



# Elliptic Curves and Cryptography

*A basic text on cryptography and elliptic curves as a mathematical concept as well as their role in cryptography and how to calculate on them efficiently*

David Ahlqvist

Examensarbete i matematik, 15 hp

Handledare: Stephan Wagner

Examinator: Martin Herschend

Juli 2022



# Contents

<b>1</b>	<b>Introduction</b>	<b>2</b>
1.1	Background . . . . .	2
1.2	Basics of cryptography . . . . .	2
1.3	Symmetric algorithms . . . . .	3
1.3.1	Stream Ciphers . . . . .	4
1.3.2	Block Ciphers . . . . .	5
1.4	Asymmetric algorithms . . . . .	5
1.4.1	The concept of a one-way function . . . . .	6
1.4.2	Diffie Hellman key exchange and the discrete logarithm problem . . . . .	6
<b>2</b>	<b>Algebraic and number-theoretical concepts</b>	<b>7</b>
2.1	Groups and fields . . . . .	7
2.2	Groups under modular multiplication and $\mathbb{F}_p^*$ . . . . .	9
2.3	Cyclic groups . . . . .	9
<b>3</b>	<b>Introduction to Elliptic Curves</b>	<b>11</b>
3.1	What is an Elliptic Curve? . . . . .	11
3.2	Arithmetic on an Elliptic Curve over $\mathbb{R}$ . . . . .	11
3.3	Elliptic Curves as a cyclic group . . . . .	14
<b>4</b>	<b>Elliptic Curve Cryptography</b>	<b>15</b>
4.1	How and why it works . . . . .	15
4.2	Why it is secure . . . . .	17
4.3	Choosing the correct curves . . . . .	18
<b>5</b>	<b>Effective calculation of exponents</b>	<b>19</b>
5.1	Double and Add, Square and Multiply . . . . .	19
5.2	Non-Adjacent Form . . . . .	21

# 1 Introduction

## 1.1 Background

Cryptology is something that has existed since 2000BC in ancient Egypt but most of its evolution as a scientific subject has been in recent time. As described in [9, Section 1.1] cryptology splits mainly into two parts, cryptography and cryptanalysis, the former focusing on building and creating secure cryptosystems to hide the real message and the latter focusing on methods to break these systems. It is fairly obvious that both of these are very important to each other, as people trying to break a cryptosystem and failing is what lets us know that the system is secure.

The study of cryptanalysis and efficiency of its methods for breaking a system lays a ground for one of the most important aspects in modern-day security, where decisions such as what key-length and what system to use are heavily influenced by how powerful current cryptanalysis methods are.

In this paper we will focus mainly on the cryptography side of things, but of course certain cryptanalysis aspects will have to be mentioned throughout the paper.

During the 20th and 21st century multiple cryptosystems have been created, some of which have failed and some of which have succeeded and are still used today, examples of successful systems include AES, 3DES, RSA, Elgamal and Elliptic Curve Cryptography (ECC).

## 1.2 Basics of cryptography

Cryptography at its core is about being able to take a piece of information, somehow distort it such that when it is sent over an unsecure/public network it is unreadable to anyone who might access it. Then when it arrives at the intended recipient, that recipient can then use some method to decrypt and access the information in its original form.

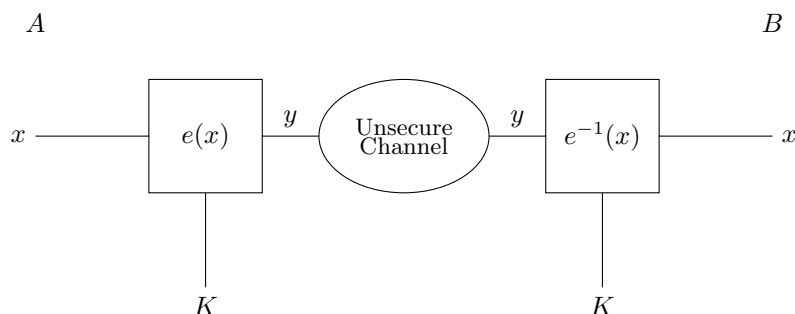


Figure 1: A very basic representation of the encryption-decryption process. Here we see recipient  $A$  send a message  $x$  to  $B$  using encryption method  $e$  and key  $K$ .

Obviously, most ciphers we see used today are a bit more convoluted and even if we can use Figure 1 as a “most zoomed out version” of the process it hardly tells the full story.

Cryptography branches into two main sections of algorithms, symmetric and asymmetric. We will focus mainly on asymmetric ciphers in this text, especially elliptic curve cryptography, but naturally we want to take a look at both.

### 1.3 Symmetric algorithms

A symmetric algorithm is what people tend to think of when they hear cryptography. As described in [7, Section 3.1], symmetric algorithms usually use the same key for encryption and decryption, or the decryption key is easily derived from the encryption key or vice versa and thus the importance of getting the key from  $A$  to  $B$  without anyone else finding it out is really important as the entire system relies on the key.

There are many symmetric algorithms, some popular ones are Twofish, IDEA, DES, 3DES and AES. There exist many more so these are just a few examples, the most widely used symmetric algorithm today is AES, which one can read more about in [9, Chapter 4].

As described by [9, Section 2.1.1], symmetric cryptography can be divided into two subsections, block ciphers and stream ciphers. Simply put, a stream cipher encrypts one bit at a time, whereas a block cipher can encrypt multiple bits, or “blocks of bits” at a time.

Further, in the same section of [9], it is explained that even though block ciphers are more commonly used for encryption in computer communication on the internet and as mentioned previously, the block cipher AES is the most commonly used symmetric cipher today, stream ciphers do have relevance due to them tending to be small and fast. Thus they are used in applications with little computational resources such as cellphones.

An example given in [9, Section 2.1.1] would be the A5/1 cipher which is part of the GSM mobile phone standard, something that is used in voice encryption. There are stream ciphers that are used in internet communications as well, an example from the same section in [9] would be the stream cipher RC4.

### 1.3.1 Stream Ciphers

A stream cipher as previously mentioned encrypts bits one at a time. This is done by adding a bit from a key stream to a plain text bit.

In [9, Section 2.1.1] it is shortly described that stream ciphers can be split up into two different versions, synchronous and asynchronous ones. For synchronous stream ciphers, the key stream depends solely on the key, whereas for asynchronous stream ciphers, the key stream also depends on the cipher text.

Further, in [9, Section 2.1.2] we are told that the process of encryption and decryption in stream ciphers is done with a simple XOR function, in more mathematical terms we call it modulo 2 addition.

Simply put, the encryption of a bit  $x_i$  to  $y_i$  is done by adding a secret key bit  $s_i$  modulo 2,

$$y_i \equiv x_i + s_i \pmod{2},$$

and the decryption is done by adding  $s_i$  again to the encrypted bit  $y_i$  modulo 2.

$$x_i \equiv y_i + s_i \pmod{2}$$

With this in mind, one might think that stream ciphers look incredibly easy, almost to the extent of questioning whether or not they can be secure. The most important part of a stream cipher is the generation of the  $s_i$ , the key stream bits. That is where all the security of the cipher lies, see [9, Section 2.1] for further reading.

In that section [9] further states that naturally for the cipher to be secure, the key stream has to appear very random to any possible attacker, else an attacker could easily guess the bits and decrypt everything.

