0.1 Basics of Security Analysis

In many cases, technical papers include the analysis on the key question: "How secure is this cryptographic algorithm?" or rather "Why this cryptographic algorithm is secure?". In this section, we will shortly describe the notation and typical construction for justifying the security of cryptographic algorithms.

Typically, the cryptographic security is defined in a form of a game between the adversary (who we call $\mathcal{A}$) and the challenger (who we call $\mathcal{C}h$). The adversary is trying to break the security of the cryptographic algorithm using arbitrary (but still efficient) protocol, while the challenger is following a simple, fixed protocol. The game is played in a form of a challenge, where the adversary is given some information and is asked to perform some task. The security of the cryptographic algorithm is defined based on the probability of the adversary to win the game.

0.1.1 Cipher Semantic Security

Let us get into specifics. Suppose that we want to specify that the encryption scheme is secure. Recall that cipher $\mathcal{E} = (E, D)$ over the space $(\mathcal{K}, \mathcal{M}, \mathcal{C})$ (here, $\mathcal{K}$ is the space containing all possible keys, $\mathcal{M}$ — all possible messages and $\mathcal{C}$ — all possible ciphers) consists of two efficiently computable methods:

- $E : \mathcal{K} \times \mathcal{M} \to \mathcal{C}$ — encryption method, that based on the provided message $m \in \mathcal{M}$ and key $k \in \mathcal{K}$ outputs the cipher $c = E(k, m) \in \mathcal{C}$.
- $D : \mathcal{K} \times \mathcal{C} \to \mathcal{M}$ — decryption method, that based on the provided cipher $c \in \mathcal{C}$ and key $k \in \mathcal{K}$ outputs the message $m = D(k, c) \in \mathcal{M}$.

Of course, we require the **correctness**:

$$(\forall k \in \mathcal{K})\,(\forall m \in \mathcal{M}) : \{D(k, E(k, m)) = m\}$$

Now let us play the following game between adversary $\mathcal{A}$ and challenger $\mathcal{C}h$:

1. $\mathcal{A}$ picks any two messages $m_0, m_1 \in \mathcal{M}$ on his choice.
2. $\mathcal{C}h$ picks a random key $k \xleftarrow{R} \mathcal{K}$ and random bit $b \xleftarrow{R} \{0, 1\}$ and sends the cipher $c = E(k, m_b)$ to $\mathcal{A}$.
3. $\mathcal{A}$ is trying to guess the bit b by using the cipher c.
4. $\mathcal{A}$ outputs the guess $\hat{b}$.

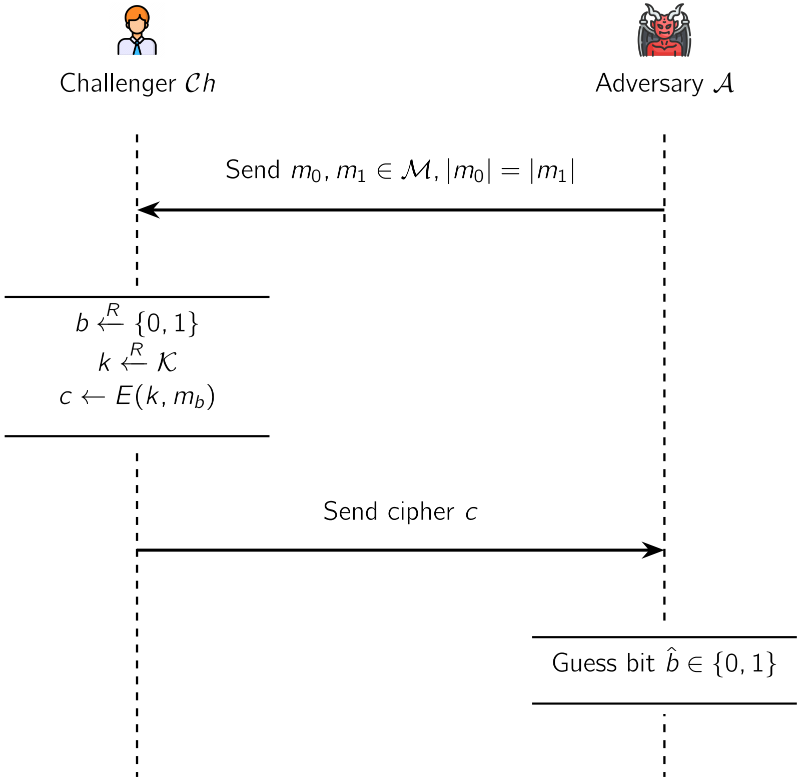
Challenger $\mathcal{C}h$
Adversary $\mathcal{A}$
Send $m_0, m_1 \in \mathcal{M}, |m_0| = |m_1|$
$b \xleftarrow{R} \{0,1\}$
$k \xleftarrow{R} \mathcal{K}$
$c \leftarrow E(k, m_b)$
Send cipher c
Guess bit $\hat{b} \in \{0,1\}$

Figure 0.1: The game between the adversary $\mathcal{A}$ and the challenger $\mathcal{C}h$ for defining the semantic security.

Now, what should happen if our encryption scheme is secure? The adversary should not be able to guess the bit b with a probability significantly higher than $1/2$ (a random guess). Formally, define the **advantage** of the adversary $\mathcal{A}$ as:

$$\mathsf{SSAdv}[\mathcal{E}, \mathcal{A}] := \left| \Pr\left[\hat{b} = b\right] - \frac{1}{2} \right|$$

We say that the encryption scheme is **semantically secure**[1] if for any efficient adversary $\mathcal{A}$ the advantage $\mathsf{SSAdv}[\mathcal{A}]$ is negligible. In other words, the adversary cannot guess the bit b with a probability significantly higher than $1/2$.

Now, what negligible means? Let us give the formal definition!

[1] This version of definition is called a **bit-guessing** version.

Definition 0.1. A function $f : \mathbb{N} \to \mathbb{R}$ is called **negligible** if for all $c \in \mathbb{R}_{>0}$ there exists $n_c \in \mathbb{N}$ such that for any $n \geq n_c$ we have $|f(n)| < 1/n^c$.

The alternative definition, which is problably easier to interpret, is the following.

Theorem 0.2. A function $f : \mathbb{N} \to \mathbb{R}$ is **negligible** if and only if for any $c \in \mathbb{R}_{>0}$, we have

$$\lim_{n \to \infty} f(n) n^c = 0$$

Example. The function $f(n) = 2^{-n}$ is negligible since for any $c \in \mathbb{R}_{>0}$ we have

$$\lim_{n \to \infty} 2^{-n} n^c = 0$$

The function $g(n) = \frac{1}{n!}$ is also negligible for similar reasons.

Example. The function $h(n) = \frac{1}{n}$ is not negligible since for $c = 1$ we have

$$\lim_{n \to \infty} \frac{1}{n} \times n = 1 \neq 0$$

Well, that is weird. For some reason we are considering a function the depends on some natural number n, but what is this number?

Typically, when defining the security of the cryptographic algorithm, we are considering the security parameter λ (e.g., the length of the key). The function is negligible if the probability of the adversary to break the security of the cryptographic algorithm is decreasing with the increasing of the security parameter λ. Moreover, we require that the probability of the adversary to break the security of the cryptographic algorithm is decreasing faster than any polynomial function of the security parameter λ.

