Elliptic curve cryptography

Author: Camilo Núñez

Master 2 Informatique et Mathématiques Discrètes
Faculty of Sciences
Aix-Marseille Université
France
November 2023

Abstract:

Contents

1 Introduction

The objective of this report is to present 3 crypto systems that make use of elliptic curves over finite fields. These crypto systems have in common the fact that their safety is based on the complexity of the discrete logarithmic problem over elliptic curves.

The last section was added in order to provide a set of tools that would permit an interested reader develop experiments with the crypto systems making use of an appropriate software.

Most of the material of this report was based on [2].

2 Preliminaries

The first theorem we state is a fundamental result in finite fields.

Theorem 2.1. *Let $q = p^r$ such that p is prime number and $r \geq 1$. There exists a finite field of cardinal q, unique up to isomorphism. Additionally, the elements of F_q are roots of the polynomial $X^q - X \in \mathbb{Z}/p\mathbb{Z}[X]$.*

Since this theorem is not part of the objectives of this report, but rather a tool, the proof will not be given. A proof of it can be found in part 1 of [1].

Remark 2.2. *On thing however, that must be understood from the proof of theorem 2.1, is the construction of the field of cardinal p^r. One construction, which can be found in part 1 of [1], takes an arbitrary irreducible polynomial $g(X)$ of degree r in F_p, and states that $F_p[X]/(g(X))$ is a field of cardinal p^r. Since the theorem also states that this field is unique up to isomorphism, we can freely use this characterization for F_q.*

From now on, whenever we refer to a finite field F_q it will be implicit that $q = p^r$ for some prime p.

[mod n]

Algorithm 2.3. *Given $x, y \in \mathbb{Z}$, the* **Extended Euclidian algorithm** *returns $a, b \in \mathbb{Z}$ such that $ax + by = (x, y)$.*

Extended Euclidean algorithm

Input: $x, y \in \mathbb{Z}, x \geq y \geq 0, x > 0$.

Output: (x, y) and a, b such that $ax + by = (x, y)$.

1. $(a, b, g, u, v, w) \leftarrow (1, 0, x, 0, 1, y)$
2. *while* $w > 0$ *:*
3. $\quad q = \lfloor g/w \rfloor$
4. $\quad (a, b, g, u, v, w) \leftarrow (u, v, w, a - qu, b - qv, g - qw)$
5. *return* y

Remark 2.4. *We point out three things:*

- *We can also use this algorithm to compute the inverse of $a \in F_p, a \not\equiv 0$ [mod p], p prime. This is because if we input (a, p) we obtain a, b such that $ax + py = 1$ and therefore, $ax \equiv 1$ [mod p].*
- *This algorithm was studied in class, so we know the complexity when computing a^{-1}[mod p] is $log^3(p)$.*
- *Following the same principle as in the first item of this remark, this algorithm can also be applied to compute the inverse of an element $p(X) \in F_q$, since by remark 2.2, $F_q \cong F_p/(g(X))$ with $g(X)$ irreducible of degree r.*

Theorem 2.5. *Let F_q be a finite field with $q = p^r$ such that p is prime, and let $a, b \in \mathbb{F}_q$. We have that every basic operation[1] between a and b takes $O(log^3(q))$ bit operations.*

Proof. Let g be an irreducible polynomial of degree r with coefficients in F_p. An element of F_q, is a polynomial with coefficients in $F_p = \mathbb{Z}/p\mathbb{Z}$ regarded modulo $g(X)$.

To multiply two such elements, we multiply the polynomials. This requires $O(r^2)$ multiplications of integers modulo p (and some additions of integers modulo p, which take much less time)[2]. And then, divide the obtained product by $g(X)$, taking the remainder polynomial as our answer. The polynomial division involves $O(r)$ divisions of integers modulo p and $O(r^2)$ multiplications of integers modulo p. Since a multiplication modulo p takes $O(log^2(p))$ bit operations[3], and a division modulo p (using the Euclidean algorithm, see 2.4) takes $O(log^3(p))$ bit operations, the total number of bit operations is: $O(r^2 + r^2log^2(p) + rlog^3(p)) = O(r^2log^2(p) + rlog^3(p)) = O((rlog(p))^3) = O(log^3(q))$.

