# Cryptography

Chun Min Tan

# Cryptography

## Contents


**1 Introduction to Cryptography** **3**

**2 Introduction to Elliptic Curve** **3**
2.1 Elliptic Curve . . . 3
2.2 Elliptic curve over finite fields . . . 7

**3 Elliptic Curve Cryptography** **7**
3.1 Elliptic Diffie-Hellman key exchange . . . 7
3.2 ElGamal public key cryptosystem . . . 8
3.3 The Double-and-Add algorithm . . . 8
3.3.1 Further improvements using ternary expansion . . . 8
3.4 How hard is ECDLP? . . . 9
3.4.1 Collision Theorem . . . 10
3.4.2 Naive Collision Algorithm . . . 11
3.4.3 Pollard $\rho$ method . . . 12
3.4.4 Solving ECDLP using Pollard's $\rho$ method . . . 16
3.4.5 Pollard's $\rho$ algorithm for factorisation . . . 17
3.4.6 Pohlig-Hellman algorthm . . . 18
3.4.7 General Attack of ECDLP . . . 20

**Appendices** **22**

**A Chinese Remainder Theorem** **22**

# Preface

This is one of the year-end projects given to Imperial 1st Year maths student in 2023. My instructor for this project is Professor Paolo Cascini. In this text, I have compiled all of my findings throughout the research journey. It includes basic intrduction to elliptic curve, some standard elliptic curve encryption method, but the main focus of this texts is to explore the algorithms used to crack elliptic curve cryptography and anaysis of their efficiency. Readers are assumed to have the foundational knowledge on modular arithmetic, probability theory, calculus, basis group theory, rings and fields, calculus, and analysis.

# Notations and Definitions

- $\mathbb{Z}/m\mathbb{Z}$ is a the integer ring modulo $m$.

$$\mathbb{Z}/m\mathbb{Z} = \{0, 1, \ldots, m-1\}$$

- When $m = p$ a prime, then the ring $\mathbb{Z}/p\mathbb{Z} = \mathbb{F}_p$ becomes a finite field because if we remove the 0 element, all element has inverses.
- We define the multiplicative group of $\mathbb{F}_p$ as $(\mathbb{F}_p)^* = \mathbb{F}_p \setminus \{0\}$ where each element in this set has a multiplicative inverse. This can also be denoted as

$$(\mathbb{F}_p)^* = (\mathbb{Z}/p\mathbb{Z})^*$$

- $\langle P \rangle := \{nP : n \in \mathbb{Z}\} \cup \{\mathcal{O}\}$
- Denote the operation that $a$ is randomly selected from a set $S$ as $a \in_R S$
- Steps: A step is the process of turning a value $A$ with any given informations and compute another value $B$ from $A$. The computation can be addition (arithmetic or elliptic curve), multiplication, finding an inverse, taking modulo, finding gcd, etc.
- Operations: The number of of a **step** is the number of computation require in order to successfully execute a step.
- $f(x) = O(g(x))$ if and only if, the following limit exists

$$\lim_{x \to \infty} \frac{f(x)}{g(x)}$$

# 1 Introduction to Cryptography

The goal of cryptography is to allows two or more people to transfer confidential information through a trusted channel or monitored medium. For many years cryptograpghy relies on the fact that on the assumption that the people attempting to communicate, call them Bob and Alice, share a secret key that their adversary does not possess. Hence, in order to start communicating by using this secret key, they must start agree upon on the same secret key secretly, which is a disadvantage. This type of cryptosystem is called *private key cryptosystem.* Using this secret key, they can both encrypt and decrypt messages, so Bob and Alice have equal (or symmetric) knowledge and abilities. For this reason, ciphers of this sort are knwon as *symmetric ciphers.*
Mathematically, a symmetric cipher uses a key $k$ chosen from a space (or a set) of possible keys $\mathcal{K}$ to encrypt a plaintext mesage $m$ chosen from a space ofpossible messages $\mathcal{M}$, and the result of the encryption process is a ciphertext $c$ belonging to a space of possible ciphertexts $\mathcal{C}$. Thus the encryption may be viewed as a function

$$e : \mathcal{K} \times \mathcal{M} \to \mathcal{C}$$

where the domain $\mathcal{K} \times \mathcal{M}$ is the set of pairs $(k, m)$ consisting of key $k$ and a plaintext $m$ and whose range is the space of ciphertexts $\mathcal{C}$. Similarly, decryption is a function

$$d : \mathcal{K} \times \mathcal{C} \to \mathcal{M}$$

It is sometimes convenient to write the dependnce on $k$ as a subscript as the $k$ is decided before the encryption. In other words we have

$$e_k : \mathcal{M} \to \mathcal{C}$$
$$d_k : \mathcal{C} \to \mathcal{M}$$

Of course, we want the decryption function to undo the encryption function, i.e

$$d_k(e_k(m)) = m$$

This also implies that $e_k$ must be a one-to-one function. It is safest to assume that anyone who isn't Alice or Bob knows the encryption method trhat is being employed. This illustrates the basic premise of modern cryptography called Kerckhoff's principle.

**Definition 1.1.** ***(Kerckhoff's Principle)*** *The security of a crypotsystem should depend only on the secrecy of the key, and not on the secrecy of the encryption algorithm itself.*

If $(\mathcal{K}, \mathcal{M}, \mathcal{C}, e, d)$ is to be a successful cipher, it must have the following properties:

1. For any key $k \in \mathcal{K}$ and plaintext $m \in \mathcal{M}$, it must be easy to compute the ciphertext $e_k(m)$
2. For any $k \in \mathcal{K}$ and ciphertext $c \in \mathcal{C}$, it must be easy to compute the plaintext $d_k(c)$.
3. Given any ciphertexts $c \in \mathcal{C}$ encrypted using key $k \in \mathcal{K}$, it must be difficult to compute $d_k(c)$ without knowing $k$.
4. Given one or more pair of plaintexts and their corresponding ciphertexts, $(m_1, c_1), (m_2, c_2), \ldots, (m_n, c_n)$, it is difficult to decrypt any ciphertext $c$ that is not in the given list without knowing $k$. This is known as security against *chosen plaintext attack*

So the next question is can be encrypt and decrypt messages without first agreeing on th same secret key? The answer is YES, and this method is first suggested by Diffie and Hellman, and how the encrpytions/decryptions works as well as any susceptible attacks will be the main focus of remaining chapters

# 2 Introduction to Elliptic Curve

## 2.1 Elliptic Curve

**Definition 2.1.** ***(Elliptic Curve)*** *An elliptic curve $E$ is the set of solutions to a Weierstrass equation*

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

*together with the extra point $\mathcal{O}$ further with $a, b$ satisfying*

$$4a^3 + 27b^2 \neq 0$$

The extra point $\mathcal{O}$ can be thought of as the point $(0, \infty)$. Before explaining why we need the condition $4a^3 + 27b^2 \neq 0$, we would need to define the addition steps $\oplus$. Let $P, Q \in E$, we start by drawing the line $L$ through $P$ and $Q$. This line $L$ intersect $E$ at $R$, then we reflect $R$ on the $x$-axis to obtain $R'$, then we define $P \oplus Q = R'$ (as illustrated in the figure below).

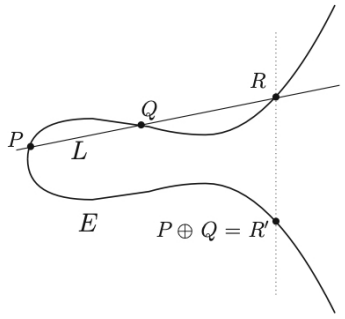



Figure 1: The addition law on an elliptic curve [1, p.281]

Above shows the geometrical interpretation of elliptic addition. We may formalise the notion of addition in the folloing theorem. (From now on, we will use $+$ instead of $\oplus$ for simplicity)

**Theorem 2.1.** ***(Elliptic Curve Addition Algorithm)***. *Let*

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

*be an elliptic curve and let $P_1, P_2$ be points on $E$. Then we have*

1. *If $P_1 = \mathcal{O}$, then $P_1 + P_2 = P_2$*
2. *If $P_1 = (x, y)$ and $P_2 = -P_1 = (x, -y)$, then $P_1 + P_2 = \mathcal{O}$*
3. *Otherwise write $P_1 = (x_1, y_1)$ and $P_2 = (x_2, y_2)$, then define $\lambda$ by*