So all in all, we can define the semantic security as follows.

Definition 0.3. The encryption scheme $\mathcal{E}$ with a security paramter $\lambda \in \mathbb{N}$ is **semantically secure** if for any efficient adversary $\mathcal{A}$ we have:

$$\left| \Pr\left[b = \hat{b} \;\middle|\; \begin{array}{c} m_0, m_1 \leftarrow \mathcal{A},\ k \xleftarrow{R} \mathcal{K},\ b \xleftarrow{R} \{0,1\} \\ c \leftarrow E(k, m_b) \\ \hat{b} \leftarrow \mathcal{A}(c) \end{array} \right] - \frac{1}{2} \right| < \mathsf{negl}(\lambda)$$

Do not be afraid of such complex notation, it is quite simple. Notation $\Pr[A \mid B]$ means "the probability of A, given that B occurred". So our inner probability is read as "the probability that the guessed bit $\hat{b}$ equals b given the setup on the right". Then, on the right we define the setup: first we generate two messages $m_0, m_1 \in \mathcal{M}$, then we choose a random bit b and a key k, cipher the message m_b, send it to the adversary and the adversary, based on provided cipher, gives $\hat{b}$ as an output. We then claim that the probability of the adversary to guess the bit b is close to $1/2$.

Let us see some more examples of how to define the security of certain crypographic objects.

0.1.2 Message recovery attacks

Essentially, message recovery attacks are types of attacks that, from given ciphertext, recover the message with significantly better probability than random guessing, which is obviously $1/|\mathcal{M}|$. Of course, any reasonable notion of security should rule out such an attack, and indeed, semantic security does.

Although this might seem intuitively obvious, we provide a formal proof to clarify the reasoning. One of the reasons for doing this is to thoroughly demonstrate the concept of a *security reduction*, which is the primary method used to assess the security of systems.

Let us play the following attack game message recovery between adversary $\mathcal{A}$ and challenger $\mathcal{C}h$. For a given cipher $\mathcal{E} = (E, D)$, defined over $\mathcal{K}, \mathcal{M}, \mathcal{C}$, and for a given adversary $\mathcal{A}$, the attack game proceeds as follows:

- The challenger computes $m \overset{R}{\leftarrow} \mathcal{M}$, $k \overset{R}{\leftarrow} \mathcal{K}$, $c \overset{R}{\leftarrow} E(k, m)$ and sends c to the adversary.
- the adversary output a message $\hat{m} \in \mathcal{M}$.

We say that $\mathcal{A}$ wins the game in this case, and we define $\mathcal{A}$'s message recovery advantage with respect to $\mathcal{E}$ as:

$$\mathrm{MRadv}[\mathcal{E}, \mathcal{A}] := \left| \Pr[\hat{m} = m] - \frac{1}{|\mathcal{M}|} \right|.$$

Definition 0.4 (Security against message recovery)**.** A cipher $\mathcal{E}$ is secure against message recovery if for all efficient adversaries $\mathcal{A}$, the value $\mathrm{MRadv}[\mathcal{E}, \mathcal{A}]$ is negligible.

Theorem 0.5. Let $\mathcal{E} = (E, D)$ be a cipher defined over $(\mathcal{K}, \mathcal{M}, \mathcal{C})$. If $\mathcal{E}$ is semantically secure then $\mathcal{E}$ is secure against message recovery.

▶ Let us prove by assuming the opposite statement. In other words, if $\mathcal{E}$ is not secure against message recovery, then $\mathcal{E}$ is not semantically secure.

Assume that $\mathcal{E}$ is not secure against message recovery. We have an efficient adversary $\mathcal{A}$ who has a significant advantage in the message recovery game and an efficient adversary $\mathcal{B}$ who is trying to compromise $\mathcal{E}$. The proof idea is to use an efficient adversary $\mathcal{A}$ as a "black box" (or oracle machine). We construct $\mathcal{B}$ as follows:

1. Adversary $\mathcal{B}$ generates $m_0, m_1 \in \mathcal{M}$ and sends them to the semantic security challenger.
2. The semantic security challenger returns ciphertext c to adversary $\mathcal{B}$. Since $\mathcal{B}$ can use the message recovery oracle, it sends c to $\mathcal{A}$.
3. $\mathcal{A}$ returns $\hat{m}$. If $\hat{m} = m_0$, then $b = 0$ otherwise, $b = 1$.
4. Sends b to the semantic security challenger.

Intuitively, this implies that SSadv$[\mathcal{E}, \mathcal{B}]$ is exactly equal to MRadv$[\mathcal{E}, \mathcal{A}]$. ◀

This means that if an adversary is capable of successfully winning the message recovery game, then they can also effectively win the semantic security game. In essence, security against message recovery is a stronger property than semantic security.

Our motivation for demonstrating proof is to illustrate in detail the notion of a security reduction, which is the main technique used to reason about the security of systems.

Security reduction is typically applied to certain computational problems. To be more precise, we will now explain how security reduction operates in general.

Definition 0.6 (Security Reduction)**.**

1. Let $\mathcal{A}$ be an efficient adversary who is trying to compromise the cryptosystem $\mathcal{E}$.
2. We construct an efficient algorithm $\mathcal{A}'$, which we call the **reduction**, that will solve the computationally hard problem $\mathcal{X}$ while using an oracle access to $\mathcal{A}$. For any input x of the problem $\mathcal{X}$, the algorithm $\mathcal{A}'$ generates the input of the cryptosystem $\mathcal{E}$ for $\mathcal{A}$. Once it wins the corresponding experiment, $\mathcal{A}'$ solves $\mathcal{X}$ for input x with a certain non-negligible probability $\mu(n)$. Typically, such probability turns out to be bounded below by $1/n^c, c \in \mathbb{N}$.
3. Thus, the algorithm $\mathcal{A}'$ solves $\mathcal{X}$ with the non-negligible probability $\varepsilon(n)\mu(n)$ (or typically $\varepsilon(n)/n^c$). This quantity is non-negligible if $\varepsilon(n)$ is assumed to be non-negligible.
4. Since the problem $\mathcal{X}$ is computationally hard, we get a contradiction.

It follows that no attacker $\mathcal{A}$ can effectively crack $\mathcal{E}$, i.e., this cryptosystem is computationally strong.

0.1.3 Discrete Logarithm Assumption

Now, let us define the fundamental assumption used in cryptography formally: the **Discrete Logarithm Assumption** (DL).

Definition 0.7. Assume that $\mathbb{G}$ is a cyclic group of prime order r generated by $g \in \mathbb{G}$. Define the following game:

1. Both challenger $\mathcal{C}h$ and adversary $\mathcal{A}$ take a description $\mathbb{G}$ as an input: order r and generator $g \in \mathbb{G}$.
2. $\mathcal{C}h$ computes $\alpha \xleftarrow{R} \mathbb{Z}_r$, $u \leftarrow g^{\alpha}$ and sends $u \in \mathbb{G}$ to $\mathcal{A}$.
3. The adversary $\mathcal{A}$ outputs $\hat{\alpha} \in \mathbb{Z}_r$.