To prove the same result for division, it suffices to show that the reciprocal of an element of $F_p/(g(X))$ can be found in time $O(log_3 q)$. As it was seen in the previous paragraph, using the Euclidean algorithm for polynomials can be used to compute the inverse of a polynomial in $F_p/(g(X))$ with complexity $O(log^3(q))$.

□

Algorithm 2.6. Repeating doubling method. *Let F be a field, $x \in F$ and $n \in \mathbb{N}$. We will present a way to compute nx that takes $O(log(n))$ steps.*

First, we factorize n as following:

- *If n is divisible by 2, divide it by 2.*
- *If n is not divisible by 2 subtract 1 and divide by 2.*
- *Repeat until arriving to 1.*

[1] Addition, subtraction, multiplication, division.

[2] This is because every element of $F_p/(g(X))$ can be regarded as the residue of a polynomial in $F_p[X]$ divided by $g(X)$ in virtue of remark 2.2.

[3] Multiplication and division of n and m in $\mathbb{Z}$ have complexity $O(log(n)log(m))$.

After obtaining a factorization of n this way, call it f, we proceed to compute the factorization f replacing the ones by x's and multiplying the innermost 2 by x as well.

Example 2.7. *Let $F = \mathbb{Q}$ and $n = 200$. If we factorize n as described in the previous algorithm, we obtain $n = 2(2(2(1 + 2(2(2(1 + 2))))$. Then, if we want to compute nx, we compute $2(2(2(x + 2(2(2(x + 2x))))$.*

Remark 2.8. *Clearly, the amount of calculations required to perform nx using the repeated doubling method is $O(log(n))$. This can be understood in the way that we have to perform at most twice as many operations (multiplications by 2 and adding 1) as many times we can divide n by 2.*

Algorithm 2.9. *Given a prime p, and $a \in F_p$ such that $a \not\equiv 0[p]$. We can follow the following algorithm to find its squared root.*

First, we check if the the number is a quadratic residue module p using **Euler's criterion***:*

$$a \text{ is a quadratic residue if, and only if, } a^{\frac{p-1}{2}} \equiv 1[p].$$

We can use for example the repeating doubling method.

Then, if a is in fact a quadratic residue we proceed to compute its square root the following way:

1. *We find a non quadratic residue modulo p, n using Euler's criterion.*
2. *We write $p - 1$ in the form $s2^\alpha$ such that s is odd.*
3. *We compute $b = n^s$ and $r = a^{\frac{s+1}{2}}$.*
4. $g \leftarrow r$
   for j in range $(0, \alpha - 2)$:
   if $(gr)^2 a^{-1} = -1$:
   $g = gg^{2^i}$
5. *Return g.*

With respect to the complexity, it's easy to see that every line from 2 to 4 takes time $O(log^3(p) + \alpha^2 log^2(p))$, and since $\alpha < log_2(p)$, lines 2 to 4 take time $O(log^4(p))$.

Regarding line 1, there is no known algorithm that can find a non residue modulo p in polynomial time. This means we must conform ourselves with trying element by element of F_p and using Euler's criterion. Every element of F_p has 50% chance of being non residue, so we may be lucky we searching it.

The good news is that we just need to find 1 and use it every time we need to find a squared root.

Example 2.10. *We give the pseudo code of the* **Rabin-Miller** *test, which is a probabilistic algorithm that, given a number q and $k \in \mathbb{N}$, returns either "q is prime with probability $1 - \frac{1}{4^k}$" or "q is not prime".*

Input: *$n \in \mathbb{N}, n$ odd and $n > 9, n - 1 = 2^s t$, and an integer $k \geq 1$.*