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

*then*

$$P_1 + P_2 = P_3 = (\lambda^2 - x_1 - x_2, \lambda(x_1 - x_3) - y_1)$$

*Proof.*

1. If $P_1 = \mathcal{O}$ then drawing a line joining $P_2$ and $\mathcal{O}$ is just a vertical line passing $P_2$, and this line intersects $E$ at $-P_2$ (as $E$ is symmetrical about the $x$-axis), hence after reflection we have $P_1 + P_2 = P_2$.
2. If $P_2 = -P_1$, then the line joining $P_1$ and $P_2$ is a vertical line, which intersect $\mathcal{O}$, upon reflection we get back $\mathcal{O}$, hence result follows.
3. Consider $P_1 = (x_1, y_1)$ and $P_2 = (x_2, y_2)$, then define the gradient of the line $L$ joining $P_1, P_2$ as $\lambda$, if $P_1 \neq P_2$, then
$$\lambda = \frac{y_2 - y_1}{x_2 - x_1}$$
if $P_1 = P_2$, then $L$ is the tangent on $E$ and $P_1$, the gradient of $E$ can be found by differentiating explicitly. Doing so gives
$$2y\frac{dy}{dx} = 3x^2 + a \implies \frac{dy}{dx} = \frac{3x^2 + a}{2y}$$
since we have $P_1 = (x_1, y_1) = P_2$, hence the gradient at $P_1$ is
$$\lambda = \frac{3x_1^2 + a}{2y_1}$$

then this gives

$$L : y = \lambda x + (y_1 - \lambda x_1)$$

now substituting this result into $E$ and we get

$$(\lambda x + y_1 - \lambda x_1)^2 = x^3 + ax + b$$
$$x^3 - \lambda^2 x^2 + (a - 2\lambda(y_1 - \lambda x_1))x + (b - (y_1 - \lambda x_1)^2) = 0$$

But we know that the two roots of this equation are $x_1, x_2$ as $L$ intersect $P_1, P_2$. Hence we know that it can be factorised into

$$x^3 - \lambda^2 x^2 + (a - 2\lambda(y_1 - \lambda x_1))x + (b - (y_1 - \lambda x_1)^2) = (x - x_1)(x - x_2)(x - x_3)$$

by Vieta's theorem, we know that the sum of roots equals $\lambda^2$, hence we have

$$x_1 + x_2 + x_3 = \lambda^2$$
$$x_3 = \lambda^2 - x_1 - x_2$$

to find $y_3$ we just substitute it back to $L$ and reflect it to get

$$y_3 = -(\lambda x_3 + (y_1 - \lambda x_1)) = \lambda(x_1 - x_3) - y_1$$

upon reflection we get hence we have

$$P_3 = (\lambda^2 - x_1 - x_2, \lambda(x_1 - x_3) - y_1)$$

□

**Theorem 2.2.** ***(Addition Law)*** *Let $E$ be an elliptic curve. Then the addition law on $E$ has the following properties:*

1. *Identity: $P + \mathcal{O} = \mathcal{O} + P = P$ for all $P \in E$*
2. *Inverse: $P + (-P) = \mathcal{O}$ for all $P \in E$*
3. *Associative: $(P + Q) + R = P + (Q + R)$ for all $P, Q, R \in E$*
4. *Commutative: $P + Q = Q + P$ for all $P, Q \in E$*

*Proof.*

1. Using the geometrical definition of addition, we have that the line joining $P$ and $\mathcal{O}$ is a vertical line passing through $P$, hence it intersect $E$ at $-P$ (other than $P$), then the reflection of $-P = P$, therefore we have $P + \mathcal{O} = P$, and by commutative property we also have $\mathcal{O} + P = P$.
2. The line joining $P$ and $-P$ is a vertical line, therefore the third point is $\mathcal{O}$, and hence reflection on the $x$-axis is still $\mathcal{O}$, then result follows.
3. If one of $P, Q, R$ is $\mathcal{O}$, then the proof is trivial. Otherwise we may write $P = (x_1, y_1), Q = (x_2, y_2), R = (x_3, y_3)$. Then using the addition algorithm, the result follows.
4. This is clear from the geometric definition of addition as line passing through $P$ and $Q$ is the same as the line passing through $Q$ and $P$, and hence intersect at the same point.

□

**Remark:** [2] Suppose we have a cubic equation

$$\alpha x^3 + \beta x^2 + ax + b$$

then we know that the discriminant of the cubic with roots $x_1, x_2, x_3$ can be written as

$$\Delta = \alpha^4 (x_1 - x_2)^2 (x_1 - x_3)^2 (x_2 - x_3)^2$$

Then by Vieta's formula [3], we have that

$$\begin{aligned} x_1 + x_2 + x_3 &= -\frac{\beta}{\alpha} \\ x_1x_2 + x_1x_3 + x_2x_3 &= \frac{a}{\alpha} \\ x_1x_2x_3 &= -\frac{b}{\alpha} \end{aligned}$$

expanding the discriminant we get

$$\Delta = \alpha^4((x_1x_2^2 + x_2x_3^2 + x_3x_1^2) - (x_1^2x_2 + x_2^2x_3 + x_3^2x_1))^2$$

If we denote $m = x_1x_2^2 + x_2x_3^2 + x_3x_1^2$ and $n = x_1^2x_2 + x_2^2x_3 + x_3^2x_1$, then $\Delta = \alpha^4(m-n)^2$. We can also verify that

$$\begin{aligned} m + n &= x_1x_2\left(x_1 + x_2\right) + x_2x_3\left(x_2 + x_3\right) + x_3x_1\left(x_3 + x_1\right) \\ &= \left(x_1 + x_2 + x_3\right)\left(x_1x_2 + x_1x_3 + x_2x_3\right) - 3x_1x_2x_3 \\ &= \frac{-\beta a + 3\alpha b}{\alpha^2}. \end{aligned}$$

Next we compute $mn$ as expressed it as the coefficients of the cubic polynomials

$$\begin{aligned} mn =& \left(x_1x_2\right)^3 + \left(x_1x_3\right)^3 + \left(x_2x_3\right)^3 + 3\left(x_1x_2x_3\right)^2 + x_1x_2x_3\left(x_1^3 + x_2^3 + x_3^3\right) \\ =& \left(x_1x_2 + x_1x_3 + x_2x_3\right)^3 - 3\left(x_1 + x_2 + x_3\right)\left(x_1x_2 + x_1x_3 + x_2x_3\right)\left(x_1x_2x_3\right) + 3\left(x_1x_2x_3\right)^2 + 3\left(x_1x_2x_3\right)^2 \\ &+ x_1x_2x_3\left(\left(x_1 + x_2 + x_3\right)^3 - 3\left(x_1 + x_2 + x_3\right)\left(x_1x_2 + x_1x_3 + x_2x_3\right) + 3x_1x_2x_3\right) \\ =& \frac{a^3}{\alpha^3} - \frac{3\beta ab}{\alpha^3} + \frac{6b^2}{\alpha^2} + \frac{\beta^3 b}{\alpha^4} - \frac{3\beta ab}{\alpha^3} + \frac{3b^2}{\alpha^2} \\ =& \frac{\alpha b^3 - 6\alpha\beta ab + 9\alpha^2b^2 + \beta^3 b}{\alpha^4}. \end{aligned}$$

Finally, we may compute $(m-n)^2 = (m+n)^2 - 4mn$, hence we have

$$\begin{aligned} \Delta = \alpha^4(m-n)^2 &= \alpha^4\left(\frac{-\beta a + 3\alpha b}{\alpha^2}\right)^2 - 4\alpha^4\left(\frac{\alpha b^3 - 6\alpha\beta ab + 9\alpha^2b^2 + \beta^3 b}{\alpha^4}\right) \\ &= \beta^2a^2 - 6\alpha\beta ab + 9\alpha^2b^2 - 4\alpha a^3 + 24\alpha\beta ab - 36\alpha^2b^2 - 4\beta^3 b \\ &= \beta^2a^2 - 4\alpha a^3 - 27\alpha^2b^2 + 18\alpha\beta ab \end{aligned}$$

Now setting $\alpha = 1, \beta = 0$, we get

$$\Delta = -4a^3 - 27b^2$$

and when $\Delta = 0$, we get repeated roots, so

$$4a^3 + 27b^2 = 0 \implies \text{reperated roots}$$

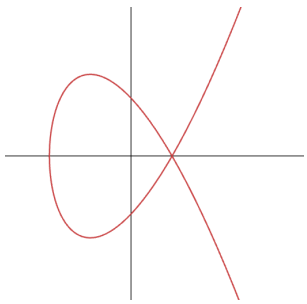


Figure 2: Repeated roots on elliptic curve

this is a problem if we define $P$ to be the repeated roots then $2P = P + P$ is undefined as there are 2 tangent lines at $P$.

**Definition 2.2.** ***(Repeated addition / Multiplication)*** *Repeated addition of a point $P \in E$ is represented as multiplication of a point by an integer,*

$$nP = \underbrace{P + P + \cdots + P}_{n \text{ copies}}$$

### 2.2 Elliptic curve over finite fields

We simply define an elliptic curve over $\mathbb{F}_p$ to be an equation of the form

$$E : y^2 = x^2 + ax + b \text{ with } a, b \in \mathbb{F}_p \text{ satisfying } 4a^3 + 27b^2 \neq 0$$

and then we look at the points on $E$ with coordinates $\mathbb{F}_p$ which we denote as

$$E(\mathbb{F}_p) = \{(x, y) : x, y \in \mathbb{F}_p \wedge y^2 = x^3 + ax + b\} \cup \{\mathcal{O}\}$$

let $P, Q \in E(\mathbb{F}_p)$ then $P + Q$ is defined using the addition algorithm . Note that in that in the algorithm, we have division, we define dividing $x$ as multiplying by the inverse of $x$, denoted as $x^{-1} \in \mathbb{F}_p$. Depending on the choices of $p$, not all elements have an inverse, hence we generally restrict $p$ to be a prime to ensure all numbers have an inverse.
The addition law also holds for $E(\mathbb{F}_p)$, and that makes $(E(\mathbb{F}_p), +)$ an abelian group.