The generation of these key streams can be achieved in different ways. For example, we have true number generators (TRNG) and cryptographically secure pseudo random number generators (CSPRNG). Cryptographically secure refers to the pseudo random number generator being designed to be unpredictable to an extent where it can be considered computationally secure. That means it is not feasible to break it using current available computational power and algorithms. For further reading, see [9, Section 2.2].

### 1.3.2 Block Ciphers

Block ciphers encrypt blocks of bits at the same time using the same key. What this means is that the encryption of one bit in a block is dependent on other bits in the same block. In practice these bit lengths usually are either 128 bits, like AES, or 64 bits, like DES or 3DES [9, Section 2.1.1].

At the foundation of today's block ciphers lie the ideas of confusion and diffusion. The former being some type of operation used to obscure the relation between the key and the cipher text, the latter meaning that even so much as one plain text symbol being changed spreads throughout the entire cipher text such that a very, very different cipher text is produced for such a minuscule plain text change, this is to hide statistical properties of the plain text.

By concatenating operations with these "qualities" one can build a secure cipher and a cipher which is a concatenation of several encryption operations is known as a product cipher, which all of today's block ciphers are [9, Section 3.1.1].

Different ciphers take different paths to achieve this diffusion and confusion and we are refraining from going into further detail about different methods as that would require a very lengthy explanation and derail too much from the purpose of this paper.

Something worth noting about block ciphers is their versatility in the sense that they are not only used as an encryption algorithm, they can be used in various ways. Examples of this are construction of hash functions, key establishing protocols and building of different block based encryption schemes [9, Chapter 5 introduction].

## 1.4 Asymmetric algorithms

Asymmetric cryptography, also known as public-key cryptography, gets the latter name from the fact that there exists both a private and a public key. In [9, Section 6.1] it is described that the public key is available for anyone to be used to encrypt a message. What is important though is that the private key, that ideally only the one who is meant to receive the encrypted messages should have access to, is needed to decrypt these messages. In short, unlike symmetric cryptography where the same key is used for both encryption and decryption, in public-key cryptography you create a one-way street where anyone can encrypt and send encrypted information but only one person with the private key can decrypt and access this encrypted information (as long as the system is secure).

Of course, no system is perfect. In the case of public-key algorithms a major drawback is often that they can be very computationally heavy compared to symmetric ciphers as stated by [9, Section 6.2.1]. Further in the same section it is stated that this means that many times the public-key algorithms are not used to actually encrypt the data itself, and instead, are rather used for key establishment, providing nonrepudiation and message integrity via digital

signatures and identification. Thus as mentioned in the same section, in practice, most protocols are hybrids, consisting of both asymmetric and symmetric algorithms.

#### 1.4.1 The concept of a one-way function

All public-key algorithms are built based on one common principle, the one-way function. Informally one can describe this as a function that is computationally easy when used, but its inverse is computationally infeasible. Simply put,

$$\begin{aligned} y = f(x) &\text{ is easy to compute,} \\ x = f^{-1}(y) &\text{ is infeasible to compute.} \end{aligned}$$

In [9, Section 6.1] we are told that in the case of asymmetrical algorithms, this boils down to two different types of functions. On the one hand we have large integer factorization, which is used in probably the most famous public-key algorithm RSA. On the other hand we have the discrete logarithm problem, which we can find in for example the Diffie-Hellman key exchange, but it is also present in what this paper will focus on mainly, elliptic curve cryptography.

#### 1.4.2 Diffie Hellman key exchange and the discrete logarithm problem

The Diffie-Hellman key exchange was the first asymmetric scheme published in the open literature, see [9, Ch 8.1] for more background.

What it does is that it provides an elegant solution to the problem of distributing a key such that two parties can share one over an insecure channel. It is built on the fact that exponentiation in  $\mathbb{F}_p^*$ ,  $p$  being prime, is a one-way function as well as the fact that exponentiation is commutative.

Diffie-Hellman is split into two protocols, the set-up protocol and the main protocol. In the set-up protocol the two parties choose a large prime  $p$  and an integer  $\alpha \in \{2, 3, \dots, p-2\}$ . Then  $p$  and  $\alpha$  are published. In other words, these are not private or secret.

The main protocol is where the actual key exchange happens, if both parties know the parameters  $p$  and  $\alpha$  they can calculate the joint secret key  $k$ .

In short,

- Person  $A$  chooses an  $a = k_{prA} \in \{2, 3, \dots, p-2\}$  and generates  $k_{pubA} \equiv \alpha^a \pmod{p}$
- Person  $B$  chooses a  $b = k_{prB} \in \{2, 3, \dots, p-2\}$  and generates  $k_{pubB} \equiv \alpha^b \pmod{p}$
- They exchange their public keys, i.e.  $k_{pubA}$  and  $k_{pubB}$

- Now person  $A$  calculates the joint secret key  $k = k_{AB}$   
by calculating  $k_{AB} = (k_{pubB})^a \mod p$
- Person  $B$  calculates the joint secret key  $k = k_{AB}$   
by calculating  $k_{AB} = (k_{pubA})^b \mod p$

The reason this works is pretty simple but yet very elegant, and the proof is very short:

$$\text{Person } A \text{ calculates } (k_{pubB})^a \equiv (\alpha^b)^a \equiv \alpha^{ab} \mod p$$

$$\text{Person } B \text{ calculates } (k_{pubA})^b \equiv (\alpha^a)^b \equiv \alpha^{ab} \mod p$$

and thus, as we see, both end up with the same secret key  $k_{AB} = \alpha^{ab} \mod p$ .

The security in Diffie-Hellman key exchange comes from the discrete logarithm problem, which essentially is that for sufficiently large numbers in a finite field  $\mathbb{F}_p^*$  determining the integer  $x$  in

$$\alpha^x \equiv \beta \mod p$$

which translates to the problem

$$x = \log_\alpha \beta \mod p$$

is very hard, whereas computing the exponentiation to find  $\beta$  is very easy if one has  $\alpha$  and  $x$ , thus we get a one-way function.

## 2 Algebraic and number-theoretical concepts

In the previous section about the Diffie-Hellman key exchange we encountered  $\mathbb{F}_p^*$  and mentioned the discrete logarithm problem. To properly understand these and other concepts in cryptography, one needs to be familiar with some basic number-theoretical and algebraic concepts such as groups and finite fields. Further, cyclic groups become of great importance when we start moving into the area of elliptic curve cryptography.

### 2.1 Groups and fields

So what is a group? A group is a set of elements  $G$  together with an operation  $\circ$  which combines the two elements of  $G$  and has the following properties, see [9, Definition 8.2.1] :

1. The group operation  $\circ$  is closed. That is, for all  $a, b \in G$ , it holds that  $a \circ b = c \in G$ .
2. The group operation is associative. That is,  $a \circ (b \circ c) = (a \circ b) \circ c$  for all  $a, b, c \in G$ .

3. There is an element  $1 \in G$ , called the neutral element (or identity element), such that  $a \circ 1 = 1 \circ a = a$  for all  $a \in G$ .
4. For each  $a \in G$  there exists an element  $a^{-1} \in G$ , called the inverse of  $a$ , such that  $a \circ a^{-1} = a^{-1} \circ a = 1$ .
5. A group is abelian (or commutative) if, furthermore,  $a \circ b = b \circ a$  for all  $a, b \in G$ .