Output: *0 if n is composite, 1 if n is prime with probability $\geq 1 - (1/4)^k$.*

for $0 \leq i < k$ *:*

Choose a random $1 \leq a \leq n - 1$ *prime to* n

$b \leftarrow a^t \bmod n$

if $b = \pm 1$ *:*

    continue

$test = 0$

for $1 \leq j < s$ *:*

    $b \leftarrow b^2 \bmod n$

    if $b = -1$ *:*

        $test = 1$

        break

    if $test = 0$ *:*

        return 0

return 1

Remark 2.11. *Since we studied this algorithm in class, we know its complexity is $O(klog^3(n))$.*

3 Basic facts

Definition 3.1. *Let K be a field. We define an* **elliptic curve** *E over K as the set of points in K satisfying an equation of the form*

$$y_2 + a_1xy + a_3y = x_3 + a_2x^2 + a_4x + a_6$$

together with a single element denoted O and called the "point at infinity".

An elliptic curve must also satisfy the following condition: Let $F(x, y) = 0$ be the implicit equation for y as a function of x. We require that every point satisfying E is **non singular***, that is, having one of its partial derivatives $\partial F/\partial x, \partial F/\partial y$ not equal to zero.*

Depending on the characteristic of K, simpler definitions of elliptic curves can be given.

If K is a field of characteristic > 3, an elliptic curve over K is the set of points (x, y) with $x, y \in K$, together with O, which satisfy the equation

$$y^2 = x^3 + ax + b \tag{1}$$

where $x^3 + ax + b$ is a cubic polynomial with no multiple roots.

If K is a field of characteristic 2, an elliptic curve over K is the set of points satisfying an equation of type either

$$y^2 + cy = x^3 + ax + b \tag{2}$$

or else

$$y^2 + xy = x^3 + ax^2 + b \tag{3}$$

(here we do not care whether or not the cubic on the right has multiple roots) together with O.

If K is a field of characteristic 3, an elliptic curve over K is the set of points satisfying the equation

$$y^2 = x^3 + ax^2 + bx + c \tag{4}$$

(where the cubic on the right has no multiple roots) together with O.

Remark 3.2. *The definitions of elliptic curves given in the particular cases are all equivalent to the general definition.*

3.1 Elliptic curves over finite fields

Let K be a finite field F_q of $q = p^r$ elements.

Remark 3.3. *Given an elliptic curve E defined over F_q, there is an algorithm, first described by René Schoof, that computes $|E|$ in $O(log^8(q))$. Since the algorithm is not simple and is not part of the objectives of the report, we did not include it.*

Definition 3.4. *Given an elliptic curve $E := y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6$ over $\mathbb{Q}$, and p a prime number such that no a_i in E in its reduced form has p in its denominator. We define E modulo p as the elliptic curve equation given by replacing each coefficient $a_i = \frac{p_i}{q_i}$ of E by $p_i[mod\ p]q_i^{-1}[mod\ p]$. It is straightforward, using the determinant of a polynomial[4], that $E[mod\ p]$ is in fact a elliptic curve over F_p except for a finite number of primes.*

We proceed to give an example of an elliptic curve over a finite field.

Example 3.5. *Let $q = 101$ ($F_q = \mathbb{Z}/101\mathbb{Z}$). We proceed to find a polynomial $p(x)$ with the form of (1), with no multiple roots. That is, $p(x) = x^3 + ax + b$ such that $4a^3 + 27b^2 \neq 0$. If we take $a = 6$ and $b = 5$ we obtain a non zero discriminant, so we consider the equation $y^2 = x^3 + 6x + 5$ as E.*

It is straight forward to check that $(2, 5)$ is a point of E.

[4]The determinant of a polynomial can be used to know the multiplicity of its roots

We proceed now to explain in which sense elliptic curves form an abelian group.

