SAPIENZA
UNIVERSITÀ DI ROMA

"Sapienza" University of Rome
Faculty of Information Engineering, Informatics and Statistics
Department of Computer Science

Cryptography

Lecture notes integrated with the book "Introduction to Modern Cryptography", J. Katz, Y. Lindell

Author
Simone Bianco

January 3, 2026

Contents

Information and Contacts

Personal notes and summaries collected as part of the *Cryptography* course offered by the degree in Computer Science of the University of Rome "La Sapienza".

Further information and notes can be found at the following link:
https://github.com/Exyss/university-notes. Anyone can feel free to report inaccuracies, improvements or requests through the Issue system provided by GitHub itself or by contacting the author privately:

- Email: **bianco.simone@outlook.it**
- LinkedIn: **Simone Bianco**

The notes are constantly being updated, so please check if the changes have already been made in the most recent version.

Suggested prerequisites:

Algebra and Computational Complexity

Licence:

1

Introduction to modern cryptography

1.1 Definitions, assumptions and notation

Cryptography is the branch of computer science that practices and studies techniques for secure communication in the presence of adversarial behavior (e.g. altering the integrity of the message, reading the content of the message, ...). Cryptography focuses on two main goals:

1. **Confidential communication**: the property of making the original message impossible to read even if an outsider finds out the *ciphertext*, i.e. the encrypted message

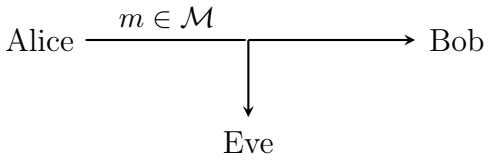

Figure 1.1: Without an encryption scheme, Eve – an evildoer – may be able to read the message that Alice is sending to Bob.

2. **Message integrity**: the capability of ensuring that the original message has not been altered even if an outsider intercepts the ciphertext

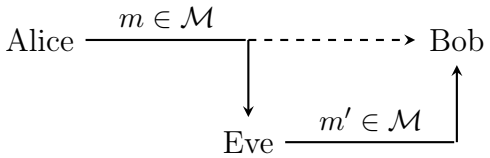

Figure 1.2: Without an encryption scheme, Eve may be able to intercept and alter the message that Alice is sending to Bob.

Before the 50's, cryptography was considered an *art* for geniouses capable of encrypting and decrypting messages written by other geniouses of the field. In modern days, cryptography became a *mathematical science* with precise definitions and proofs. These mathematical tools separate in two types: **unconditional proofs** and **conditional proofs**.

The former type referres to proofs where no assumptions are made, focusing on showing that something is possible without caring about it being inefficient (hence we have theoretically infinite resources at our disposal). The latter, instead, uses real-world assumptions that are believed to be true in order to prove real-world results. The typical example is the $\mathsf{P} \neq \mathsf{NP}$ assumption, i.e. that there are some problems that are verifiable in polynomial time but not solvable in polynomial time. Many cryptosystems are based on the assumption that prime factorization is hard to solve but easy to verify. This is equivalent to assuming that FACTORING $\in \mathsf{NP} - \mathsf{P}$ (only FACTORING $\in \mathsf{NP}$ has been proven), making the assumption $\mathsf{P} \neq \mathsf{NP}$ fundamental. Making a crytosystem easy to verify allows us to make it impossible to access a resource without using knowing the solution to the authentication phase.

Theorem 1.1

Every cryptosystem based on prime factorization is "secure" if FACTORING $\notin \mathsf{P}$

Proof (sketch). By contrapositive, suppose that there is a cryptosystem Π that is not "secure", meaning that there is a polynomial time algorithm A that is capable of "breaking" Π. Then, for each number $n \in \mathbb{N}$ we can forge a message m based on n and use the output $A(m)$ to find the prime factors of n, concluding that $\mathsf{FACTORING} \in \mathsf{P}$. □

These two goals will be discussed under two main types of cryptosystems:

- **Symmetric cryptography**: both ends of the communication share a single common key that is random and unknown to any outsider.
- **Asymmetric cryptography**: both ends of the communication have each their own key pair (pk, sk) where pk is *public*, i.e. known by everyone, and sk is *secret*, i.e. known only by the owner.

Throughout this work we'll use the following notation to talk about cryptographic systems:

- $\mathcal{M}$ is the message space, i.e. the set of all strings of messages that can be sent between two parties.
- $\mathcal{C}$ is the ciphertext space, i.e. the set of all strings of ciphertexts that can be produced by a cryptosystem.
- $\mathcal{K}$ is the key space, i.e. the set of all strings of keys that can be used in a cryptographic system.
- $\mathcal{H}$ is the hashing function space, i.e. the set of all hash functions that can be used by a cryptosystem.

2

Information-theoretic cryptography

2.1 Perfect secrecy and Shannon's theorem

For now, we'll focus on symmetric cryptography, i.e. cryptosystems where a shared secret key is used for both encryption and decryption. It is widely used due to its speed and efficiency, especially in encrypting large volumes of data. A core model of this approach is **Symmetric-key Encryption (SKE)**, which includes three main components:

- A shared *secret key* $K \in_R \mathcal{K}$ chosen uniformly at random
- An *encryption function* $\text{Enc} : \mathcal{K} \times \mathcal{M} \to \mathcal{C}$ that transforms plaintext into ciphertext
- A *decryption function* $\text{Dec} : \mathcal{K} \times \mathcal{C} \to \mathcal{M}$ that transforms a ciphertext into plaintext

In order to be functional, SKEs must be **correct**, meaning that if a message $m \in \mathcal{M}$ gets encrypted with $K \in_R \mathcal{K}$ obtaining the ciphertext $c \in \mathcal{C}$, the decryption process over c using the same key K must give back the original message.

Definition 2.1: Correctness in SKEs

An SKE $\Pi = (\text{Enc}, \text{Dec})$ is said to be correct if $\forall m \in \mathcal{M}, \forall K \in_R \mathcal{K}$ it holds that:

$$\text{Dec}(K, \text{Enc}(K, m)) = m$$

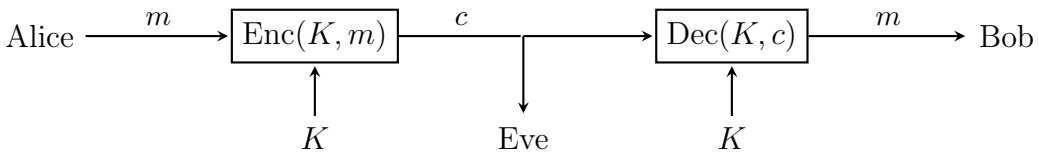

Figure 2.1: Example of SKE with perfect secrecy. Even if Eve intercepts the ciphertext, she cannot obtain the original message since she doesn't know the key.

Originally proposed in the 19th century by the homonym cryptographer, **Kerckhoffs's principle** is a foundational concept in cryptography which asserts that the security of a cryptographic system should depend only on the secrecy of the key. In other words, a system should be secure even if everything about the system is known publicly, except for the secret key. The principle was later succinctly restated by Claude Shannon as "the enemy knows the system".

During his work, Shannon also proposed a formal definition of **perfect secrecy**, i.e. a property that fully respects the concept behind Kerckhoff's principle. Shannon's formulation states that the probability of m being the communicated message is equal to the probability of m being the communicated message even when the corresponding ciphertext c is known. In other words, no ciphertext reveals additional information over any message.

Definition 2.2: Perfect secrecy

Let $\Pi = (\text{Enc}, \text{Dec})$ be an SKE. Let M be a random variable over $\mathcal{M}$ and let C be another random variable defined as $C = \text{Enc}(K, M)$, for some key $K \in_R \mathcal{K}$. We say that Π has perfect secrecy when $\forall m \in \mathcal{M}$ and $\forall c \in \mathcal{C}$ with $\Pr[C = c] > 0$ it holds that:

$$\Pr[M = m] = \Pr[M = m \mid C = c]$$

Shannon proved that such definition is achievable by some cryptosystems, but it comes with inheritent practical limitations. Suprisingly, even a simple SKE as the **One Time Pad (OTP)** system has perfect secrecy. In this system we assume that everything is a binary string of the same length, i.e. that $\mathcal{M} = \mathcal{K} = \mathcal{C} = \{0,1\}^n$ for some $n \in \mathbb{N}$. The encryption and decryption functions are defined as follows:

$$\text{Enc}(K, m) = K \oplus m \qquad \text{Dec}(K, c) = K \oplus c$$

By properties of the bit-wise XOR function, it's easy to see that OTP is complete:

$$\text{Dec}(K, \text{Enc}(K, m)) = K \oplus (K \oplus m) = m$$

In order to prove that OTP has perfect secrecy, we start by proving two equivalent definitions of perfect secrecy. In particular, states that for every pair of messages every ciphertext has the same probability of being the output of the encoding function when applied of both messages.

Lemma 2.1: Perfect secrecy (eq. definitions)

Let $\Pi = (\text{Enc}, \text{Dec})$ be an SKE. Let M be a random variable over $\mathcal{M}$ and let C be another random variable defined as $C = \text{Enc}(K, M)$, for some key $K \in_R \mathcal{K}$. The following statements are equivalent:

1. Π has perfect secrecy
2. M and C are independent
3. $\forall m, m' \in \mathcal{M}$ and $\forall c \in \mathcal{C}$ it holds that:

$$\Pr_{K \in_R \mathcal{K}}[\text{Enc}(K, m) = c] = \Pr_{K \in_R \mathcal{K}}[\text{Enc}(K, m') = c]$$

Proof.