We define $\mathcal{A}$'s **advantage in solving the discrete logarithm problem in** $\mathbb{G}$, denoted as $\mathsf{DLadv}[\mathcal{A}, \mathbb{G}]$, as the probability that $\hat{\alpha} = \alpha$.

Definition 0.8. The **Discrete Logarithm Assumption** holds in the group $\mathbb{G}$ if for any efficient adversary $\mathcal{A}$ the advantage $\mathsf{DLadv}[\mathcal{A}, \mathbb{G}]$ is negligible.

Informally, this assumption means that given u, it is very hard to find α such that $u = g^{\alpha}$. But now we can write down this formally!

0.1.4 Computational Diffie-Hellman

Another fundamental problem in cryptography is the **Computational Diffie-Hellman** (CDH) problem. It states that given g^{α}, g^{β} it is hard to find $g^{\alpha\beta}$. This property is frequently used in the construction of cryptographic protocols such as the Diffie-Hellman key exchange.

Let us define this problem formally.

Definition 0.9. Let $\mathbb{G}$ be a cyclic group of prime order r generated by $g \in \mathbb{G}$. Define the following game:

1. Both challenger $\mathcal{C}h$ and adversary $\mathcal{A}$ take a description $\mathbb{G}$ as an input: order r and generator $g \in \mathbb{G}$.
2. $\mathcal{C}h$ computes $\alpha, \beta \xleftarrow{R} \mathbb{Z}_r$, $u \leftarrow g^{\alpha}$, $v \leftarrow g^{\beta}$, $w \leftarrow g^{\alpha\beta}$ and sends $u, v \in \mathbb{G}$ to $\mathcal{A}$.
3. The adversary $\mathcal{A}$ outputs $\hat{w} \in \mathbb{G}$.

We define $\mathcal{A}$'s **advantage in solving the computational Diffie-Hellman**

problem in $\mathbb{G}$, denoted as $\mathsf{CDHadv}[\mathcal{A}, \mathbb{G}]$, as the probability that $\hat{w} = w$.

Definition 0.10. The **Computational Diffie-Hellman Assumption** holds in the group $\mathbb{G}$ if for any efficient adversary $\mathcal{A}$ the advantage $\mathsf{CDHadv}[\mathcal{A}, \mathbb{G}]$ is negligible.

0.1.5 Decisional Diffie-Hellman

Now, we loosen the requirements a bit. The **Decisional Diffie-Hellman** (DDH) problem states that given $g^{\alpha}, g^{\beta}, g^{\alpha\beta}$ it is "hard" to distinguish $g^{\alpha\beta}$ from a random element in $\mathbb{G}$. Formally, we define this problem as follows.

Definition 0.11. Let $\mathbb{G}$ be a cyclic group of prime order r generated by $g \in \mathbb{G}$. Define the following game:

1. Both challenger $\mathcal{C}h$ and adversary $\mathcal{A}$ take a description $\mathbb{G}$ as an input: order r and generator $g \in \mathbb{G}$.
2. $\mathcal{C}h$ computes $\alpha, \beta, \gamma \xleftarrow{R} \mathbb{Z}_r$, $u \leftarrow g^{\alpha}$, $v \leftarrow g^{\beta}$, $w_0 \leftarrow g^{\alpha\beta}$, $w_1 \leftarrow g^{\gamma}$. Then, $\mathcal{C}h$ flips a coin $b \xleftarrow{R} \{0,1\}$ and sends u, v, w_b to $\mathcal{A}$.
3. The adversary $\mathcal{A}$ outputs the predicted bit $\hat{b} \in \{0,1\}$.

We define $\mathcal{A}$'s **advantage in solving the Decisional Diffie-Hellman problem in** $\mathbb{G}$, denoted as $\mathsf{DDHadv}[\mathcal{A}, \mathbb{G}]$, as

$$\mathsf{DDHadv}[\mathcal{A}, \mathbb{G}] := \left| \Pr\left[b = \hat{b}\right] - \frac{1}{2} \right|$$

Now, let us break this assumption for some quite generic group! Consider the following example.

Theorem 0.12. Suppose that $\mathbb{G}$ is a cyclic group of an even order. Then, the Decision Diffie-Hellman Assumption does not hold in $\mathbb{G}$. In fact, there is an efficient adversary $\mathcal{A}$ that can distinguish $g^{\alpha\beta}$ from a random element in $\mathbb{G}$ with an advantage $1/4$.

Proof. If $|\mathbb{G}| = 2n$ for $n \in \mathbb{N}$, it means that we can split the group into two subgroups of order n, say, $\mathbb{G}_1$ and $\mathbb{G}_2$. The first subgroup consists of elements in a form g^{2k}, while the second subgroup consists of elements in a form g^{2k+1}.

Now, if we could efficiently determine, based on group element $g \in \mathbb{G}$, whether $g \in \mathbb{G}_1$ or $g \in \mathbb{G}_2$, we essentially could solve the problem. Fortunately, there is such a method! Consider the following lemma.

Lemma 0.13. Suppose $u = g^{\alpha}$. Then, α is even if and only if $u^n = 1$.

Proof. If α is even, then $\alpha = 2\alpha'$ and thus

$$u^n = (g^{2\alpha'})^n = g^{2n\alpha'} = (g^{2n})^{\alpha'} = 1^{\alpha'} = 1$$

Conversely, if $u^n = 1$ then $u^{\alpha n} = 1$, meaning that $2n \mid \alpha n$, implying that α is even. Lemma is proven.

Now, we can construct our adversary $\mathcal{A}$ as follows. Suppose $\mathcal{A}$ is given (u, v, w). Then,

1. Based on u, get the parity of α, say $p_{\alpha} \in \{\text{even}, \text{odd}\}$.
2. Based on v, get the parity of β, say $p_{\beta} \in \{\text{even}, \text{odd}\}$.
3. Based on w, get the parity of γ, say $p_{\gamma} \in \{\text{even}, \text{odd}\}$.
4. Calculate $p'_{\gamma} \in \{\text{even}, \text{odd}\}$ — parity of $\alpha\beta$.
5. Return $\hat{b} = 0$ if $p'_{\gamma} = p_{\gamma}$, and $\hat{b} = 1$, otherwise.

Suppose γ is indeed $\alpha \times \beta$. Then, condition $p'_{\gamma} = p_{\gamma}$ will always hold. If γ is a random element, then the probability that $p'_{\gamma} = p_{\gamma}$ is $1/2$. Therefore, the probability that $\mathcal{A}$ will guess the bit b correctly is $3/4$, and the advantage is $1/4$ therefore. ■

Why is this necessary? Typically, it is impossible to prove the predicate "for every efficient adversary $\mathcal{A}$ this probability is negligible" and therefore we need to make assumptions, such as the Discrete Logarithm Assumption or the Computational Diffie-Hellman Assumption. In turn, proving the statement "if X is secure then Y is also secure" is manageable and does not require solving any fundamental problems. So, for example, knowing that the probability of the adversary to break the Diffie-Hellman assumption is negligible, we can prove that the Diffie-Hellman key exchange is secure.

