-time bounded algorithm B and for every sufficiently large n,

$$pr_{x\in\Sigma^n}[f(B(f(x),r,n))=f(x)]<\frac{1}{q(n)}.$$

Then in particular we have that,

$$pr_{x\in\Sigma^n}[f(B(f(x),r,n))\neq f(x)]>1-\frac{1}{q(n)}.$$

Let us take a polynomial $p(n)=1-\frac{1}{q(n)}$, which is a positive polynomial, since $0<\frac{1}{q(n)}<1$. Let us now consider the polynomial $s(n)$ such that,

$$s(n)=\frac{q(n)}{q(n)-1}=\frac{1}{p(n)}.$$

It is easy to see that $s(n)$ is also positive, since it is the inverse of a positive polynomial. Then we have the following,

$$pr_{x\in\Sigma^n}[f(B(f(x),r,n))\neq f(x)]<\frac{1}{s(n)}.$$

And we can conclude that f is a weak one-way function.

2. If f is a weak one-way function, then we have that for every polynomial $t(.)$, there is a polynomial $q(.)$ such that for every probabilistic t-time bounded algorithm B and for every sufficiently large n we have

$$pr_{x\in\Sigma^n}[f(B(f(x),r,n))\neq f(x)]>\frac{1}{q(n)}.$$

Then in particular the inequality holds for every deterministic algorithm B and we conclude that f is a deterministic one-way function. Note that in this case, r is irrelevant.

It is clear that any strong one-way function is a weak one-way function. It is also easy to see that there are weak one-way functions that are not strong one-way functions. The intersection result about these functions concerns existence. It is well known that weak one-way functions exist if and only if strong one-way functions exist *i.e.* from a weak one-way function one can construct a strong one-way function. Furthermore from Longpré et al. [1991] they exist if and only if **P** $\neq$ **NP**. □

The next result relates the notion of strong one-way function, presented in Definition 2.25 and Kolmogorov complexity. This result will be used later to establish an upper bound on the number of trapdoors a function can have.

Theorem 2.27. Let f be an injective and polynomial time computable function. If f is a strong one-way function, then for every constant c and for every polynomial $t(.)$, the expected value of $K^t_f(x|f(x), r, n)$ over pairs $(x, r) \in \Sigma^n \times \Sigma^{t(n)}$, is larger than $c \log n$ for every sufficiently large n.

Proof. Assume by absurd, that for some constant c and some polynomial $t(.)$, we have $E[K^t_f(x|f(x), r, n) \leq 2c \log n]$ infinitely often. Using Markov's inequality we get:

$$pr_{(x,r)\in\Sigma^n\times\Sigma^{t(n)}}[K^t_f(x|f(x), r, n) \leq 2c \log n] > 1 - \frac{c \log n}{2c \log n} = \frac{1}{2}.$$

We define an algorithm Q that on input $(f(x), r)$ tries to invert $f(x)$, and succeeds for the cases where $K^t_f(x|f(x), r, n) \leq 2c \log n$. This algorithm runs all programs of size up to $2c \log n$ for at most t steps, using the random string r with input $f(x)$. For each such program, Q tests if the output is an inverse of $f(x)$, and if it is, outputs that inverse. If, for the pair (x, r) it happens that $K^t_f(x|f(x), r, n) \leq 2c \log n$, then Q will find a suitable shortest program and output the correct x. Therefore, it succeeds with probability 1.

Since there are most $2^{2c \log n+1} = 2n^{2c}$ programs of length at most $2c \log n$ and each of them runs for a polynomial number of steps, then Q runs in polynomial time. By construction, we know that for infinitely many n's.:

$$pr_{(x,r)\in\Sigma^n\times\Sigma^{t(n)}}[Q(f(x), r, n) = x] > \frac{1}{2}.$$

Thus, f is not a strong one-way function. □

We will now present an approach to define one-way functions using Kolmogorov complexity. This approach is presented in Antunes et al. [2013].

Definition 2.28. Let $f : \Sigma^* \to \Sigma^*$ be an injective and polynomial time computable function such that $|f(x)| = m(n)$ for all $x \in \Sigma^n$, where m is some polynomial. We say that f is a *Kolmogorov one-way function* if for every polynomial $t(.)$, for every positive integer c, for every sufficiently large n and for every x of length n,

$$K^t_f(x|n) - K^t_f(x|f(x), n) \leq c \log n.$$

We can easily relate a Kolmogorov one-way function with a deterministic one-way function. This result is presented in the next theorem.

Theorem 2.29. If f is a Kolmogorov one-way function then f is a deterministic one-way function.

Proof. We prove this theorem by contraposition. Assume that f is not a deterministic one-way function. Thus, there is a deterministic polynomial time algorithm B such that for every polynomial $q(.)$ and for every n_0, there is an $n \geq n_0$, for which:

$$\#\{x \in \Sigma^n : B(f(x), n) = x\} \geq 2^n - \frac{2^n}{q(n)}.$$

Thus, for an infinity of n's, B inverts at least one x such that $|x| = n$, $K_f^t(x|n) > \sqrt{n}$, due to the Incompressibility Theorem 2.15. For these x, we have that $K_f^t(x|n) > \sqrt{n}$ and $K_f^t(x|f(x), n) \leq c'$, where c' is a constant that includes the description of B. Taking those x of sufficiently large n such that for every c, $\sqrt{n} > c \log n + c'$, we have that:

$$\begin{aligned} K_f^t(x|n) - K_f^t(x|f(x), n) &> \sqrt{n} - c' \\ &> c \log n + c' - c' \\ &= c \log n. \end{aligned}$$

□

For the moment there exists no results relating Kolmogorov one-way function with strong one-way function or weak one-way function.

2.4 Trapdoor One-Way Functions

We will look in detail to the case where our one-way function has a trapdoor that provides extra information. The trapdoor will be important to extract some extra information about the function in study. We will define these functions through Kolmogorov complexity and prove some results.

We will study one-way functions that map objects from a set of arity n to a set of arity $m(n)$, meaning $f : \Sigma^n \to \Sigma^{m(n)}$, where m is some polynomial. We will consider these functions as a family $\{f_n\}_{n \in \mathbb{N}}$. The following definitions represent the new insight this work brings to the field of Kolmogorov complexity and are the main definitions developed in this work.

Definition 2.30. Let $\{f_n\}_{n \in \mathbb{N}}$ be a family of Kolmogorov one-way functions, such that:

$$f_n : \Sigma^n \to \Sigma^{m(n)},$$

where m is some polynomial. We say that $\{f_n\}_{n \in \mathbb{N}}$ is a *trapdoor Kolmogorov one-way function family* if there is a function:

$$h : \mathbb{N} \to \Sigma^{t(n)},$$

for t polynomial time function, such that

$$K_{f_n}^t(x|f_n(x), h(n), n) \in \mathcal{O}(1).$$

We call h our *trapdoor function*.

Taking advantage of this new definition we can define strong, weak and deterministic one-way functions with the trapdoor property.

Definition 2.31. We say that a family $\{f_n\}_{n\in\mathbb{N}}$ is a family of *trapdoor strong one-way function*, *trapdoor weak one function* or *trapdoor deterministic one-way function* if each element f_n of the family is respectively a strong one-way function, weak one-way function or deterministic one-way function and if there is a computable function $h : \mathbb{N} \to \Sigma^{t(n)}$ such that:

$$Pr_{(x,h(n))\in\Sigma^n\times\Sigma^{t(n)}}[f_n(B(f_n(x), h(n), n)) = f_n(x)] = 1.$$

From now on, every time we say that f trapdoor one-way function, we will be referring to a function f that belongs to a family of trapdoor one-way functions.

We can prove a result similar to the result of Theorem 2.29 for this new definition.

Proposition 2.32. If f is a trapdoor Kolmogorov one-way function, then f is a trapdoor deterministic one-way function.