1. $\implies$ 2. Suppose that Π has perfect secrecy. Then, through definition of conditional probability we get that:

$$\Pr[M = m] = \Pr[M = m \mid C = c] = \frac{\Pr[M = m, C = c]}{\Pr[C = c]}$$

which implies that:

$$\Pr[M = m] \cdot \Pr[C = c] = \Pr[M = m, C = c]$$

concluding that M and C are independent

2. $\implies$ 3. Suppose that M and C are independent. Fix $m, m' \in \mathcal{M}$ and $c \in \mathcal{C}$. Through event manipulation and independency of M and C we obtain that:

$$\begin{aligned}\Pr_{K \in_R \mathcal{K}}[\text{Enc}(K, m) = c] &= \Pr_{K \in_R \mathcal{K}}[\text{Enc}(K, M) = c \mid M = m] \\ &= \Pr_{K \in_R \mathcal{K}}[C = c \mid M = m] \\ &= \Pr_{K \in_R \mathcal{K}}[C = c]\end{aligned}$$

Proceding in the same way with $\text{Enc}(K, m')$, we conclude that:

$$\Pr_{K \in_R \mathcal{K}}[\text{Enc}(K, m) = c] = \Pr_{K \in_R \mathcal{K}}[\text{Enc}(K, m') = c]$$

3. $\implies$ 1. Assume the third statement holds. Fix $m \in \mathcal{C}$ and $c \in \mathcal{C}$.

Claim: $\Pr[C = c] = \Pr[C = c \mid M = m]$

Proof of the claim. Through the probability sum principle, we get that:

$$\begin{aligned}\Pr[C=c] &= \sum_{m'\in\mathcal{M}} \Pr[C=c, M=m] \\ &= \sum_{m'\in\mathcal{M}} \Pr[C=c \mid M=m'] \cdot \Pr[M=m'] \\ &= \sum_{m'\in\mathcal{M}} \Pr[\mathrm{Enc}(K,M')=c \mid M=m'] \cdot \Pr[M=m'] \\ &= \sum_{m'\in\mathcal{M}} \Pr[\mathrm{Enc}(K,m')=c] \cdot \Pr[M=m']\end{aligned}$$

Through the hypothesis we also get that:

$$\begin{aligned}\Pr[C=c] &= \sum_{m'\in\mathcal{M}} \Pr[\mathrm{Enc}(K,m')=c] \cdot \Pr[M=m'] \\ &= \sum_{m'\in\mathcal{M}} \Pr[\mathrm{Enc}(K,m)=c] \cdot \Pr[M=m'] \\ &= \Pr[\mathrm{Enc}(K,m)=c] \cdot \sum_{m'\in\mathcal{M}} \Pr[M=m'] \\ &= \Pr[\mathrm{Enc}(K,m)=c] \cdot 1 \\ &= \Pr[\mathrm{Enc}(K,M)=c \mid M=m] \\ &= \Pr[C=c \mid M=m]\end{aligned}$$

□

By definition of conditional probability, we know that:

$$\Pr[M=m \mid C=c]\cdot\Pr[C=c] = \Pr[M=m, C=c] = \Pr[C=c \mid M=m]\cdot\Pr[M=m]$$

which implies that:

$$\Pr[M=m] = \frac{\Pr[M=m \mid C=c] \cdot \Pr[C=c]}{\Pr[C=c \mid M=m]}$$

Finally, through the claim we conclude that:

$$\Pr[M=m] = \frac{\Pr[M=m \mid C=c] \cdot \Pr[C=c]}{\Pr[C=c \mid M=m]} = \Pr[M=m \mid C=c]$$

□

Proposition 2.1

OTP has perfect security.

Proof. We'll prove that the third definition of Lemma 2.1 holds for OTP. Fix two messages $m, m' \in \mathcal{M}$ and a ciphertext $c \in \mathcal{C}$. Through definition of the OTP system and by the properties of the XOR function, we have that:

$$\Pr_{K \in_R \mathcal{K}}[\text{Enc}(K, m) = c] = \Pr_{K \in_R \mathcal{K}}[K \oplus m = c] = \Pr_{K \in_R \mathcal{K}}[K = m \oplus c] = 2^{-n}$$

Through the same argument, we also get that:

$$\Pr_{K \in_R \mathcal{K}}[\text{Enc}(K, m') = c] = 2^{-n}$$

□

We'll now show the inheritent limitations of perfect secrecy. The key idea is simple: in order for M and C to be independent, the key cannot be shorter than the message. Otherwise, there will be some ciphertexts that are unreachable by some messages, revealing information on the system.

Theorem 2.1: Shannon's perfect secrecy theorem

Let $\Pi = (\text{Enc}, \text{Dec})$ be a non-trivial perfectly secret SKE. Then, it holds that $|\mathcal{K}| \geq |\mathcal{M}|$

Proof. Suppose that Π is a non-trivial perfectly secret. Fix any $c \in \mathcal{C}$ such that $\Pr[C = c] > 0$ (this is where non-triviality is required). Let $\mathcal{M}'$ be the set of possible decryptions over c, i.e. $\mathcal{M}' = \{\text{Dec}(K, c) \mid K \in \mathcal{K}\}$. We observe that $\mathcal{M}'$ contains at most one decryption for each key (some keys may yield the same decryption). By way of contradiction, suppose that $|\mathcal{K}| < |\mathcal{M}|$. Then, we get that $|\mathcal{M}'| \leq |\mathcal{K}| < |\mathcal{M}|$.

This implies that $\exists m \in \mathcal{M} - \mathcal{M}'$. Thus, there is a message that cannot be the result of applying the decryption function on c, meaning that $\Pr[M = m \mid C = c] = 0$. However, we know that, when no additional information is given, every message is uniform, hence $\Pr[M = m] = \frac{1}{\mathcal{M}}$, contradicting the fact that Π has perfect secrecy. □

2.2 Message authentication codes

We'll now focus on the second key goal of cryptography: *message integrity*. The simplest way to reach such goal is through **Message Authentication Codes (MACs)**, an additional piece of information sent with the message that enables the receiver to assert that the message hasn't been altered.

For now, we'll start with a simple model that doesn't care about secrecy, but only about integrity. This type of MACs use a deterministic **tagging function** (usually implemented as an *hash function*) $\text{Tag} : \mathcal{K} \times \mathcal{M} \to \mathcal{T}$, where $\mathcal{T}$ is the tag space, i.e. the set of all tag strings.

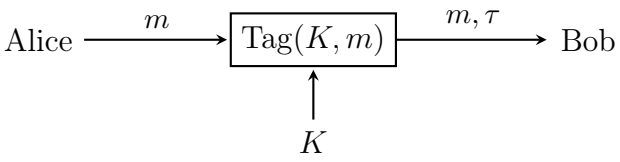

Figure 2.2: Example of an integrity-only MAC.

The idea behind tagging functions is simple: once the message and the tag have been received, Bob can re-compute the tag using the same key-message pair and compare it to the received tag. If the two tags are equal, Bob is sure that the message hasn't been altered. However, this simple idea can only work under the assumption of **unforgeability**:

- It should be hard to forge a valid tag τ for a message m when the key K is not known
- It should be hard to forge a valid tag τ for a message m even when a pair (m', τ') is known

In other words, unforgeability states that no pair should reveal no information about how the tags are computed. Without this property, an adversarial entity may be able to infer information on the shared key and/or the tagging function, using them to forge valid tags. If an entity is capable of forging a valid tag, it may intercept the message-tag pair, alter the message, forge a valid tag for the new message and send a new pair, fooling the receiver. We give a formal definition of a property that reflects this unforgeability aspect.

Definition 2.3: t-time ε-statistical security

We say that a MAC $\Pi = (\text{Tag})$ has t-time ε-statistical security when $\forall m, m_1, \ldots, m_t \in \mathcal{M}$ and $\forall \tau, \tau_1, \ldots, \tau_t \in \mathcal{T}$, where $m \neq m_i$ and $m_i \neq m_j$ for each $i \neq j$, it holds that:

$$\Pr_{K \in_R \mathcal{K}}[\text{Tag}(K, m) = \tau \mid \text{Tag}(K, m_1) = \tau_1, \ldots, \text{Tag}(K, m_t) = \tau_t] \leq \varepsilon$$

In simple terms, the above property states that, even when t message-tag pairs $(m_1, \tau_1), \ldots, (m_t, \tau_t)$ are known, the probability of a message-tag pair (m, τ) being possible is at most ε. Optimally, we want ε to be as small as possible and t to be as large as possible. However, it's easy to see that $\forall t \in \mathbb{N}$ it is impossible to get $\varepsilon = 0$ since a random $\tau \in \mathcal{T}$ always has probability at least $\frac{1}{|\mathcal{T}|}$ of being correct. Just as we did with perfect secrecy, we'll show that the notion of good statistical security is achievable, but it's highly inefficient in terms of key size.

Theorem 2.2

Any t-time $2^{-\lambda}$-statistically secure MAC must have a key of size $(t+1)\lambda$

Proof. Omitted. □

A good enough 1-time statistically secure MAC is achievable through **pairwise independent hash functions**, i.e. a family of hash functions where each pair of functions forms a pair of independent random variables. This idea can also be expressed through joint uniform distributions, as in the below definition.

Definition 2.4: Pairwise independent hash functions

Let $\mathcal{H} = \{h_K : \mathcal{M} \to \mathcal{T}\}_{K \in \mathcal{K}}$ be a family of hash functions. We say that $\mathcal{H}$ is pairwise independent if $\forall m, m' \in \mathcal{M}$ with $m \neq m'$ it holds that the distribution $(h_K(m), h_K(m'))$ is uniform over $\mathcal{T} \times \mathcal{T}$ when $K \in_R \mathcal{K}$.

Note: $h_K(m)$ and $h_K(m')$ denote two random variables in this context.

Theorem 2.3

Let $\mathcal{H} = \{h_K : \mathcal{M} \to \mathcal{T}\}_{K \in \mathcal{K}}$ be a family of pairwise indepependent hash functions. Then, $\mathcal{H}$ induces a 1-time $\frac{1}{|\mathcal{T}|}$-statistically secure MAC.