## 3 Elliptic Curve Cryptography

In elliptic cryptography, the hardness of the elliptic curve discrete logarithm problem (ECDLP) is essential for the secuity of all elliptic curve cryptographic schemes.

**Definition 3.1.** ***(ECDLP)*** *The elliptic curve discrete logarithm problem (ECDLP) is: given an elliptic curve $E$ defined over a fininte field $\mathbb{F}_p$, a point $P \in E(\mathbb{F}_p)$ of order $N$, and a point $Q \in \langle P \rangle$, find the integer $n \in [0, N-1]$ such that $Q = nP$.*

**Definition 3.2.** ***(Discrete Logarithm)*** *Suppose $P, Q \in E(\mathbb{F}_p)$, then the integer $n$ such that $Q = nP$ is known as the discrete logarithm of $Q$ to the base $P$, denoted as*

$$n = \log_P Q$$

### 3.1 Elliptic Diffie-Hellman key exchange

Suppose Alice and Bob wants send each other encrypted messages. They agree to use a parituclar elliptic curve $E(\mathbb{F}_p)$ and a parituclar point $P \in E(\mathbb{F}_p)$. Alice chooses a secret integer $n_A$ and Bob chooses a secret integer $n_B$, then each of them compute $Q_A, Q_B$ respectively where

$$Q_A = n_A P$$
$$Q_B = n_B P$$

then Alice and Bob exchange the values $Q_A, Q_B$. Alice uses her secret integer to compute $n_A Q_B = n_A n_B P$, and Bob uses his secret integer to compute $n_B Q_A = n_B n_A P = n_A n_B P$ . Now Alice and Bob both have the common values, and with this common values, a symmetric ciphers can be used. A summary of elliptic Diffie-Hellman key exchange is given below

| **Public parameter creation** | |
|---|---|
| A trusted party chooses and publishes a (large) prime $p$, an elliptic curve $E$ over $\mathbb{F}_p$, and a point $P$ in $E(\mathbb{F}_p)$. | |
| **Private computations** | |
| **Alice** | **Bob** |
| Chooses a secret integer $n_A$. Computes the point $Q_A = n_A P$. | Chooses a secret integer $n_B$. Computes the point $Q_B = n_B P$. |
| **Public exchange of values** | |
| Alice sends $Q_A$ to Bob $\longrightarrow$ $Q_A$ | |
| $Q_B$ $\longleftarrow$ Bob sends $Q_B$ to Alice | |
| **Further private computations** | |
| **Alice** | **Bob** |
| Computes the point $n_A Q_B$. | Computes the point $n_B Q_A$. |
| The shared secret value is $n_A Q_B = n_A(n_B P) = n_B(n_A P) = n_B Q_A$. | |

Figure 3: Diffie-Hellman key exchange summary [1, p.297]

## 3.2 ElGamal public key cryptosystem

**Potential problems:**

1. Man-in-the-middle attack: In order to figure out the common value, one only needs to figure out $n_A$ or $n_B$, suppose say we want to find $n_A$. After obtaining the value $n_A$, simply computing $n_A Q_B$ yields the common value. In other words, the middle-man needs to solve the **elliptic curve discrete logarithm problem (ECDLP)** to get the value $n_A$ or $n_B$
2. Efficiency of computing $n_A P, n_B P$ for Alice and Bob: Alice and Bob know their own private key, if they compute $nP$ using repeated addition, then it's no easier than the middle-man to crack the encryption. Hence is there an easier method to compute $nP$ from known $n$?

## 3.3 The Double-and-Add algorithm

As mentioned in the last part, the efficiency of computing $nP$ from known $n$ is an important problem. The algorithm works as follows:

1. We write $n$ in binary form as
$$n = n_0 + n_1 \cdot 2 + n_2 \cdot 2^2 + \cdots n_r \cdot 2^r$$
where $n_i \in \{0, 1\}$ for $i = 0, 1, \ldots, r$
2. Next we compute the following quantities
$$Q_0 = P, \;\; Q_1 = 2Q_0, \;\; Q_2 = 2Q_1, \;\; \cdots \;\; , Q_r = 2Q_{r-1}$$
Notice that $Q_i$ is simply just twice the previous $Q_{i-1}$, so
$$Q_i = 2Q_{i-1} = 2^i Q_0$$
**Note that each term only requires one more step to compute from the previous term.**
3. Finally, we compute $nP$ using **at most $r$ additional additions**,
$$nP = n_0 Q_0 + n_1 Q_1 + \cdots + n_r Q_r$$
as each $n_i \in \{0, 1\}$.

It is not hard to see that this algorithm requires at most $2r$ step in $E(\mathbb{F}_p)$ ($r$ additions as menioned above, and $r$ multiplication of 2 as $Q_0 = P$ does not require ay multiplication to start with.). Since $r = \lfloor \log_2 n \rfloor \leq \log_2 n$ (because index starts from 0), we have that the number of step is no more than $\boxed{2 \log_2 n}$.

### 3.3.1 Further improvements using ternary expansion

**Definition 3.3.** ***(Ternary Expansion)*** *In this context, we define the ternary expansion of $n$ as writing $n$ as a sum of positive and negative powers of 2.*

Writing $n$ in ternary expansion is a good idea as it can be proven that at most half of the digits in ternary expansion are nonzero, which decreases the number of addition step. Furthermore, substracting two points is relatively easy in elliptic curve as $-(x, y) = (x, -y)$

**Proposition 3.0.1.** *Let $n$ be a posiive integer and $k = \lfloor \log_2 n \rfloor + 1$. Then we can always write $n$ as*

$$n = u_0 + u_1 \cdot 2 + \cdots u_k \cdot 2^k$$

*where $u_i \in \{-1, 0, 1\}$ and at most $\frac{k}{2}$ of the $u_i$ are nonzero. (Note that $k$ is the number of digits in binary expansion, and in this ternary expansion the number of digits is at most 1 more than the binary expansion)*

*Proof.* This will be a constructive proof. We start by writing $n$ in binary form

$$n = n_0 + n_1 \cdot 2 + \cdots + n_{k-1} \cdot 2^{k-1}$$

Working from left to right, we look for the first occurence of two or more consecutive nonzero $n_i$ coefficients. For example, suppose that

$$n_s = n_{s+1} = \cdots = n_{s+t-1} = 1 \qquad \text{and} \qquad n_{s+t} = 0$$

for some $t \geq 2$ (in order for 2 or more consecutive nonzero $n_i$ to happen). In other words, the expression

$$2^s + 2^{s+1} + \cdots + 2^{s+t-1} + 0 \cdot 2^{s+t}$$

appears in the binary expansion. We can write this as

$$2^s + 2^{s+1} + \cdots + 2^{s+t-1} + 0 \cdot 2^{s+t} = 2^s(1 + 2 + 4 + \cdots + 2^{t-1}) = 2^s(2^t - 1) = -2^s + 2^{s+t}$$

Repeating this we get the ternary expansion of $n$, in which no two consecutive $u_i$ are nonzero. □

**Comparison** :

1. Binary expansion: The upper bound for the number step is $\boxed{2 \log_2 n}$ and on average we expect the binary expansion to have half of digits 1 and the other half 0. Hence, the average number of step required will be $\boxed{\frac{3}{2} \log_2 n}$ where ( $\frac{1}{2} \log_2 n$ are addition and $\log_2 n$ are doubling).
2. Ternary expansion: The upper limit is $\lfloor \log_2 n \rfloor + 1$ doubling and at most $\lfloor (\lfloor \log_2 n \rfloor + 1)/2 \rfloor + 1$ addition which gives an upper bound of $\boxed{\frac{3}{2} \log_2 n + \frac{5}{2}}$. On average $n$ has $\frac{2}{3}$ of the digits in ternary expansion being 0, hence on average we would expect $\lfloor \log_2 n \rfloor + 1$ doubling and $\frac{1}{3}(\lfloor \log_2 n \rfloor + 1)$. Therefore on average we would expect $\boxed{\frac{4}{3} \log_2 n + \frac{7}{3}}$ step.