Proof. From Theorem 2.29 we know that if f is a kolmogorov one-way function, then f is a deterministic one-way function. We just have to proof the trapdoor property. Since f is a trapdoor Kolmogorov one-way function we know that:

$$K^t_{f_n}(x|f_n(x), h(n), n) \in \mathcal{O}(1).$$

Then for any x there is an algorithm A that with $f_n(x), h(n)$ and n, it outputs x, i.e. for a universal Turing machine U we have that $U(A(f_n(x), h(x), n) = x$. We know that any A has constant size. Let us take $c = max\{|A| : \forall x, U(A(f_n(x), h(x), n) = x\}$. There are 2^c possible algorithms A. Let us consider B to be an algorithm such that it runs all possible algorithms A with $f(x), h(n)$ and n and it checks whether the output is x or not. Then the following holds:

$$Pr_{x\in\Sigma^n}[(B(f_n(x), h(n), n)) = x] = 1.$$

It remains to show that B runs in polynomial time. Since c is a constant we have that 2^c is a constant and since A runs in polynomial time, we have that B runs in $2^cTIME(A)$ which is also polynomial, where by $TIME$ we understand the running time of A. □

For a given one-way function f, if for every $s \in \mathbb{N}$, we have that s is a trapdoor for f, then f is not a good encryption function. Therefore we aim to find one-way functions with a limited number of trapdoors. The next theorem imposes some limits to the number of trapdoors for each problem. We prove this theorem for strong one-way functions. From Proposition 2.26 we can extend this result to weak and deterministic one-way functions.

Proposition 2.33. Let f be a trapdoor strong one-way function and

$$H = \{r \in \Sigma^{t(n)} : K^t_f(x|f(x), r, n) \in \mathcal{O}(1)\}$$

Then for n large enough and $x \in \Sigma^n$ we have that:

$$\#H = h < 2^{t(n)} - \frac{c\log(n)q(n)}{2^{n-1}}$$

Proof. From Theorem 2.27 we know that if f is injective and a strong one-way function, then for every constant c and for every polynomial $t(.)$ we have that:

$$E[K_f^t(x|f(x), r, n)] > c\log(n)$$

Let us explore the expected value:

$$\sum_{x\in\Sigma^n}\sum_{r\in\Sigma^{t(n)}} Pr(B(f(x), r, n) = x)K_f^t(x|f(x), r, n) > c\log(n)$$

$$\Leftrightarrow$$

$$\sum_{x\in\Sigma^n}\sum_{r\in\Sigma^{t(n)}} \frac{1}{q(n)}K_f^t(x|f(x), r, n) > c\log(n)$$

$$\Leftrightarrow$$

$$\sum_{x\in\Sigma^n}\sum_{r\in H} \frac{1}{q(n)}K_f^t(x|f(x), r, n) + \sum_{x\in\Sigma^n}\sum_{r\in\Sigma^{t(n)}\setminus H} \frac{1}{q(n)}K_f^t(x|f(x), r, n) > c\log(n)$$

$$\Leftrightarrow$$

$$2^{n-1}h\mathcal{O}(1) + \sum_{x\in\Sigma^n}\sum_{r\in\Sigma^{t(n)}\setminus H} K_f^t(x|f(x), r, n) > c\log(n)q(n)$$

$$\Leftrightarrow$$

$$h\mathcal{O}(1) + (2^{t(n)} - h)(n + \mathcal{O}(1)) > \frac{c\log(n)q(n)}{2^{n-1}}$$

$$\Leftrightarrow$$

$$-hn > \frac{c\log(n)q(n)}{2^{n-1}} - 2^{t(n)}(\mathcal{O}(1) + n)$$

$$\Leftrightarrow$$

$$h < 2^{t(n)} - \frac{c\log(n)q(n)}{n2^{n-1}}$$

□

This proposition tells us that for each trapdoor Kolmogorov one-way function, the number of trapdoors is always lower than the totality of trapdoor possibilities. In fact it decreases by a polynomial fraction.

Chapter 3

Elliptic curves cryptography

3.1 Introduction

In this chapter we will introduce the topic of elliptic curves cryptography. Our goal is to build an El Gamal cryptographic system based on elliptic curves. To this end we first provide an overview on how elliptic curves work. In particular we will build the group of rational points of an elliptic curve, we will study in deep this group when the elliptic curve is defined over a finite field and explain how one can use this group as a tool for the elliptic curves and ultimately as a cryptographic scheme, as the group of rational points will be the main set that we will work on when defining a cryptographic system. We will present a technique to count the number of points in the rational group, to intuitively provide reasoning for the difficulty of the problem and hence of the system itself. Finally in the last section of this chapter we provide some algorithms that justify the simplicity of setting the scheme.

3.2 Preliminaries

We will denote by $\mathbb{K}$ the field on where our elliptic curve will be defined and denote by $\mathbb{A}$ the affine space where our solutions lies.

Definition 3.1. An equation of the form:

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

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

For a long Weierstrass normal form E we can set f such that $f(X, Y) = Y^2 + a_1XY + a_3Y - X^3 - a_2X^2 - a_4X - a_6$ as the function defining E.

Definition 3.2. Given a long Weierstrass normal form E, we define by $E(\mathbb{K})$ the *set of rational points*

over the field $\mathbb{K}$ and represent it as:

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

Before defining what an elliptic curve is, we need to introduce two other concepts: The idea of *point at infinity* and the concept of *singular point*. The point at infinity is related with the representation of our long Weierstrass normal form in the projective space $\mathbb{P}$. To this end, we first need to define what we understand by *homogeneous polynomial*.

Definition 3.3. Consider the following $(n+1)$-tuples: $(x_0, ..., x_n) \in \mathbb{P}^n$ and the equivalence relation given by:

$$(x_0, ..., x_n) \sim (y_0, ..., y_n)$$

if and only if there is $\lambda \in \mathbb{K} \setminus \{0\}$ such that $x_i = \lambda y_i$ for all i. The equivalence class of $(x_0, ..., x_n)$ is written as $[x_0, ..., x_n]$.
A *Projective n-space over* $\mathbb{K}$, denoted by $\mathbb{P}^n(\mathbb{K})$ is the set of all $(n + 1)$-tuples of the form presented above, such that at least one x_i is non-zero module the equivalence relation. A projective n-space is represented as:

$$\mathbb{P}^n = \{[x_0, ..., x_n] \in \mathbb{K}^{n+1} : \text{exists i, such that: } x_i \neq 0\}.$$

Definition 3.4. Let $\mathbb{K}$ be a field. A polynomial $P(x, y, z)$ over $\mathbb{K}$ is said to be an *homogeneous polynomial* of degree $n \in \mathbb{N}$ if it is the sum of terms of the form $x^i y^j z^k$ such that $i + j + k = n$.

Example 3.5. *Consider the polynomial* $P(x, y, z) = x^3 + 2y^2z + 7xyz$*. This is a homogeneous polynomial of degree 3 since the degree of each monomial is constant and equal to 3.* ▷

A polynomial $f \in \mathbb{K}[x, y]$ can be homogenized or transformed into an homogeneous polynomial $F \in \mathbb{K}[x, y, z]$ by defining:

$$F(x, y, z) = zf\left(\frac{x}{z}, \frac{y}{z}\right).$$

Similarly, a homogeneous polynomial $F \in \mathbb{K}[x, y, z]$ can be dehomogeneized into a polynomial $f \in \mathbb{K}[x, y]$ using the following transformation:

$$f(x, y) = F(x, y, 1).$$

Given a long WeierstrassWeierstrass normal form E defined by a function f, we denote by F its representation over the projective space $\mathbb{P}^2$, where F is a homogeneous polynomial constructed using the transformation presented before. The representation of the equation presented in Definition 3.1 in the projective space is of the form:

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

In the projective form, we have one equivalence class of points in the infinity, the point $\mathcal{O} = [0, 1, 0]$. In the affine version, this equivalence class is represented by the point $\mathcal{O} = (\infty, \infty)$, which is commonly

called *base point*.

Example 3.6. *Let us consider the following curves in Weierstrass form:*

- $E_1 : Y^2 + Y = X^3 + X.$
- $E_2 : Y^2 = X^3 + X.$

The corresponding projective curves are:

- $E_1 : Y^2Z + YZ^2 = X^3 + XZ^2.$
- $E_2 : Y^2Z = X^3 + XZ^2$

Both E_1 and E_2 have the rational points over $\mathbb{K}$ or $\mathbb{K}$-rational points $P = (0, 0)$ and $\mathcal{O}$, which, in projective representation are $P = [0, 0, 1]$ and $\mathcal{O} = [0, 1, 0]$. ▷

It is easy to see that the points $Z = 0$ in the projective space of the projective Weierstrass normal form are mapped into infinity in the affine version of the Weierstrass normal form. From now on for simplicity of presentation we will be working with the affine version of the long Weierstrass normal form.

The notion of singular point is also essential to define what an elliptic curve is.