Example 3.6. *Let F_q be a finite field and E a elliptic curve over it. We state that $(E \cup \{O\}, +)$ is a group under the following operations:*

- *The added element O operates as the neutral element.*
- *We define $-(x_1, y_1) = (x_1, -y_1)$, that is, $(x_1, y_1) + (x_1, -y_1) = O$.*
- *Let $(x_1, y_1), (x_2, y_2) \in E$. We define*

$$(x_1, y_1) + (x_2, y_2) = ((\frac{y_2 - y_1}{x_2 - x_1})^2 - x_1 - x_2, -y_1 + (\frac{y_2 - y_1}{x_2 - x_1})(x_1 - x_3)$$

To verify that $E \cup \{O\}$ is closed under $+$, it suffices to show that the equation defining E holds for $(x_1, y_1) + (x_2, y_2)$. Associativity and commutativity follow form the fact that F_q is a field.

Remark 3.7. *It can be shown that the structure of the group formed by the point of an elliptic curve is the product of $p-$primary groups of the form $\mathbb{Z}/p^\alpha\mathbb{Z} \times \mathbb{Z}/p^\beta\mathbb{Z}$, where the product is taken over primes dividing N (here $\alpha \geq 1$, $\beta \geq 0$).*

4 The cryptosystems

Definition 4.1. *The elliptic curve analog of multiplication in $\mathbb{F}_q*$ is addition. Thus, multiplication by $k \in \mathbb{Z}$ is the analog of the k^{th} power. Given $e \in E$, we say that the elements of the form $ke = e + \cdots + e$(k times) are* **multiple of a e**.

Theorem 4.2. *Suppose that an elliptic curve E is defined by a Weierstrass equation (equation (1), (2) or (3) in the last section) over a finite field $\mathbb{F}_q$. Given $e \in E$, the coordinates of kP can be computed in $O(log(k)log^3(q))$ bit operations.*

Proof. Using the equations that were constructed din example 3.6 for the addition of elements in E, we can infer that adding two elements of E requires $k < 15$ basic operations (addition, multiplication, subtraction, division) in F_q. From theorem 2.5 we know that each one of the k basic operations will take $O(log^3(q))$ bit operations, and thus, the total amount of basic operations will take $O(log^3(q))$ bit operations.

Finally, from remark 2.8 we know that there are $O(log(k))$ steps in the computing of kP. Thus, the total amount of bit operations is $O(log(k))O(log^3(q)) = O(log(k)log^3(q))$.

□

4.1 Embedding plain texts

We will now see how to encode a message m as a set of points of a given curve E. Notice that any user of the system should be able to recognize m just by having it's encoding as a set of points of E. In other words, an encoding is not a ciphering, and encoding is just the first step before ciphering since we need to present our messages of our abelian group of interest.

Remark 4.3. *1. There is no known polynomial time (in $\log(q)$) deterministic algorithm that for writing down a large number of points of E.*

2. There are probabilistic algorithms for which the chance of failure is very small.

Example 4.4. *We proceed to describe a probabilistic algorithm that takes as input a message m and output with very high probability a consistent encoding of m.*

Let m be the desired message to be transmitted. For the ease of the encoding we will assume m is an integer such that $0 \leq m < M$ for some fixed $M \in \mathbb{N}$ (M is a bound for every message). We want to be able to encode every message (integer) between 0 and $M-1$.

Let $k \in \mathbb{N}$ such that we're satisfied with a probability of failure of $\frac{1}{2^k}$ when were encoding our message. In practice $k = 40$ is usually enough.

Finally, we must also establish from the beginning a finite field F_q, $q = p^r$, such that $Mk < q$, and an elliptic curve $E := y^2 = x^3 + ax + b$ over F_q.

1. We write the integers $mk + j$ where $1 \leq j \leq k$, $0 \leq m < M$. We denote $\mathcal{A} = \{mk + j : 1 \leq j \leq k \, 0 \leq m < M\}$.

2. We define a injective function $f : \mathcal{A} \to F_q$ in the following way: Let $a \in \mathcal{A}$, then a has the form $a = mk + j$.

   (a) We write $a = (a_0 \dots a_{r-1})_p$ in base p taking into account r components (this is enough since $mk + s \leq q = p^r$).

   (b) $(a_0 \dots a_{r-1})_p$ can be regarded as a polynomial of degree $r-1$ on F_p. By theorem 2.1, we know $(a_0 \dots a_{r-1})_p$ is an element of F_q.