## 3.4 How hard is ECDLP?

As described in section 2.1, if one can solve $Q_A = n_A P$, then they are able to decrypt the message as the attacker in the man-in-the-middle attack. Therefore essentially solving $Q = nP$ is the core to how secure elliptic curve cryptography is. A susceptible attack is as follows: Since $P, Q_A, p, E(\mathbb{F}_p)$ are public keys (from now on we will write $Q$ instead of $Q_A$ as for generality of ECDLP), suppose Eve (the middle-woman) chooses random integers $j_1, \ldots, j_r$ and $k_1, \ldots, k_r$ between 1 and $p$. Then Eve constuct the following lists

$$\begin{aligned} &\text{List 1 : } j_1 P, j_2 P, \ldots, j_r P \\ &\text{List 2 : } k_1 P + Q, k_2 P + Q, \ldots, k_r P + Q \end{aligned}$$

Computing such term can be spid up by using Double-and-Add algorithm. As soon as Eve finds a match (collision) between two lists, then she is done because suppose

$$j_\alpha P = k_\beta P + Q \implies (j_\alpha - k_\beta) P = Q$$

and since $p$ is a prime, $j_\alpha - k_\beta \equiv n \pmod{p}$ as the unique modulo, hence the ECDLP is solved. The next question is how large should $r$ be to have a good chance for collision? In other words, how secure the encryption depends on how large $r$ needs to be to provide a good chance for collision. This brings us to the next topic in probability theory.

### 3.4.1 Collision Theorem

**Proposition 3.0.2.** *For all $x \in \mathbb{R}$, we have*

$$e^{-x} \geq 1 - x$$

*Proof.* Consider $f(x) = e^{-x}$, then by Taylor's Theorem we have

$$f(x) = f(0) + f'(0)x + \frac{f^{(2)}(c)}{2!}x^2$$

where $c$ is between 0 and $x$. Hence we have

$$f(x) = 1 - x + \frac{e^{-c}}{2!}x^2$$

and since $e^{-c} > 0$ for all $c \in \mathbb{R}$ and $x^2 \geq 0$ for all $x \in \mathbb{R}$, hence

$$f(x) = 1 - x + \frac{e^{-c}}{2!}x^2 \geq 1 - x$$

□

**Theorem 3.1.** ***(Collision Theorem)*** *An urn contain $N$ balls, of which $n$ are red and $N - n$ are blue. Bob randomly selects a ball from the urn, replaces it in the urn, randomly selects a second ball, replaces it, and so on. He does this until he has looked at a total of $m$ balls. Then if $X$ is the random variable of the number of red balls observed, then*

$$P(X \geq 1) \geq 1 - e^{-mn/N}$$

*Proof.* We have that $P(X \geq 1) = 1 - P(X = 0)$, and $P(X = 0)$ is quite easy to compute as it is the probability of oberving $m$ blue balls from the $m$ observations. Hence we have

$$P(X = 0) = \left(\frac{N - n}{N}\right)^m = \left(1 - \frac{n}{N}\right)^m$$

therefore we have

$$P(X \geq 1) = 1 - \left(1 - \frac{n}{N}\right)^m$$

by Proposition 2.0.2, we have that

$$e^{-n/N} \geq 1 - \frac{n}{N} \implies e^{-nm/N} \geq \left(1 - \frac{n}{N}\right)^m$$

which implies

$$P(X \geq 1) \geq 1 - e^{-mn/N}$$

□

### 3.4.2 Naive Collision Algorithm

**Definition 3.4.** ***(Order of points on elliptic curve)*** *Let $P \in E(\mathbb{F}_p)$, the order of $P$ is denoted as $ord(P)$ and is defined as*

$$ord(P) = \min\{n \in \mathbb{Z}_{>0} : nP = \mathcal{O}\}$$

Since we know that $E(\mathbb{F}_p)$ is a finite group, therefore we have that the order of $P \in E(\mathbb{F}_p)$ is finite since the set generated by $P$ only takes finitely many points. Consider $Q = nP$ where $Q, P \in E(\mathbb{F}_p)$ and here is how we can use the naive algorithm to solve for $n$.

1. We first compute $N = \text{ord}(P)$ (if it isn't given in the public key). This can be done by brute force for relatively small $p$ or by Pollard $\rho$ algorithm to find the order by solving $nP = \mathcal{O}$. (But it doesn't make sense to use Pollard $\rho$ algorithm for naive algorithm as we could solve it directly using Pollard $\rho$ method as well)

2. We begin choosing $y_1, y_2, \ldots, y_r$ where $1 \leq y_i \leq N$ and computing the values

   $$y_1P, y_2P, \ldots, y_rP \in \langle P \rangle \subseteq E(\mathbb{F}_p)$$

   by using Double-and-Add algorithm.

3. Next we choose additional random integers $z_1, \ldots, z_r$ again where $1 \leq z_i \leq N$ then we compute the following list

   $$z_1P + Q, z_2P + Q, \ldots, z_rP + Q \in \langle P \rangle \subseteq E(\mathbb{F}_p)$$

   note that $z_iP + Q \in \langle P \rangle$ because if we assume that $Q = nP$ has a solution, then there exists some $n_0 : Q = n_0P \in \langle P \rangle$.

4. Now we search for a collision, and **if** a collision occurs, then we have solved the problem as

   $$y_\alpha P = z_\beta P + Q \implies Q = (y_\alpha - z_\beta)P$$

   and therefore $y_\alpha - z_\beta \equiv n \pmod N$ is the solution to ECDLP.

**So how likely is such a collision occur?** Suppose we consider picking $r \approx 3\sqrt{N}$. Now if we use the analogy of collision theorem, we treat the set $\langle P \rangle$ as the urn. Where we have

$$\langle P \rangle = \{P, 2P, \ldots, NP\} = \{\mathcal{O}, P, 2P, \ldots, (N-1)P\}$$

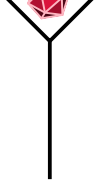
then first we pick $y_1, \ldots, y_r$ from this set (urn), we treat this as a way of colouring $r$ balls red. Then we replace all these balls into the urn, now we pick $z_1P + Q, \ldots, z_rP + Q$ again from the set(urn). Then the we have

$$P(\text{at least one collision/pairing}) = 1 - \left(1 - \frac{r}{N}\right)^r \geq 1 - e^{-r^2/N}$$

but since we have $r \approx 3\sqrt{N}$, hence this gives

$$P(\text{at least one collision/pairing}) \geq 1 - e^{-9} \approx 99.98\%$$

Therefore is it **almost certain that there will be pairing**.



**How long does it take for us to find this solution?**
In the first list, we required $r$ multiplication of $y_i$ to $P$, and from Double-and-Add algorithm, we know that each multiplication takes $2\log_2 y_i$ steps, and since $1 \leq y_i \leq N$, the upper bound for computing the first list is $2r\log_2 N$.
In the second list, we also required $r$ multiplication of $z_i$ and further another $r$ addtion of $Q$, therefore the number of steps needed here is $2r\log_2 N + r$.
Next we wish to check for collision. One way to do that is to first sort the first list, then for each element in the second list we do a binary search on the first list to a matching pair. Sorting the list using merge sort on average takes $r\log_2 r$, then the binary search for each element requires $\log_2 r$ steps. Hence in total this process takes $2r\log_2 r$

Since $r \approx 3\sqrt{N}$, therefore the total number of steps we need is

$$\begin{aligned} 2r\log_2 N + 2r\log_2 N + r + 2r\log_2 r &= 4r\log_2 N + 2r\log_2 r + r \\ &= r\log_2 N^4 r^2 + r \\ &\approx 3\sqrt{N}\log_2 N^4 \cdot 9N + 3\sqrt{N} \\ &= \boxed{O(\sqrt{N}\log_2 N)} \end{aligned}$$

### 3.4.3 Pollard $\rho$ method

**Definition 3.5.** ***(Forward Orbit)*** *Let $S$ be a finite set and let $f : S \to S$, suppose we start with some element $x \in S$, then we define*

$$\begin{aligned} x_0 &= x \\ x_i &= f(x_{i-1}) = f^i(x) \qquad i \geq 1 \end{aligned}$$

*Then the sequence $(x_n)_{n\geq 0}$ is called the forward orbit of $x$ by map $f$ and is denoted by $O_f^+(x)$*

**Remark:** The set $S$ is fintie, so eventually there must be a some element of $S$ that appears twice in the orbit $O_f^+(x)$. This illustrated below

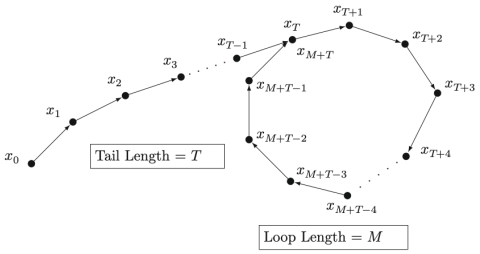



Figure 4: Pollard's $\rho$ method [1, p.235]

Before stating and proving Pollard's $\rho$ method, we first we will the following lemma

**Lemma 3.2.** *Let $F(t)$ be a function such that $F(t) \geq 0$ for $t \geq 0$ and has a continuous derivative with the property that*

$$\int_0^\infty F(t)dt \qquad \textit{converges}$$

*and for all $n$*

$$\frac{1}{n}\sum_{i=0}^{\infty} F\left(\frac{i}{n}\right) \qquad \textit{exists}$$