Definition 3.7. Let E be a long Weierstrass normal form and $E(\mathbb{K})$ its set of rational points. Take $P = (x_0, y_0) \in E(\mathbb{K})$. We say that P is a *singular point* of E if and only if:

$$\frac{\partial f}{\partial X}(P) = \frac{\partial f}{\partial Y}(P) = 0.$$

A Weierstrass form is said to be *non singular* if it has no singular points. On the other hand a Weierstrass form is said to be *singular* if it has at least one singular point.

We are finally ready to introduce the concept of Elliptic curve. These are Weierstrass curves with no singular points and one point at infinity. In the next section, we will see that the set of rational points of an elliptic is in fact an abelian group.

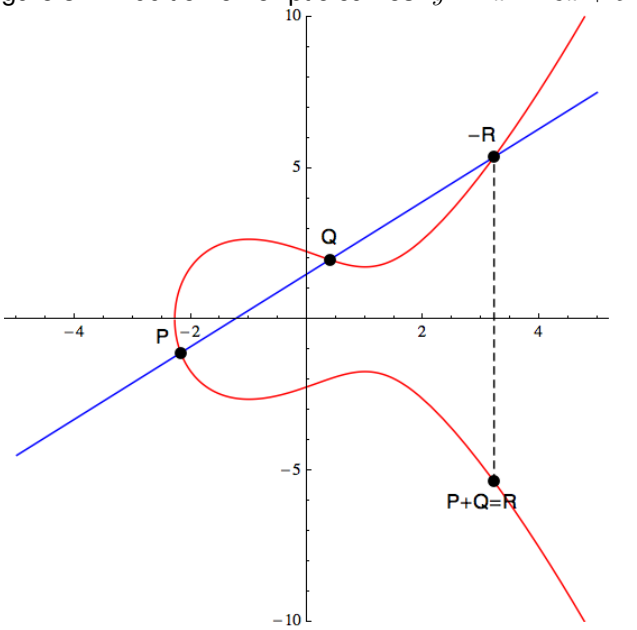
Definition 3.8 (Elliptic Curve)**.**
By *Elliptic Curve* E over a field $\mathbb{K}$, we understand a curve given by a long Weierstrass normal form with no singular points and a point at infinity, that we will also call base point. We will denote by $E(\mathbb{K})$ the set of rational points of the elliptic curve E over the field $\mathbb{K}$.

3.3 Rational points group law

The set of rational points of an elliptic curve E forms a group under addition. This addition is slightly different from the common notion of addition. Before presenting the new notion of addition operation, we will provide some geometric intuition.

Consider two rational points of the equation, say P, Q. Consider the line r passing through P and Q. Since E is a cubic curve r can intersects E in a third point, say $R = (x_3, y_3)$. Consider now the line r' through R and $\mathcal{O}$, then r' intersects E in a third point that we call R' and we define $P + Q = R'$. The following image exemplifies this notion.

Figure 3.1: Addition on elliptic curves. $y^2 = x^3 - 3x + 5$.



Let us now consider the case of summing the point with itself, i.e. calculate $2P$. To this end we will consider t the tangent to the curve E in the point P. If t intersects E in another point $R = (x_2, y_2)$, then $2P = (x_2, -y_2)$. If t does not intersect E in any other point, then we say that $2P = \mathcal{O}$. This notion of doubling a point will be fundamental do define a cryptographic system based on elliptic curves. One can find an example of doubling a point in Figure 3.2.

Finally let us consider the case where we have points $R, P \in E$, the line r that intersects R, P and $\mathcal{O}$. In this case, the point $R + \mathcal{O} = R$. Similarly, for this case we say that $P + R = \mathcal{O}$. One can find an example of doubling a point in Figure 3.3.

We will define precisely this geometric interpretation in the next definition. It is worthwhile mentioning that since any long Weierstrass equation E has degree 3, any line crossing E, intersects it in at most 3 points.

Definition 3.9. Let E be an elliptic curve over $\mathbb{K}$ and $P_1, P_2 \in E(\mathbb{K})$. The line through P_1 and P_2 intersects the elliptic curve in a third point P_3'. We consider the line through P_3' and $\mathcal{O}$. This line intersects E in a third point P_3, then we define:

$$P_1 + P_2 = P_3.$$

This definition yields the following proposition.

Figure 3.2: Point doubling on elliptic curves. $y^2 = x^3 - 3x + 5$.

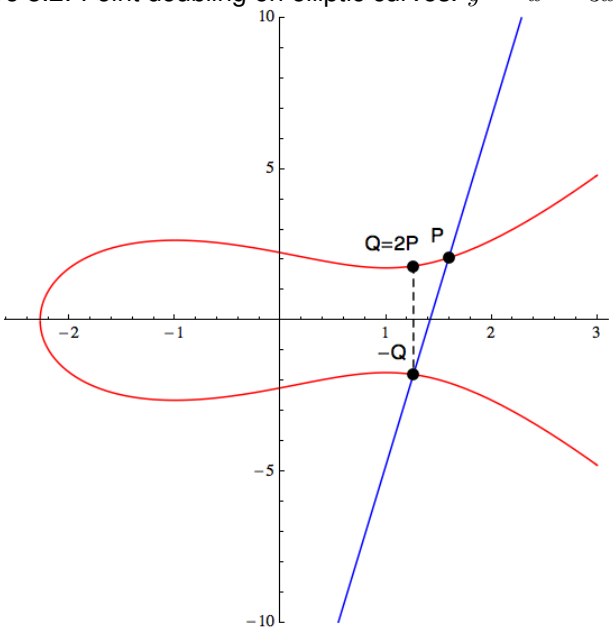
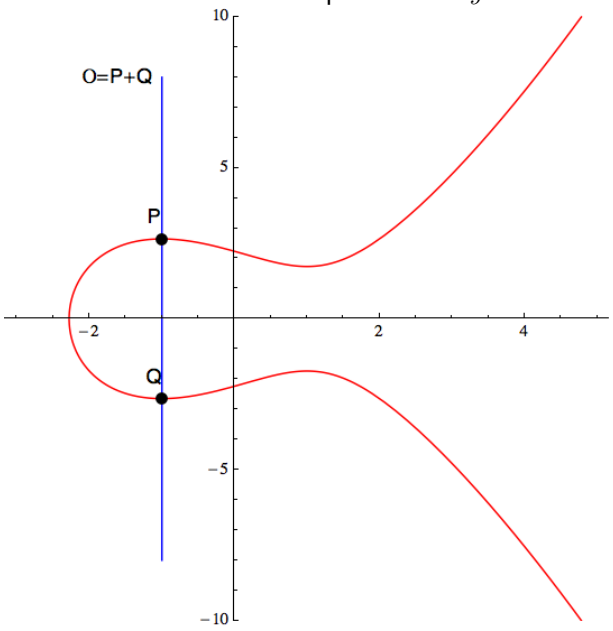


Figure 3.3: Point inversion on elliptic curves. $y^2 = x^3 - 3x + 5$.



Proposition 3.10. Let E be an elliptic curve over $\mathbb{K}$ and $E(\mathbb{K})$ the set of rational points of E. Then $E(\mathbb{K})$ is closed under the addition defined above.

Proof. Take $P_1, P_2 \in E(\mathbb{K})$. Take r to be the line crossing E at P_1 and P_2. If r intersects no other point in E then $P_1 + P_2 = \mathcal{O}$. Otherwise r intersects E in a third point P_3 which is obviously in $E(\mathbb{K})$. Consider now r' the line passing through P_3 and $\mathcal{O}$. This line crosses E in a third point $P_3' \in E(\mathbb{K})$ and we set $P_1 + P_2 = P_3'$. □

We can prove that the set of rational points is actually stronger than just closed under addition, it is a group. We prove this result in the next theorem.

Theorem 3.11. Let E be an elliptic curve over a field $\mathbb{K}$. Then $E(\mathbb{K})$ is an additive abelian group with $\mathcal{O}$ being the identity element.

Proof. In order to prove this theorem we must prove that $E(\mathbb{K})$ is closed under addition, it has an identity element, there exists an inverse for each element in the group and the sum is commutative and associative. We will proof each bullet point of the theorem separately.

