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**:

$$(\forall k \in \mathcal{K})\,(\forall m \in \mathcal{M}) : \{D(k, E(k, m)) = m\} \tag{1}$$

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 $\hat{b}$.

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:

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

We say that the encryption scheme is **semantically secure**[1] if for any efficient adversary $\mathcal{A}$ the advantage $\mathrm{SSAdv}[\mathcal{A}]$ 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.

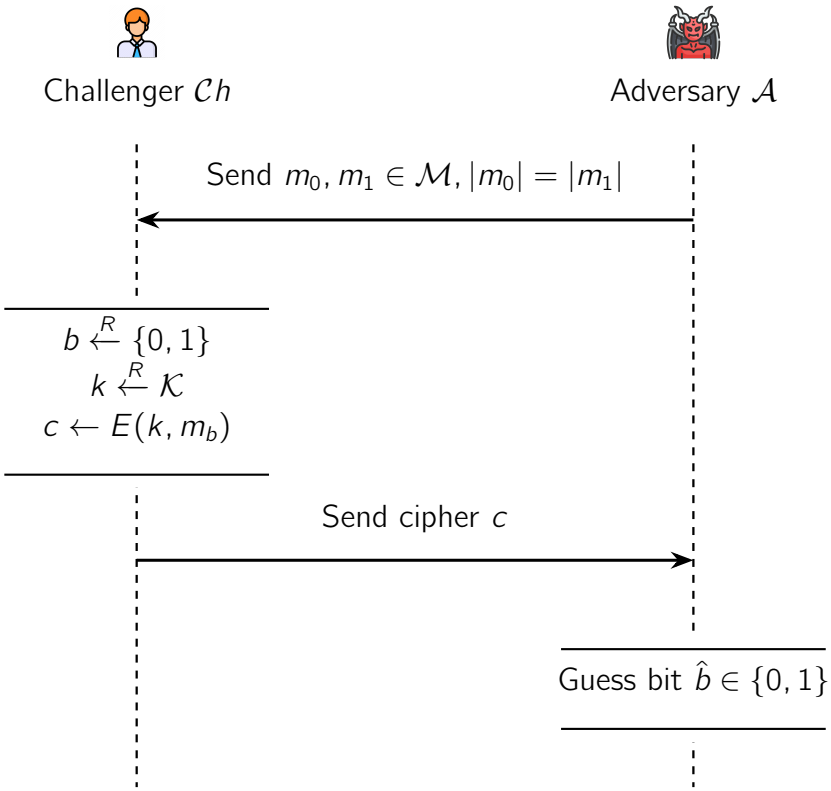

Figure 0.1: The game between the adversary $\mathcal{A}$ and the challenger $\mathcal{C}h$ for defining the semantic security.

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

$$\lim_{n\to\infty} f(n)n^c = 0 \tag{3}$$

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

$$\lim_{n\to\infty} 2^{-n}n^c = 0 \tag{4}$$

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

$$\lim_{n\to\infty} \frac{1}{n} \times n = 1 \neq 0 \tag{5}$$

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:

$$\left| \Pr\left[b = \hat{b} \;\middle|\; \begin{array}{c} m_0, m_1 \leftarrow \mathcal{A},\ k \xleftarrow{R} \mathcal{K},\ b \xleftarrow{R} \{0,1\} \\ c \leftarrow E(k, m_b) \\ \hat{b} \leftarrow \mathcal{A}(c) \end{array} \right] - \frac{1}{2} \right| < \mathrm{negl}(\lambda) \tag{6}$$

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 Discrete Logarithm Assumption (DL)

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

Definition 0.4. 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 0.5. 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!

0.1.3 Computational Diffie-Hellman (CDH)

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 0.6. 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 0.7. 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.

0.1.4 Decisional Diffie-Hellman (DDH)

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 0.8. 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

$$\mathsf{DDHadv}[\mathcal{A}, \mathbb{G}] := \left| \Pr\left[b = \hat{b}\right] - \frac{1}{2} \right| \tag{7}$$

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

Theorem 0.9. 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 0.10. 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

$$u^n = (g^{2\alpha'})^n = g^{2n\alpha'} = (g^{2n})^{\alpha'} = 1^{\alpha'} = 1 \tag{8}$$

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

some intro word

0.2.1 Introduction to number theory

Definition 0.11. An integer a is divisible by a nonzero integer b, denoted $b \mid a$, or b is divisor of a, if and only if there exists an integer $k \in \mathbb{Z}$ such that $a = k \cdot b$.