*Then for large values of $n$ we have*

$$\frac{1}{n}\sum_{i=0}^{\infty} F\left(\frac{i}{n}\right) \approx \int_0^\infty F(t)dt$$

*Proof.* We start with the defnite integral of $F(t)$ over the interval $0 \leq t \leq A$ for some $A \in \mathbb{N}$. Then by using standard partition we have that

$$\lim_{n\to\infty} \frac{1}{n}\sum_{i=0}^{An} f\left(\frac{i}{n}\right) = \int_0^A F(t)dt$$

Let $\varepsilon > 0$ be arbitrary, then there exists sufficiently large $n_0$ such that $\forall n \geq n_0$ we have

$$\left| \frac{1}{n} \sum_{i=0}^{A_n} f\left(\frac{i}{n}\right) - \int_0^A F(t)dt \right| < \frac{\varepsilon}{3}$$

further given that

$$\frac{1}{n} \sum_{i=0}^{\infty} F\left(\frac{i}{n}\right)$$

exists, this implies that there exists some sufficiently large $A_1 \in \mathbb{R}$ such that $\forall A \geq A_1$ we have

$$\left| \frac{1}{n} \sum_{i=0}^{\infty} F\left(\frac{i}{n}\right) - \frac{1}{n} \sum_{i=0}^{An} F\left(\frac{i}{n}\right) \right| < \frac{\varepsilon}{3}$$

Further we also know that $\int_0^\infty F(t)dt$ exists, hence there also exists some sufficiently large $A_2 \in \mathbb{R}$ such that $\forall A \geq A_2$ we have

$$\left| \int_0^A F(t)dt - \int_0^\infty F(t)dt \right| < \frac{\varepsilon}{3}$$

Now consider for $n \geq n_0$ and $A \geq \max(A_1, A_2)$, we have

$$\begin{aligned}
\left| \frac{1}{n} \sum_{i=0}^{\infty} F\left(\frac{i}{n}\right) - \int_0^\infty F(t)dt \right| &= \left| \frac{1}{n} \sum_{i=0}^{\infty} F\left(\frac{i}{n}\right) - \frac{1}{n} \sum_{i=0}^{An} F\left(\frac{i}{n}\right) + \frac{1}{n} \sum_{i=0}^{An} F\left(\frac{i}{n}\right) - \int_0^A F(t)dt + \right. \\
&\quad \left. \int_0^A F(t)dt - \int_0^\infty F(t)dt \right| \\
&\leq \left| \frac{1}{n} \sum_{i=0}^{\infty} F\left(\frac{i}{n}\right) - \frac{1}{n} \sum_{i=0}^{An} F\left(\frac{i}{n}\right) \right| + \left| \frac{1}{n} \sum_{i=0}^{An} F\left(\frac{i}{n}\right) - \int_0^A F(t)dt \right| + \\
&\quad \left| \int_0^A F(t)dt - \int_0^\infty F(t)dt \right| \\
&< \frac{\varepsilon}{3} + \frac{\varepsilon}{3} + \frac{\varepsilon}{3} = \varepsilon
\end{aligned}$$

□

**Lemma 3.3.** ***(Gaussian Integral)***

$$\int_{-\infty}^{\infty} e^{-x^2} dx = \sqrt{\pi}$$

*Proof.* Here is the classic way of deriving the result. Denote

$$I = \int_{-\infty}^{\infty} e^{-x^2} dx$$

then consider

$$I^2 = \int_{-\infty}^{\infty} e^{-x^2} dx \int_{-\infty}^{\infty} e^{-y^2} dy = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} e^{-(x^2+y^2)} dxdy$$

by change of coordiantes to polar coordinates, we have that

$$x = r\cos\theta$$
$$y = r\sin\theta$$

then we have that the Jacobian is

$$J = \begin{bmatrix} \frac{\partial x}{\partial r} & \frac{\partial x}{\partial \theta} \\ \frac{\partial y}{\partial r} & \frac{\partial y}{\partial \theta} \end{bmatrix} = \begin{bmatrix} \cos\theta & -r\sin\theta \\ \sin\theta & r\cos\theta \end{bmatrix}$$

hence we have that

$$I^2 = \int_0^{2\pi}\int_0^{\infty} e^{-r^2}|J|drd\theta$$

and computing $|J| = r$ , hence we have

$$I^2 = \int_0^{2\pi}\int_0^{\infty} e^{-r^2} r drd\theta$$

by using the substitution $u = r^2$, we have that $du = 2rdr$. This gives

$$I^2 = \int_0^{2\pi}\int_0^{\infty} \frac{1}{2}\cdot e^{-u}dud\theta = \frac{1}{2}\int_0^{2\pi} d\theta = \pi$$

therefore we have that

$$I = \sqrt{\pi}$$

□

**Lemma 3.4.**

$$\int_0^{\infty} t^2 e^{-t^2/2}dt = \sqrt{\frac{\pi}{2}}$$

*Proof.* From Lemma 2.3, we have

$$I = \int_{-\infty}^{\infty} e^{-x^2}dx = \sqrt{\pi}$$

now since $e^{-x^2}$ is even, we by symmetry we have that

$$I = 2\int_0^{\infty} e^{-x^2}dx = \sqrt{x} \implies \tilde{I} = \int_0^{\infty} e^{-x^2}dx = \frac{\sqrt{\pi}}{2}$$

Then consider the substitution $x = \sqrt{a}t$, this gives

$$\tilde{I} = \int_0^{\infty} \sqrt{a}e^{-at^2}dt$$

which implies

$$\int_0^{\infty} e^{-at^2}dt = \frac{1}{2}\sqrt{\frac{\pi}{a}}$$

differentiating both sides with respect to $a$ gives

$$\frac{d}{da}\int_0^{\infty} e^{-at^2}dt = -\int_0^{\infty} t^2 e^{-at^2}dt = -\frac{1}{4}\frac{\sqrt{\pi}}{a^{3/2}}$$

Setting $a = \frac{1}{2}$, we get

$$\int_0^\infty t^2 e^{-t^2/2} dt = \sqrt{\frac{\pi}{2}}$$

□

**Theorem 3.5.** ***(Pollard's $\rho$ method)*** *Let $S$ be a finite set containing $N$ elements, let $f : S \to S$ be a map, and let $x \in S$ be an initial point.*

1. ***(Floyd's cycle-detection algorithm)*** *Suppose that the forward $O_f^+(x)$ of $x$ has a tail of length $T$ and a loop of length $M$, then*

$$x_{2i} = x_i \quad \textit{for some } 1 \leq i < T + M$$

2. *If the map $f$ is sufficiently random, then expected value of $T + M$ is*

$$E(T + M) = \sqrt{\frac{\pi N}{2}} \approx 1.2533\sqrt{N}$$

*Proof.* 1. Clearly from figure 4 we have that for $j > i$

$$x_j = x_i \qquad \text{if and only if } i \geq T \text{ and } j \equiv i \pmod{M}$$

therefore we have that

$$x_{2i} = x_i \iff 2i \equiv i \pmod{M}$$

and this implies $M \mid 2i - i = i$. Therefore we have that $x_{2i} = x_i$ exactly when $M \mid i$, to achieve lowest possible $i$ value, it must be the first multiple of $M$. Since one of the numbers in $T, T + 1, \ldots, T + M - 1$ must be divisble by $M$ as they are $M$ consecutive integer, therefore there must exists some $i$ where $T \leq i < T+M$ such that $x_{2i} = x_i$, hence $x_{2i} = x_i$ for some $1 \leq i < T+M$

2. Suppose that we have computed the first $k$ values $x_0, x_1, \ldots, x_{k-1}$. Now we consider the probability that we do not get any matches. If we assume that the successive $x_i'$ are randomly chosen from the set $S$ ($f$ is sufficiently random), then we can compute this probability as

$$P\left(\substack{x_0, x_1, \ldots, x_{k-1} \\ \text{are all different}}\right) = \prod_{i=1}^{k-1} P\left(\substack{x_i \neq x_j \text{for} \\ \text{all } 0 \leq j < i} \middle| \substack{x_0, x_1, \ldots, x_{i-1} \\ \text{are all different}}\right)$$

But note that we have the following

$$P\left(\substack{x_i \neq x_j \text{for} \\ \text{all } 0 \leq j < i} \middle| \substack{x_0, x_1, \ldots, x_{i-1} \\ \text{are all different}}\right) = \frac{N - i}{N}$$

because given that $x_0, x_1, \ldots, x_{i-1}$ are all different, then there are $N - i$ elements in $S$ that is available for us to pick for $x_i$ that is distinct to all previous elements. Hence we have

$$P\left(\substack{x_0, x_1, \ldots, x_{k-1} \\ \text{are all different}}\right) = \prod_{i=1}^{k-1} \frac{N - i}{N} = \prod_{i=1}^{k-1} \left(1 - \frac{i}{N}\right)$$