Proof. Fix $m \in \mathcal{M}$ and $\tau \in \mathcal{T}$. Let $\Pi = (\text{Tag})$ be the MAC defined as $\text{Tag}(K, m) = h_K(m)$. Since the joint probability of each pair of hash functions in $\mathcal{H}$ is pairwise uniformly distributed, we get that each hash function is also individually uniformly distributed. Hence, we derive that:

$$\Pr[\text{Tag}(K, m) = \tau] = \Pr[h_K(m) = \tau] = \frac{1}{|\mathcal{T}|}$$

Similarly, by pairwise independence $\forall m, m' \in \mathcal{M}$ with $m \neq m'$ and $\forall \tau, \tau' \in \mathcal{T}$ it holds that:

$$\begin{aligned}
\Pr_{K \in_R \mathcal{K}}[\text{Tag}(K, m') = \tau', \text{Tag}(K, m) = \tau] &= \Pr_{K \in_R \mathcal{K}}[\text{Tag}(K, m') = \tau'] \cdot \Pr_{K \in_R \mathcal{K}}[\text{Tag}(K, m) = \tau] \\
&= \Pr_{K \in_R \mathcal{K}}[h_K(m') = \tau'] \cdot \Pr_{K \in_R \mathcal{K}}[h_K(m) = \tau] \\
&= \frac{1}{|\mathcal{T}|} \cdot \frac{1}{|\mathcal{T}|}
\end{aligned}$$

Putting the two results together we get that:

$$\Pr[\text{Tag}(K, m') = \tau' \mid \text{Tag}(K, m) = \tau] = \frac{\Pr_{K \in_R \mathcal{K}}[\text{Tag}(K, m') = \tau', \text{Tag}(K, m) = \tau]}{\Pr[\text{Tag}(K, m) = \tau]} = \frac{1}{|\mathcal{T}|}$$

□

After proving that pairwise independent hash function families form a good enough MAC, we're left with proving that such families exist. Suprisingly, these families can be easily constructed through prime numbers.

Proposition 2.2

Given a prime $p \in \mathbb{P}$, let $\mathcal{M} = \mathcal{T} = \mathbb{Z}_p$ and $\mathcal{K} = \mathbb{Z}_p^2$. Then, the family $\mathcal{H} = \{h_{(a,b)}\}_{(a,b)\in\mathbb{Z}_p^2}$ where $h_{(a,b)}(m) = am + b \pmod p$ is pairwise independent.

Proof. Fix $m, m' \in \mathbb{Z}_p$ with $m \neq m'$ and $\tau, \tau' \in \mathbb{Z}_p$. We'll show that the joint probability over the hash functions is uniform. First, we observe that:

$$\begin{aligned}\Pr_{(a,b)\in_R\mathbb{Z}_p^2}[h_{(a,b)}(m) = \tau, h_{(a,b)}(m') = \tau'] &= \Pr_{(a,b)\in_R\mathbb{Z}_p^2}[am + b = \tau, am' + b = \tau'] \\ &= \Pr_{(a,b)\in_R\mathbb{Z}_p^2}\left[\begin{pmatrix} m & 1 \\ m' & 1 \end{pmatrix}\begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} \tau \\ \tau' \end{pmatrix}\right]\end{aligned}$$

Since $m \neq m'$, we know that $\det \begin{pmatrix} m & 1 \\ m' & 1 \end{pmatrix} = m - m' \neq 0$, thus the matrix is invertible. This allows us to further manipulate the event and rewrite it in terms of the key:

$$\begin{aligned}\Pr_{(a,b)\in_R\mathbb{Z}_p^2}[h_{(a,b)}(m) = \tau, h_{(a,b)}(m') = \tau'] &= \Pr_{(a,b)\in_R\mathbb{Z}_p^2}\left[\begin{pmatrix} m & 1 \\ m' & 1 \end{pmatrix}\begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} \tau \\ \tau' \end{pmatrix}\right] \\ &= \Pr_{(a,b)\in_R\mathbb{Z}_p^2}\left[\begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} m & 1 \\ m' & 1 \end{pmatrix}^{-1}\begin{pmatrix} \tau \\ \tau' \end{pmatrix}\right] \\ &= \frac{1}{\left|\mathbb{Z}_p^2\right|}\end{aligned}$$

concluding that $\mathcal{H}$ is pairwise independent. □

2.3 Randomness extraction

We discussed how randomness can be used to generate secret keys over a probability distribution, but how a random value be generated? As we will discuss later, randomness is a crucial concept for secure cryptography. Clearly, there is no such concept as *real randomness*: even if the whole universe may appear chaotic, everything is technically deterministic. The process of generating a random-enough value is called **randomness extraction** and it is typically achieved by measuring physical quantities (noise, air humidity, ...) in order to produce a short unpredictable sequence of bits (e.g. 256 bits), which is expensive to generate and not necessarily uniform. This short "truly random" sequence gets usually expanded to any desired length – as long as it is polynomial with respect to the original lenght – through the use of a **pseudorandom generator (PRG)**. However, this process requires some strict computational assumptions, which we'll discuss later.

For now, we'll focus on understanding how to extract randomness from an unpredictable secure variable X. The first **extractor** that sparked the idea is Von Neumann's Extractor, which yields a fair random coin from an unpredictable unfair one. Let $B \in \{0, 1\}$ be the

random variable describing the unpredictable unfair coin, where $\Pr[B = 0] = p < \frac{1}{2}$. Let $Y \in \{0, 1\}$ be the random variable describing our new coin. The value of Y is determined by the following procedure. Sample two values b_1, b_2 from B at different times. If $b_1 = b_2$, Y assumes no value (marked as $Y =?$) and we repeat the sampling process. If not, $Y = 1$ if $b_1 = 0$ and $b_2 = 1$, otherwise $Y = 0$ if $b_1 = 1$ and $b_2 = 0$.

If the sampling process succeeds, i.e. Y assumes a value, we have that $\Pr[Y = 0] = p(1-p)$ and $\Pr[Y = 1] = (1 - p)p$, thus $\Pr[Y = 0] = \Pr[Y = 1]$. Moreover, we observe that the probability of $Y == ?$ being true for m consecutive tries is at most:

$$\Pr[Y =? \text{ for } m \text{ tries}] = (\Pr[Y == ?])^m = (1 - \Pr[Y = 0 \cup Y = 1])^m \leq (1 - 2p(1 - p))^m$$

As m grows to infinity, the latter probability goes to 0, making the even negligible. Implying that $\Pr[Y = 0]$ and $\Pr[Y = 1]$ tend to be $\frac{1}{2}$ due to them having the same probability and them being the only two outcomes. This makes Y a fair enough coin.

Our goal is to generalize this concept to any desider value, i.e. design an extractor Ext that uses a random variable X to output any desided uniform distribution $\text{Ext}(X)$. A good eye may recognize that this is clearly impossible to achieve unless the source is already truly unpredictable, which makes the extractor useless since we already have a truly random source. As for many computational and probabilistic concepts, we can only hope to achieve something that is good-enough for our purposes. This goodness is measured through a concept known as **min-entropy**, that being the largest value m having the property that each observation of X provides at least m bits of information.

Definition 2.5: Min-entropy

Given a random variable X, we define the min-entropy of X as:

$$H_\infty(X) = -\log \max_x \Pr[X = x]$$

One way to justify the name of this operator is to compare it with the more standard definition of *entropy*, i.e. the expected value of $\log\left(\frac{1}{p_i}\right)$ over a distribution P:

$$H(P) = \sum_i p_i \log\left(\frac{1}{p_i}\right)$$

In the min-entropy, instead, we take take the minimum value of $\log\left(\frac{1}{p_i}\right)$, where:

$$\min_i \log\left(\frac{1}{p_i}\right) = \log\left(\min_i \frac{1}{p_i}\right) = -\log \max_i p_i$$

Consider a random variable $X \sim \mathcal{U}_n$, where $\mathcal{U}_n$ is an uniform distribution over $\{0, 1\}^n$. In this case, we have that $\Pr[X = x] = 2^{-n}$ for all value x assumable by X. Hence, the min-entropy is:

$$H_\infty(X) = -\log \max_x \Pr[X = x] = -\log(2^{-n}) = n$$

This concludes that each observation of X provides at least n bits of information. Consider now a constant random variable X', i.e. $\Pr[X' = x^*] = 1$ for some fixed value x^* and $\Pr[X' = x] = 0$ for every $x^* \neq x$. It's easy to see that:

$$H_\infty(X') = -\log \max_{x'} \Pr[X' = x] = -\log 1 = 0$$

Concluding that each observation of X' provides at least 0 bits of information. This information-theoretic measure opens a new question regarding extractors: is there an extractor Ext^* that for any random variable X outputs an uniform distribution $Y = \text{Ext}(X)$ such that $H_\infty(X) \geq k$ for some value $k > 0$? Even under these constraints, the answer to this question is still negative. In particular, it's impossible to define such extractor even if we restrict our interest to extracting only one bit while revealing at least one bit less than the input length.

Proposition 2.3

There is no extractor Ext such that for every random variable X over $\{0,1\}^n$ with $H_\infty(X) \geq n - 1$ it holds that $\text{Ext}(X)$ is a uniform distribution over $\{0,1\}$.

Proof. Let $\text{Ext} : \{0,1\}^n \to \{0,1\}$ be any extractor and let $b \in \{0,1\}$ be the output maximizing the cardinality of the preimage of the extractor, i.e. the set of inputs for which the extractor outputs b.

$$b = \arg\max_{b' \in \{0,1\}} \left|\text{Ext}^{-1}(b)\right|$$

By the pidgeonhole principle, we have that $\left|\text{Ext}^{-1}(b)\right| \geq \frac{|\{0,1\}^n|}{2} = 2^{n-1}$. Let X be a random variable uniform over $\text{Ext}^{-1}(b)$. Since X is uniform, we have that $H_\infty(X) \geq n-1$. However, we know that $\text{Ext}(X)$ isn't uniform due to the output being always b. This concludes that any extractor has always a bad input uniform random variable X that returns a non-uniform distribution $\text{Ext}(X)$. □

Since we can't define an extractor that yields an uniform distribution, the best we can hope for is a distribution that is close enough to an uniform one. We use a standard measure for distribution similarity, called **ε-closeness**.