3. For each $j \in \{0, \dots, k\}$ and $m \in \{0, \dots, M-1\}$ we compute $h(x) = x^3 + ax + b$ with $x = f(mk + j)$ and we find the square root of $h(x)$, if it exists, following the algorithm explained in 2.9.

Remark 4.5. *We point out the two following facts about our encoding:*

- *Notice that for every $m \in \{0, \dots, M-1\}$, the algorithm has k chances of finding a point in the elliptic curve to encode m.*

- *Given an encoding of m, one can always decode m from the point (x, y) by the formula $m = \frac{\overline{x}-1}{k}$, where $\overline{x}$ is the integer corresponding to x under the 1-to-1 correspondence between integers and elements of F_q.*

4.2 Discrete logarithm on E

Definition 4.6. *Let $\mathbb{F}_q$ be a finite field, E an elliptic curve over F_q, and $B \in E$. The* **discrete log problem** *on E to the base B is, given $P \in E$, the problem consisting of finding $k \in \mathbb{Z}$ such that $kB = P$.*

Remark 4.7. *Like in the case of* **discrete log problem** *in $\mathbb{Z}/n\mathbb{Z}$, there is no algorithm with a reasonable complexity that can solve a general case of this problem. However, there are some instances of this problem in which there are efficient algorithms to solve the* **discrete log problem on** *E. This is why it is also be important to know the elliptic curve and finite field where we are working on in order to keep our crypto system safe.*

4.3 Analog of the Diffie-Helman key exchange

Let A (Anna) and B (Bernard) be two people that want to share a key privately through a given system where there are multiple other members who are trying find out to find out what this key is.

We proceed to explain a protocol Anna and Bernard can follow in order to share a key privately. This protocol is known as **Diffie-Hellman key exchange method variant for elliptic curves.**

1. A and B choose publicly a finite field F_q and a elliptic curve E over it.
2. A and B publicly choose a point $P \in E$.
3. A chooses privately $a \in \mathbb{Z}$, and make public the element aP. Also B chooses a random $b \in \mathbb{Z}$ and makes public bP.
4. A and B compute abP and this is their secret key.

Remark 4.8. *Any member of the system trying to compute abP would need to be able to compute $log_P(aP)$ or $log_P(bP)$. This means that the security of this protocol is based on the complexity of the discrete logarithm problem over elliptic curves.*

4.4 Analog of Massey-Omura

We proceed to describe a public key crypto-system that will let A send a message m to B without sharing a private key. We suppose that the process described in example 4.4 of embedding m as a point P_m of E is of public knowledge, as well as the elliptic curve E and the finite field F_q where E is defined.

We also suppose the number N of points of E has been computed previously say, with Schoof's algorithm (see remark 3.3), and is also of public knowledge.

1. A and B privately choose numbers a and b such that $1 \leq a, b \leq N$ and $g.c.d(a, N) = g.c.d(b, N) = 1$.

2. Using the euclidean algorithm, both compute a' and b' such that $aa' = 1(mod\ \ N)$ and $bb' = 1(mod\ \ N)$.

3. A sends the point aP_m, B returns the point baP_m to Alice.

4. A sends the point $a'baP_m = bP_m$, and B is already able to infer P_m from bP_m since he knows b'.

Remark 4.9. *Massey-Omura's crypto system's safety is also based in the complexity of the discrete logarithm problem over elliptic curves.*

4.5 Analog of ElGamal

We proceed to describe a second method that permits A send P_m to B safely.

Again, we suppose that the process described in example 4.4 used to encode m as a point P_m of E is of public knowledge, as well as the elliptic curve E and the finite field F_q where E is defined.

However, this time there is no need to know the number N of points of E.

1. A chooses privately $a \in \mathbb{Z}$ and B chooses publicly $b \in \mathbb{Z}$.

2. A sends to B the pair of points $(aP_m, P_m + abP_m)$ to B.

3. B multiplies the first component by b and obtain abP_m and then subtract it to the second component to obtain P_m.

Again, the discrete discrete logarithm problem over elliptic curves guarantees safety here.