Here we will use the approximation that for small $t$ we have

$$1 - t \approx e^{-t}$$

and since $N$ is usually large (in cryptography) therefore we have

$$P\left(\substack{x_0, x_1, \ldots, x_{k-1} \\ \text{are all different}}\right) \approx \prod_{i=1}^{k-1} e^{-i/N} = e^{-(1+2+\cdots+(k-1))/N} = e^{-k(k-1)/2N}$$

given that for large $k$ we have

$$\frac{k(k-1)}{2} \approx \frac{k^2}{2}$$

then we have that

$$P\left(\substack{x_0,x_1,\ldots,x_{k-1} \\ \text{are all different}}\right) \approx e^{-k^2/2N}$$

(we will provide more justification at the end of the proof). We now know the probability that $x_0, x_1, \ldots, x_{k-1}$ are all distinct. Assuming that they are distinct, then the probability that the next term is a match is given by

$$P(x_k \text{ is a match} \mid x_0, \ldots, x_{k-1} \text{ are distnct}) = \frac{k}{N}$$

As picking any first $k$ terms will lead to a match. Then we have that

$$\begin{aligned} P(x_k \text{ is the first match}) &= P(x_k \text{ is a match AND } x_0, \ldots, x_{k-1} \text{ are all distinct}) \\ &= P(x_k \text{ is a match} \mid x_0, \ldots, x_{k-1} \text{ are distinct}) \cdot P(x_0, \ldots, x_{k-1} \text{ are distinct}) \\ &= \frac{k}{N} e^{-k(k-1)/2N} \end{aligned}$$

define the random variable $X$ to be the index for which the first match occurs, then we have that

$$\begin{aligned} \mathrm{E}(X) = \sum_{x=0}^{\infty} xP(X=x) &= \sum_{x=1}^{\infty} xP(X=x) \\ &= \sum_{k=1}^{\infty} k \cdot \frac{k}{N} e^{-k(k-1)/2N} = \sum_{k=1}^{\infty} \frac{k^2}{N} e^{-k(k-1)/2N} \end{aligned}$$

now note that since $N$ is large, the first terms in the series are insignificant, further we have further into the series, the larger the $k$ values, we have the following approximation

$$\frac{k(k-1)}{2} \approx \frac{k^2}{2}$$

therefore with all the above information we deduce that

$$\mathrm{E}(X) = \sum_{k=1}^{\infty} \frac{k^2}{N} e^{-k^2/N}$$

then if we let $F(x) = x^2 e^{-x^2/2}$, we have by Lemma 2.2

$$\sum_{k=1}^{\infty} \frac{k^2}{N} e^{-k^2/N} = \sum_{k=1}^{\infty} F\left(\frac{k}{\sqrt{N}}\right) \approx \sqrt{N} \cdot \int_0^{\infty} t^2 e^{-t^2/2} dt$$

from Lemma 2.4, we can conclude that

$$\mathrm{E}(X) = \sqrt{N} \cdot \sqrt{\frac{\pi}{2}} \approx 1.2533\sqrt{N}$$

given that the expected first match is $\mathrm{E}(X)$, then this implies that we form the $\rho$ shape in $\mathrm{E}(X)$ number of terms, hence

$$\mathrm{E}(T+M) \approx 1.2533\sqrt{N}$$

□

### 3.4.4 Solving ECDLP using Pollard's $\rho$ method

The method explained is referenced from [4, §4.1.2]Let $P \in E(\mathbb{F}_q)$ and let $Q \in \langle P \rangle$, suppose $P$ has **prime** order $N = p$. The main idea of Pollard's $\rho$ to solve the ECDLP problem $Q = \tilde{n}P$ is to find $c', d', c'', d'' \in \mathbb{Z}/N\mathbb{Z}$ such that

$$c'P + d'Q = c''P + d''Q \implies (c' - c'')P = (d'' - d')Q = (d'' - d')\tilde{n}P$$

hence this implies

$$c' - c'' \equiv (d'' - d')\tilde{n} \pmod{p}$$

We may assume that $d'' - d' \neq 0$ (reasons explained later), then $d'' - d' \in \mathbb{F}_p^*$ hence we have

$$\tilde{n} \equiv (c' - c'')(d'' - d')^{-1} \pmod{p}$$

A naive method for finding such pairs $(c', d'), (c'', d'')$ is to select random integers $c, d \in_R [0, p-1]$ and store the triples $(c, d, cP + dQ)$ in a table sorted by third component until a point $cP + dQ$ is obtained for a second time (collision). But the drawback of this algorithm is the storage required is large as $p$ increases.
Pollard's $\rho$ algorithm finds $(c', d'), (c'', d'')$ in roughly the same expected time as the naive method, but requires significantly less storage (negligible). Idea is to first define a sufficiently random endomorphism function $f : \langle P \rangle \to \langle P \rangle$. Next we partition the set $\langle P \rangle$ to be $\{S_1, \ldots, S_L\}$. Next we define the partition function $H$,

$$H(X) = j \qquad \text{if } X \in S_j$$

Now let $a_j, b_j \in_R [0, p-1]$ for $1 \leq j \leq L$. Then

$$\begin{aligned} f : \langle P \rangle &\to \langle P \rangle \\ X &\mapsto X + a_j P + b_j Q \qquad \text{where } j = H(X) \end{aligned}$$

Note that for any point $X \in \langle P \rangle$ we can write $X = cP + dQ$ as $Q = \tilde{n}P$, further we have $f(X) = \overline{c}P + \overline{d}Q$ where $\overline{c} = c + a_j \pmod{p}$ and $\overline{d} = d + b_j \pmod{p}$. Now we let $X_0 \in_R \langle P \rangle$, then we define the sequence $(X_n)_{n \geq 0}$ as $X_i = f(X_{i-1})$ for $i \geq 1$. Then by Pollard $\rho$ and Floyd's cycle-finding algorithm, we compute the pair $(X_i, X_{2i})$ until $X_i = X_{2i}$. After each computation a new pair, the previous may be discarded; thus storage requirements are negligible because a match is ought to occur in one of the pair (guaranteed by Pollard $\rho$ method).

**How fast can we find a solution using Pollard's $\rho$ method:**
This method is referenced from Given that the expected tail length and cycle length is

$$\mathrm{E}(S + T) = \sqrt{\frac{p\pi}{2}}$$

Therefore we have that the expeccted number of evaluation of $f$ is $\sqrt{p\pi/2}$, and in each evaluation we need to compute $\overline{c}P + \overline{d}Q$, in each step of evaluation, it includes addition and reduction modulo $p$ to compute $\overline{c} = c + a_j \pmod{p}$ and $\overline{d} = d + b_j \pmod{p}$. So it takes approximately $2 \cdot 2 \cdot \sqrt{p\pi/2}$ steps, hence it takes $\boxed{O(\sqrt{p})}$.

### 3.4.5 Pollard's $\rho$ algorithm for factorisation

This is referrenced from a MIT lecture. Not to be confused with Pollard's $\rho$ method, which is used to solve ECDLP. Pollard $\rho$ algorithm is used to factor a composite number. Suppose $N$ is the number to be factored, write $N = pq$ where $p \leq \sqrt{N}$ is a prime and $q$ may be composite (note that for all composite $N$ there always exists a prime divisor $p \leq \sqrt{N}$, this exercise is left to the reader :D). Now consider picking $x_1, \ldots, x_r$ uniformly at random in $\mathbb{Z}/N\mathbb{Z}$ (the choice of $r$ will be determined later). Now suppose that there exists $1 \leq i < j \leq r$ such that

$$x_i \equiv x_j \pmod{p}$$

then $p \mid x_i - x_j$, and since $p \mid n$, we also have that

$$p \mid \gcd(x_i - x_j, n) > 1 \text{ as } p \text{ is prime}$$

Moreover since $x_i, x_j \in \mathbb{Z}/N\mathbb{Z}$ which implies $-N < x_i - x_j < N$ and we may also assume $x_i \neq x_j$ (as $n$ is large compared to $r$, hence we may assume $x_1, \ldots, x_r$ are distinct), therefore we have that

$$1 < \gcd(x_i - x_j, N) < N$$

(the condition $x_i \neq x_j$ removes the possibility that $\gcd(x_i - x_j, N) = N$). This $\gcd(x_i - x_j, N)$ thus provide a nontrivial factor of $N$ . Again keep in mind that we don't know what the value of $p$ is, but if we compute

$$\gcd(x_i - x_j, N) \text{ for every pair } 1 \leq i < j \leq r$$

we will find a nontrivial factor. To implement this, we will use the function

$$f : \mathbb{Z}/N\mathbb{Z} \to \mathbb{Z}/N\mathbb{Z}$$
$$x \pmod N \mapsto x^2 + 1 \pmod N$$