Definition 2.6: ε-closeness

Let X, X' be two random variables defined over the same set. We say that X and X' are ε-close, written as $X \sim_\varepsilon X'$ when their *statistical distance* $\text{SD}(X; X')$ is at most ε, where:

$$\text{SD}(X; X') = \frac{1}{2} \sum_x |\Pr[X = x] - \Pr[X' = x]|$$

The concept of ε-closeness between two random variables is equivalent to saying that every *unbounded adversary* A, i.e. an entity with unlimited computational power that

wants to break our system, cannot distinguish whether a value x has been sampled from X or X'.

$$|\Pr[A(x) = 1 : x \in_R X] - \Pr[A(x) = 1 : x \in_R X']| \leq \varepsilon$$

Definition 2.7: Deterministic extractor

Let S be a random variable, referred to as *seed*. We say that $\text{Ext} : \{0,1\}^d \times \{0,1\}^n \to \{0,1\}^\ell$ is a (k, ε)-extractor if for every random variable X with $H_\infty(X) \geq k$ it holds that $(S, \text{Ext}(S, X)) \sim_\varepsilon (S, \mathcal{U}_\ell)$ when $S \sim \mathcal{U}_d$.

We notice that the condition $(S, \text{Ext}(S, X)) \sim_\varepsilon (S, \mathcal{U}_\ell)$ implies that the seed must be *public*. This requirement is forced in order to avoid trivial extractors such as $\text{Ext}(S, X) = S$. The idea is to use a source of randomness with high min-entropy in order to force that the output of a "good" hash function is statistically close to uniform – even if the input seed wasn't. This result is known as the **left-over hash lemma**.

The goodness of the hash functions is measured in terms of **low collision probability**, defined as the expected value that a random variable Y over a set $\mathcal{Y}$ and a copy Y' of Y, giving two i.i.d (identical and independend distributions) variables, return the same result:

$$\text{Col}(Y) = \Pr[Y = Y']$$

In particular, we observe that since Y and Y' are i.i.d. we get that:

$$\text{Col}(Y) = \Pr[Y = Y'] = \sum_{y \in \mathcal{Y}} \Pr[Y = y, Y' = y] = \sum_{y \in \mathcal{Y}} \Pr[Y = y]^2$$

Before proving the left-over hash lemma, we prove the following relationship commonly used in statistical analysis.

Proposition 2.4

Let Y be a random variable over a set $\mathcal{Y}$ and assume $\text{Col}(Y) = \frac{1}{|\mathcal{Y}|}(1 + 4\varepsilon^2)$ for some $\varepsilon \in \mathbb{R}_{>0}$. Then, it holds that $\text{SD}(Y, U) \leq \varepsilon$, where U is the uniform distribution over $\mathcal{Y}$.

Proof. We start by observing that:

$$\text{SD}(Y; U) = \frac{1}{2} \sum_{y \in \mathcal{Y}} |\Pr[Y = y] - \Pr[U = y]| = \frac{1}{2} \sum_{y \in \mathcal{Y}} \left| \Pr[Y = y] - \frac{1}{\mathcal{Y}} \right|$$

For each $y \in \mathcal{Y}$, fix $q_y = \Pr[Y = y] - \frac{1}{|Y|}$ and $s_y = \text{sign}(q_y)$. Let $\overrightarrow{q} = \begin{bmatrix} q_{y_1} & \cdots & q_{y_{|Y|}} \end{bmatrix}$ and $\overrightarrow{s} = \begin{bmatrix} s_{y_1} & \cdots & s_{y_{|Y|}} \end{bmatrix}$. We notice that:

$$\text{SD} = \frac{1}{2} \sum_{y \in \mathcal{Y}} \left| \Pr[Y = y] - \frac{1}{\mathcal{Y}} \right| = \frac{1}{2} \sum_{y \in \mathcal{Y}} q_y s_y = \frac{1}{2} \langle \overrightarrow{q}, \overrightarrow{s} \rangle$$

By the *Cauchy-Swartz inequality* (which we won't prove), we get that:

$$\mathrm{SD}(Y;U) = \frac{1}{2}\left\langle \vec{q}, \vec{s} \right\rangle \leq \frac{1}{2}\sqrt{\left\langle \vec{q}, \vec{q} \right\rangle \left\langle \vec{s}, \vec{s} \right\rangle} = \frac{1}{2}\sqrt{\left(\sum_{y\in\mathcal{Y}} q_y^2\right)|\mathcal{Y}|}$$

Claim: $\sum_{y\in\mathcal{Y}} q_y^2 \leq \frac{4\varepsilon^2}{|Y|}$

Proof of the claim. Through algebraic manipulation we get that:

$$\begin{aligned}
\sum_{y\in\mathcal{Y}} q_y^2 &= \sum_{y\in\mathcal{Y}} \left(\Pr[Y=y] - \frac{1}{\mathcal{Y}}\right)^2 \\
&= \sum_{y\in\mathcal{Y}} \left(\Pr[Y=y]^2 - \frac{2}{|\mathcal{Y}|}\Pr[Y=y] + \frac{1}{|\mathcal{Y}|^2}\right) \\
&= \sum_{y\in\mathcal{Y}} \Pr[Y=y]^2 - \sum_{y\in\mathcal{Y}} \frac{2}{|\mathcal{Y}|}\Pr[Y=y] + \sum_{y\in\mathcal{Y}} \frac{1}{|\mathcal{Y}|^2} \\
&= \sum_{y\in\mathcal{Y}} \Pr[Y=y]^2 - \frac{2}{|\mathcal{Y}|} + \frac{1}{|\mathcal{Y}|} \\
&= \mathrm{Col}(Y) - \frac{1}{\mathcal{Y}}
\end{aligned}$$

By hypothesis, we know that the collision probability of Y is bounded, concluding that:

$$\sum_{y\in\mathcal{Y}} q_y^2 = \mathrm{Col}(Y) - \frac{1}{\mathcal{Y}} \leq \frac{1}{|\mathcal{Y}|}(1+4\varepsilon^2) - \frac{1}{\mathcal{Y}} = \frac{4\varepsilon^2}{|\mathcal{Y}|}$$

□

Through the claim we directly conclude that:

$$\mathrm{SD}(Y;U) = \frac{1}{2}\sqrt{\left(\sum_{y\in\mathcal{Y}} q_y^2\right)|\mathcal{Y}|} \leq \frac{1}{2}\sqrt{\frac{4\varepsilon^2}{|\mathcal{Y}|}|Y|} = \varepsilon$$

□

Lemma 2.2: Left-over hash lemma

Let $\mathcal{H} = \{h_S : \{0,1\}^n \to \{0,1\}^\ell\}_{S\in\{0,1\}^d}$ be a pairwise independent hash function. Let X be a random variable with $H_\infty(X) \geq k$. Then, for each $S \in \{0,1\}^d$ it holds that $\mathrm{Ext}(S,X) = h_S(X)$ is a (k,ε)-extractor for $k \geq \ell + 2\log\left(\frac{1}{\varepsilon}\right) - 2$

Proof. Fix $S \in \{0,1\}^d$ and two random variables X, X'. Let $Y = (S, \text{Ext}(S, X))$ and let Y' be a copy of Y defined as and $Y' = (S', \text{Ext}(S', X'))$. We observe that:

$$\begin{aligned}\text{Col}(Y) &= \Pr[Y = Y'] \\ &= \Pr[S = S', \text{Ext}(S, X) = \text{Ext}(S', X')] \\ &= \Pr[S = S', h_S(X) = h_{S'}(X')]\end{aligned}$$

Since S and S' are independend from X and X', we get that:

$$\begin{aligned}\text{Col}(Y) &= \Pr[S = S', h_S(X) = h_{S'}(X')] \\ &= \Pr[S = S', h_S(X) = h_S(X')] \\ &= \Pr[S = S'] \cdot \Pr[h_S(X) = h_S(X')] \\ &= 2^{-d} \cdot \Pr[h_S(X) = h_S(X')] \\ &= 2^{-d} \cdot (\Pr[X = X', h_S(X) = h_S(X')] + \Pr[X \neq X', h_S(X) = h_S(X')]) \\ &= 2^{-d} \cdot (\Pr[X = X'] + \Pr[X \neq X', h_S(X) = h_S(X')]) \\ &= 2^{-d} \cdot (\Pr[X = X'] + 2^{-\ell})\end{aligned}$$

By hypothesis, we know that $H_\infty(X) \geq k$. This implies that $\Pr[X = X'] \leq 2^{-k}$ since an adversary cannot guess whether a sample comes from X or X' with probability greater than 2^{-k}.

$$\begin{aligned}\text{Col}(Y) &= 2^{-d} \cdot (\Pr[X = X'] + 2^{-\ell}) \\ &= 2^{-d} \cdot (2^{-k} + 2^{-\ell}) \\ &= \frac{1}{2^{d+\ell}}(2^{\ell-k} + 1)\end{aligned}$$

Finally, by hypothesis we have conclude that:

$$\text{Col}(Y) \leq \frac{1}{2^{d+\ell}}(2^{\ell-k} + 1) \leq \frac{1}{2^{d+\ell}}\left(2^{2-2\log\left(\frac{1}{\varepsilon}\right)} + 1\right) = \frac{4\varepsilon^2 + 1}{|\mathcal{Y}|}$$

□

2.4 Solved exercises

Problem 2.1

Prove or disprove: a SKE $\Pi = (\text{Enc}, \text{Dec})$ has perfect secrecy if and only if $\forall c_0, c_1 \in \mathcal{C}$ it holds that $\Pr[C = c_0] = \Pr[C = c_1]$.

Proof. The claim is false. To disprove it, consider the SKE Π such that $\mathcal{M} = \mathcal{K} = \{0,1\}^n$ and $\mathcal{C} = \{0,1\}^{n+1}$, where:

$$\text{Enc}(K, m) = 0 \,||\, (m \oplus K) \qquad \text{Dec}(K, b \,||\, \widehat{c}) = \widehat{c} \oplus K$$

Let $(*)$ be the special property of the claim. We observe that $\forall \widehat{c} \in \{0,1\}^n$ it holds that $\Pr[C = 1 \mid\mid \widehat{C}] = 0$, while $\Pr[C = 0 \mid\mid \widehat{C}] > 0$, implying that $(*)$ doesn't hold. Nonetheless, we can easily prove that Π has perfect secrecy.