0.2 Basic Number Theory

As mentioned earlier, the cryptography must work with integer-based formats. One of the earliest and most fundamental branches of mathematics is number theory, which deals with the properties of the integer set $\mathbb{Z}$. Although, as we will see later, we will primarily need the notion of *prime/finite field* further, the basic concepts of number theory will still be used extensively nonetheless.

0.2.1 Introduction to number theory

We start with the most basic definition of number theory — *divisibility*.

Definition 0.14. An integer a is divisible by a non-zero integer b, denoted $b \mid a$ (or *b is a divisor of a*), if and only if there exists an integer $k \in \mathbb{Z}$ such that $a = k \cdot b$.

Let us consider some basic properties of this relation.

Lemma 0.15 (Divisibility properties)**.** For all $a, b, c \in \mathbb{Z}$:

1. $1 \mid a$ (any number is divisible by 1)
2. If $a \neq 0$, then $a \mid a$ (any non-zero integer divides itself).
3. If $a \neq 0$, then $a \mid 0$ (any non-zero integer divides 0).
4. If $b \mid a$ and $c \mid b$, then $c \mid a$ (formally called *transitivity*).
5. $b \mid a \Leftrightarrow b \cdot c \mid a \cdot c$ for any $c \neq 0$.
6. If $c \mid a$ and $c \mid b$, then $c \mid (\alpha \cdot a \pm \beta \cdot b)$, for any $\alpha, \beta \in \mathbb{Z}$

But what happens if a number a is not divisible by the given integer b, meaning there is no integer k that satisfies the condition $a = b \cdot k$? In such cases, a new theorem comes into play.

Theorem 0.16 (Division theorem)**.**

$$\forall a, b \in \mathbb{Z}\ \exists! q, r \in \mathbb{Z} \text{ such that } a = b \cdot q + r \text{ with } 0 \leq r < |b|$$

To prove this theorem, all we need to prove is the existence and uniqueness of such a decomposition. To not make the book too long, we will not prove this theorem here, but you can find the proof in any number theory textbook.

This theorem allows us to define two new operations. Suppose a, b, q, r are given as in the Theorem 0.16. Then,

- **Floor Operation** ($\lfloor a/b \rfloor$ or $a \text{ div } b$) is defined as q. This operation is a standard `div` opeartion commonly used in programming languages.
- **Mod Operation** ($a \bmod b$) is defined as r. This operation is a standard `mod` operation commonly used in programming languages.

In division operations, it is common to check for any shared factors between two numbers. This is where the concept of the **greatest common divisor** (or **gcd** for short) comes into play.

Definition 0.17 (GCD)**.** For any $a, b \in \mathbb{Z}$, the **greatest common divisor** $\gcd(a, b)$ is defined as an integer $d \in \mathbb{N}$ such that:

1. $d \mid a$ and $d \mid b$.

2. d is a maximal integer that satisfies the first condition.

One might right the above definition more concisely:

$$\gcd(a, b) = \max\{d \in \mathbb{N} : d \mid a \text{ and } d \mid b\}.$$

Definition 0.18. Two numbers a and b are **coprime** if and only if $\gcd(a, b) = 1$.

As with any previous concept, let us check some basic properties of the GCD operation.

Lemma 0.19 (Greatest common divisor properties)**.**

1. $\gcd(a, b) = b \Leftrightarrow b \mid a$
2. If $a \neq 0$, then $\gcd(a, 0) = a$
3. If there exists $\delta \in \mathbb{Z}$ such that $\delta \mid a$ and $\delta \mid b$, then $\delta \mid \gcd(a, b)$
4. If $c > 0$, then $\gcd(ac, bc) = c \cdot \gcd(a, b)$
5. Suppose $d = \gcd(a, b)$. Then, $\gcd(a/d, b/d) = 1$

Lemma 0.20. For any $a, b \in \mathbb{Z}$, we have $\gcd(a, b) = \gcd(b, a - b)$.

The aforementioned lemma will be further extensively used for the Euclidean algorithm.

Corollary 0.21. As a corollary, for any $a, b \in \mathbb{Z}$ we also have $\gcd(a, b) = \gcd(b, a \mod b)$.

All these properties are interesting and useful from the theory standpoint, but you may wonder how to practically find $\gcd(a, b)$ (say, if you were to implement this function in the programming language). *Euclidean algorithm* is an efficient method for computing the greatest common divisor. The main idea of Euclidean algorithm is to recursively apply the Corollary 0.21. The concrete implementation in Python is specified below.

```python
def gcd(a: int, b: int) -> int:
    return gcd(b, a % b) if b != 0 else a
```

Notice that the algorithm can be easily implemented in a single line.

While the greatest common divisor (gcd) focuses on finding the largest shared factor between two numbers, the least common multiple (lcm) deals with finding the smallest multiple that both numbers have in common. The LCM is particularly useful when we need to synchronize cycles or work with fractions.

Definition 0.22 (LCM)**.** For any $a, b \in \mathbb{Z}$, the **least common multiple** $\mathrm{lcm}(a, b)$ is defined as an integer $m \in \mathbb{N}$ such that:

1. $a \mid m$ and $b \mid m$
2. m is a minimal integer that satisfies the first condition

One might right the above definition more concisely:

$$\mathrm{lcm}(a, b) = \min\{m \in \mathbb{N} : a \mid m \text{ and } b \mid m\}.$$

Lemma 0.23 (Least Common Multiple Properties)**.**

1. We assume that $\mathrm{lcm}(a, 0)$ is undefined.
2. $\mathrm{lcm}(a, b) = a \Leftrightarrow b \mid a$.
3. If a and b are coprime, then $\mathrm{lcm}(a, b) = a \cdot b$.
4. Any common divisor δ of a and b satisfies $\delta \mid \mathrm{lcm}(a, b)$.
5. For any $c > 0$, we have $\mathrm{lcm}(a \cdot c, b \cdot b) = c \cdot \mathrm{lcm}(a, b)$.
6. Integers $\mathrm{lcm}(a, b)/a$ and $\mathrm{lcm}(a, b)/b$ are coprime.

Theorem 0.24. For any $a, b \in \mathbb{N}$, we have $\gcd(a, b) \cdot \mathrm{lcm}(a, b) = ab$.

One interpretation of the above theorem is that no additional algorithm is required for $\mathrm{lcm}(a, b)$ if we already have an algorithm for $\gcd(a, b)$. Indeed, we can simply use the formula $\mathrm{lcm}(a, b) = ab/\gcd(a, b)$ with the previously computed $\gcd(a, b)$.

The reasonable question is how to generalize the gcd and lcm operations to more than two arguments. For that reason, we provide the following algorithm:

Definition 0.25 (GCD and LCM for multiple arguments)**.** We define the **greatest common divisor** $\gcd(a_1, a_2, \ldots, a_n)$ and the **least common multiple** $\mathrm{lcm}(a_1, a_2, \ldots, a_n)$ for any set of integers $a_1, a_2, \ldots, a_n \in \mathbb{Z}$ as follows:

$$\gcd(a_1, a_2, \ldots, a_n) = \max\{d \in \mathbb{N} : d \mid a_1, d \mid a_2, \ldots, d \mid a_n\}.$$
$$\mathrm{lcm}(a_1, a_2, \ldots, a_n) = \min\{m \in \mathbb{N} : a_1 \mid m, a_2 \mid m, \ldots, a_n \mid m\}.$$

Theorem 0.26 (Computational Properties Of GCD and LCM For Multiple Arguments.)**.** The following two statements are true:

- $\forall a, b \in \mathbb{N} : \gcd(a, b, c) = \gcd(\gcd(a, b), c) = \gcd(a, \gcd(a, b))$.
- $\forall a, b \in \mathbb{N} : \operatorname{lcm}(a, b, c) = \operatorname{lcm}(\operatorname{lcm}(a, b), c) = \operatorname{lcm}(a, \operatorname{lcm}(a, b))$.