Further, a group is called a finite group if it has a finite number of elements. The cardinality of a group  $G$  is denoted  $|G|$ , see for instance [9, Definition 8.2.2].

For us, the groups of interest tend to be additive groups or multiplicative groups where the group operation is, as their names suggest, addition or multiplication. Here the inverse operation clearly for the additive group becomes subtraction and for the multiplicative group becomes division, or more precisely, multiplication with the inverse element [9, Ch 4.3].

If one creates a structure which contains all four basic arithmetic options, one gets a set which contains an additive group and a multiplicative group. This is called a field and in [12, Definition 1.3] we are given the following definition:

**Definition 1.** *A field is a set with two operations  $(G, \cdot, +)$  such that*

- $(G, +)$  is an abelian group with identity denoted by 0,
- $(G \setminus \{0\}, \cdot)$  is an abelian group,
- $(G, \cdot, +)$  satisfies the distributivity law,  $(a + b) \cdot c = a \cdot c + b \cdot c$

For most of the requirements, it is pretty easy to follow along and see that they are fulfilled in concrete examples. In an additive group containing 0, it is easy to see that it has a neutral element. For example, if the group operation is addition modulo a number, we know that any addition we do will not "leave the group", thus the requirement for being closed is fulfilled. For the other requirements it is usually easy to see that they are fulfilled given the more traditional group operations such as addition, multiplication etc.

However, determining if there exists a multiplicative inverse for all elements can get a little tricky. If we take the group of all non-zero real numbers then it is easy to know that all multiplicative inverses exists, we simply use the "regular" definition  $a^{-1} = \frac{1}{a}$  which in that case is perfectly valid. However, when one talks about sets consisting only of integers with operation multiplication modulo a number, for example  $\mathbb{Z}_{15} = \{0, 1, 2, 3, 4, \dots, 13, 14\}$  it is not as clear cut if there exists a multiplicative inverse for every element and thus if it is a group or not.

But there exists a rule for this. In [9, Section 1.4.2] it is stated that the multiplicative inverse for an element  $a \in \mathbb{Z}_n$  exists if and only if  $\gcd(a, n) = 1$ . In other words,  $a$  and  $n$  are relatively prime or also known as co-prime. Finding the

greatest common divisor for two numbers can be difficult if brute-forced when dealing with large numbers, and finding the inverse of a number when they become really big is even worse. For both of these, we instead use the Euclidean algorithm (and extended Euclidean algorithm). For further reading about it, see [9, Section 6.3]. In short the Euclidean algorithm allows us to first of all determine the greatest common divisor, and then its extended version allows us to compute modular inverses, which is a very important thing in public-key cryptography.

## 2.2 Groups under modular multiplication and $\mathbb{F}_p^*$

In the previous section we spoke about groups and the requirement of the elements being relatively prime to the modulus for there to exist a multiplicative inverse. For example  $\mathbb{Z}_9 = \{0, 1, 2, 3, 4, 5, 6, 7, 8\}$  is not a multiplicative group since 3 does not have a multiplicative inverse since  $\gcd(3, 9) = 3$ . Thus the  $*$  notation in  $\mathbb{Z}_n^*$  is used to specify a set containing all  $a \in \mathbb{Z}_n^*$  with  $\gcd(a, n) = 1$ , so for example  $\mathbb{Z}_9^* = \{1, 2, 4, 5, 7, 8\}$ . What we get then is a set which forms a group under modular multiplication, which more explicitly stated in [9, Theorem 8.2.1] can be summarized as:

**Theorem 2.1.** *The set  $\mathbb{Z}_n^*$  which consists of all integers  $i \in \{0, 1, \dots, n-1\}$  for which  $\gcd(i, n) = 1$  forms an abelian group under multiplication modulo  $n$ . The identity element is  $e = 1$ .*

However, not all of these groups are of importance to cryptography, the groups that we do care about particularly are those where the modulus is a prime. In that case, we obtain a field and then we use the notation  $\mathbb{F}_p^*$ . We care about these cases since as one might realize at this point, a prime number will have  $\gcd(a, p) = 1$  for all values of  $1 \leq a < p$ .

## 2.3 Cyclic groups

We mentioned previously what a finite group is. Further on, the order of an element in a group is defined as follows, see [9, Definition 8.2.3]:

**Definition 2.** *The order  $\text{ord}(a)$  of an element  $a$  of a group  $(G, \circ)$  is the smallest positive integer  $k$  such that  $a^k = \underbrace{a \circ a \circ \dots \circ a}_{k \text{ times}} = 1$ , where 1 is the identity element of  $G$ .*

With this we give the definition for a cyclic group, see [9, Definition 8.2.4]:

**Definition 3.** *A group  $G$  which contains an element  $\alpha$  with maximum order  $\text{ord}(\alpha) = |G|$  is said to be cyclic. Elements with maximum order are called primitive elements or generators.*

To add to this, there also exists a theorem, see [9, Theorem 8.2.2], that states the following:

**Theorem 2.2.** *For every prime  $p$ ,  $(\mathbb{F}_p^*, \cdot)$  is an abelian finite cyclic group.*

Further, we state a theorem from [9, Theorem 8.2.3].

**Theorem 2.3.** *Let  $G$  be a finite group. Then for every  $a \in G$  it holds that:*

1.  $a^{|G|} = 1$
2.  $\text{ord}(a)$  divides  $|G|$

The first property is a generalization of Fermat's little theorem for all cyclic groups. The second property tells us that in a cyclic group, only element orders which divide the group cardinality exist.

Finally for this chapter, we state one more theorem, see [9, Theorem 8.2.4].

**Theorem 2.4.** *Let  $G$  be a finite cyclic group. Then it holds that*

1. *The number of primitive elements of  $G$  is  $\phi(|G|)$ , where  $\phi$  is Euler's phi function.*
2. *If  $|G|$  is prime, then all elements  $a \neq 1 \in G$  are primitive.*

On a last minor note before we move on to elliptic curves, we want to quickly mention that the group operation does not necessarily have to be the regular multiplication or addition. As we will see with elliptic curve cryptography, there will be a difference how the arithmetic works compared to the familiar arithmetic of the real numbers. Thus it is important to remember that we are just operating at that point in a different type of group.

## 3 Introduction to Elliptic Curves

### 3.1 What is an Elliptic Curve?

An elliptic curve can be described by the following definition, see [6, Chapter 6, Definition 1]:

**Definition 4.** Let  $K$  be a field of characteristic  $\neq 2, 3$ , and let  $x^3 + ax + b$  (where  $a, b \in K$ ) be a cubic polynomial with no multiple roots. An elliptic curve over  $K$  is the set of points  $(x, y)$  with  $x, y \in K$  which satisfy the equation

$$y^2 = x^3 + a \cdot x + b$$

together with a single element denoted  $\mathcal{O}$  and called the "point at infinity".

Here we note that the characteristic of a field is defined as follows, see [6, Chapter 2]:

**Definition 5.** If adding the multiplicative identity 1 to itself in  $\mathbb{F}$  never gives 0, then we say that  $\mathbb{F}$  has characteristic zero,  $\text{char}(\mathbb{F}) = 0$ . In that case,  $\mathbb{F}$  contains a copy of the rational numbers. Otherwise, there is a prime number  $p$  such that  $\underbrace{1 + 1 + \dots + 1}_{p \text{ times}} = 0$ , and  $p$  is called the characteristic of the field  $\mathbb{F}$ . In that case,  $\mathbb{F}$  contains a copy of the field  $\mathbb{F}_p$ , which is called its prime field.