1. (Closed under addition) We saw this property in Proposition 3.10.
2. (Identity element) The identity is $id = \mathcal{O}$ by definition.
3. (Existence of inverse) For each element $P \in E$ consider the line intersecting P and $\mathcal{O}$, this line intersects a third point $R \in E$, then the sum $P + R = \mathcal{O}$ and we can conclude that R is the inverse of P.
4. (The sum is commutative) If $P_1 = P_2$ then there is nothing to proof. Otherwise Let us consider the line that passes through P_1 and P_2. This line also passes through the point $-(P_1 + P_2)$, as we have seen in the previous bullet point, but this is the same as $-(P_2 + P_1)$, which implies that $P_1 + P_2 = P_2 + P_1$.
5. (The sum is associative) We want to prove that for $P_1, P_2, P_3 \in E(\mathbb{K})$ we have that,

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

This is the same as saying,

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

To prove the equality we will define the following lines:

- L_1: Line through P_1 and P_2. This line intersects the curve in a third point $-(P_1 + P_2)$.
- L_2: Line through P_3 and P_1+P_2. This line intersects the curve in a 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 a third point $-(P_2 + P_3)$.
- L_1': Line through P_3 and P_2. This line intersects the curve in a third point $-(P_2 + P_3)$.
- L_2': Line through P_1 and $(P_2 + P_3)$. This line intersects the curve in a 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 a 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 the union of three lines). Bézout Theorem tells us that if we have two plane curves A, B with no common components, then they

have at most $deg(A).deg(B)$ points in common. Applying Bézout Theorem, we know that the 9 points that E and C have in common are precisely:

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

The curve C' intersects at the first 8 of the common points between C and E. Therefore C' intersects also at the 9-th common point. On the other hand, applying Bézout Theorem again for C' and E we get the following common points:

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

Hence,

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

□

The next theorem states the sum between points in the rational group through algebraic formulas.

Theorem 3.12. Let E be an elliptic curve over a field K. Let $P_1 = (x_1, y_1)$ and $P_2 = (x_2, y_2)$ be rational points of E. Then we have the following formulas:

1. The inverse of P_1, denoted by $-P_1$, is defined as:

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

2. If $P_1 = -P_2$, then:

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

3. Let $P_1 \neq -P_2$ and $x_1 \neq x_2$, then we consider the following constants:
   - $\lambda = \dfrac{y_2 - y_1}{x_2 - x_1}$.
   - $\mu = \dfrac{y_1x_2 - y_2x_1}{x_2 - x_1} = y_1 - \lambda x_1$.
4. If $P_1 \neq -P_2$ and $x_1 = x_2$, then we consider the following constants:
   - $\lambda = \dfrac{3x_1^2 + 2a_2x_1 + a_4 - a_1y_1}{2y_1 + a_1x_1 + a_3}$.
   - $\mu = \dfrac{-x_1^3 + a_4x_1 + 2a_6 - a_3y_1}{2y_1 + a_1x_1 + a_3}$.

Using the constants presented in 3 and 4 and denoting

$$P_3 = (x_3, y_3) = P_1 + P_2 \neq \mathcal{O}.$$

We have that x_3, y_3 are defined as follow:

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

- $y_3 = -(\lambda + a_1)x_3 - \mu - a_3$.

Proof. Consider the elliptic curve E given by the following curve:

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

1. Consider the point $P = (x_1, y_1) \in E(K)$ and the line $L : X = x_1$. P_1 and $\mathcal{O}$ lie on L. Let us now consider the point $P' = (x'_1, y'_1)$ to be the intersection point of E and L. We will compute P'.

$$\begin{aligned} f(x_1, Y) &= Y^2 + (a_1x_1 + a_3)Y - (x_1^3 + a_2x_1^2 + a_4x_1 + a_6) \\ &= (Y - y_1)(Y - y'_1) \\ &= Y^2 + (-y_1 - y'_1)Y + y_1y'_1. \end{aligned}$$

   Comparing coefficients, we see that

$$y'_1 = -y_1 - a_1x_1 - a_3.$$

   The third intersection point of L with E is therefore

$$P'_1 = (x_1, -y_1 - a_1x_1 - a_3).$$

2. Follows from 1.

3. Consider the line L through P_1 and P_2,

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

   Therefore we have,

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

4. We assume that $P_1 \neq -P_2$ and $x_1 = x_2$, that means that $P_1 = P_2$. The tangent at P_1 is given by:

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

   with 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 that $f_Y(x_1, y_1) \neq 0$. Therefore we write:

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

Using the results obtained in the point 3 and 4 we can conclude the proof of the theorem. The third intersection point of the line L with E is a point $P_3' = (x_3', y_3')$. We now compute this point:

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

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

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

The point $P_3 = (x_3, y_3) = P_1 + P_2$ is $-P_3'$. According to the 1 this has the coordinates:

$$\begin{array}{rcccl} x_3 & = & x_3' & = & \lambda^2 + a_1\lambda - a_2 - x_1 - x_2 \\ y_3 & = & -y_3' - a_1x_3' - a_3 & = & -(\lambda + a_1)x_3 - \mu - a_3. \end{array}$$

□

To end this section we define the multiplication of a rational point by an integer.

Definition 3.13. Let E be an elliptic curve over a filed $\mathbb{K}$, $m \in \mathbb{Z}$ and $P \in E(\mathbb{K})$. We define mP as follow:

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

This notion will be fundamental to define a cryptographic scheme over an elliptic curves.

3.4 Elliptic curves over finite fields

We will now study the case where the ground field $\mathbb{K}$ is finite field. In this case we will identify $\mathbb{K}$ by $\mathbb{F}_q$ where $q \in \mathbb{N}$. Since we are working over a finite field, we can count the number of rational points and

denote by $\#E(\mathbb{F}_q)$ the number of such points. Given an elliptic curve E, we will denote the number of rational points on E by $\#E(\mathbb{F}_q)$. We will also define a quantity called *trace of Frobenius* and a map called *Frobenius map* or *Frobenius endomorphism*. These notions will be useful in due course in our study, as they play a main role counting the number of rational points of an elliptic curve. To ease on notation, we will use $\mathcal{P}$ when referring to the set of prime numbers.

Definition 3.14. Given an elliptic curve E over a finite field $\mathbb{F}_q$, we call *trace of Frobenius* to the quantity t defined by the equation below:

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

Definition 3.15. Given an elliptic curve E over a field $\mathbb{F}_q$, by q^{th} *power Frobenius map*, we understand the following map:

$$\begin{aligned} \varphi : E(\mathbb{F}_q) &\rightarrow E(\mathbb{F}_q) \\ (x, y) &\mapsto (x^q, y^q) \\ \mathcal{O} &\mapsto \mathcal{O}. \end{aligned}$$

The map φ is a group endomorphism, usually mentioned as the *Frobenius endomorphism*.

In order to build algorithms that count the number of rational points on an elliptic curve over a finite field, we will need first to present two important results. The first one is presented in the next proposition and it is an explicit way to calculate the trace of Frobenius.

Proposition 3.16. Given an elliptic curve E over a finite field $\mathbb{F}_q$, for any rational point $P = (x, y)$ over the elliptic curve, we have that the following equation holds:

$$\varphi^2(P) - t\varphi(P) + q(P) = \mathcal{O}$$

where we denote φ by the q-th Frobenius power.

The second important result for counting the number of points is known as *Hasse's Estimate*. This shows an approximation to the order of $\#E(\mathbb{F}_q)$ which uses the trace of Frobenius as we have presented in Definition 3.14.

Theorem 3.17 (Hasse's Estimate)**.**
The trace of Frobenius t of an elliptic curve E over a field $\mathbb{F}_q$ satisfies the following:

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

We will not provide a proof for any of the two results presented before since there proof is out of the scope of this thesis, however the interested reader can find the proofs on Schmitt and Zimmer [2003].

The notion of trace of Frobenius yields a new way of characterizing elliptic curves, the concept of *supersingular* curves. As we will see later, this notion is important in terms of security for cryptosystems based on elliptic curves.

Definition 3.18. Let E be an elliptic curve over a finite field $\mathbb{F}_q$ of characteristic p with $\#E(\mathbb{F}_q) = q+1-t$. The curve is called *supersingular*, if $p|t$.

With these tools at hand, we are able to build an algorithm that count the number of rational points of an elliptic curve. There are several algorithms that do this task, but we will only present one, the so called *Schoof algorithm*. This computes the order of the group, of rational points, modulo small primes and uses the Chinese Remainder Theorem to obtain the exact order. One can find more on the Chinese Remainder Theorem in Fernandes and Ricou [2004]. The q-Frobenius endomorphism presented in Definition 3.15 is also important in this algorithm.

We are now going to take a deeper look at this algorithm. From Hasse's estimate, we know that $|t| \leq 2\sqrt{q}$, where t is the Frobenius trace of an elliptic curve E over a finite field $\mathbb{F}_q$. Consider all primes $2 \leq l \leq l_{max}$, where

$$l_{max} = min\{p' \in \mathcal{P} : \prod_{l \in \mathcal{P}, l \leq p'} l > 4\sqrt{q}\}.$$

From Prime Number Theorem it easily follows that the number of primes needed is $\mathcal{O}(\frac{\log q}{\log \log q})$ and that the size of $l_{max} = \mathcal{O}(\log q)$. Taking all such primes we use the Chinese Remainder Theorem to calculate the value of t.