In conclusion, from these two theorems there is no necessity for specific algorithms for gcd and lcm when dealing with many arguments. There are more specialized algorithms for each when considering a specific number of arguments, but unfortunately, such topics are beyond the scope of this book.

0.2.2 Extended Euclidean algorithm

In this section, we will introduce the Extended Euclidean Algorithm and an important lemma related to the GCD. You might have reasonable question: why do we even need an extended version? One of the primary reasons is that this algorithm will help in finding inverse modular elements, introduced in the subsequent sections.

Lemma 0.27 (Bezout identity)**.** For any two given integers $a, b \in \mathbb{N}$ with $d = \gcd(a, b)$ there exists such $u, v \in \mathbb{Z}$ that $d = au + bv$.

Corollary 0.28 (From Bezout identity)**.**

1. Integers u and v are of different signs (excluding the case when either $u = 0$ or $v = 0$).
2. *Generalization for multiple integers:* Suppose $d = \gcd(a_1, a_2, \dots, a_n)$, then there exist such integers $u_1, u_2, \dots, u_n \in \mathbb{Z}$ that $d = u_1a_1 + u_2a_2 + \cdots + u_na_n$.

The integers u and v are called **Bezout coefficients**. The first corollary can be understood intuitively: if all coefficients are non-negative, the result will be much larger than necessary. Similarly, if they are non-positive, the result must be negative, but the GCD was defined to be positive. The second consequence follows from the fact that we can decompose gcd, thus sequentially derive this sequence. Also note that we can find Bezout coefficients on each Euclidean algorithm step.

Now we introduce the **extended Euclidean algorithm.** The extended Euclidean algorithm finds the Bezout coefficients together with the GCD efficiently.

Algorithm 1: Extended Euclidean algorithm

```
Input  : $a, b \in \mathbb{N}$. Without loss of generality, $a \geq b$
Output: $(\gcd(a, b), u, v)$
$r_0 \leftarrow a$; $r_1 \leftarrow b$
$u_0 \leftarrow 1$; $u_1 \leftarrow 0$
$v_0 \leftarrow 0$; $v_1 \leftarrow 1$
while $r_{i+1} \neq 0, \ i = 1, 2, \dots$ do
    $q_i \leftarrow r_{i-1}$ div $r_i$
    $u_{i+1} \leftarrow u_{i-1} - u_i q_i$
    $v_{i+1} \leftarrow v_{i-1} - v_i q_i$
    $r_{i+1} \leftarrow a u_{i+1} + b v_i$
end
return $(r_i, u_i, v_i)$
```

The time complexity of the Extended Euclidean algorithm is $O(\log a \log b)$ bit operations, which is very efficient. Although we already know how the Extended Euclidean Algorithm works, this method is still not very human-friendly. For that reason, let us consider an example of an easy way to find the GCD.

Example (Extended Euclidean algorithm example)**.**
First, we you need to find $d = \gcd(a, b)$. Then, knowing the sequence of expansions, find the Bezout coefficients. Note that this can be done simultaneously.

1. $125 = 93 \cdot 1 + 32$
2. $93 = 32 \cdot 2 + 29$
3. $32 = 29 \cdot 1 + 3$
4. $29 = 3 \cdot 9 + 2$
5. $3 = 2 \cdot 1 + 1$
6. $2 = 1 \cdot 2 + 0$

i	0	1	2	3	4	5	6
q_i	×	1	2	1	9	1	2
u_i	1	0	1	-2	3	-29	32
v_i	0	1	-1	3	-4	**?**	**?**

Here, each corresponding cell is calculated with the following formula:

$$u_{i-1} - q_i u_i = u_{i+1}$$
$$v_{i-1} - q_i v_i = v_{i+1}$$

Knowing that, try to finish this example by filling in the missed cells marked by **?**. After finding v_6, be sure to check yourself.

0.2.3 Prime numbers

Prime numbers are fundamental in mathematics due to their role of the building blocks of all natural numbers. Every integer greater than 1 can be uniquely factored into primes, a concept known as the *Fundamental Theorem of Arithmetic*, which we introduce in the next section. This property makes primes central to number theory, and they play a crucial role in various mathematical proofs and structures.

Definition 0.29. Number $n \in \mathbb{N}$ is **prime** if and only if its only two divisors are 1 and n.

Definition 0.30. Number $n \in \mathbb{N}$ is **composite** iff there exists an integer $a \in \mathbb{N}$, which is not 1 or n, for which $a \mid n$. In other words, it is not prime.

Remark. One might ask: what to do with the number 1? We consider it to be neither prime nor composite.

Lemma 0.31. For all $n \in \mathbb{N}$, we have $\gcd(n, n+1) = 1$

Theorem 0.32 (Euclidean theorem)**.** If $P = \{p_1, p_2, p_3, \dots p_k\} \subset \mathbb{N}$ is a finite set, consisting of prime numbers, then there exists a prime number p such that $p \notin P$.

Corollary 0.33. The set of prime numbers has an infinite cardinality.

Lemma 0.34. For all $n \in \mathbb{N}$, where n is composite, if there exists a minimal divisor $d > 1$ of n, then d is a prime number.

Let's say we have a large number and we need to find out if it is prime, how do we do it? In other words, are there any methods for checking a number for prime? Yes, there are, starting with logical reasoning, you can check numbers with brute force, although this method is not practically applicable. There are probabilistic prime tests that will fail with some error. In 2003, an effective deterministic test of simplicity was found, which moved the problem to the class P.

It is worth to be mentioned, there are also different forms of primes that find their application in some fields. For example, Mersenne primes, factorial primes, Euclidean primes, Fibonacci primes, and many other types of primes. We will consider one of the most famous types of primes — Mersenne primes.

Definition 0.35 (Mersenne primes)**.** The prime number of form $2^p - 1$ is called **Mersenne prime**, where p is a prime number.

Mersenne primes, are important in both number theory and cryptography. They have unique mathematical properties that make them useful for testing primality and generating large prime numbers. Mersenne primes are also crucial in the construction of efficient algorithms for error correction in coding theory and for generating random numbers in cryptographic applications, etc. Beside, next theorem importan says that we can know the form of a prime numbers.

Theorem 0.36 (Dirichlet's theorem)**.** For any $a, b \in \mathbb{N}$ if $\gcd(a, b) = 1$, then infinite primes number form of $am + b$ exists, where $m \in \mathbb{N}$.

In other words, every infinite arithmetic progression whose first term and difference are positive integers contains an infinite number of primes.