5 How to easily choose E and $e \in E$.

This section was included in order to leave a concrete support for a reader willing to develop the crypto systems described in section 4. Given a finite field F_q it is not a trivial question to compute the equation of an elliptic curve and neither it is to find a element of a given elliptic curve.

If one wants to implement one of the previous crypto systems, this is a fundamental problem that must be solved algorithmically. We begin this section by doing so.

Algorithm 5.1. Random selection of E, B. *We describe an algorithm that returns an elliptic curve E and an element B of it (which can be used as the base of the discrete logarithm).*

Take an arbitrary finite field F_q. We assume F_q of characteristic grater than 2 for the ease of the construction[5]*.*

1. *Arbitrarily choose 3 elements $a, c, d \in F_q$.*

[5] The equation of elliptic curves in this cases is simpler.

2. *Set $b = d^2 - c^3 + ac$ and check if the equation $x^3 + ax + b$ has no multiple roots, which is equivalent to have $4a^3 + 27b^2 \neq 0$.*

3. *If $4a^3 + 27b^2 \neq 0$ is not met make another choice of $a, c, d \in F_q$ and repeat.*

4. *if $4a^3 + 27b^2 \neq 0$ is met, take as elliptic curve $y^2 = x^3 + ax + b$, and (c, d) is a point of the same curve.*

We already know from remark 3.3 that the number of elements in the curve we just created can be computed effectively.[6]

This question turns out to be particularly important since the complexity of the discrete logarithmic problem on E may depend on $|E|$. In particular, if for example E is cyclic and B is a generator of E, if the prime factorization of $|E|$ does not contain a 'large' prime, the security of all the previously described crypto systems would be compromised since the **pohlig-Silver-Hellman** method can be used to solve the discrte logarithm problem in this case.

Now our goal will be to create an elliptic curve E along with a base $B \in E$ such that B has a large enough prime factor so we can use the previous crypto systems safely.

In order to do this, we will have to go through some other constructions before we reach our final solution. We will see first another method to compute a an elliptic curve along with a base.

Algorithm 5.2. Reducing a global (E, B) modulo p. *Let E be a elliptic curve defined over the field of the rational numbers and $B \in E$ an element of infinite order.*

1. *Take a large prime p and consider the elliptic curve E' over F_p and $B' \in E'$ as E and B modulo p respectively. We know that E' will be an elliptic curve over F_p and $B' \in E'$ for every prime p except for a finite number of cases.*

Since this new method permits us create an infinite amount of elliptic curves along with an element in it, we propose the following solution for the safety of our system: If we find a prime p such that $|E'|$ is itself also prime, E' would be cyclic, and every element of E' except for O would be a base of E'. This would imply 2 important things:

- The fact that $|E'|$ is prime means that we don't have to worry anymore about having only small primes in our prime factorization.
- The B' given in the construction 5.2 would always be safe for out crypto systems.

We proceed now to describe the process of constructing such pair (E', B').

[6] This algorithm requires time $O(log^6(q))$.

Algorithm 5.3. *We just need to keep trying with different primes until we find a curve E' such that $|E'|$ is prime, which can be tested with the primality test described in theorem 2.10.*

Remark 5.4. *There is obviously the question of how long it would take until we find the desired elliptic curve. Eventhough there is no answer for this, it is conjectured that that the probability of choosing the correct be is $O(\frac{1}{log(p)})$.*

6 Conclusions

The nature of elliptic curves makes the development of a safe crypto system based on them a harder task (see for example the comment after construction 5.1). However, in section 5 we provided tools that let us efficiently overcome these challenges.

It must be remarked as well that for every finite field, many elliptic curves can be defined, and thus, many group structures can be defined through them. This richness in terms of possibilities of choices for spaces in which we can define the discrete logarithm problem could be an advantage in security or in future mathematical research.

References

[1] Marc Hindry. *Cours d'arithmétique (M1)*. 2005.

[2] Neal Koblitz. *A course in number theory and cryptography*, volume 114. Springer Science & Business Media, 1994.