Examples of possible fields  $K$  include (see [6]) the field  $\mathbb{R}$  of real numbers, the field  $\mathbb{Q}$  of rational numbers, the field  $\mathbb{C}$  of complex numbers, or the finite field  $\mathbb{F}_q$  of  $q = p^r$  elements. In our case we will first work with elliptic curves over the field  $\mathbb{R}$ , this to be able to give a background of the geometrical approach that lays the foundation for the group operation, to later move over to elliptic curves over finite prime fields  $\mathbb{F}_p$ , since these are the ones that are of interest when doing cryptography.

### 3.2 Arithmetic on an Elliptic Curve over $\mathbb{R}$

Arithmetic on an elliptic curve consists of a group operation usually referred to as point addition. One can visualize this operation fairly well by a more geometrical approach, see [5, Section 6.1]. Simply put, if we represent the elliptic curve on a real plane and take two points on the curve  $P$  and  $Q$ , then the third point from adding  $P$  and  $Q$  together can be derived by mirroring the third intersection point on the curve by the line that is created when connecting  $P$  and  $Q$ , as seen in Figure 2. So we have a rather intuitive way of doing addition on an elliptic curve that is very different from addition e.g. of real numbers. We will denote the addition on an elliptic curve using ' $\oplus$ '. Clearly, in practice when doing calculations over an elliptic curve, drawing lines is not a very precise or good way to do addition. Instead we want to use a more analytical approach, which becomes more understandable with the geometrical background previously given.

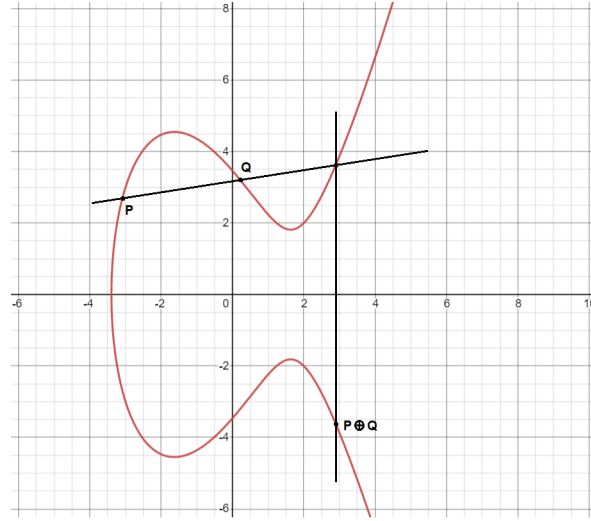


Figure 2: Figure showing the correlation between intersecting lines and addition of arbitrary points on the elliptic curve  $y^2 = x^3 - 8x + 12$  (Curve image generated using desmos.com)

Following [2, Section 13.2.1a], we have an elliptic curve given by the equation as per Definition 4:

$$y^2 = x^3 + a \cdot x + b$$

We assume that we have two points,  $P = (x_1, y_1)$  and  $Q = (x_2, y_2)$ ,  $P \neq \pm Q$  such that  $P \oplus Q = (x_3, y_3)$ . Now define  $\lambda = \frac{y_1 - y_2}{x_1 - x_2}$ . Then our coordinates  $(x_3, y_3)$  are given by

$$x_3 = \lambda^2 - x_1 - x_2 \text{ and } y_3 = \lambda(x_1 - x_3) - y_1.$$

Further on, we have a special case of point addition called point doubling, it is essentially adding a point to itself. So we would write  $P \oplus P = 2P$ . As explained in [5, Section 6.1] the operations are rooted in a geometrical interpretation of intersecting lines on the elliptic curve in a real plane. Thus we get a slightly different formula in this case. As shown in Figure 3, when doubling, we treat the addition as if the point we add on is a point that is moving infinitely close to  $P$ . So our line intersecting the points to be added in this case becomes a tangent at  $P$ .

Because of this we alter the formulae to calculate  $(x_3, y_3)$ , see [2, Section 13.2.1a]. Let  $\lambda = \frac{3x_1^2 + a}{2y_1}$ . Then  $(x_3, y_3)$  is given by

$$x_3 = \lambda^2 - 2x_1 \text{ and } y_3 = \lambda(x_1 - x_3) - y_1$$

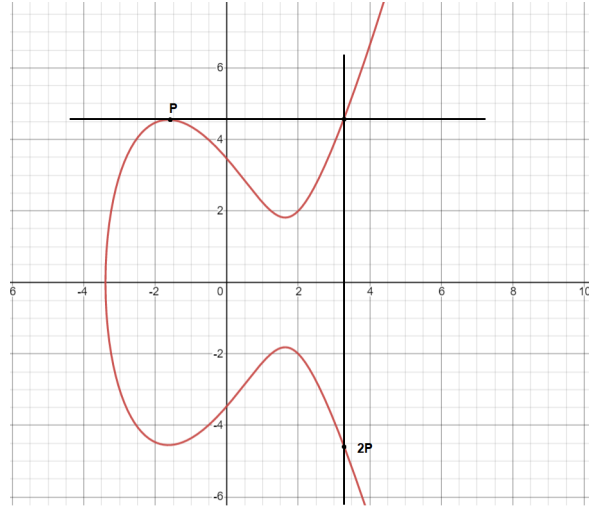


Figure 3: The figure shows point doubling on the elliptic curve  $y^2 = x^3 - 8x + 12$  (curve image generated using desmos.com). We use its tangent to create a line that will then intersect another point on the curve which we mirror just like in addition on an elliptic curve.

Recall the  $\mathcal{O}$  mentioned in Definition 4. It was described as an "imaginary point at infinity". To further elaborate on this, see [5, Section 6.1]. Consider two points  $P = (x_1, y_1)$  and  $P' = (x_1, -y_1)$  such that  $P'$  is the reflection on the  $x$ -axis to  $P$ . If we were to add these two together we would get a problem as this would create a straight line that does not intersect the curve a third time. This seemingly "breaks" our operation of point addition, but the solution to this is to create the extra point  $\mathcal{O}$ . This point exists at infinity, or as further described by [5], it does not exist in the  $xy$ -plane, but we pretend that it lies on every vertical line such that

$$P \oplus P' = \mathcal{O}.$$

What is created is an additive neutral element. With this in mind we further note that  $P'$  in this case is the additive inverse of  $P$  or the "negative of  $P$ ". This becomes apparent if we return to looking at it geometrically since if we add together  $P$  and  $\mathcal{O}$  we create a straight vertical line that intersects at a third point  $P'$ , which when reflected on the  $x$ -axis returns us back to  $P$ . Thus

$$P \oplus \mathcal{O} = P$$

and since this is the case, it holds true that  $P'$  is the additive inverse of  $P$  since by definition  $P \oplus P' = \mathcal{O}$ .

To finish this section, we state some properties of the operation  $\oplus$ , see [5, Theorem 6.5].