For $l = 2$ we have to consider two cases: When 2 divides q and when it does not. For the first case, by Theorem 3.5 on Schmitt and Zimmer [2003] if $2|q$ then $t \equiv 0 \mod 2$ if and only if E is supersingular. This case is not of special interest since, as we will see later, supersingular curves can be attacked and therefore are not of interest for cryptography applications. For the remaining case we have that if 2 does not divide q, then $t \equiv 0 \mod 2$ if and only if there exists a non trivial point of order 2. From Silverman [2009] we know that if E is not supersingular and does not have characteristic 2, then is of the type:

$$Y^2 = X^3 + aX^2 + bX + c$$

and therefore it has a non trivial point of order 2.

Let us now recall the Frobenius endomorphism mentioned in Definition 3.14 and Proposition 3.16, let uLet uss also denote by $E[l]$ the $E(\mathbb{F}_q) \mod l$. Take φ_q as the q-th Frobenius endomorphism, then for all $P \in E(\mathbb{F}_q)$ we know that,

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

If there exists a $\tau \in \{0, 1, ..., l-1\}$ such that for $P \in E[l] \setminus \mathcal{O}$ we have,

$$\varphi_q^2(P) + q_l(P) = \tau\varphi_q(P),$$

where $q_l \equiv q \mod l$, then $t \equiv \tau \mod l$.

Schoof Algorithm is presented in 2. The importance of this algorithm is based on the fact that one can calculate the number of rational points in polynomial time, since the complexity Schoof algorithm is $\mathcal{O}(\log^8 q)$. This result can be found in Blake et al. [1999].

3.5 Cryptographic system

Public key cryptographic systems are systems where the enciphering function is public. We will see, as an example, the El Gamal method for elliptic curves. This is a special method for the *discrete logarithm problem* as we present in the next definition.

Definition 3.19. Let $G = \langle g \rangle$ be a cyclic abelian finite group and $h \in G$. The *discrete logarithm problem (DLP)* is the following: Knowing G, g, h and finding $x \in \mathbb{Z}$ such that $h = g^x$.

Definition 3.20. Let E be an elliptic curve over a field $\mathbb{F}_q$ and P a point in $E(\mathbb{F}_q)$, then the *Elliptic Curve Discrete Logarithm Problem* or *ECDLP* on E is the following:

- Instance: Given a base point $P \in E(\mathbb{F}_q)$ and a point $Q \in E(\mathbb{F}_q)$.
- Question: Find an integer $x \in \mathbb{Z}$ such that $xP = Q$, if such an integer exists.

It is interesting to see that if one has access to factorization into primes p_i of:

$$n = |G| = \prod_{i=1}^{k} p_i^{e_i}$$

then one can reduce the ECDLP into a DLP. In fact, one has to reduce G into $G \mod p_i$ for each prime factor p_i of n and then apply the Chinese Remainder Theorem to build a DLP. In Chapter 4 we will find an example of such reduction. Using this reduction one can easily conclude that studying the security of an ECDLP is the same as studying the security of the DLP from which the ECDLP reduces to.

Definition 3.20 yields the cryptosystem that we will study. Let E be an elliptic curve over $\mathbb{F}_q$ and $P \in E(\mathbb{F}_q)$. The ECDLP is the question to know if a given point $Q \in E(\mathbb{F}_q)$, there exists an integer n with $Q = nP$ and if one can compute this n. The main point of this El Gamal method is that the DLP is hard to solve. The interaction in cryptosystem between the sender and the receiver is described as follows:

1. Sender has a secret message m that wants to send to the receiver.
2. Receiver makes $P \in E(\mathbb{F}_q)$ public.
3. Receiver chooses $n \in \mathbb{Z}$ secretly, computes nP and sends it to the sender.
4. Sender picks $k \in \mathbb{Z}$ secretly, computes kP and $m + k(nP)$ and sends it to the receiver.
5. The receiver computes $m + k(nP) - n(kP) = m$ and obtains the secret message.

Later, we will explore this cryptography system. We will be referring to n and k as private keys and as P by public key. There are some necessary conditions for an elliptic curve to be secure. Although we will not study these conditions deeply, we will give some intuition on each of them and we will take them into consideration when building cryptosystems.

An elliptic curve E over $\mathbb{F}_p$ is said to be *anomalous* if $\#E(\mathbb{F}_p) = p$. In this case one can build an isomorphism between $E(\mathbb{F}_p)$ and the additive group of $\mathbb{F}_p$. Knowing this isomorphism, one can build an algorithm that solves the ECDLP in polynomial time. This attack is known as the *anomalous attack*. The interested reader can find more on this attack in Blake et al. [1999].

Using a technique called Weil Pairing, one can embed $E(\mathbb{F}_q)$ in the multiplicative group of the field $\mathbb{F}_{q^k}$ for some integer k. This reduces the ECDLP in $E(\mathbb{F}_q)$ to DLP in $E(\mathbb{F}^*_{q^k})$. For this technique to work, it is necessary that $\#E(\mathbb{F}_q)|q^k - 1$ for all $1 \leq k \leq C$ where C is large enough so that it is computationally infeasible to find a discrete logarithm by brute force attack in $\mathbb{F}^*_C$. All supersingular curves fall under this assumption and therefore they are not secure curves. This is known as the *MOV attack* due to Menezes, Okamoto and Vanstone. One can find more on this attack on Blake et al. [1999].

As we will see in the next chapter the order of the group of rational points has to be divided by a very large prime, otherwise using techniques that we will present later, one can break the system.

These two attacks give us some restrictions to the curves that we want to work with in order to have a secure cryptosystem.

The question now is how to generate a curve E that satisfies all these conditions. The preferred method for generating a *good* curve suitable for cryptographic applications is based on selecting curves at random, and determining group orders until a curve satisfying the desired conditions is found.

The method draws a random elliptic curve E and checks if this is secure against the anomalous attack and the MOV attack. When we find a curve that is secure against these attacks, we check if the size of the group of rational points is factored as $\#E(\mathbb{F}_q) = s'.r$, where s' is an integer smaller or equal to a given integer s and r is a very large prime. This method is outlined in Algorithm 1 .

In the next chapter we will study the security and Kolmogorov complexity of an El Gamal cryptographic system that uses elliptic curves generated via Algorithm 1 .

Chapter 4

Kolmogorov complexity and cryptography

4.1 Introduction

In this chapter we will use the knowledge presented in Chapter 2 and the cryptosystem based on elliptic curves presented in Chapter 3. In the first section we will present an example on how the cryptosystem works and based on this example we will impose restrictions on the cryptographic system in order to ensure security. We will consider a family of elliptic curves, each of which will have a cryptographic system associated to it. On the second section of this chapter we will build a function f that emulates the ECDLP presented in the previous chapter and we will prove that if ECDLP $\notin$ **P**, this function is a Kolmogorov one-way function as presented in Definition 2.28. Finally on the third section of this chapter we will extend the notion of family of Kolmogorov one-way functions associated to a cryptosystem to the notion of trapdoor Kolmogorov one-way function presented in Definition 2.30 and we will see that each f has this property too.

4.2 Security and Kolmogorov complexity

As we have stated, the problem of ECDLP can be reduced to the DLP. Pohlig and Hellman noticed that to solve the DLP in a finite abelian group G, one needs only to solve the DLP in a subgroup of G of prime power order. The original DLP is then solved by applying the Chinese Remainder Theorem.

Let G be a finite abelian group having an order divisible by a prime p, take $Q, P \in G$ and suppose we wish to solve the following DLP. We will denote elements $m \in \mathbb{N}$ by $[m]$, this will help us differentiate natural and elements in G.

$$Q = [m]P.$$

If G has order n, then the problem can be restricted to a subgroup that has the same order as an element

$P \in G$ by solving

$$Q' = [n']Q = [m_0]([n']P) = [m_0]P'.$$

Where $n' = \frac{n}{p^{c-1}}$ and p^c is the largest power of p dividing n. P' is a point of order p. Solving this problem will determine the value of $m_0 \equiv m \mod p$.

Continuing in a similar manner by solving the DLP in subgroups of order p, we eventually determine $m \mod p^c$. After computing $m \mod p^c$ for all primes divisors p of n, the initial solution m to the original DLP can be obtained using the Chinese Remainder Theorem.