Now we let $x_0 \in_R \mathbb{Z}/N\mathbb{Z}$, then define the sequence $(x_n)_{n\geq 0}$ with $x_i = f(x_{i-1})$ for $i \geq 1$. Hence we get the following sequence of elements

$$\mathcal{A} : x_0, x_1, \ldots$$

Now consider taking modulo $p$ (hypothetically as this is still not possible because we don't know $p$) and get the following new list

$$\mathcal{B} : x_0 \mod p, x_1 \mod p, \ldots$$

We shall denote this new sequnce as $(x_n \pmod p)_{n\geq 0}$. This sequence contains at most $p$ different values, hence it will loop at certain point by Pollard's $\rho$ method. From the result in Pollard's $\rho$ method (since $p$ is prime), there will be a collision in expected number of steps of $O(\sqrt{p})$ where

$$x_i \pmod p \equiv x_j \pmod p$$

but note that

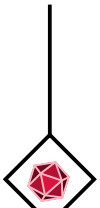
$$x_i \pmod p = x_j \pmod p \implies x_i = x_j$$

But this doesn't matter as long as we have the collision in $\mathcal{B}$, we get

$$x_i \pmod p = x_j \pmod p \implies x_i \equiv x_j \pmod p$$

then as described above, this will lead to a nontrivial factor of $N$. But note that since $p$ is unknown, we may get $\gcd(x_i - x_j, N) = 1$, and in this case the test fail, and we repeat by picking a new $x_0$. This can summarised as

```
Let y_0 = x_0
For i = 1, 2, ...
    a) x_i = f(x_{i-1})
    b) y_i = f(f(y_{i-1}))
    c) If gcd(x_i - y_i, N) ≠ 1, return this discovered factor
```



**How fast can $N$ be factorised?**
Note that finding a factor depends on the size of the prime factor of $N$. If after finding each factor we may repeat this step to factorise each factor until we get a prime factorisation.

1. Therefore if $N$ has $\alpha$ prime factors (including repetition), and if $p$ is the largest prime divisor $N$, we get that it requires $\alpha\sqrt{p}$ steps, i.e $\boxed{O(\sqrt{p})}$
2. Another bound can be obtained by observing that we always use largest prime less than $\sqrt{N}$, so this also imply another upper bound is $\alpha\sqrt{p} \leq \alpha N^{1/4}$, hence $\boxed{O(N^{1/4})}$

### 3.4.6 Pohlig-Hellman algorthm

The algorithm presented here is referenced from [4, §4.1.1] The Pohlig-Hellman algorithm efficiently reduces the computation of $n = \log_P Q$ to computation of discrete logarithm in the prime order subgroups of $\langle P\rangle$, **therefore it is very efficient in solving ECDLP with composite order points**. Suppose we have an elliptic curve $E(\mathbb{F}_q)$ and let $P, Q \in E(\mathbb{F}_q)$ where $Q \in \langle P\rangle$, and we wish to solve $Q = nP$. Since $E(\mathbb{F}_q)$ is a finite group, hence order of $P$ is finite, say the $\text{ord}(P) = N$. Suppose the prime factorisation of $N$ is

$$N = p_1^{e_1} p_2^{e_2} \cdots p_r^{e_r}$$

Since $Q \in \langle P\rangle$ then there exists some integer $\tilde{n} \in \mathbb{Z}/N\mathbb{Z}$ such that $Q = \tilde{n}P$. The Pohlig-Hellman algorithm shows us that in order to solve for $\tilde{n}$, we can do as follows, first we take modulo with respect to each prime power factors in $N$, i.e we get the following systems of equation

$$\begin{aligned}
\tilde{n} &\equiv n_1 \pmod{p_1^{e_1}} \\
\tilde{n} &\equiv n_2 \pmod{p_2^{e_2}} \\
&\vdots \\
\tilde{n} &\equiv n_r \pmod{p_r^{e_r}}
\end{aligned}$$

then Chinese Remainder Theoerem guarantees us a unique solution $[\tilde{n}]_N$ (refer to appendix A ). Now we show how the computation of each $n_i$ can be reduced to DLP. Now let's consider $n_i$ for some $1 \leq i \leq r$. Then consider writing $n_i \pmod{p_i^{e_i}}$ in $p_i$-adic expansion, and we obtain

$$n_i \equiv z_0 + z_1 p_i + \cdots + z_{e_i-1} p_i^{e_i} \pmod{p_i^{e_i}}$$

where each $z_i \in [0, p_i - 1]$. The digits $z_0, z_1, \ldots, z_{e_i-1}$ are computed one at time as follows. We first compute $P_0 = (N/p_i)P$ and $Q_0 = (N/p_i)Q$. Since the order of $P_0$ is $p_i$ we have

$$Q_0 = \frac{N}{p_i} Q = \frac{N}{p_i} \tilde{n} P = \tilde{n} \left( \frac{N}{p_i} P \right) = \tilde{n} P_0$$

since $\tilde{n} \equiv n_i \pmod{p_i^{e_i}}$, this implies $\tilde{n} \equiv n_i \pmod{p_i}$, and since $P_0$ has order $p_i$ hence we have

$$Q_0 = \tilde{n} P_0 = [\tilde{n}]_{p_i} P_0 = z_0 P_0$$

therfore we get $z_0 = \log_{P_0} Q_0$ which can be solved by Pollard's $\rho$ method as $P_0$ has order $p_i$ prime. Next, we compute $Q_1 = (N/p_i^2)(Q - z_0 P)$. We have

$$\begin{aligned} Q_1 &= \frac{N}{p_i^2}(Q - z_0 P) = \frac{N}{p_i^2}(\tilde{n} - z_0) P = (\tilde{n} - z_0) \left( \frac{N}{p_i^2} P \right) \\ &= (z_0 + z_1 p_i - z_0) \left( \frac{N}{p_i^2} P \right) = z_1 \left( \frac{N}{p_i} P \right) = z_1 P_0 \end{aligned}$$

Now note that $(N/p_i^2)P$ has order $p_i^2$, therefore we have

$$\begin{aligned} Q_1 &= (\tilde{n} - z_0) \left( \frac{N}{p_i^2} P \right) = [\tilde{n} - z_0]_{p_i^2} \left( \frac{N}{p_i^2} P \right) = (z_0 + z_1 p_i - z_0) \left( \frac{N}{p_i^2} P \right) \\ &= z_1 p_i \left( \frac{N}{p_i^2} P \right) = z_1 \left( \frac{N}{p_i} P \right) = z_1 P_0 \end{aligned}$$

hence again we have $z_1 = \log_{P_0} Q_1$ (which can be solve by Pollard's $\rho$ method yet again because $P_0$ has order $p_i$ prime). In general, if the digits $z_0, z_1, \ldots, z_{t-1}$ have been computed, then $z_t = \log_{P_0} Q_t$ where

$$Q_t = \frac{N}{p_i^{t+1}} \left( Q - z_0 P - z_1 p_i P - \cdots - z_{t-1} p_i^{t-1} P \right)$$

can be computed by using Pollard's $\rho$ method where $0 \leq t \leq e_i - 1$.

Now after computing $z_0, z_1, \ldots, z_{e_i-1}$ we get $n_i$, repeating these for $n_1, \ldots, n_r$, then by Chinese Remainder Theorem, we compute $\tilde{n}$, thus solving ECDLP.

**How fast can we find a solution using Pohlig-Hellman algorithm?**
Suppose that $P \in E(\mathbb{F}_p)$ has order $N$, if the prime factorisation is given as

$$N = \prod_{i=1}^{r} p_i^{e_i}$$

Then recall that we first write out the systems of $r$ congruences. Now let's focus on the $1 \leq i \leq r$ congruences as before, notice that for each coefficients, we compute

$$z_t = \log_{P_0} Q_t$$

and this is just an ECDLP where $P_0$ has a prime order $p_i$, hence this can be solved by using Pollard's $\rho$ method and in $O(\sqrt{p_i})$ steps for each $0 \leq t \leq e_i - 1$, hence it takes about $O(e_i \sqrt{p_i})$ **steps** for solving one congruences in the system. But now notice that we also have to compute $Q_t$ first before solving $Q_t = z_t P_0$ DLP. But notice that $z_0, z_1, \ldots, z_{t-1}$ are computed in previous iterations, so for each $t$ we just need to compute

$$Q_t = \frac{N}{p_i^{t+1}} \tilde{X}$$

where $\tilde{X} \in E(\mathbb{F}_p)$. By Double-and-Add algorithm, this takes at most $2 \log_2 \frac{N}{p_i^{t+1}}$, hence $O(\log_2(N/p_i^{t+1}))$ which is obviously bounded above by $O(\log_2 N)$ steps for each $0 \leq t \leq e_i - 1$. Therefore, solving for

each linear congruences in the systems of congruences requires

$$O(e_i(\log_2 N + \sqrt{p_i}))$$

but now since we have $r$ linear congruences in the systems, therefore it takes

$$O\left(\sum_{i=1}^{r} e_i(\log_2 N + \sqrt{p_i})\right)$$

i.e

$$\boxed{O\left(\sum_{i=1}^{r} e_i(\log N + \sqrt{p_i})\right)}$$

[5, §3.1][6, §3.6.4; Fact 3.65]