**Theorem 3.1.** *Let  $E$  be an elliptic curve. Then the addition law on  $E$  has the following properties:*

- $P \oplus \mathcal{O} = \mathcal{O} \oplus P = P$  for all  $P \in E$  (Identity)
- $P \oplus (-P) = \mathcal{O}$  for all  $P \in E$  (Inverse)
- $(P \oplus Q) \oplus R = P \oplus (Q \oplus R)$  for all  $P, Q, R \in E$  (Associative)
- $P \oplus Q = Q \oplus P$  for all  $P, Q \in E$  (Commutative)

Point 1, 2 and 4 follow from the definitions whereas point 3 requires a rather lengthy proof for general cases but can also be checked on a case by case basis, for further reading see [3] or [11, Chapter III.2-3]. The content of this theorem will be of importance when discussing elliptic curves as a cyclic group.

### 3.3 Elliptic Curves as a cyclic group

As shown by Theorem 3.1 elliptic curves form a group with operation  $\oplus$ . It turns out we can even form cyclic groups, see [2, Section 13.1.3]:

**Theorem 3.2.** *Consider an elliptic curve over a finite prime field  $\mathbb{F}_p$ . The set of points  $E(\mathbb{F}_p)$  together with  $\mathcal{O}$  form a group that is either cyclic or a product of two cyclic groups.*

We note that we now use elliptic curves over finite fields  $\mathbb{F}_p$ , this means that we use a slightly modified definition, see [9, Definition 9.1.1]:

**Definition 6.** *The elliptic curve over  $\mathbb{F}_p$ ,  $p > 3$  is the set of all pairs  $(x, y) \in \mathbb{F}_p$  which fulfill*

$$y^2 \equiv x^3 + ax + b \pmod{p}$$

*together with an imaginary point of infinity  $\mathcal{O}$ , where  $a, b \in \mathbb{F}_p$  and the condition  $4 \cdot a^3 + 27 \cdot b^2 \not\equiv 0 \pmod{p}$  is satisfied.*

In addition to this, we note that the order of a group like this is important when setting up a discrete logarithm cryptosystem, see [9, Section 9.2]. While knowing the exact number of points can be difficult when dealing with curves over fields based on large primes we can know the approximate amount thanks to a theorem known as Hasse's theorem, see [9, Theorem 9.2.2].

**Theorem 3.3.** *Given an elliptic curve  $E$  modulo  $p$ , the number of points on the curve is denoted by  $\#E$  and is bounded by:*

$$p + 1 - 2\sqrt{p} \leq \#E \leq p + 1 + 2\sqrt{p}$$

In practice what this gives us is that the number of points is roughly in the range of the prime  $p$ , for example, this means that if we want a curve with  $2^{160}$  elements, we have to use a prime with a length of about 160 bit.

## 4 Elliptic Curve Cryptography

### 4.1 How and why it works

We recall the discrete logarithm problem from Chapter 1 and that it is difficult to calculate the exponent  $x$ . When using elliptic curves over finite prime fields, we get a cyclic group which lets us construct a discrete logarithm problem in a similar way as the one given in Chapter 1, see [5, Chapter 6.3]. Essentially we choose two points  $P$  and  $Q$  in  $E(\mathbb{F}_p)$ , and the secret integer  $n$  is the integer that makes

$$Q = \underbrace{P \oplus P \oplus \cdots \oplus P}_{n \text{ times}} = nP.$$

We can summarize this with the following definition, see [5, Chapter 6.3].

**Definition 7.** Let  $E$  be an elliptic curve over the finite field  $\mathbb{F}_p$  and let  $P$  and  $Q$  be points in  $E(\mathbb{F}_p)$ . The Elliptic Curve Discrete Logarithm Problem (ECDLP) is the problem of finding an integer  $n$  such that  $Q = nP$ . By analogy with the discrete logarithm problem for  $\mathbb{F}_p^*$  we denote this integer  $n$  by

$$n = \log_P(Q)$$

and we call  $n$  the elliptic discrete logarithm of  $Q$  with respect to  $P$ .

We take a look at an example to get a more concrete picture. From [9, Example 9.5] we take the curve

$$y^2 \equiv x^3 + 2x + 2 \pmod{17}$$

All the points on this curve form a cyclic group where the order is  $\#E = 19$ . Since the group order is prime we can make use of Theorem 2.4 which tells us that all elements except the neutral one are primitive elements, also known as generators.

For easier understanding of future sections we start off by wanting to find all the points on the curve. Remember that we said that there will be 19 elements, and that we can use any point on the curve within  $E(\mathbb{F}_p)$  to generate the other elements. So just as in [9, Example 9.5] we start with point  $P = (5, 1)$ . Now when we start performing point addition of continuously adding  $P$  to the previous point we get a list as shown in Table 1, see [9, Example 9.5]. With the entire group listed like this, we will be able to refer back to it to understand certain things when working with elliptic curve cryptography.

For example, we can perform a Diffie-Hellman key exchange which we recall from Section 1.4.2. We choose a point on the curve, let us use  $P = (5, 1)$  as our primitive element. Thus we have our domain parameters, the prime 17 (recall the modulus for the curve), the coefficients  $a = 2$  and  $b = 2$  from the curve and our primitive element  $P = (5, 1)$ .

$P = (5, 1)$	$11P = (13, 10)$
$2P = P \oplus P = (5, 1) \oplus (5, 1) = (6, 3)$	$12P = (0, 11)$
$3P = 2P \oplus P = (6, 3) \oplus (5, 1) = (10, 6)$	$13P = (16, 4)$
$4P = 3P \oplus P = (3, 1)$	$14P = (9, 1)$
$5P = (9, 16)$	$15P = (3, 16)$
$6P = (16, 13)$	$16P = (10, 11)$
$7P = (0, 6)$	$17P = (6, 14)$
$8P = (13, 7)$	$18P = (5, 16)$ , notice here that $(5, 16) \equiv (5, -1)$ which is the additive inverse of $P$ , also known as $-P$ .
$9P = (7, 6)$	$19P = 18P \oplus P = -P \oplus P = \mathcal{O}$
$10P = (7, 11)$	$20P = 19P \oplus P = \mathcal{O} \oplus P = P = (5, 1)$

Table 1: This table showcases the 19 elements of the elliptic curve  $y^2 \equiv x^3 + 2x + 2 \pmod{17}$  and how they are generated using  $P = (5, 1)$  as the primitive element, notice that when we reach  $20P$  we have started over and thus are back at  $P$ , which gives nice context to why we call this group cyclic

Now, we simply follow the main protocol from our previous section on Diffie-Hellman key exchange with the slight adjustment that instead of exponentiation we use the group operation  $\oplus$  and we remember that  $nP$ , where  $n$  is an integer, is a notation for repeated usage of the group operation:

- Alice chooses a  $k_{prA} \in \{2, 3, \dots, \underbrace{19-1}_{\#E-1}\}$ , for this example we let  $k_{prA} = 5$ .  
With this, Alice generates her public key  $k_{pubA} = k_{prA}P = 5P = (9, 16) = A$ .
- Bob chooses a  $k_{prB} \in \{2, 3, \dots, 18\}$ , for this example we let  $k_{prB} = 11$ . With this, Bob generates his public key  $k_{pubB} = k_{prB}P = 11P = (13, 10) = B$ .
- They exchange their public keys  $k_{pubA}$  and  $k_{pubB}$ .
- Alice calculates the joint secret key  $k = k_{AB}$  by calculating  $k_{prA}(k_{pubB}) = 5(13, 10) = 5B$ .