0.2.4 Fundamental Theorem of Arithmetic

The Fundamental Theorem of Arithmetic states that every integer greater than 1 can be uniquely factored into primes, which is crucial for understanding the structure of numbers. It plays a key role in areas like number theory, cryptography, and simplifying calculations involving divisibility. But before formal description of the theorem, for better understanding it is good to know the following Lemma 0.37.

Lemma 0.37 (Euclidean)**.** If p is a prime and $p \mid ab$, then $p \mid a$ or $p \mid b$.

Proof. ▶ Let $p \mid ab$, but $p \nmid a$, then $\gcd(a, p) = 1$ because there are no common divisors. In which case, by Bezout identity Lemma 0.27, there exist such $u, v \in \mathbb{Z}$ that $au + pv = 1$. Let's multiply the left and right sides by b: $abu + pbv = b$, but $p \mid ab$ and $p \mid pb$, therefore their sum is also divisible by $p \mid abu + pbv$. ◀

Now, we are ready to introduce the central Number Theory theorem — Fundamental Theorem of Arithmetic.

Theorem 0.38 (Fundamental theorem of arithmetic)**.** Any integer $n > 1$

can be decomposed in the unique way into a product of prime numbers:

$$n = p_1^{\alpha_1} p_2^{\alpha_2} \cdots p_t^{\alpha_t} = \prod_{j=1}^{t} p_j^{\alpha_j},$$

where $p_1, \dots, p_t$ are prime numbers and $\alpha_1, \dots, \alpha_t \in \mathbb{N}$.

Proof. ▶ The theorem state equation then need to proof equation existence and uniqueness. Since the existence property is easy to prove, let us make the small exception and show it. For other proves (in particular, for the uniqueness case, see literature)

Existence. Suppose the theorem statement is false and there exist n that do not have such a representation. Let n_0 be the smallest of them. If n_0 is prime, it can be represented as $p_1 = n_0$, $\alpha_1 = 1$, which satisfies the representation. Therefore, n_0 must be composite, which means $\exists a, b \in \mathbb{N}, 1 < a, b < n_0$ such that $n = ab$. Since n_0 is the smallest non-decomposable number and $a, b < n_0$, then a and b can be represented as

$$a = p_1^{\alpha_1} \dots p_t^{\alpha_t}$$
$$b = q_1^{\beta_1} \dots q_k^{\beta_k}.$$

Therefore, $n_0 = ab = p_1^{\alpha_1} \dots p_t^{\alpha_t} q_1^{\beta_1} \dots q_k^{\beta_k}$. Thus, the contradiction. ◀

Corollary 0.39. Suppose n is decomposed as in Theorem 0.38. Then,

1. If $d \mid n$, then $d = p_1^{\beta_1} p_2^{\beta_2} \dots p_t^{\beta_t}$, where $0 \le \beta_i \le \alpha_i$.
2. Suppose $a = p_1^{\alpha_1} \dots p_t^{\alpha_t}$ and $b = p_1^{\beta_1} \dots p_t^{\beta_t}$. Then, the GCD can be evaluated as $\gcd(a, b) = \prod_{i=0}^{t} p^{\min\{\alpha_i, \beta_i\}}$.
3. Suppose $a = p_1^{\alpha_1} \dots p_t^{\alpha_t}$ and $b = p_1^{\beta_1} \dots p_t^{\beta_t}$. Then, the LCM can be evaluated as $\operatorname{lcm}(a, b) = \prod_{i=0}^{t} p^{\max\{\alpha_i, \beta_i\}}$.
4. If $b \mid a$ and $c \mid a$ and $\gcd(b, c) = 1$, then $bc \mid a$.

The implications and applications of the Fundamental Theorem of Arithmetic are very large and important, and a number of "obvious" ones are listed above. However, you should also be aware that the problem of factorization, i.e., knowing the decomposition of a number into prime factors, is in a NP class (complexity) and is the basis of some cryptosystems, although it is gradually being abandoned, including due to the potential of quantum computers.

0.2.5 Modular arithmetic

Now we are ready for practical applications of number theory, starting with modulo congruence. In this section, we will review the idea of congruence, learn how to use it, and explore its key properties. Understanding these properties will help simplify calculations and solve problems more efficiently in fields such as cryptography, algorithms, and number theory.

Definition 0.40. Two integers $a, b \in \mathbb{Z}$ are said to be **congruent** modulo $n \in \mathbb{N}$ (congruence modulo n is denoted as $a \equiv b \pmod{n}$), if one of the following conditions is met:

1. There exists such $t \in \mathbb{Z}$ that $a = b + nt$
2. $a \bmod n = b \bmod n$
3. $n \mid (a - b)$

It is fairly easy to see that all conditions are equivalent to each other. Next, as always, it is necessary to formally describe the properties, although some of them may seem intuitive, they need to be formally defined.

Lemma 0.41 (Reflexivity, Symmetry, and Transitivity)**.**

1. $a \equiv a \pmod{n}$
2. If $a \equiv b \pmod{n}$, then $b \equiv a \pmod{n}$
3. If $a \equiv b \pmod{n}$ and $b \equiv c \pmod{n}$, then $a \equiv c \pmod{n}$

The lemma states three basic properties of congruence modulo n: reflexivity, symmetry, and transitivity. These properties confirm that congruence modulo n is an equivalence relation (for general equivalence relation definition, see **??**).

Lemma 0.42. Suppose we have $a \equiv b \pmod{n}$ and $c \equiv d \pmod{n}$, then we have

1. $a \pm c \equiv b \pm d \pmod{n}$
2. $ac \equiv bd \pmod{n}$

In turn, this lemma states that we can perform addition, subtraction, and multiplication on the congruent numbers modulo n similarly to the usual arithmetic.

Lemma 0.43. If $ca \equiv cb \pmod{n}$ and $\gcd(c, n) = 1$, then $a \equiv b \pmod{n}$.

This means that we can cancel out the same left and right parts respectively, but with a certain requirement. You may ask why this is necessary, the answer is in the following example.

Example. Let us try to simplify the following equation: $6 \equiv 2 \pmod 4$. Suppose we divide both sides by 2, then we have the false statement $3 \equiv 1 \pmod 4$. That being said, the requirement $\gcd(c, n) = 1$ is mandatory.

Remark. In particular, the example above shows why the division operation is fundamentally different from the regular real/rational numbers. We will see that, in fact, the arithmetic modulo n is not always what mathematicians call a **field**. You will see more details in **??**.

Lemma 0.44. Modulo congruence can be scaled in both directions:

1. If $k \neq 0, a \equiv b \pmod n$, then $ak \equiv bk \pmod{nk}$
2. If $d = \gcd(a, b, n)$ and $a = a_1 d, b = b_1 d, n = n_1 d, a \equiv b \,(\text{mod } n)$, then $a_1 = b_1 \,(\text{mod } n_1)$

Lemma 0.45. If $d \mid n$ and $a \equiv b \,(\text{mod } n)$, then $a \equiv b \,(\text{mod } d)$.

This property is very convenient when it comes to large calculations, if you know its decomposition. It allows you to significantly reduce them, and then restore the result by a large modulus.