### 3.4.7 General Attack of ECDLP

The best general-attack of ECDLP currently is a combination of Pohlig-Hellman algorithm and Pollard $\rho$ algorithm (not to be confused with Pollard $\rho$ method). This is said to have a fully-exponential running time of

$$\boxed{O(\sqrt{p})}$$

where $p$ is the largest prime divisr of $n$. Therefore to resist this attack, one should use elliptic curve with point order $N$ where all the prime divisor of $N$ is sufficiently large. [4, §4.1]

**Unsolved and potentially million dollar question(?)**

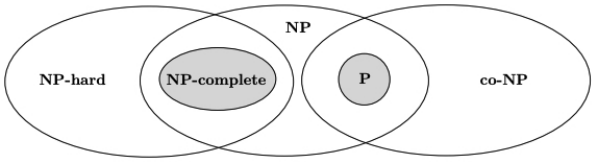



Figure 5: The complexity classes venn diagram [7, §2.3.1; p.35]

- Instances are the input to the decsion problem, a YES-instance is an input that produces YES as the solution and a NO-instance is the opposite.
- $P$ (polynomial time) is complexity class used to classify decision problems that are solvable deterministically in polynomial time.
- $NP$ (non-deterministic polynomial time) is the complexity class used to classify decision problems that can't be solved by a deterministic Turing Machine in polynomial time but can be solved by a nondeterministic Turing Machine in polynomial time. Furthermore, a YES-instance of a problem in $NP$ is verifiable in polynomial time by a deterministic Turing Machine. **Every problem in this class is DECIDABLE**.
- $NP-$complete or $NPC$ (non-deterministic polynomial-time complete): A problem $p$ is in $NP$-complete if every decision problems in $NP$ can be reduced to $p$ in a polynomial time. Thus, if we can find a deterministic algorithm that solves one $NP$-complete problem, then we can solve all problems in $NP$ is polynomial time, this would imply $P = NP$. I.e $NPC$ **are the hardest problems in** $NP$ **as solving one implies solving all** $NP$
- $NP$-hard (non-deterministic polynomial-time hard) Class of problems which are at least as hard as the hardest problems in NP. Problems that are NP-hard do not have to be elements of NP; indeed, they may not even be decidable. Note that there is a mistake in the diagram, we should interpret it as $NP-\text{hard} \cap NP \cap \overline{NP-\text{complete}} = \emptyset$
- co-$NP$ is the class of decision problems where a NO-instance is verifiable in polynomial time.

ECDLP is in NP as there is currently no deterministic polyomial time algorithm that can solve it. It is believed that ECDLP is not NP-hard either as the decision version of ECDLP is known to be in NP ∩ co-NP .

**Definition 3.6.** ***(Decision ECDLP)*** *Given $(E(\mathbb{F}_p), N, d, Q, P)$ where $E$ is an elliptic curve, $Q \in E(\mathbb{F}_p)$ and $P \in E(\mathbb{F}_p)$ with order $N$ and $d \leq N$ is an integer. The the decision problem is:*

$$\textit{Is there an integer } k \leq d : Q = kP \textit{?}$$

Thus, if one can show that there does not exists a deterministic polynomial time algorithm that solves ECDLP, then $P \neq NP$[4, §4.1]. It is also suspected that ECDLP is not in NP-complete, because if there exists a problem that is in NP ∩ co-NP ∩ NP-C, then NP=co-NP (which would be an unexpected result) this is because if a problem is in NP ∩ co-NP then it's YES and NO-instances can be verifiable in polynomial time. Further if this problem is in NP-C, then that means all NP can be reduced to it, meaning that all NP problems can be verifiable in polynomial time, i.e NP = co-NP

One thing to note is that although there does not exists a deterministic polynomial time algorithm that solves ECDLP on deterministic Turing machine, but there does exists deterministic algorithm in polynomial that solves ECDLP in a **quantum Turing machine.** This makes ECDLP to be categorised into the complexity class BQP (bounded-error quantum polynomial time). The algorithm will be a modified version of Shor's algorithm. [8]

**Remarks on problem reduction:**
We say that a problem $A$ can be reduced to problem $B$ if

1. $A$ is at most as hard as $B$, i.e $B$ is at least as hard as $A$
2. If we can solve $B$ then we can solve $A$ (i.e if $B$ is decidable then so is $A$)
3. If $A$ is undecidable then so is $B$, otherwise if $B$ is decidable, then since $A$ can be reduce to $B$, this would imply $A$ is decidable hence a contradiction.

# Appendices

## A Chinese Remainder Theorem

**Theorem A.1.** ***(Chinese Remainder Theorem)*** *Let $m_1, m_2, \ldots, m_k$ be a collection of pairwise relatively prime integers, i.e*

$$\gcd(m_i, m_j) = 1 \text{ for all } i \neq j$$

*Let $a_1, \ldots, a_k$ be arbitrary integers. Then the systems of simultaneous congruences*

$$\begin{aligned} x &\equiv a_1 \pmod{m_1} \\ x &\equiv a_2 \pmod{m_2} \\ &\vdots \\ x &\equiv a_k \pmod{m_k} \end{aligned}$$

*has a unique solution modulo $m_1 m_2 \cdots m_k$. I.e if $x = c$ and $x = c'$ are solutions to the systems of equations, then*

$$c \equiv c' \pmod{m_1 m_2 \cdots m_k}$$

*Proof.* This will be construction proof. Clearly $a_i$ is the solution to the congruence $x \equiv a_i \pmod{m_i}$. Now suppose we define

$$M = \prod_{i=1}^{k} m_i$$

and define

$$M_i' = \frac{M}{m_i}$$

note that $\gcd(M_i', m_i) = 1$ as $\gcd(m_i, m_j) = 1$ when $i \neq j$. Hence we have that there exists $M_i'^{-1} \in \mathbb{Z}/m_i\mathbb{Z}$. Therefore we have that

$$a_i M_i' M_i'^{-1} \pmod{m_d} = \begin{cases} a_i & \text{if } d = i \\ 0 & \text{if } d \neq i \end{cases}$$

the second case is because $m_d \mid M_i'$ for all $1 \leq d \leq k$ and $d \neq i$. Therefore if we define

$$x = \sum_{i=1}^{k} a_i M_i' M_i'^{-1}$$

then $x$ clearly satisfies all the congruences relation.
Now suppose $x = x_1$ and $x = x_2$ are solution to the systems of congruences, i.e

$$\begin{aligned} x_1 &\equiv a_1 \pmod{m_1} \\ x_1 &\equiv a_2 \pmod{m_2} \\ &\vdots \\ x_1 &\equiv a_k \pmod{x_k} \end{aligned} \qquad \begin{aligned} x_2 &\equiv a_1 \pmod{m_1} \\ x_2 &\equiv a_2 \pmod{m_2} \\ &\vdots \\ x_2 &\equiv a_k \pmod{m_k} \end{aligned}$$

then clearly subtracting each of the linear congruences from each other's corresponding modulo we get

$$\begin{aligned} x_1 - x_2 &\equiv 0 \pmod{m_1} \\ x_1 - x_2 &\equiv 0 \pmod{m_2} \\ &\vdots \\ x_1 - x_2 &\equiv 0 \pmod{m_k} \end{aligned}$$

which implies

$$x_1 \equiv x_2 \pmod{m_i} \quad \text{for all } 1 \leq i \leq k$$

which implies also that

$$x_1 \equiv x_2 \pmod{M}$$

as $(m_i, m_j)$ are relatively coprime for all $i \neq j$. Thus this finishes the proof. □

# References

[1] Jeffrey Hoffstein, Jill Pipher, Joseph H Silverman, and Joseph H Silverman. *An introduction to mathematical cryptography*, volume 1. Springer, 2008.

[2] Brilliant.org. Cubic discriminant, 2023-06-02. URL `https://brilliant.org/wiki/cubic-discriminant/`.

[3] Wikipedia contributors. Vieta's formulas — Wikipedia, the free encyclopedia, 2023. URL `https://en.wikipedia.org/w/index.php?title=Vieta%27s_formulas&oldid=1154999372`. [Online; accessed 12-June-2023].

[4] Darrel Hankerson, Alfred J Menezes, and Scott Vanstone. *Guide to elliptic curve cryptography*. Springer Science & Business Media, 2006.

[5] Martin Lysoe Sommerseth and Haakon Hoeiland. Pohlig-hellman applied in elliptic curve cryptography, 2015.

[6] Alfred J Menezes, Paul C Van Oorschot, and Scott A Vanstone. *Handbook of applied cryptography*. CRC press, 2018.

[7] Anton Pierre De Villiers. *Edge criticality in secure graph domination*. PhD thesis, Stellenbosch: Stellenbosch University, 2014.

[8] Wikipedia contributors. Shor's algorithm — Wikipedia, the free encyclopedia, 2023. URL `https://en.wikipedia.org/w/index.php?title=Shor%27s_algorithm&oldid=1158928892`. [Online; accessed 12-June-2023].