To calculate this point in the most basic of ways we note that we can start off by doubling twice and then add once, that is,  $5B = B \oplus B \oplus B \oplus B \oplus B = \underbrace{2B \oplus 2B}_{2(2B)} \oplus B$  (thanks to the associativity law). In fact, there is an algo-

rithm to use to determine how we proceed when we want to multiply a scalar with a point, this algorithm is known as the double and add algorithm and is something we will touch further in Chapter 5. However, for now we just note that we can double twice and then add once to get  $5B$ . Since  $B = 11P$  we get that  $2B = 11P \oplus 11P = 22P = 3P$ ,

then  $2B \oplus 2B = 3P \oplus 3P = 6P$  and finally that  
 $4B \oplus B = 6P \oplus 11P = 17P = (6, 14)$ .

- Bob calculates the joint secret key  $k = k_{AB}$  by calculating  $k_{prB}(k_{pubA}) = 11(9, 16) = 11A$ . We note that we can split this up into doubling twice, adding, then doubling and then adding again, that is,  
 $11A = 2(2(2A) \oplus A) \oplus A$ .  
Since  $A = 5P$  we get  $2A = 10P$ ,  $4A = 20P = P$ ,  
 $5A = 4A \oplus A = P \oplus A = P \oplus 5P = 6P$ ,  
 $10A = 6P \oplus 6P = 12P$ ,  
 $11A = 10A \oplus A = 12P \oplus 5P = 17P = (6, 14)$ .
- As we see both of the parties have now calculated the joint secret key  $k$ , which works thanks to the group operation being associative and thus  $aB = a(bP) = k$  and  $bA = b(aP) = k$ . In practice, a common use would be that one of the coordinates in  $k$  would be used as a session key, for example,  $x$  being used as a symmetric key, see [9, Section 9.3].

## 4.2 Why it is secure

Elliptic curves are used because the ECDLP has very good one-way characteristics, see [9, Section 9.4]. If an attacker wants to break the elliptic curve Diffie-Hellman key exchange (ECDH), they have the following information: the curve  $E$ , the prime  $p$ , the primitive element  $P$ , the public keys  $A$  and  $B$  and to calculate the joint secret key  $k$  he has to solve either

$$a = \log_P A$$

or

$$b = \log_P B.$$

If the correct curves are chosen, then we currently do not have attacks against the ECDLP that match the strength of the best known attacks against DLP modulo  $p$  or the best factoring algorithms used for attacks against RSA. What this means in practice is that the prime  $p$  chosen for an elliptic curve can be considerably shorter than for example the prime factors used for an RSA encryption.

An especially noteworthy example are the index-calculus algorithms which are powerful algorithms used against the DLP modulo  $p$ , which are not applicable for elliptic curves. For properly selected curves, the attacks that are available are generic discrete logarithm algorithms such as the baby-step giant-step method and the rho method which are known as square root methods. As explained in [2, Chapter 19.1], the term square root methods refers to the fact that they have an upper bound  $\sqrt{|G|}$  steps to solve the discrete logarithm problem for a cyclic group  $G$ . For further reading about individual methods, see [2, Chapter 19].

What we emphasize with this is that for an elliptic curve we can have a secure system with for example 256-bit length elliptic curves, since these provide a security level of  $\sqrt{2^{256}} = 2^{128}$ , so 128-bit, whereas achieving the same security level using RSA or DL problem modulo  $p$  would require 3072 bit, see [9, Table 6.1].

### 4.3 Choosing the correct curves

When choosing curves for ECC it is important that you pick a correct type, this mainly to avoid certain cryptographic weaknesses that otherwise can exist. For general guidelines, see [10]. An example of the type of curves we have been talking about mainly are Weierstrass curves, which in [10, Section 4.2.1] are described as curves of the form:

$$E : y^2 \equiv x^3 + ax + b \pmod{p}$$

with :

- The prime modulus  $p$
- The coefficient  $a$ 
  - For pseudorandom curves,  $a = -3$  was taken for reasons of efficiency; see IEEE std 1363-2000
- The coefficient  $b$ 
  - For pseudorandom curves, the coefficient  $b$  satisfies  $b^2c \equiv -27 \pmod{p}$
- The base point  $G$  with  $x$ -coordinate  $G_x$  and  $y$ -coordinate  $G_y$

An example of such a curve is P-256. In [10, Section 4.2.1.3] it is described as a Weierstrass curve  $W_{a,b}$  defined over the prime field  $GF(p)$  that has order  $n$ , where  $n$  is a prime number. Its parameters are as follows:

$$\begin{aligned}
p : & \quad 2^{256} - 2^{224} + 2^{192} + 2^{96} - 1 \\
& \quad = 115792089210356248762697446949407573530 \backslash \\
& \quad \quad 086143415290314195533631308867097853951 \\
n : & \quad 115792089210356248762697446949407573529 \backslash \\
& \quad \quad 996955224135760342422259061068512044369 \\
a : & \quad -3 \\
& \quad = 115792089210356248762697446949407573530 \backslash \\
& \quad \quad 086143415290314195533631308867097853948 \\
b : & \quad 41058363725152142129326129780047268409 \backslash \\
& \quad \quad 114441015993725554835256314039467401291 \\
G_x : & \quad 48439561293906451759052585252797914202 \backslash \\
& \quad \quad 762949526041747995844080717082404635286 \\
G_y : & \quad 36134250956749795798585127919587881956 \backslash \\
& \quad \quad 611106672985015071877198253568414405109
\end{aligned}$$

As described in [1], P-256 has been used extensively in *critical infrastructure* (such as the internet) projects and is being used in Elliptic Curve Digital Signature Algorithms for Autonomous System path signing and verification in the BGPSEC protocol (Border Gateway Protocol with security). For further reading about these, see [1] and the references therein.

## 5 Effective calculation of exponents

### 5.1 Double and Add, Square and Multiply

The Square and Multiply and the Double and Add algorithms are actually the same algorithm with different names specifying whether it is being used for calculating powers or scalar multiples respectively. The idea is to get an effective way to perform arithmetic in the respective fields. We will mainly focus on it from an “elliptic curve point of view”.

Recall from Section 4.1 when being shown an example of Diffie-Hellman key exchange that we wanted to calculate  $5B$  and  $11A$ . In that case the scalars were so small that the usage of the Double and Add algorithm may have not been very noticeable in terms of efficiency compared to if we had just simply done regular addition 5 and 11 times respectively. However, when working with very large scalars it becomes important to do calculations effectively, as calculating  $P, 2P, 3P, 4P, \dots, nP$  would be a very slow process. For example, if  $n = 2^{160}$ , we would require  $2^{160} - 1$  steps to calculate  $nP$ . To avoid this we use the Double and Add algorithm.