Before presenting an example, we will present another technique to solve the DLP. This is called the *baby step giant step method*. Again for a finite abelian group G, we aim to find m such that for $P, Q \in G$ we have:

$$Q = [m]P.$$

This technique uses the Euclidean division to calculate the value of m, since by Euclidean division we know that:

$$m = \lceil \sqrt{n} \rceil a + b$$

where $0 \leq a, b < \lceil \sqrt{n} \rceil$. The goal of the baby step giant step method is finding the values of a and b in the previous equation. We start by rewriting the equation in the following form:

$$Q - [b]P = [a]([\lceil \sqrt{b} \rceil]P).$$

The first step in this method is called baby step. We compute a table for all values R_b of the following equation:

$$R_b = Q - [b]P.$$

This table is sorted and stored in memory so that it can be efficiently searched by using a binary search method.
After having computed the baby steps, the giant steps are performed. For each a we compute the following equation:

$$S_a = [a]([\lceil \sqrt{n} \rceil]P).$$

On each computation of a giant step, it is seen whether S_a occurs in the table. If it does then the value of a and b are recovered.

The complexity of this method is roughly $\mathcal{O}(\sqrt{n})$ as this much time is necessary to compute the baby steps and in the worst case this much time to compute the giant steps. We have ignored the time needed to perform the table look up but in the worst case scenario this search is $\mathcal{O}(\log n)$. However the main problem with this method is that it requires the storage of $\mathcal{O}(\sqrt{n})$ group elements.

We will now present an example of an attack to the elliptic curve discrete logarithm problem that uses these two methods.

Example 4.1. *Consider the following elliptic curve,*

$$E : Y^2 = X^3 + 71X + 602$$

over the finite field $\mathbb{F}_{1009}$*. The group order of* $E(\mathbb{F}_{1009})$ *is* $1060 = 2^2.5.53$*. Consider the following two points:*

$$P = (1, 237), \quad Q = (190, 271)$$

First notice that P *has order* $530 = 2.5.53$ *in the group* $E(\mathbb{F}_{1009})$*. Hence by the above reduction of Pohlig and Hellman, the computation of* m *can be reduced to the computation of* m *modulo 2,5 and 53. Let us start by computing the solution modulo 2.*

One multiplies P *and* Q *by* $\frac{n}{2} = \frac{530}{2} = 265$*. This leads to:*

$$P_2 = [265]P = (50, 0)$$

$$Q_2 = [265]Q = (50, 0).$$

The system to find $m \mod 2$ *is simply given by:*

$$Q_2 \equiv [m \mod 2]P_2,$$

hence $m \equiv 1 \mod 2$.

We will now do the same for $m \mod 5$*. One multiplies* P *and* Q *by* $\frac{530}{5} = 106$*. This leads to the following:*

$$P_5 = [106]P = (639, 160)$$

$$Q_5 = [106]Q = (639, 849)$$

$$Q_5 \equiv [m \mod 5]P_5.$$

Hence $m \equiv 4 \mod 5$.*, since we obtain the symmetric value of the previous result.*

Finally we do the same for modulo 53. We multiply P *and* Q *by* $\frac{530}{53} = 10$ *and obtain the following:*

$$P_{53} = [10]P = (32, 737)$$

$$Q_{53} = [10]Q = (592, 97).$$

Clearly we could use brute force to calculate the value of m *module 53 but instead we will use the baby*

step giant step method.

As we have seen, we have to calculate the value of a and b in the following equation:

$$m = \lceil \sqrt{n} \rceil a + b.$$

In this case we take $n = 53$ and $\lceil \sqrt{53} \rceil = 8$, this means that one needs 8 baby steps. After computing the baby steps one computes one giant step at a time and compares it with the baby steps computed before.
One notices an identity with $a = 6$ and $b = 0$, which leads to:

$$\begin{aligned} m &= 8a + b \\ &= 8.6 + 0 \\ &= 48 \\ \implies & m \equiv 48 \mod 53. \end{aligned}$$

Using the three results and Chinese Remainder Theorem one has that $m = 419$. This example can be found in Blake et al. [1999]. ▷

Based on this example, we can draw some conclusions on the security of the cryptographic system based on elliptic curves. To obtain, in principal a more secure curve, assumptions can be made on the type and size of the private keys as we will see in the following results.

Proposition 4.2. Let us take an elliptic curve E over $\mathbb{F}_q$, p the largest prime dividing $\#E(\mathbb{F}_q)$ and an ECDLP associated to E and m a private key to be used in that curve. If we take $m \equiv 0, 1, -1 \mod p$ then the ECDLP is in **P**.

Proof. Without loss of generality let us assume that $m \equiv 1 \mod p$, where p is the largest prime dividing $\#E(\mathbb{F}_q)$. By Pohlig and Hellman reduction one can find the value of m. This is done in polynomial time. The other cases are similar to prove. □

Example 4.1 establishes a cryptosystem based on a single elliptic curve. From now on, we will be interested in working with a chain or family of elliptic curves. For each elliptic curve we will have a cryptosystem associated as the one presented in Example 4.1.

As before let us take $\mathcal{P}$ to be set of prime numbers in $\mathbb{N}$. For each prime $p \in \mathcal{P}$, we consider the following family of elliptic curves:

$$\{E(\mathbb{F}_p)\}_{p \in \mathcal{P}}$$

where E is generated using Algorithm 1 and for $i, j \in \mathcal{P}$, if $i > j$, then $\#E(\mathbb{F}_i) > \#E(\mathbb{F}_j)$. To ease on notation, from now on, we will denote $E(\mathbb{F}_i)$ by E_i and $\#E(\mathbb{F}_i)$ by n_i.

For each elliptic curve E_i, we will associate a cryptographic system as the one presented in Example 3.5. For each E_i we will fix a pair $(m_i, l_i) \in \mathbb{N}^2$ of private keys and an element $P_i \in E_i$, the public key, and we will denote the cryptographic system by $(E_i, (m_i, l_i), P_i)$. Our goal is to assure that each $(E_i, (m_i, l_i), P_i)$ is secure. For this and based on Example 4.1 we will impose some restrictions on the set of private keys as the following proposition will denote.

Proposition 4.3. Consider $(E_i, (m_i, l_i), P_i)$ and a polynomial time function t such that $K^t(m_i) \in \mathcal{O}(\log \log n_i)$, where $K^t(m_i)$ is the Kolmogorov Complexity of m_i. Then the ECDLP associated to this system is in **P**.

Proof. Let us take $P, Q \in E_i$ such that $Q = [m_i]P$ and consider the polynomial time function t. We denote $\#E_i$ by n_i. We know that

$$K^t(m_i) \in \mathcal{O}(\log \log n_i).$$

Consider the algorithm A that for each candidate x for m_i, tests if

$$Q = xP.$$

There are $2^{c \log \log n_i}$ possible candidates, for $c \in \mathbb{R}^+$. Manipulating this result we get that:

$$\begin{aligned} 2^{c \log \log n_i} &= 2^{\log \log {n_i}^c} \\ &= \log {n_i}^c \\ &= c \log n_i. \end{aligned}$$

Therefore there is a polynomial number of candidates and one can easily conclude that $A \in$ **P**. □

Based on this result, for each elliptic curve E_i, we associate a cryptosytem $(E_i, (m_i, l_i), P_i)$ such that $m_i, l_i \in \mathcal{O}(\log n_i)$. One can also notice that Proposition 4.2 is a particular case of Proposition 4.3.

4.3 A Kolmogorov one-way function candidate

In this section we will build a parallelism between a function f that emulates the ECDLP and a Kolmogorov one-way function presented in Definition 2.28. To this end we will build a function f that simulates the ECDLP, is honest as in Definition 2.21, injective and computable in polynomial time. For the sake of presentation and without loss of generality, we will prove each property focusing on one individual curve. Later in this chapter we will revisit the idea of family of elliptic curves as it plays an important role in the study the trapdoor Kolmogorov one-way functions case.

Let us consider the elliptic curve E and establish as basis for the presentation a cryptosystem $(E, (m, l), P)$. Recall the cryptosytem given in Section 3.5. We need to consider a function f such that for a fixed elliptic curve E we have that for a fixed $P \in E(\mathbb{F}_q)$ and for $m, l \in \mathbb{N}$ that:

$$\begin{aligned} f : E(\mathbb{F}_q) &\rightarrow E(\mathbb{F}_q)^4 \\ x &\mapsto (x + [l][m]P, [l]P, [m]P, P). \end{aligned} \tag{4.1}$$

This function emulates the last interaction of the cryptosystem presented in Section 3.5.