Lemma 0.12 (Divisibility properties)**.** For all $a, b, c \in \mathbb{Z}$:

1. $1 \mid a$
2. $a \neq 0,\ a \mid a$
3. $a \neq 0,\ a \mid 0$
4. $b \mid a,\ c \mid b \Rightarrow c \mid a$
5. $b \mid a \Leftrightarrow b \cdot c \mid a \cdot c,\ c > 0$
6. $c \mid a,\ c \mid b \Rightarrow c \mid (\alpha \cdot a \pm \beta \cdot b)$, for any $\alpha, \beta \in \mathbb{Z}$

You might wonder if there are cases where a number is not divisible by any integer, meaning there is no integer k that satisfies the condition. Well, in such cases, a new theorem comes into play.

Theorem 0.13. $\forall a, b \in \mathbb{Z} : \exists!\ q, r$, where $q \in \mathbb{Z},\ r \in \mathbb{N},\ 0 \leq r < |b|$, that $a = b \cdot q + r$.

To prove this theorem, all we need to prove is the existence and uniqueness of such a decomposition. Basically, this new theorem allows to define new operation that accept two integers and return two integers $q = \left\lfloor \frac{a}{b} \right\rfloor,\ r = a - b \cdot \left\lfloor \frac{a}{b} \right\rfloor$. Thus, $a \bmod b$ represents the remainder when a is divided by b.

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 (gcd) comes into.

Definition 0.14. $\forall a, b \in \mathbb{Z}, \gcd(a, b) = d$ such that:

1. $d \mid a,\ d \mid b$
2. $d \in \mathbb{N}$ is a maximal integer that satisfies the first condition

Definition 0.15. Two numbers a and b **coprime** if and only if $\gcd(a, b) = 1$.

Lemma 0.16 (Greatest common divisor properties)**.**

1. $\gcd(a, b) = b \Leftrightarrow b \mid a$
2. $a \neq 0,\ \gcd(a, 0) = a$
3. $\exists \delta\colon\ \delta \mid a, \delta \mid b \Rightarrow \delta \mid \gcd(a, b)$
4. $c > 0,\ \gcd(a \cdot c, b \cdot c) = c \cdot \gcd(a, b)$
5. $d = \gcd(a, b) \Rightarrow \gcd(\frac{a}{d}, \frac{b}{d}) = 1$

Lemma 0.17. $\gcd(a, b) = \gcd(b, a - b)$

Corollary 0.18. $\gcd(a, b) = \gcd(b, a \mod b)$

All these properties are interesting and useful, but you may wonder how to actually find $gcd(a, b)$. Euclidean algorithm is an efficient method for computing the greatest common divisor, and we will describe extended version, but later.

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 0.19. $\forall a, b \in \mathbb{Z}, lcm(a, b) = m$ such that:

1. $a \mid m$, $a \mid m$
2. $m \in \mathbb{N}$ is a maximal integer that satisfies the first condition

Lemma 0.20 (Least common multiple properties)**.**

1. $\text{lcm}(a, 0)$ - undefined
2. $\text{lcm}(a, b) = a \Leftrightarrow b \mid a$
3. a, b - coprime $\Rightarrow \text{lcm}(a, b) = a \cdot b$
4. Any common divisor of a and b $\delta \mid \text{lcm}(a, b)$
5. $\forall c > 0 : \text{lcm}(a \cdot c, b \cdot b) = c \cdot \text{lcm}(a, b)$
6. $\frac{\text{lcm}(a,b)}{a}$ and $\frac{\text{lcm}(a,b)}{b}$ - are coprime

Theorem 0.21. $\forall a, b \in \mathbb{N} : \gcd(a, b) \cdot \text{lcm}(a, b) = a \cdot b$

This theorem states that no additional algorithm is required for $\text{lcm}(a, b)$.