Lemma 0.46. Suppose $a \equiv b \pmod{n_1}$, $a \equiv b \pmod{n_2}$, $\dots$, $a \equiv b \pmod{n_k}$. Then, the following statement is true:

$$a \equiv b \pmod{\operatorname{lcm}(n_1, n_2, \dots, n_k)}.$$

Lemma 0.47. If $a \equiv b \,(\text{mod } n)$, then $\gcd(a, n) = \gcd(b, n)$

Definition 0.48. The congruence class or residues of k modulo n is defined as a set $k + n\mathbb{Z} = \{k + nt \mid t \in \mathbb{Z}\}$

Definition 0.49. The **complete residue system modulo** (or residue ring) n is a set of integers, where every integer is congruent to a unique member of the set modulo n, usually denoted as $\mathbb{Z}_n = \{0, 1, 2, \dots, n - 1\}$.

Remark. Sometimes, one might also encounter the notation $\mathbb{Z}/n\mathbb{Z}$, which is more frequently used in Abstract Algebra.

The aforementioned lemmas and definitions describe the properties of addition, subtraction, and multiplication. But what about division? To define the division (if such operation is valid at all), we need to consider the so-called *modular multiplicative inverse*, which is usually denoted as $a^{-1} \pmod{n}$, similarly to real/rational numbers.

Definition 0.50. A modular multiplicative inverse of an integer $a \in \mathbb{Z}$ modulo $n \in \mathbb{N}$ is such an integer a^{-1} that satisfies $a \cdot a^{-1} \equiv a^{-1}a \equiv 1 \pmod{n}$.

Remark. (**Caution**): not all numbers have an inverse number!

Remark. The inverse (if exists) behaves similarly to the usual inverse over rations/reals. For example, $(a^2)^{-1} = (a^{-1})^2$, which means, we can first find the inverse, then the squared value, and vice versa.

The question then is when does the number have the inverse? Consider the following theorem.

Theorem 0.51. Modular inverse $a^{-1} \pmod{n}$ exists if and only if $\gcd(a, n) = 1$.

Based on Extended Euclidean Algorithm 1 and Theorem 0.51, we build the algorithm that can verify an existence and find the inverse value.

Algorithm 2: Modular multiplicative inverse algorithm

```
Input  : $a, n$
Output: $a^{-1}$
$(d, u, v) \leftarrow$ ExtendedEuclideanAlgorithm$(a, n)$ /* See Algorithm 1 */
if $d \neq 1$ then
    return inverse does not exist.
end
return $u$
```

Remark. Note that the complexity of this algorithm is the same as for Algorithm 1, which is $O(\log a \log n)$ bit operations.

Definition 0.52. The **multiplicative group of integers** modulo n, denoted as $\mathbb{Z}_n^\times$, is a set of natural numbers that are coprime to n and less than n. In other words, $\mathbb{Z}_n^\times = \{a \in \mathbb{N} : \gcd(a, n) = 1\}$.

The $\mathbb{Z}_n^\times$ is a fundamental object in number theory and plays a crucial role in cryptography, forming the basis almost for every cryptographic primitive.

Definition 0.53. Euler's Totient Function $\varphi(n)$ is the cardinality of the multiplication group of integers $\mathbb{Z}_n^\times$. In other words, $\varphi(n) = |\mathbb{Z}_n^\times|$.

Remark. The alternative Euler's totient function intuition is following: $\varphi(n)$ counts all coprimes with n in range $[1, n]$.

Lemma 0.54 (Euler's totient function properties)**.**

1. $\varphi(1) = 1$.
2. $\varphi(p) = p - 1$, p where is prime.
3. $\varphi(pq) = \varphi(p) \cdot \varphi(q)$, where p, q are primes.
4. $\varphi(p^\alpha) = p^\alpha - p^{\alpha-1}$, where p is prime.

Corollary 0.55. For any number $n = p_1^{\alpha_1} p_2^{\alpha_2} \dots p_t^{\alpha_t}$ general formula for Euler's totient function is:

$$\varphi(n) = \prod_{i=1}^{t} \left(p_i^{\alpha_i} - p_i^{\alpha_i - 1}\right) = n \prod_{i=1}^{t} \left(1 - \frac{1}{p_i}\right).$$

Example.

- $\varphi(107) = 106$, because 107 is prime.
- $\varphi(123) = \varphi(3 \cdot 41) = \varphi(3)\varphi(41) = 2 \cdot 40 = 80$
- $\varphi(729) = \varphi(3^6) = 3^6 - 3^5 = 486$

0.2.6 Chinese Remainder Theorem

The Chinese Remainder Theorem is a fundamental result in number theory that provides an efficient method for solving congruence systems. It allows to decompose a complex problem with large integers into several simpler problems with smaller modules. This decomposition reduces the required computational complexity, especially when working with large numbers, and is widely used in areas such as modular arithmetic, cryptography, and algorithmic number theory.

Theorem 0.56 (Chinese Remainder Theorem)**.** Suppose there is the following system

$$\begin{cases} x \equiv x_1 \pmod{n_1} \\ x \equiv x_2 \pmod{n_2} \\ x \equiv x_3 \pmod{n_3} \\ \dots \\ x \equiv x_t \pmod{n_t} \end{cases}$$

where the n_i are pairwise coprime and $x_1, x_2, \cdots x_t$ are fixed numbers. Then there is unique solution in modulo $N = n_1 n_2 \cdots n_t$.

In fact, we can easily find such solution. Let $N_i = N/n_i$ since n_i are pairwise coprime one, for all i has $\gcd(N_i, n_i) = 1$, and by the Extended Euclidean Algorithm, there exists an integer a_i such that $a_i N_i \equiv 1 \pmod{n_i}$. Thus, the solution is given by

$$x = a_1 N_1 x_1 + a_2 N_2 x_2 + \cdots + a_t N_t x_t.$$

This expression provides the unique solution modulo N, where $N = n_1 n_2 \cdots n_t$. Beside, using this idea, we can write an efficient algorithm to compute x iteratively.

0.2.7 Euler's Theorem

Theorem 0.57 (Euler's)**.** For all $n \in \mathbb{N}$ and $a \in \mathbb{Z}_n^\times$ holds $a^{\varphi n} \equiv 1 \pmod{n}$.

▶ To prove this fact, we prove the following auxiliary lemma.

Lemma 0.58. Suppose $a \in \mathbb{Z}_n^\times$. Denote by $a\mathbb{Z}_n^\times = \{ax : x \in \mathbb{Z}_n^\times\}$. Then, $\mathbb{Z}_n^\times = a\mathbb{Z}_n^\times$. In other words, elementwise multiplication by a permutes the elements of $\mathbb{Z}_n^\times$.

Proof. Our statement is equivalent to claiming that the function $f : \mathbb{Z}_n^\times \to a\mathbb{Z}_n^\times$ defined as $f(x) = ax \pmod{n}$ is a **bijection**. To show this, we need to prove the following three staments:

1. **Correctness** is obvious since ax is in $a\mathbb{Z}_n^\times$ if x is in $\mathbb{Z}_n^\times$.
2. **Injectivity**: we need to prove that if $f(x_1) = f(x_2)$ for two $x_1 \neq x_2 \in \mathbb{Z}_n^\times$, then $x_1 = x_2$. From the function definition, we have $ax_1 = ax_2 \pmod{n}$. Following the Lemma 0.44, we can cancel a modulo n as a is coprime to n. Thus, $x_1 = x_2$.
3. **Surjectivity**: We need to prove that every element $y \in a\mathbb{Z}_n^\times$ has at least

one pre-image $f^{-1}(y)$. Indeed, in this case we have $ax \equiv y \pmod{n}$. Then, set $x := a^{-1}y \pmod{n}$. This expression is well-defined as a is coprime to n and has an inverse.