Fix any pair $m, c \in \mathcal{M} \times \mathcal{C}$ such that $\Pr[C = c] > 0$. Since $\Pr[C = c] > 0$, it must hold that $c = 0 \mid\mid \widehat{c}$ for some $\widehat{c} \in \{0,1\}^n$. Therefore, we have that:

$$\Pr[M = m \mid C = c] = \Pr[M = m \mid C = 0 \mid\mid \widehat{c}] = \Pr[M = m]$$

since $c = 0 \mid\mid \widehat{c}$ doesn't reveal any additional information for the plaintext due to how Enc is defined. □

Problem 2.2

Prove that a SKE $\Pi = (\text{Enc}, \text{Dec})$ has perfect secrecy if and only if for every unbounded adversary A it holds that $\text{Game}^{\text{1-time}}_{\Pi,A}(\lambda, 0) \equiv \text{Game}^{\text{1-time}}_{\Pi,A}(\lambda, 1)$

Proof. We observe that $\text{Game}^{\text{1-time}}_{\Pi,A}(\lambda, 0) \equiv \text{Game}^{\text{1-time}}_{\Pi,A}(\lambda, 1)$ if and only if for all adversaries A' it holds that:

$$\left|\Pr[A'(b') = 1 : b' \leftarrow \text{Game}^{\text{1-time}}_{\Pi,A}(\lambda, 0)] - \Pr[A'(b') = 1 : b' \leftarrow \text{Game}^{\text{1-time}}_{\Pi,A}(\lambda, 1)]\right| = 0$$

$$\iff \Pr[A'(b') = 1 : b' \leftarrow \text{Game}^{\text{1-time}}_{\Pi,A}(\lambda, 0)] = \Pr[A'(b') = 1 : b' \leftarrow \text{Game}^{\text{1-time}}_{\Pi,A}(\lambda, 1)] = \frac{1}{2}$$

This can happen if and only if for all choices $m_0, m_1 \in \mathcal{M}$ made by any adversary A and for all possible answers $c^* \in \mathcal{C}$ that may be returned by the challenger it holds that $\Pr[\text{Enc}(K, m_0) = c^*] = \Pr[\text{Enc}(K, m_1) = c^*]$, which is exactly the third equivalent definition of perfect secrecy. □

Problem 2.3

Given a prime $p \in \mathbb{P}$, let $\mathcal{M} = \mathcal{T} = \mathbb{Z}_p$ and $\mathcal{K} = \mathbb{Z}_p^2$. Prove that the hash family $\mathcal{H} = \{h_{(a,b)}\}_{(a,b)\in\mathbb{Z}_p^2}$, where $h_{(a,b)}(m) = am + b \pmod p$, cannot be a 2-time statistically secure MAC

Solution. Suppose that the values $h_{(a,b)}(m_1) = \tau_1$ and $h_{(a,b)}(m_2) = \tau_2$, where $m_1 \neq m_2$, are known to an adversary. Hence, the adversary knows that:

$$\begin{cases} am_1 + b \equiv \tau_1 \pmod p \\ am_2 + b \equiv \tau_2 \pmod p \end{cases} \implies a(m_1 - m_2) \equiv \tau_1 - \tau_2 \pmod p$$

Since $m_1 \neq m_2$, the value $m_1 - m_2$ is invertible in $\mathbb{Z}_p$, the value $Q \equiv (\tau_1 - \tau_2)(m_1 - m_2)^{-1} \pmod p$ (notice that $a \equiv Q \pmod p$) can be easily computed. Once Q is known, b can be easily computed as $b \equiv \tau_1 - Qm_1 \pmod p$. Therefore, we have that:

$$\Pr_{K\in_R\mathbb{Z}_p}[h_K(m) = \tau \mid h_K(m_1) = \tau_1, h_K(m_2) = \tau_2] = \begin{cases} 1 & \text{if } \tau \equiv Q(m - m_1) + \tau_1 \pmod p \\ 0 & \text{otherwise} \end{cases}$$

making $\mathcal{H}$ statistically insecure for every value $0 < \varepsilon < 1$.

Problem 2.4

Construct a 3-wise independent hash function family and prove it's correctness.

Solution. Given a prime $p \in \mathbb{P}$, let $\mathcal{M} = \mathcal{T} = \mathbb{Z}_p$ and $\mathcal{K} = \mathbb{Z}_p^3$. Consider the hash family $\mathcal{H} = \{h_{(a,b,c)}\}_{(a,b,c)\in\mathbb{Z}_p^3}$, where $h_{(a,b,c)}(m) = am^2 + bm + c \pmod p$.

Fix three values $x_1, x_2, x_3 \in \mathbb{Z}_p$ such that $x_i \neq x_j$ for all $i \neq j$. Fix $\tau_1, \tau_2, \tau_3 \in \mathbb{Z}_p$. We observe that:

$$\begin{aligned}\Pr_{(a,b,c)\in_R\mathbb{Z}_p^3}\begin{bmatrix} h_{(a,b,c)}(x_1) = \tau_1 \\ h_{(a,b,c)}(x_2) = \tau_2 \\ h_{(a,b,c)}(x_3) = \tau_3 \end{bmatrix} &= \Pr_{(a,b,c)\in_R\mathbb{Z}_p^3}\begin{bmatrix} ax_1^2 + bx_1 + c = \tau_1 \\ ax_2^2 + bx_2 + c = \tau_2 \\ ax_3^2 + bx_3 + c = \tau_3 \end{bmatrix} \\ &= \Pr_{(a,b,c)\in_R\mathbb{Z}_p^3}\left[\begin{pmatrix} x_1^2 & x_1 & 1 \\ x_2^2 & x_2 & 1 \\ x_3^2 & x_3 & 1 \end{pmatrix}\begin{pmatrix} a \\ c \\ b \end{pmatrix} = \begin{pmatrix} \tau_1 \\ \tau_2 \\ \tau_3 \end{pmatrix}\right]\end{aligned}$$

Since $x_i \neq x_j$ for all $i \neq j$, we observe that the matrix is invertible:

$$\begin{aligned}\det\begin{pmatrix} x_1^2 & x_1 & 1 \\ x_2^2 & x_2 & 1 \\ x_3^2 & x_3 & 1 \end{pmatrix} &= x_1^2 \det\begin{pmatrix} x_2 & 1 \\ x_3 & 1 \end{pmatrix} - x_2^2 \det\begin{pmatrix} x_1 & 1 \\ x_3 & 1 \end{pmatrix} + x_3^2 \det\begin{pmatrix} x_1 & 1 \\ x_2 & 1 \end{pmatrix} \\ &= x_1^2x_2 - x_1^2x_3 - x_2^2x_1 + x_2^2x_3 + x_3^2x_1 - x_3^2x_2 \\ &= (x_1 - x_2)(x_1 - x_3)(x_2 - x_3) \\ &\neq 0\end{aligned}$$

concluding that $\mathcal{H}$ is 3-wise independent:

$$\begin{aligned}\Pr_{(a,b,c)\in_R\mathbb{Z}_p^3}\begin{bmatrix} h_{(a,b,c)}(x_1) = \tau_1 \\ h_{(a,b,c)}(x_2) = \tau_2 \\ h_{(a,b,c)}(x_3) = \tau_3 \end{bmatrix} &= \Pr_{(a,b,c)\in_R\mathbb{Z}_p^3}\left[\begin{pmatrix} x_1^2 & x_1 & 1 \\ x_2^2 & x_2 & 1 \\ x_3^2 & x_3 & 1 \end{pmatrix}\begin{pmatrix} a \\ c \\ b \end{pmatrix} = \begin{pmatrix} \tau_1 \\ \tau_2 \\ \tau_3 \end{pmatrix}\right] \\ &= \Pr_{(a,b,c)\in_R\mathbb{Z}_p^3}\left[\begin{pmatrix} a \\ c \\ b \end{pmatrix} = \begin{pmatrix} x_1^2 & x_1 & 1 \\ x_2^2 & x_2 & 1 \\ x_3^2 & x_3 & 1 \end{pmatrix}^{-1}\begin{pmatrix} \tau_1 \\ \tau_2 \\ \tau_3 \end{pmatrix}\right] \\ &= \frac{1}{|\mathbb{Z}_p^3|}\end{aligned}$$

3

Computational security

3.1 One-way functions and Impagliazzo's worlds

In the previous chapter we discussed how, without computational assumptions, symmetric encryption and random generation can be achieved with some strong limitations. For privacy and integrity, we saw how the message length and the key length must be at least equal in order to guarantee a 1-time security (no guarantees for t-time security with $t > 1$). For randomness, instead, we saw how we can't extract more than k bits when the min-entropy is k.

Our next objective is to overcome all these limitation (or at least reduce them). This can be achieved at the price of two very strong assumptions:

1. The adversary is *computationally bounded*, i.e. it has limited resources
2. There are *hard* problems

In other words, we'll prove results of the following form. "if problem X is hard against efficient solvers then cryptosystem Π is secure against efficient adversary". The direct consequence of these results is their contrapositive: if the system Π were ever to be proved insecure, there would be an efficient solution for problem X. Depending on the "level of hardness" of problem X, this would have groundbreaking consequences (e.g. X could be a problem that implies $\mathsf{P} = \mathsf{NP}$, factoring is easy, discrete-log is easy, ...).

Someone may ask "can't we just assume that $\mathsf{P} \neq \mathsf{NP}$ and prevent all of this?" The answer is *yes*, we could, but this assumption would be *too weak* for our necessities. In fact, to achieve good security we require something that isn't directly implied by $\mathsf{P} \neq \mathsf{NP}$: the existence of **one-way functions (OWF)**. In simple terms, a functions $f : \{0,1\}^n \to \{0,1\}^m$ is said to be *one-way* when it can be efficiently computed but hard to invert (meaning that f^{-1} cannot be efficiently computed).