Theorem 0.22. $\forall a, b \in \mathbb{N} : \gcd(a, b, c) = \gcd(\gcd(a, b), c) = \gcd(a, \gcd(a, b))$

Theorem 0.23. $\forall a, b \in \mathbb{N} : \text{lcm}(a, b, c) = \text{lcm}(\text{lcm}(a, b), c) = \text{lcm}(a, \text{lcm}(a, b))$

In conclusion, from these two theorems, we already know that 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 extended version. The simple answer is that it will later help us find inverse elements, which will be introduced in a subsequent section.

Lemma 0.24 (Bezout identity)**.** For all $a, b \in \mathbb{N}, d = \gcd(a, b)$ $\exists u, v \in \mathbb{Z}, d = a \cdot u + b \cdot v$

Corollary 0.25 (From Bezout identity)**.**

1. $d = au + bv$, where one of the numbers is necessarily non-negative and the other is non-positive.
2. $d = \gcd(a_1, a_2, \ldots, a_n) \Rightarrow \exists\, u_1, u_2, \ldots, u_n \in \mathbb{Z} :\ d = u_2 a_1 + u_2 a_2 + \cdots + u_n a_n$
3. For all $i \in \mathbb{N}$ $\exists\, u_i, v_i \in \mathbb{Z} : r_i = au_i + bv_i$

The integers u and v are called coefficients Bezout. The first corollary seems obvious,

because if all coefficients are non-negative, the result will be much larger than necessary, and if they are non-positive, the result must be negative, but the GCD $\in \mathbb{N}$. The second consequence follows from the fact that we can decompose gcd thus sequentially derive this sequence. The last one stand that we can find coefficients Bezout on each euclidean algorithm step.

The extended Euclidean algorithm finds the Bezout coefficients together with the GCD.

Algorithm 1: Extended Euclidean algorithm

Input : $a, b \in \mathbb{N}, a \geq b$
Output: $(gcd(a, b),\ u,\ v)$

$r_0 \leftarrow a$; $r_1 \leftarrow b$
$u_0 \leftarrow 1$; $u_1 \leftarrow 0$
$v_0 \leftarrow 0$; $v_1 \leftarrow 1$
for $i \in \{1, \ldots, \log(b)\}$ **do**
  $q_i \leftarrow \lfloor r_{i-1} \,/\, r_i \rfloor$
  $u_{i+1} \leftarrow u_{i-1} - u_i q_i$
  $v_{i+1} \leftarrow v_{i-1} - v_i q_i$
  $r_{i+1} \leftarrow a u_{i+1} + b v_i$
  if $r_{i+1} = 0$ **then**
    return (r_i, u_i, v_i)
  end
end

Hence, we already know how the Extended Euclidean Algorithm works, but this method is not human-friendly. Next, example of a easy way to find the GCD.

Example (Extended Euclidean algorithm example)**.**
First, we you need to find d = gcd(a,b), then, knowing this sequence of expansions, find the coefficients of Bezout, although 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	$\times$	1	2	1	9	1	2
u_i	1	0	1	-2	3	-29	32
v_i	0	1	-1	3	-4		

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. After finding v_6 and be sure to check yourself.

0.2.3 Prime numbers

0.2.4 Fundamental theorem of arithmetic

0.2.5 Linear congruence

0.2.6 Modular multiplicative inverse

0.2.7 Chinese remainder theorem

0.2.8 Schwartz-Zippel Lemma

Lemma 0.26. 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 $P(x) = Q(x) \rightarrow H(x) = 0$. Applying Schwartz-Zippel lemma, the probability of $H(x) = 0$ for $x \xleftarrow{R} \mathbb{F}$ 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 ($\mathrm{negl}(\lambda)$).

Exercise 2. Consider the cipher $\mathcal{E} = (E, D)$ with encryption function $E : \mathcal{K} \times \mathcal{M} \rightarrow \mathcal{C}$ 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 $\mathrm{MRadv}[\mathcal{A}, \mathcal{E}]$ is negligible. Which of the following statements is a valid interpretation of the message recovery advantage?

a) $\mathrm{MRadv}[\mathcal{A}, \mathcal{E}] := \left|\Pr[m = \hat{m}] - \frac{1}{2}\right|$

b) $\mathrm{MRadv}[\mathcal{A}, \mathcal{E}] := |\Pr[m = \hat{m}] - 1|$.

c) $\mathrm{MRadv}[\mathcal{A}, \mathcal{E}] := \Pr[m = \hat{m}]$

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

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