The algorithm itself is very straightforward, we map our step by step process by looking at the binary version of  $n$ . To elaborate on this we start off with an example:

Let  $n = 177 = (10110001)_2$  and let  $P$  be an arbitrary point on an elliptic curve  $E$  such that  $P$  is a generator. Now, if we wanted to calculate  $nP = 177P$  we would use the binary form of  $n$  as our “guide” on how to do this effectively:

$$\begin{aligned}
1P &= (1)_2P \\
2P &= (10)_2P \\
4P &= (100)_2P \\
5P &= (101)_2P \\
10P &= (1010)_2P \\
11P &= (1011)_2P \\
22P &= (10110)_2P \\
44P &= (101100)_2P \\
88P &= (1011000)_2P \\
176P &= (10110000)_2P \\
177P &= (10110001)_2P
\end{aligned}$$

What happens is that we start off with the leading 1 in the binary representation, then, to add a zero to the right, we double, and to convert the right-most zero, we add. We do this until we have “recreated”  $n$  in its binary form. As we see from the example above, unlike having to do 177 steps if one would just add constantly, we only have to do 10 steps.

To give a more precise description of the algorithm, we present the pseudocode given in [9, Section 9.2]:

**Input:** elliptic curve  $E$  together with an elliptic curve point  $P$ , a scalar  $d =$

$$\sum_{i=0}^t d_i 2^i \text{ with } d_i \in \{0, 1\} \text{ and } d_t = 1$$

**Output:**  $T = dP$

**Initialization:**

$$T = P$$

**Algorithm:**

FOR  $i = t - 1$  DOWNT0 0

$$T = T + T \mod p$$

IF  $d_i = 1$

$$T = T + P \mod p$$

RETURN  $T$

As mentioned before, the square and multiply algorithm works exactly the same, with the difference being that we use different terminology since we are dealing with exponents instead of scalars.

This algorithm has a logarithmic complexity, where in the best-case scenario, when the scalar/exponent is a power of 2, the amount of steps is equal to the bit length minus one, and in the worst-case scenario, it is the bit length minus one times two, see [5, Section 6.3.1]. This is because the amount of doublings we have to do is always equal to the bit length minus one, i.e., 8 bits means we get  $8 - 1 = 7$  doublings, whereas the amount of adding is dependent on how many nonzero terms there are in the binary expansion of the number. This amount is known as the number's *Hamming weight*, see [2, Section 9.1.1]. Thus we get an interval of

$$\lfloor \log_2(n) \rfloor \leq \# \text{ of steps} \leq 2 \cdot \lfloor \log_2(n) \rfloor.$$

Since it is expected in cryptography to use random numbers in the sense that no number should be expected more than another, it is fair to say that approximately, we can expect to need  $1.5 \cdot \log_2(n)$  steps on average to calculate the scalar multiple/power, see [9, Section 7.4]. For really large numbers, one can ignore the floor function when estimating as it only reduces the estimated amount of steps by at most 2, which for really large numbers is fairly redundant.

## 5.2 Non-Adjacent Form

As numbers get larger, it becomes of interest to save computing resources by calculating them as efficiently as possible. In the case of elliptic curves and using the double and add algorithm, this translates into ways to represent a number with as few nonzero terms as possible. Thus we present NAF (Non-Adjacent Form). In short, NAF takes the regular binary representation and allows for the coefficients to be not only 0 and 1 but also  $-1$ . It gets its name from the fact that with this way of writing a number, using  $\{-1, 0, 1\}$  as coefficients for each term in the binary expansion, we can write every number in such a way that no two nonzero terms are allowed to be adjacent to each other. The existence of this form for every number is stated in [4, Theorem 2]:

**Theorem 5.1.** *Every integer  $x$  has exactly one NAF.*

A more accurate definition for NAF is given in [4, Section 2.3]:

**Definition 8.** *Consider representations*

$$x = \sum_{i=0}^{\infty} d_i 2^i$$

*with  $d_i \in \{-1, 0, 1\}$  for all  $i$ . A Non-Adjacent Form (NAF) is a representation in which  $d_i \cdot d_{i+1} = 0$  for all  $i \geq 0$ .*

We denote a  $-1$  coefficient as  $\bar{1}$  when writing in NAF form.

To determine the NAF representation of a number  $n$  we can use an algorithm that makes it simple:

1. To start off, take  $n$  and
  - If  $n$  is even (equivalent to 0 or 2 mod 4), note down 0 and divide  $n$  by 2, let the result be the new  $n$ .
  - If  $n$  is odd and equivalent to 1 mod 4, note down a 1 and then divide  $n - 1$  by 2, let the result be the new  $n$ .
  - If  $n$  is odd and equivalent to 3 mod 4, note down a  $\bar{1}$  and then divide  $n + 1$  by 2, let the result be the new  $n$ .
2. Repeat step one but for the new  $n$  received by dividing, noting down the new digits to the left of the most recent digit.
3. When you reach 1, add a last 1 and halt.

To further explain, we show an example with determining the NAF representation for  $n = 27$ .

$$\begin{array}{llll}
n = 27 \equiv 3 \pmod{4} & \bar{1} & \frac{27+1}{2} = 14 & \\
n = 14 \equiv 2 \pmod{4} & 0\bar{1} & \frac{14}{2} = 7 & \\
n = 7 \equiv 3 \pmod{4} & \bar{1}0\bar{1} & \frac{7+1}{2} = 4 & \\
n = 4 \equiv 0 \pmod{4} & 0\bar{1}0\bar{1} & \frac{4}{2} = 2 & \\
n = 2 \equiv 2 \pmod{4} & 00\bar{1}0\bar{1} & \frac{2}{2} = 1 & \\
n = 1 \equiv 1 \pmod{4} & 100\bar{1}0\bar{1} & \text{Algorithm halts since } n = 1. & 
\end{array}$$

This may not be the most optimal way of doing it, since for example we know that after every nonzero digit, a zero must follow, so an algorithm which takes advantage of that property could probably yield a faster result. Nonetheless, we get a very clear and simple way of how to systematically write out a number in its NAF representation.

It becomes obvious that when using NAF representation of a number, at most the Hamming weight of a number is going to be about half of the bit length of the NAF representation. More precisely, for an even bit length  $t$  we get at most a Hamming weight of  $\frac{t}{2}$  and for an odd bit length  $t$  we get at most a Hamming weight of  $\lceil \frac{t}{2} \rceil$ . This means that when dealing with large numbers, a sizeable amount of computing power can be saved since we minimize the Hamming weight. In fact, on average we expect only a third of the digits to be nonzero, see [4, Section 2.3]. Something that is important to note is that the NAF representation does in many cases have a bit length of  $t + 1$  for numbers that otherwise in regular binary have a bit length of  $t$ . This means that for small numbers, the extra doubling step this causes may cancel out any reduction in nonzero digits that may have been achieved, but for larger numbers, one extra bit will not matter compared to a severe reduction in nonzero digits.

The only adjustment that has to be made to the double and add algorithm when using NAF representation is that we subtract instead of add when we have a  $-1$  coefficient. We present the pseudocode for the double and add algorithm adjusted for NAF.

**Input:** elliptic curve  $E$  together with an elliptic curve point  $P$ , a scalar  $d = \sum_{i=0}^t d_i 2^i$  with  $d_i \in \{-1, 0, 1\}$  and  $d_t = 1$

**Output:**  $T = dP$

**Initialization:**

$T = P$

**Algorithm:**

FOR  $i = t - 1$  DOWNT0 0

$T = T + T \mod p$

IF  $d_i = 1$

$T = T + P \mod p$

ELSEIF  $d_i = -1$

$T = T - P \mod p$

RETURN  $T$

As one can see, this is almost identical to the algorithm for regular double and add, with the small difference of adding  $-1$  as a possible  $d_i$  value in the input as well as adding the “ELSEIF” section for the case of  $d_i = -1$ .