As we said in the beginning of this section, our goal is to build a Kolmogorov one-way function. We will start by showing that f is an honest function.

Proposition 4.4. Function f as in 4.1 is honest.

Proof. Recall that by $|x|$ we understand the length of the binary string representing x. Let us take $P = (x_1, y_1) \in E(\mathbb{F}_q)$ and assume that:

$$|P| = |x_1| + |y_1|.$$

Let us now take $f(x) = (x + [l][m]P, [l]P, [m]P, P)$ and consider the following:

- $[l][m]P = P_1$
- $[l]P = P_2 \in E(\mathbb{F}_q)$.
- $[m]P = P_3 \in E(\mathbb{F}_q)$.

Then the length of the binary string representing $f(x)$ is given by:

$$|f(x)| = |x + [l][m]P| + |P_2| + |P_3| + |P| \leq |x| + |P_1| + |P_2| + |P_3| + |P|.$$

Let us now set $\#E(\mathbb{F}_q) = n$. We know that $|n| \approx \log n$, then for any $Q \in E(\mathbb{F}_q)$, we have that $|Q| \leq 2\lceil \log n \rceil$. Following the same rational, for $m, l \in \mathbb{N}$ and for t polynomial time function we have:

$$K^t(m), K^t(l) \leq \log n.$$

Therefore one can easily see that $|m| \leq \log n$ and the same for l. Then we have that:

- $|P_1| \leq 2 \log n$.
- $|P_2| \leq 2 \log n$.
- $|P_3| \leq 2 \log n$.
- $|P| \leq 2 \log n$.

We will consider $k = 8\lceil \log n \rceil$, then one can easily see that:

1. $|f(x)| \leq |x| + |P_1| + |P_2| + |P_3| + |P| \leq (|x|)^k + k$.
2. On the other hand since $|x| < k$ by definition of k and $|f(x)|$>0, we have that $|x| \leq |f(x)|^k + k$.

The two arguments ensure that f is honest. □

Our next step is to prove that f as in 4.1 is injective. Notice that for a fixed $P \in E(\mathbb{F}_q)$ and for $m, l \in \mathbb{N}$, we have that:

$$f(x) = (x + [l][m]P, [l]P, [m]P, P)$$

where $[l]P$, $[m]P$ and P are independent from x and therefore always take the same value independently of x. Therefore when studying the injectivity of f, one only has to consider the first entry of the output. Let us consider the function:

$$\begin{aligned} f' : E(\mathbb{F}_q) &\rightarrow E(\mathbb{F}_q) \\ x &\mapsto x + [l][m]P \end{aligned} \tag{4.2}$$

where $P \in E(\mathbb{F}_q)$ is a fixed element and $l, m \in n$ are also fixed. If we prove that f' is injective, then f is obviously injective.

Proposition 4.5. Function f' as in 4.2 is injective.

Proof. Let us take $a, b \in E(\mathbb{F}_q)$ such that $a \neq b$ and consider

- $f'(a) = a + [l][m]P$.
- $f'(b) = b + [l][m]P$.

Let us assume by absurd that $f'(a) = f'(b)$, then this implies that

$$a + [l][m]P = b + [l][m]P$$

This is the same as saying that

$$a + [l][m]P - (b + [l][m]P) = \mathcal{O}.$$

From Theorem 3.11, we know that $E(\mathbb{F}_q)$ is associative and commutative, therefore we have that

$$\begin{aligned} & a - b + [l][m]P - [l][m]P = \mathcal{O} \\ \Leftrightarrow \quad & a - b = \mathcal{O} \\ \Leftrightarrow \quad & a = b, \end{aligned}$$

which is a contradiction with our initial assumption. □

One can then easily conclude that f as in 4.1 is injective.

The next property we will show is that f is computable in polynomial time. The computations of $[l][m]P, [l]P$ and $[m]P$ are pre-computed and therefore one does not consider them when calculating the computational power needed for computing the function f. However, they are computed in polynomial

time as the interested reader can see in Chapter IV of Blake et al. [1999].
The computation of $x + [l][m]P$ is given by the formulas presented in Theorem 3.12 and therefore are computed polynomial time. One can then easily conclude that f is computed in polynomial time.

We have seen that f presented in 4.1 is honest, injective and computable in polynomial time.

We are finally at stage where we have all the machinery to build a connection between our function f and the notion of Kolmogorov one-way function presented in Definition 2.28. The next theorem states that under the assumption that the ECDLP $\notin$ **P** then f is a Kolmogorov one-way function. We will consider again our family of elliptic curves and to each curve E_i we will associate a function f_i that behaves as the function f that we have just built.

Theorem 4.6. Consider a family of elliptic curves

$$\{E_i\}_{i\in\mathcal{P}}.$$

For each curve E_i take $n_i = \#E_i$ and consider a function f_i that emulates the ECDLP associated to E_i and is honest, injective, computable in polynomial time and there exists a polynomial m such that $m_i(|x|) = |f_i(x)|$.

If the ECDLP $\notin$ **P** then there is a polynomial time function t and an infinitly set of keys (m_i, l_i) such that:

- $K^t(m_i) > \mathcal{O}(\log\log n_i)$,
- $K^t(l_i) > \mathcal{O}(\log\log n_i)$,
- $K^t(m_i l_i) > \mathcal{O}(\log\log n_i)$,

and for each E_i the function f_i is a Kolmogorov one-way function.

Proof. We will prove this theorem by contraposition. Suppose that there exists $i \in \mathbb{N}$ such that for all $j > i$ one has:

$$(E_j, (m_j, l_j), P_j)$$

where $min\{K^t(m_j), K^t(l_j), K^t(m_j l_j)\} \leq \mathcal{O}(\log\log n_j)$ for some t polynomial time function. Take $E = E_h$, such that $h > i$ and consider the following,

$$\begin{aligned} K^t_f(x|f(x),n) &= K^t_f(x|x+lmP, lP, mP, P, n) \\ &\leq K^t_f(lmP|lP, mP, P, n) \\ &\leq min\{K^t_f(l|mP, lP, P, n), K^t_f(m|mP, lP, P, n)\} \quad (*) \end{aligned}$$

By Proposition 4.3 there exists an algorithm A that solves equation $(*)$ in polynomial time, hence the following is true,

$$K^t_f(x|f(x),n) \in \mathcal{O}(1).$$

One can conclude that f is not Kolmogorov one-way function. □

Corollary 4.7. Under the assumption that ECDLP $\notin$ **P**, f is a deterministic one-way function.

Corollary 4.7 is an immediate result from Theorem 2.29.

4.4 A trapdoor Kolmogorov one-way function candidate

From Example 4.1 we understand that the security of an El Gamal cryptosystem over an elliptic curve is based on the fact that it is hard to solve the logarithm problem. In this section, we will study the case where we are provided with extra information, a trapdoor, that helps us solve the problem.

As a result of last section we will consider the following set up for our problem.

Consider a family of elliptic curves of the following form,

$$\{E_i\}_{i\in\mathbb{N}}$$

where E_i is obtained as in Algorithm 1. we set $\#E_i = n_i$ and for $i, j \in \mathbb{N}$ with $i > j$, we have that $n_i > n_j$. For each elliptic curve we consider an El Gamal cryptosystem represented as:

$$(E_i(m_i, l_i), P_i),$$

such that for a polynomial time function t we have that

- $K^t(m_i) \in \mathcal{O}(\log n_i)$.
- $K^t(l_i) \in \mathcal{O}(\log n_i)$.
- $K^t(l_i m_i) \in \mathcal{O}(\log n_i)$.

Each cryptosystem $(E_i, (m_i, l_i), P_i)$ has a function f_i associated that emulates the last interaction in the cryptosystem and is given by,

$$\begin{aligned} f_i : E_i &\to E_i^4 \\ x &\mapsto (x + [l_i][m_i]P, [l_i]P, [m_i]P, P). \end{aligned}$$

As a result of Section 4.3, we have seen that f_i is injective, honest, computable in polynomial time and if ECDLP $\notin$ **P**, then f_i is a Kolmogorov one-way function.

It is important to note that if one has access to the value of m_i, l_i or $m_i l_i$, then one can easily extract the value of x. We will denote a_i by trapdoor for the system $(E_i, (m_i, l_i), P_i)$.