It's easy to see that assuming the existence of OWFs implies that $\mathsf{P} \neq \mathsf{NP}$. This is due to the contrapositive statement: if $\mathsf{P} = \mathsf{NP}$ then OTWs cannot exist since for each function f we can efficiently verify if $f(x) = y$ in order to compute $f^{-1}(y) = x$. However, the

converse statement doesn't hold: assuming $\mathsf{P} \neq \mathsf{NP}$ could still imply that OWFs don't exist. In the article A Personal View of Average-Case Complexity presented at the 1995 Complexity Conference, Russell Impagliazzo describes **five possible worlds** that we may be living in and their implications to computer science:

- *Algorithmica*: there are no hard problems ($\mathsf{P} = \mathsf{NP}$) and OWFs cannot exist.
- *Heuristica*: there are hard problems ($\mathsf{P} \neq \mathsf{NP}$) but can they solved efficiently on average
- *Pessiland*: there are hard problems ($\mathsf{P} \neq \mathsf{NP}$) but OWFs don't exist
- *Minicrypt*: one-way functions exist but public-key cryptography is impossible
- *Cryptomania*: one-way functions exist and public-key cryptography is possible

Impagliazzo does not guess which world we live in, but describes *Pessiland* as the worst one out of all possible worlds: in this scenario, not only can we not solve hard problems on average but we apparantly do not get any cryptographic advantage from the hardness of these problems. Today, most computer scientists believe (and hope) that our world corresponds to either *Cryptomania* or *Minicrypt*.

All the results will be discussed with respect to a fixed standard computational model: the *Turing machine* (TM). From now on, *efficient computation* will refer to a computation made under a polynomial amount of steps with respect to the input size, i.e. if the input x has length n then poly(n)-time referres to at most n^c steps, for some $c \in \mathbb{N}$.

As mentioned before, the adversary we'll also assume that the adversary has limited resources. However, we'll be generous with them: the adversary is allowed to use a poly(n) amount of time and a poly(n) amount of *random bits*. This implies that their computational model is a *probabilistic polynomial time (PPT)* TM. To formally define the concept of OWFs, we'll use **negligible functions**, i.e. functions $\varepsilon : \mathbb{N} \to \mathbb{R}$ such that $\forall c \in \mathbb{N}$ there is a value $\lambda_0 \in \mathbb{N}$ for which $\forall \lambda > \lambda_0$ it holds that

$$\varepsilon(\lambda) < \frac{1}{\lambda^c}$$

Definition 3.1: One-way functions

We say that a function $f : \{0,1\}^n \to \{0,1\}^m$ is a one-way function if:

- f is computable in poly(n)-time
- for every PPT algorithm A there is a negligible function negl(n) such that:

$$\Pr_{x \in_R \mathcal{U}_n}[f(x') = y : y = f(x); x' \leftarrow A(y)] \leq \text{negl}(n)$$

Note: $x \leftarrow A(y)$ is read as "x is the output of $A(y)$"

3.2 Pseudorandom generators

In the previous chapter we discussed how randomness is essential for good cryptography, even though true randomness cannot be achieved. To get a good enough approximation of true randomness, modern systems use **pseudo-randomness**, i.e. efficient deterministic algorithms that generate a sequence of bits that look unpredictable but aren't truly random. By definition, if enough time and resources are permitted an adversary is clearly able to learn how a pseudorandom algorithm work through bruteforce. Hence, we'll restrict that our adversary doesn't have such time and resources, i.e. their computation must also be efficient.

The idea behind pseudo-randomness is to generate a "non-random" probability distribution (this doesn't really mean anything, but you get the idea) that is **computationally indistinguishable** from a truly random probability distribution. The formal definition of computational indistinguishability refers to *probability ensembles*, i.e. infinite sequences of probability distributions $\mathcal{X} = \{X_n\}_{n\in\mathbb{N}}$, where each X_n is a distribution over $\{0,1\}^n$. This multi-length definition takes into account the fact that the distribution may change over the input length.

Definition 3.2: Computational indistinguishability

Let $\mathcal{X} = \{X_n\}_{n\in\mathbb{N}}$ and $\mathcal{Y} = \{Y_n\}_{n\in\mathbb{N}}$ be two probability ensambles. We say that $\mathcal{X}$ and $\mathcal{Y}$ are computationally indistinguishable, written as $\mathcal{X} \approx_c \mathcal{Y}$, if for every PPT algorithm D there is a negligible function $\text{negl}(n)$ such that $\forall n \in \mathbb{N}$ it holds that:

$$|\Pr[D(z) = 1 : z \in_R X_n] - \Pr[D(z) = 1 : z \in_R Y_n]| \leq \text{negl}(n)$$

We observe that we already discussed a definition similar to the one above when we talked about ε-closeness. In fact, this can be seen as a practical computational counterpart of that definition (notice how the adversary is now restricted to PPT algorithms and the distributions vary over input lengths). Moreover, it's esy to see that the relation $\approx_c$ is *transitive*, i.e. if $\mathcal{X} \approx_c \mathcal{Y}$ and $\mathcal{Y} \approx_c \mathcal{Z}$ then $\mathcal{X} \approx_c \mathcal{Z}$, and closed under *reductions*, i.e. for any PPT function f if $\mathcal{X} \approx_c \mathcal{Y}$ then $f(\mathcal{X}) \approx_c f(\mathcal{Y})$.

Definition 3.3: Pseudorandom generator (PRG)

Let $G : \{0,1\}^n \to \{0,1\}^{n+\ell}$ be a function where $\ell \geq 1$. We say that G is a pseudorandom generator (PRG) with expansion factor ℓ if:

- G can be computed in $\text{poly}(n)$-time
- $G(\mathcal{U}_n) \approx_c \mathcal{U}_{n+\ell}$.

The above definition requires a bit of discussion in order to be fully understood. First, we observe that G is deterministic, making $G(\mathcal{U}_n) \approx_c \mathcal{U}_{n+\ell}$ a very strong property. Then, we observe that the idea behind a PRG is to generate new bits that look like they are truly random, as long as the initial seed is already truly random (in practice, we use

a seed that is computationally indistinguishable from a truly random one). Hence, we can use a randomness extractor to get an initial seed and then pass it to a PRG in order to "expand the seed" up to any desired length: after constructing a simple PRG $G : \{0,1\}^n \to \{0,1\}^{n+1}$, it can be used to construct a new PRG $G : \{0,1\}^n \to \{0,1\}^{n+\ell}$ up to any polynomial value $\ell > 0$.

To prove this result, we use a technique known as **hybrid argument**. We define a sequence $H_0, \ldots, H_\ell$ of distributions such that:

- $H_0 \approx_c G^\ell(\mathcal{U}_n)$ and $H_\ell \approx_c \mathcal{U}_{n+\ell}$
- $H_i \approx_c H_{i+1}$ for each $i \in [0, \ell - 1]$
- Each H_i is an hybrid of truly random and pseudorandom bits, where the number of truly random bits increases as i grows, getting replacing the pseudorandom bits

Theorem 3.1: PRG stretching

Let $G : \{0,1\}^n \to \{0,1\}^{n+1}$ be a PRG. Then, for any $\ell = \text{poly}(n)$ there is a PRG $G^\ell : \{0,1\}^n \to \{0,1\}^{n+\ell}$.

Proof. For each $i \in [0, \ell]$, let $f_i(s) = b_1 \,||\, \ldots \,||\, b_\ell \,||\, s_\ell$ – where $||$ is the string concatenation operator – be the function computed through the following algorithm (see Figure 3.1).

f_i = "On input $s \in \{0,1\}^n$:

1. Set $s_i = s$
2. For each $j \leq i$ set b_j as a truly random bit, i.e. $b_j \in_R \mathcal{U}_1$
3. For each $j > i$ set s_j and b_j as the values such that $G(s_{j-1}) = s_j \,||\, b_j$, where $s_j \in \{0,1\}^n$ and $b_j \in \{0,1\}$.
4. Return $b_1 \,||\, \ldots \,||\, b_\ell \,||\, s_\ell$.

It's easy to see that each $i \in [0, \ell]$ the function f_i is PPT-computable since G is a PPT algorithm by hypothesis and $\ell = \text{poly}(n)$. Moreover, f_0 is deterministic since it doesn't generate random bits. Let $H_0, \ldots, H_\ell$ be the distributions over the outputs of the functions $f_0, \ldots, f_\ell$. We observe that $H_\ell = \mathcal{U}_{n+\ell}$ since every bit is truly random in f_ℓ. Hence, by letting $G^\ell = f_0$ we get that $H_0 = G^\ell(\mathcal{U}_n)$ and $H_\ell = \mathcal{U}_{n+\ell}$.

Claim: For each $i \in [0, \ell - 1]$ it holds that $H_i \approx_c H_{i+1}$.

Proof of the claim. Fix $i \in [0, \ell-1]$. We prove the claim by reducing the distinguishability of $G(\mathcal{U}_n)$ from $\mathcal{U}_{n+1}$ to the distinguishability of H_i from H_{i+1}. By way of contradiction, suppose that $H_i \not\approx_c H_{i+1}$, meaning that there is a distinguisher D_i such that:

$$|\Pr[D_i(z) = 1 : z \in_R H_i] - \Pr[D_i(z) = 1 : z \in_R H_{i+1}]| > \frac{1}{n^c}$$

for some $c > 0$. Let D be the algorithm defined as follows:

D = "On input $w \in \{0,1\}^{n+1}$:

1. Set s and b as the values such that $w = s \,||\, b$, where $s \in \{0,1\}^n$ and $b \in \{0,1\}$.
2. Compute $f_{i+1}(s) = b_1 \,||\, \ldots \,||\, b_\ell \,||\, s_\ell$.
3. Set $z' = b_1 \,||\, \ldots \,||\, b_{i-1} \,||\, b \,||\, b_{i+1} \,||\, \ldots \,||\, b_\ell \,||\, s_\ell$.
4. Return $D_i(z_w)$."