Therefore, f is bijective and thus $\mathbb{Z}_n^\times = a\mathbb{Z}_n^\times$.

So we know $a\mathbb{Z}_n^\times = \mathbb{Z}_n^\times$, how do we proceed? Notice that from the set equality follows the equality of the products of all elements. Let us write this down:

$$\prod_{i=1}^{\varphi(n)} x_i \equiv \prod_{i=1}^{\varphi(n)} ax_i \pmod{n} \implies \prod_{i=1}^{\varphi(n)} x_i \equiv a^{\varphi(n)} \prod_{i=1}^{\varphi(n)} x_i \pmod{n}$$

Since all x_i are coprime to n by definition, we can cancel out the product of all x_i from both sides, leading to the conclusion of Euler's theorem. ◀

Corollary 0.59. If p is a prime and a is not divisible by p, then $a^{p-1} \equiv 1 \pmod{p}$.

This result is commonly known as Fermat's Little Theorem. However, to maintain generality, we state the theorem in a more general form below.

Theorem 0.60 (Fermat's Little Theorem.)**.** Let p is prime number. For any integer a, the following holds $a^p \equiv a \pmod{p}$.

Note that Euler's theorem is a generalization of Fermat's Little Theorem.

0.2.8 Schwartz-Zippel Lemma

Lemma 0.61. Let $\mathbb{F}$ be a field. Let $f(x_1, x_2, ..., x_n)$ be a polynomial of total degree d. Suppose that f is not the zero polynomial. Let S be a finite subset of $\mathbb{F}$. Let $r_1, r_2, ...r_n$ be chosen at random uniformly and independently from S. Then the probability that $f(r_1, r_2, ..., r_n) = 0$ is $\leq \frac{d}{|S|}$.

Example. Let $F = \mathbb{F}_3$, $f(x) = x^2 - 5x + 6$, $S = F$, $r \xleftarrow{R} \mathbb{F}_3$.
Schwartz-Zippel lemma says that the probability that $f(r) = 0$ is $\leq \frac{2}{3}$.

Given two polynomials P, Q with degree d in a field $\mathbb{F}_p$, for $r \xleftarrow{R} \mathbb{F}_3$: $\Pr[P(r) == Q(r)] \leq \frac{d}{p}$. For large fields, where $\frac{d}{p}$ is negligible, this property allows to succinctly check the equality of polynomials. Let $H(x) := P(x) - Q(x)$. Than for each $P(x) = Q(x) \to H(x) = 0$. Applying Schwartz-Zippel lemma, the probability of $H(x) = 0$ for $x \overset{R}{\leftarrow} \mathbb{F}$ is $\leq \frac{d}{|S|}$.

0.3 Exercises

Exercise 1. Suppose that for the given cipher with a security parameter λ, the adversary $\mathcal{A}$ can deduce the least significant bit of the plaintext from the ciphertext. Recall that the advantage of a bit-guessing game is defined as $\mathsf{SSAdv}[\mathcal{A}] = \left|\Pr\left[b = \hat{b}\right] - \frac{1}{2}\right|$, where b is the randomly chosen bit of a challenger, while $\hat{b}$ is the adversary's guess. What is the maximal advantage of $\mathcal{A}$ in this case?

Hint: The adversary can choose which messages to send to challenger to further distinguish the plaintexts.

a) 1
b) $\frac{1}{2}$
c) $\frac{1}{4}$
d) 0
e) Negligible value ($\mathsf{negl}(\lambda)$).

Exercise 2. Consider the cipher $\mathcal{E} = (E, D)$ with encryption function $E : \mathcal{K} \times \mathcal{M} \to \mathcal{C}$ over the message space $\mathcal{M}$, ciphertext space $\mathcal{C}$, and key space $\mathcal{K}$. We want to define the security that, based on the cipher, the adversary $\mathcal{A}$ cannot restore the message (*security against message recovery*). For that reason, we define the following game:

1. Challenger chooses random $m \xleftarrow{R} \mathcal{M}, k \xleftarrow{R} \mathcal{K}$.
2. Challenger computes the ciphertext $c \leftarrow E(k, m)$ and sends to $\mathcal{A}$.
3. Adversary outputs $\hat{m}$, and wins if $\hat{m} = m$.

We say that the cipher $\mathcal{E}$ is secure against message recovery if the **message recovery advantage**, denoted as $\mathsf{MRadv}[\mathcal{A}, \mathcal{E}]$ is negligible. Which of the following statements is a valid interpretation of the message recovery advantage?

a) $\mathsf{MRadv}[\mathcal{A}, \mathcal{E}] := \left|\Pr[m = \hat{m}] - \frac{1}{2}\right|$
b) $\mathsf{MRadv}[\mathcal{A}, \mathcal{E}] := |\Pr[m = \hat{m}] - 1|$.
c) $\mathsf{MRadv}[\mathcal{A}, \mathcal{E}] := \Pr[m = \hat{m}]$
d) $\mathsf{MRadv}[\mathcal{A}, \mathcal{E}] := \left|\Pr[m = \hat{m}] - \frac{1}{|\mathcal{M}|}\right|$

Exercise 3. Suppose that f and g are negligible functions. Which of the following functions is not neccessarily negligible?

a) $f + g$

b) $f \times g$

c) $f - g$

d) f/g

e) $h(\lambda) := \begin{cases} 1/f(\lambda) & \text{if } 0 < \lambda < 100000 \\ g(\lambda) & \text{if } \lambda \geq 100000 \end{cases}$

Exercise 4. Suppose that $f \in \mathbb{F}_p[x]$ is a d-degree polynomial with d **distinct** roots in $\mathbb{F}_p$. What is the probability that, when evaluating f at n random points, the polynomial will be zero at all of them?

a) Exactly $(d/p)^n$.

b) Strictly less that $(d/p)^n$.

c) Exactly nd/p.

d) Exactly d/np.

Exercise 5-6. To demonstrate the idea of Reed-Solomon codes, consider the toy construction. Suppose that our message is a tuple of two elements $a, b \in \mathbb{F}_{13}$. Consider function $f : \mathbb{F}_{13} \to \mathbb{F}_{13}$, defined as $f(x) = ax + b$, and define the encoding of the message (a, b) as $(a, b) \mapsto (f(0), f(1), f(2), f(3))$.

Question 5. Suppose that you received the encoded message $(3, 5, 6, 9)$. Which number from the encoded message is corrupted?

a) First element (3).

b) Second element (5).

c) Third element (6).

d) Fourth element (9).

e) The message is not corrupted.

Question 6. Consider the previous question. Suppose that the original message was (a, b). Find the value of $a \times b$ (in $\mathbb{F}_{13}$).

a) 4

b) 6

c) 12

d) 2

e) 1