Since we are working with a family of elliptic curves, we want to define a function, that given an elliptic curve returns the value of a trapdoor of the cryptosystem associated to our elliptic curve. We will consider the following:

$$\begin{aligned} \varphi : \mathcal{P} &\rightarrow \mathbb{N} \\ i &\mapsto a_i \end{aligned} \tag{4.3}$$

Recall the definition of trapdoor Kolmogorov one-way function presented in Definition 2.30. We will show in the next theorem, that if the ECDLP is not in **P**, then the family of functions $\{f_i\}_{i\in\mathbb{N}}$ that emulates cryptosystems are trapdoor Kolmogorov one-way functions.

Theorem 4.8. Assume that the ECDLP is not in **P** then there are infinitely many m_i such that each function of the family of $\{f_i\}_{i\in\mathbb{N}}$ associated to the $(E_i, (m_i, l_i), P_i)$ is a trapdoor Kolmogorov one-way function as in Definition 2.30.

Proof. For each system $(E_i, (m_i, l_i), P_i)$ consider the function f_i that emulates the system. From Theorem 4.6, we know that if ECDLP $\notin$ **P**, then f_i is a Kolmogorov one-way function.

Consider the function φ 4.3 presented before. We will take φ to be our trapdoor function. We can consider the following godelization

$$g : \mathbb{N} \rightarrow \mathcal{P}.$$

To ease on notation we will take

$$f_i(x) = (x + [l_i][m_i]P_i, [l_i]P_i, [m_i]P_i, P_i) = (f_{i,1}, f_{i,2}, f_{i,3}, f_{i,4}).$$

Consider the algorithm A that receives $(g(\varphi(i)), f_i(x))$ as an input and returns:

$$f_{i,1} - g(\varphi(i))f_{i,2} = x.$$

Clearly A runs in polynomial time, hence $K^t_{f_i}(x|f_i(x), \varphi(i), i) \in \mathcal{O}(1)$ and we conclude that $\{f_i\}_{\ in\mathbb{N}}$ is a trapdoor Kolmogorov one-way function family. □

Corollary 4.9. Assume that ECDLP is not in **P**, then each f_i is a trapdoor deterministic one-way function.

Corollary 4.9 is an immediate result from Proposition 2.32.

Chapter 5

Conclusions

5.1 Achievements

In this work we have introduced the concept of trapdoor Kolmogorov one-way function family and proved that for each function of this family, the number of trapdoors is always lower, (by a polynomial fraction), than the number of possible trapdoors. This is a new way of looking at trapdoor one-way functions and relate it with Kolmogorov complexity.

We have also presented a public key cryptographic system based on elliptic curves and we defined a function f that emulates the system.

Based on results from Kolmogorov complexity, we provided restrictions on the set of private keys of the cryptographic system. These restrictions ensure securer system against possible attacks.

With the assumption at hand that ECDLP is not in **P** we have shown that every function that emulates our cryptographic system is in fact a Kolmogorov one-way function. Furthermore we have seen that each of these functions is an element of a family of trapdoor Kolmogorov one-way functions.

These results leads us to an individual way to approach security that might not rely on a computational hardness assumption.

5.2 Future Work

As part of future work, interesting open questions consist in relating the notion of Kolmogorov one-way functions with non deterministic one-way functions for instance, establish relations between Kolmogorov one-way functions and strong and weak one-way functions.

Explore a more restrict result for the number of trapdoors, each trapdoor Kolmogorov one-way function has is in our interest for upcoming researches.

Finding trapdoor Kolmogorov one-way function candidates using different encryption schemes as well as using Kolmogorov complexity to ensure a more secure system is another possibility for future work.

Bibliography

Luis Filipe Coelho Antunes, Armando Matos, Alexandre Pinto, Andre Souto, and Andreia Teixeira. One-way functions using algorithmic and classical information theories. *Theory Comput. Syst.*, 52(1): 162–178, 2013.

R. Balasubramanian. *Elliptic curves and cryptography.*, pages 325–345. New Delhi, IN: Hindustan Book Agency, 2003. ISBN 81-85931-42-9.

Ian F. Blake, Gadiel Seroussi, and Nigel Paul Smart. *Elliptic curves in cryptography*. London Mathematical Society lecture note series. Cambridge University Press, Cambridge, New York, 1999. ISBN 0-521-65374-6. URL `http://opac.inria.fr/record=b1095617`. Autres tirages : 2000, 2001, 2002, 2004.

R.L. Fernandes and M. Ricou. *Introdução à álgebra*. Ensino da ciência e da tecnologia. IST Press, 2004. ISBN 9789728469276.

Oded Goldreich. *The Foundations of Cryptography - Volume 1, Basic Techniques*. Cambridge University Press, 2001. ISBN 0-521-79172-3.

H. Imai and Y. Zheng. *Public Key Cryptography: First International Workshop on Practice and Theory in Public Key Cryptography, PKC'98, Pacifico Yokohama, Japan, February 5-6, 1998, Proceedings*. Lecture Notes in Artificial Intelligence. Springer, 1998. ISBN 9783540646938.

Troy Jeffrey Lee. Kolmogorov complexity and formula size lower bounds. 2006.

M. Li and P.M.B. Vitányi. *An Introduction to Kolmogorov Complexity and Its Applications*. Texts in computer science. Springer, 2009. ISBN 9780387498201.

L. Longpré, O. Watanabe, and Mass.). College of Computer Science Northeastern University (Boston. *On Symmetry of Information and Polynomial Time Invertibility*. Technical report (Northeastern University (Boston, Mass.). College of Computer Science)). College of Computer Science, Northeastern University, 1992.

Luc Longpre and Sarah Mocas. Symmetry of information and one-way functions. *Inform. Proc. letters*, 46:95–100, 1993.

S. Schmitt and H.G. Zimmer. *Elliptic Curves: A Computational Approach*. De Gruyter studies in mathematics. Walter de Gruyter, 2003. ISBN 9783110168082.

J.H. Silverman. *The Arithmetic of Elliptic Curves*. Graduate texts in mathematics. Springer, 2009. ISBN 9780387094946.

Andre Souto, Luis Antunes, Paulo Mateus, Andreia Teixeira, and Sophie Laplante. Witness hiding without extractors or simulators. *Personal communication*.

D.R. Stinson. *Cryptography: Theory and Practice, Third Edition*. Discrete Mathematics and Its Applications. Taylor & Francis, 2002. ISBN 9781584882060.

Appendix A

Algorithms

The algorithms presented through out the text are presented below.

Algorithm 1 Generating an Elliptic Curve

INPUT: A large finite field $\mathbb{F}_q$, a small positive integer s.
OUTPUT: An elliptic curve E over $\mathbb{F}_q$ such that $E(\mathbb{F}_q) = s'r$.
1. Draw E at random with coefficients in $\mathbb{F}_q$.
2. Determine $\#E(\mathbb{F}_q)$
3. Check the conditions for the MOV and anomalous attack. If any of these fail go back to step 1.
4. Attempt to factor $\#E(\mathbb{F}_q)$ in *reasonable* time. If the attempt fails go back to step 1.
if $\#E(\mathbb{F}_q) = s'.r$ for r prime and $s' \leq s$ **then**
  Return E.
else
  Go back to step 1.
end if

Algorithm 2 *Shoof algorithm*

INPUT: The prime power q and E elliptic curve defined over $\mathbb{F}_q$.
OUTPUT: The trace of Frobenius t
$l_{max} = min\{p \in \mathcal{P} : \prod_{l \in \mathcal{P}, l \leq p} l > 4\sqrt{q}$
if $2|q$ **then**
  if E is supersingular **then**
    $t \equiv 0 \mod 2$
  else
    $t \equiv 1 \mod 2$
  end if
else
  if $\#E[2] = 0$ **then**
    $t \equiv 0 \mod 2$
  else
    $t \equiv 1 \mod 2$
  end if
end if
for all $l \in \mathcal{P}, 3 \leq l \leq l_{max}$ **do**
  Take a random point $P \in E[l] \setminus \mathcal{O}$
  Compute $\varphi_q^2(P) + q_l(P)$ with $0 \leq q_l \leq q, q_l \equiv q \mod l$
  for $\tau = 0$ to l **do**
    if $\tau\varphi_q(P) = \varphi_q^2(P) + q_l P$ **then**
      $t_l = \tau$
    end if
  end for
end for
Use Chinese Remainder Theorem to determine t with $|t| \leq 2\sqrt{q}$ and $t \equiv t_l \mod l$ for all $l \in \mathcal{P}, 2 \leq l \leq l_{max}$.
return t