We observe that H_i and H_{i+1} differ from each other only by one bit: in f_i, the bit b_i is the last bit of $G(s_{i-1})$, while in f_{i+1} it is a truly random bit. Hence, when D is given an input $w \in G(\mathcal{U}_n)$, the last bit of w is generated through G, making the string z' built by D an input sampled from H_i. Vice versa, when D is given an input $w \in_R \mathcal{U}_{n+1}$, the last bit of w is truly random, making the string z' built by D an in input sampled from H_{i+1}. Therefore, we get that:

$$\begin{aligned} &|\Pr[D(w) = 1 : w \in_R G(\mathcal{U}_n)] - \Pr[D(w) = 1 : w \in_R \mathcal{U}_{n+1}]| \\ =&|\Pr[D_i(z') = 1 : z' \in_R H_i] - \Pr[D_i(z') = 1 : z' \in_R H_{i+1}]| \\ >&\frac{1}{n^c} \end{aligned}$$

concluding that D is a distinguisher for G, raising a contradiction over G being a PRG. □

By transitivy of $\approx_c$ and the above claim, we conclude that $G^\ell(\mathcal{U}_n) = H_0 \approx_c \ldots \approx_c H_\ell = \mathcal{U}_{n+\ell}$, meaning that G^ℓ is a PRG. □

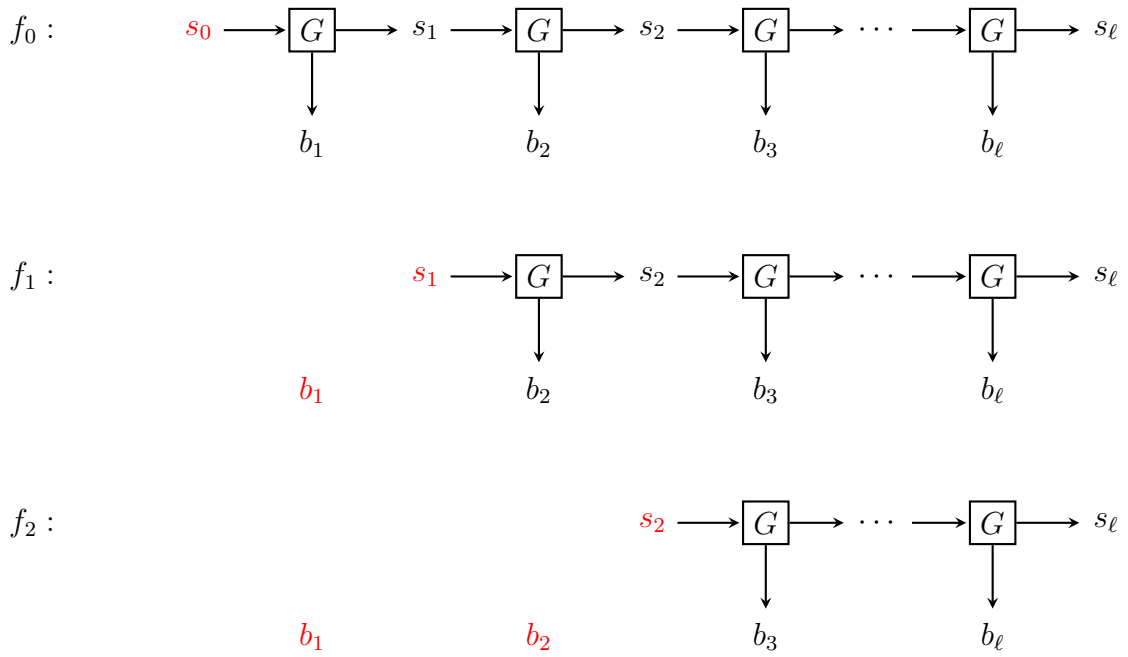

Figure 3.1: The idea behind the construction of the functions $f_0, \ldots, f_\ell$. Red strings represent truly random bits.

3.3 Hard-core predicates

In the previous chapter we saw how a PRG with stretch 1 is – in theory – sufficient to construct a PRG with stretch poly(n). The next natural objective is to build such PRG with stretch 1. In theoretical terms, we'll see that such PRG can be constructed from any OWF. However, it can be proven that the existence of OWF and PRG is equivalent, i.e. that there exists an OWF if and only if there is a PRG. This theoretical result makes building PRG as hard as building a OWF, which we aren't even sure it exists. To bypass (only partially) this problem, in practical applications we build good-enough PRGs using heuristic constructions.

Theorem 3.2: From PRGs to OWFs

Let $G : \{0,1\}^n \to \{0,1\}^{2n}$ be a PRG. Then, there is a OWF $f : \{0,1\}^{2n} \to \{0,1\}^{2n}$.

Proof. Let f be the function that runs G on the first half of the input:

$$f(x_1 \,||\, \ldots \,||\, x_{2n}) = G(x_1 \,||\, \ldots \,||\, x_n)$$

By way of contradiction, suppose that f is not a OWF, meaning that there is an adversary A such that:

$$\Pr_{x \in_R \mathcal{U}_n}[f(x') = y : y = f(x); x' \leftarrow A(y)] \geq \frac{1}{n^c}$$

for some $c > 0$. We build an adversary D defined as:

D = "On input $z \in \{0,1\}^{2n}$:

1. Compute $A(z) = b_1 \,||\, \ldots \,||\, b_{2n}$
2. If $G(b_1 \,||\, \ldots \,||\, b_n) = z$ accept, otherwise reject."

If $z \in_R \mathcal{U}_{2n}$, the adversary D accepts with probability at most 2^{-n} since the first half of the output $A(z)$ is also uniform at random. If $z \in_R G(\mathcal{U}_n)$, instead, the adversary D accepts with probability at least n^{-c} since it first inverts the output and then recomputes it to distinguish it. Hence, we conclude that:

$$|\Pr[D(z) = 1 : z \in_R G(\mathcal{U}_n)] - \Pr[D(z) = 1 : z \in_R \mathcal{U}_{2n}]| \geq \frac{1}{n^c} - \frac{1}{2^n} \geq \frac{1}{n^c}$$

contradicting the assumption of G being a PRG. □

To prove that the implication also holds from right to left, thus that OWF imply PRGs, we need a fundamental theoretical concept called **hard-code bits**. Given a function f, these bits contain information about an input x that is hard to obtain from the output $f(x)$, meaning that any adversary A cannot do better than guessing with 50% chance – up to some negligible error – if the bit comes from x or not. Hard-core bits are given by a function h – called *predicate* – that extract an hard-core bit from any input x.

Definition 3.4: Hard-core predicate

Let $f : \{0,1\}^n \to \{0,1\}^m$ be a PPT-computable function. We say that a function $h : \{0,1\}^n \to \{0,1\}$ is a hard-core predicate for f if for every PPT adversary A it holds that:

$$\Pr[A(y) = h(x) : x \in_R \mathcal{U}_n, y = f(x)] \leq \frac{1}{2} + \text{negl}(n)$$

From the very definition that we just gave, it's easy to get an equivalent formulation: h is hard-core for f if and only if $(f(x), h(x)) \approx_c (f(x), \mathcal{U}_1)$ with $x \in_R \mathcal{U}_n$ (proof is left as an exercise).

We observe that both definitions we gave don't require that f is a OWF. However, hard-core predicates are – intuitively – what allows a function to be one-way: if we can establish an hard-core bit for every input x, any adversary will have a hard time inverting $f(x)$ due to this very bit. From these observation, we ask three questions:

1. Is there a universal hard-core predicate that is hard-core for every OWF?
2. Is every function with an hard-core predicate also one-way?
3. Does every one-way function have an hard-core predicate?

For the first question, the answer is *no*. Suppose that such h^* exists. Fix a OWF f and let $\widetilde{f}$ be defined $\widetilde{f}(x) = h^*(x) \,||\, f(x)$. It's easy to see that the adversary A^* that always returns the first bit of the output $\widetilde{f}(x)$ is such that:

$$\Pr[A^*(y) = h^*(x) : x \in_R \mathcal{U}_n, y = \widetilde{f}(x)] = 1$$

concluding that h^* is not an hard-core predicate for $\widetilde{f}$. By way of contradiction, suppose that $\widetilde{f}$ is not a OWF, i.e. there is an adversary $\widetilde{A}$ such that:

$$\Pr_{x \in_R \mathcal{U}_n}[\widetilde{f}(x') = y : y = \widetilde{f}(x); x' \leftarrow \widetilde{A}(y)] \geq \frac{1}{n^c}$$

for some $c > 0$. We build an adversary A such that $A(y) = \widetilde{A}(b \,||\, y)$, where $b \in_R \mathcal{U}_1$. By construction, we have that:

$$\begin{aligned}
&\Pr_{x \in_R \mathcal{U}_n}[f(x') = y : y = f(x); x' \leftarrow A(y)] \\
\geq &\Pr_{x \in_R \mathcal{U}_n}[b = h^*(x), f(x') = y : y = f(x); x' \leftarrow \widetilde{A}(b \,||\, y); b \in_R \mathcal{U}_1] \\
= &\frac{1}{2} \Pr_{x \in_R \mathcal{U}_n}[f(x') = y : y = f(x); x' \leftarrow \widetilde{A}(h^*(x) \,||\, y)] \\
= &\frac{1}{2} \Pr_{x \in_R \mathcal{U}_n}[\widetilde{f}(x') = h^*(x') \,||\, y : y = f(x); x' \leftarrow \widetilde{A}(h^*(x) \,||\, y)] \\
= &\frac{1}{2n^c}
\end{aligned}$$

which contradicts the assumption of f being OWF. Hence, $\widetilde{f}$ must be an OWF without h^* as an hard-predicate.

For the second question, the answer is also *no*. Let f be the function such that drops the last bit of the input, i.e. $f(x_1 \,||\, \ldots \,||\, x_n) = x_1 \,||\, \ldots \,||\, x_{n-1}$. Let h be the function that returns only the last bit of the input, i.e. $h(x_1 \,||\, \ldots \,||\, x_n) = x_n$. By construction, h is trivially an hard-core predicate for f since the last bit is independent from the others when $x \in_R \mathcal{U}_n$. However, f is not a OWF since an adversary that picks a random bit $b \in_R \mathcal{U}_1$ has 50% chance of guessing $b = x_n$, inverting the output.