As stated in [8, Chapter 1], the benefits are not as clear cut when we are in the square and multiply case, i.e., when we are dealing with exponents, since then instead of point subtraction we would be dividing which in itself can be a rather costly operation since finding multiplicative inverses can be hard. In the case of elliptic curves though, as we know, the inverse to a point on the curve is very easy to find and therefore computationally very cheap. Thus the NAF representation becomes very advantageous when operating on an elliptic curve since we can overall reduce the amount of steps while still performing equally costly operations as in the regular binary representation case.

To showcase NAF’s efficiency, we give an example. Let  $n = 1013$  and  $P$  be an arbitrary point on an arbitrary curve  $E$  of appropriate type.

$n = (1111110101)_2$				$n = (100000\bar{1}0101)_{\text{NAF}}$			
$P$	$(1)_2$	$62P$	$(111110)_2$	$P$	$(1)_{\text{NAF}}$	$252P$	$(100000\bar{1}00)_{\text{NAF}}$
$2P$	$(10)_2$	$63P$	$(111111)_2$	$2P$	$(10)_{\text{NAF}}$	$253P$	$(100000\bar{1}01)_{\text{NAF}}$
$3P$	$(11)_2$	$126P$	$(1111110)_2$	$4P$	$(100)_{\text{NAF}}$	$506P$	$(100000\bar{1}010)_{\text{NAF}}$
$6P$	$(110)_2$	$252P$	$(11111100)_2$	$8P$	$(1000)_{\text{NAF}}$	$1012P$	$(100000\bar{1}0100)_{\text{NAF}}$
$7P$	$(111)_2$	$253P$	$(11111101)_2$	$16P$	$(10000)_{\text{NAF}}$	$1013P$	$(100000\bar{1}0101)_{\text{NAF}}$
$14P$	$(1110)_2$	$506P$	$(111111010)_2$	$32P$	$(100000)_{\text{NAF}}$		
$15P$	$(1111)_2$	$1012P$	$(1111110100)_2$	$64P$	$(1000000)_{\text{NAF}}$		
$30P$	$(11110)_2$	$1013P$	$(1111110101)_2$	$63P$	$(100000\bar{1})_{\text{NAF}}$		
$31P$	$(11111)_2$			$126P$	$(100000\bar{1}0)_{\text{NAF}}$		

As we can see, using the NAF representation we can finish the calculation in 13 steps, whereas regular binary form requires 16 steps, despite the fact that the NAF representation in this case is one bit longer. Even if we were to choose different numbers in the range of  $512 - 1023$  (10 bit long numbers in binary representation, 10 or 11 bit long numbers in NAF representation), the worst-case scenario for a number in NAF would at most be 15 steps, whereas for binary the worst-case scenario would be 18 steps. If we assume completely random numbers within the given range, we should on average get  $\approx 14$  steps in the binary case and  $\approx 13$  steps in the NAF case, since as pointed out earlier in the paper, for binary we expect on average half nonzero digits whereas for NAF we expect on average a third nonzero digits.

## References

- [1] Mehmet Adalier. “Efficient and Secure Elliptic Curve Cryptography Implementation of Curve P-256”. In: *Workshop on Elliptic Curve Cryptography Standards*. <https://csrc.nist.gov/csrc/media/events/workshop-on-elliptic-curve-cryptography-standards/documents/papers/session6-adalier-mehmet.pdf>. National Institute of Standards and Technology (NIST), 2015.
- [2] Henri Cohen et al., eds. *Handbook of elliptic and hyperelliptic curve cryptography*. Discrete Mathematics and its Applications (Boca Raton). Chapman & Hall/CRC, Boca Raton, FL, 2006, pp. xxxiv+808. ISBN: 978-1-58488-518-4; 1-58488-518-1.
- [3] Stefan Friedl. *An elementary proof of the group law for elliptic curves*. [https://www.uni-regensburg.de/Fakultaeten/nat\\_Fak\\_I/friedl/papers/elliptic\\_2017.pdf](https://www.uni-regensburg.de/Fakultaeten/nat_Fak_I/friedl/papers/elliptic_2017.pdf). Accessed 2022-04-25. 2017.
- [4] Daniel M. Gordon. “A survey of fast exponentiation methods”. In: *J. Algorithms* 27.1 (1998), pp. 129–146. ISSN: 0196-6774. DOI: 10.1006/jagm.1997.0913. URL: <https://doi.org/10.1006/jagm.1997.0913>.
- [5] Jeffrey Hoffstein, Jill Pipher, and Joseph H. Silverman. *An introduction to mathematical cryptography*. Second. Undergraduate Texts in Mathematics. Springer, New York, 2014, pp. xviii+538. ISBN: 978-1-4939-1710-5; 978-1-4939-1711-2. DOI: 10.1007/978-1-4939-1711-2. URL: <https://doi.org/10.1007/978-1-4939-1711-2>.
- [6] Neal Koblitz. *A course in number theory and cryptography*. Second. Vol. 114. Graduate Texts in Mathematics. Springer-Verlag, New York, 1994, pp. x+235. ISBN: 0-387-94293-9. DOI: 10.1007/978-1-4419-8592-7. URL: <https://doi.org/10.1007/978-1-4419-8592-7>.
- [7] Czeslaw Koscielny, Mirosław Kurkowski, and Marian Srebrny. *Modern Cryptography Primer - Theoretical Foundations and Practical Applications*. Springer, 2013. ISBN: 978-3-642-41385-8. DOI: 10.1007/978-3-642-41386-5. URL: <https://doi.org/10.1007/978-3-642-41386-5>.
- [8] François Morain and Jorge Olivos. “Speeding up the computations on an elliptic curve using addition-subtraction chains”. In: *RAIRO Inform. Théor. Appl.* 24.6 (1990), pp. 531–543. ISSN: 0988-3754. DOI: 10.1051/ita/1990240605311. URL: <https://doi.org/10.1051/ita/1990240605311>.
- [9] Christof Paar and Jan Pelzl. *Understanding Cryptography - A Textbook for Students and Practitioners*. Springer, 2010. ISBN: 978-3-642-04100-6. DOI: 10.1007/978-3-642-04101-3. URL: <https://doi.org/10.1007/978-3-642-04101-3>.
- [10] Andrew Regenscheid et al. “Recommendations for Discrete-Logarithm Based Cryptography: Elliptic Curve Domain Parameters”. In: (Oct. 2019). DOI: 10.6028/nist.sp.800-186-draft. URL: <https://doi.org/10.6028/nist.sp.800-186-draft>.

- [11] Joseph H. Silverman. *The arithmetic of elliptic curves*. Second. Vol. 106. Graduate Texts in Mathematics. Springer, Dordrecht, 2009, pp. xx+513. ISBN: 978-0-387-09493-9. DOI: 10 . 1007 / 978 - 0 - 387 - 09494 - 6. URL: <https://doi.org/10.1007/978-0-387-09494-6>.
- [12] Nigel Smart. *Cryptography: An Introduction*. <https://www.cs.umd.edu/~waa/414-F11/IntroToCrypto.pdf>. Accessed 2022-05-04. 2016.