What about the third question? In this case, the answer is *yes, kinda*. Goldreich and Levin showed that every OWF can be trivially modified to obtain a new OWF that has a specific hard-core predicate. For any OWF $f : \{0,1\}^n \to \{0,1\}^m$, the **Goldreich-Levin alteration** of f is the function $g : \{0,1\}^{2n} \to \{0,1\}^{m+n}$ defined as $g(x,r) = f(x) \,||\, r$, where r is supposed to be taken as $r \in_R \mathcal{U}_n$. We omit the proof due to it being too convoluted.

Theorem 3.3: The Goldreich-Levin theorem

Let f be an OWF and let g be obtained through the Goldreich-Levin construction. Then, g is an OWF function with an hard-core predicate $h(x,r) = \bigoplus_{i=1}^{n} x_i r_i$.

Proof. Omitted. □

Surprisingly, OWFs can be combined with an hard-core predicate in order to generate a PRG. The easiest case of such property is given by **One-way Permutations (OWPs)**, that being bijective OWFs.

Proposition 3.1: From OWP to PRG

Let f be a OWP with an hard-core predicate h. Then, $G(s) = f(s) \,||\, h(s)$ is a PRG with expansion factor 1.

Proof. Consider the first n bits of the output $G(s)$, i.e. the bits $f(s)$. When s is taken uniformly at random, the output $f(s)$ will also be uniform at random due to f being a permutation. For the last $n+1$ bit of the output, instead, we know that $h(s)$ is by definition computationally hard to distinguish from a bit $b \in_R \mathcal{U}_1$. This concludes that:

$$G(\mathcal{U}_n) \equiv (f(\mathcal{U}_n), h(\mathcal{U}_n)) \approx_c (\mathcal{U}_n, \mathcal{U}_1) \equiv \mathcal{U}_{n+1}$$

□

The above proposition can be strenghtened by generalizing to every OWF instead of OWPs (the idea is to mix multiple outputs of f with their hard-core bits).

Theorem 3.4: From OWF to PRG

For every OWF f there is a PRG G with expansion factor 1.

Proof. Omitted. □

3.4 Solved exercises

Problem 3.1

For $m > n$, let $G : \{0,1\}^n \to \{0,1\}^m$ be a secure PRG. Consider G' defined as $G'(s) = G(s) \oplus (0^{m-n} \,||\, s)$. Prove or disprove that G' is a secure PRG.

Solution:

G' is not necessarily a secure PRG. Let $G(s) = G^*(s_{1:n-1}) \,||\, s_n$, where $G^* : \{0,1\}^n \to \{0,1\}^{m-1}$ is a PRG, $s_{1:n-1}$ denotes the first $n-1$ bits of s and s_n is the last bit of s. It can be proven that this chosen G is indeed a PRG, making it a valid choice. Now, we observe that:

$$\begin{aligned} G'(s) &= G(s) \oplus (0^{m-n} \,||\, s) \\ &= (G^*(s_{1:n-1})_{1:m-n} \oplus 0^{m-n}) \,||\, (G^*(s_{1:n-1})_{m-n+1:m-1} \oplus s_{1:n-1}) \,||\, (s_n \oplus s_n) \\ &= G^*(s_{1:n-1})_{1:m-n} \,||\, (G^*(s_{1:n-1})_{m-n+1:m-1} \oplus s_{1:n-1}) \,||\, 0 \end{aligned}$$

In other words, the last bit of $G'(s)$ will always be 0. Consider now the adversary A that checks the input $z \in \{0,1\}^m$ and returns 1 if $z_m = 0$ and 0 otherwise. Then:

$$|\Pr[A(z) = 1 : z \in_R G'(\mathcal{U}_n)] - \Pr[A(z) = 1 : z \in_R \mathcal{U}_m]| = \left|1 - \frac{1}{2}\right| = \frac{1}{2}$$

concluding that A is an adversary that distinguishes $G'(\mathcal{U}_n)$ from $\mathcal{U}_m$ with non-negligible probability.

Problem 3.2

Let $f_1, f_2, f_3 : \{0,1\}^n \to \{0,1\}^n$ be arbitrary functions. You know that at least one of them is a OWF, but you don't know which one. Design a OWF f using f_1, f_2, f_3 in a black-box manner.

Solution:

Let $f : \{0,1\}^{3n} \to \{0,1\}^{3n}$ be defined by $f(x_1 \,||\, x_2 \,||\, x_3) = f_1(x_1) \,||\, f_2(x_2) \,||\, f_3(x_3)$. By way of contradiction, suppose that there is an adversary A_f^{OWF} that inverts f with non-negligible probability. We define three adversaries $A_{f_1}^{\text{OWF}}, A_{f_2}^{\text{OWF}}, A_{f_3}^{\text{OWF}}$ as follows (with $i \in [3]$ fixed):

1. C_f^{OWF} sends the output y to $A_{f_i}^{\text{OWF}}$
2. $A_{f_i}^{\text{OWF}}$ sets $y'_i = y$ and $y'_j = 0^n$ for $j \in [3] - \{i\}$. They then send $y'_1 \,||\, y'_2 \,||\, y'_3$ to A_f^{OWF}
3. A_f^{OWF} returns $x'_1 \,||\, x'_2 \,||\, x'_3$ and $A_{f_i}^{\text{OWF}}$ forwards x'_i to C^{OWF}

Since at least one of f_1, f_2, f_3 is a OWF, there is at least one index i such that $A_{f_i}^{\text{OWF}}$ forms a reduction from the OWF game of f to the OWF game of f_i, raising a contradiction. Therefore, f must be an OWF.

4

Symmetric-key encryption

4.1 Games as security proofs

In Chapter 2 we gave the initial definitions of secure cryptographic schemes, without caring about the computation being achieved efficiently. In Chapter 3, instead, we discussed the basics of secure efficient computation. Now, we're ready to merge the two ideas.

We start by discussing secure SKEs. Using the concept of pseudorandom generator (PRG) that we introduced in the previous chapter, we can define a *PRG One Time Pad (PRG-OTP)*. Given $\mathcal{K} = \{0,1\}^\lambda$ and $\mathcal{M} = \mathcal{C} = \{0,1\}^{\lambda+\ell}$, we define:

$$\text{Enc}(K, m) = G(K) \oplus m \qquad \text{Dec}(K, c) = G(K) \oplus c$$

where $G : \{0,1\}^\lambda \to \{0,1\}^{\lambda+\ell}$ is a PRG and $\ell = \text{poly}(\lambda)$. The use of a PRG allows us to "bypass" Shannon's key-length constraint since the "pseudorandom key" generated through $G(K)$ is still at least as long as the message. However, this implies that our SKE doesn't have real perfect secrecy, otherwise the theorem would be false. This idea moves us into defining a new concept of **computationally secure SKE**.

In this and many other cases, security definitions will be given in terms of **games** between two entities: a challenger and an adversary. The two players play the game sending messages to each other. The challger's objective is proving that the scheme is secure by answering the questions posed the adversary, while the latter's objective is to break the scheme. The challenger wins if it is capable of responding to any possible series of questions without making the adversary able to break the scheme. If the challenger has a strategy to win the game, we consider the scheme to be secure. We assume that the two players are allowed only efficient computation.

Consider the following example. Let $\text{Game}^{\text{1-time}}_{\Pi,A}(\lambda, b)$ be the 2-player game defined as follows:

1. The adversary A sends two messages $m_0, m_1 \in \mathcal{M}$ to the challenger C
2. C generates a key $K \in \mathcal{K}$ and sends $\text{Enc}_\Pi(K, m_b) = c$ back to A

3. A analyzes c and tries to guess if it was obtained through m_0 or m_1 by sending a bit $b' \in \{0,1\}$
4. If $b' \neq b$, the challenger wins. Otherwise, the adversary wins.

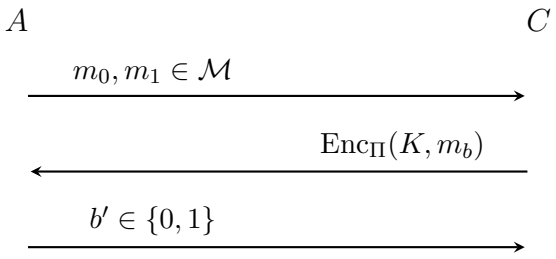

Figure 4.1: Graphical representation of $\text{Game}^{\text{1-time}}_{\Pi,A}(\lambda, b)$

By the very definition of the game, it's easy to see that if, for every pair of messages, the adversary is not capable of distinguishing which message got encrypted by the challenger, i.e. if they are playing $\text{Game}^{\text{1-time}}_{\Pi,A}(\lambda, 0)$ or $\text{Game}^{\text{1-time}}_{\Pi,A}(\lambda, 1)$, then the scheme can be considered **1-time computationally secure SKE**.

Definition 4.1: 1-time computational security

Let $\Pi = (\text{Enc}, \text{Dec})$ be a SKE. We say that Π is 1-time computationally secure if:

$$\text{Game}^{\text{1-time}}_{\Pi,A}(\lambda, 0) \approx_c \text{Game}^{\text{1-time}}_{\Pi,A}(\lambda, 1)$$

for all PPT adversaries A.

To see how this game can be used for specific cryptosystems, we prove that PRG-OTP has 1-time computationally security.

Proposition 4.1

PRG-OTP has 1-time computational security.

Proof. Let $\ell = \text{poly}(n)$ be the expansion factor of the PRG G used inside PRG-OTP. We use an argument similar to the hybrid argument. Let $\text{Hyb}(\lambda, b)$ be the 2-player game defined as follows:

1. The adversary A sends two messages $m_0, m_1 \in \{0,1\}^\lambda$ to the challenger C
2. C picks $z \in_R \mathcal{U}_{\lambda+\ell}$ and sends $z \oplus m_b = c$ back to A
3. A analyzes c and tries to guess if it was obtained through m_0 or m_1 by sending a bit $b' \in \{0,1\}$
4. If $b' \neq b$, the challenger wins. Otherwise, the adversary wins.

We observe that the difference between $\text{Game}^{\text{1-time}}_{\Pi,A}(\lambda, b)$ and $\text{Hyb}(\lambda, b)$ is given by $G(K)$ being replaced by a uniform at random string $z \in_R \mathcal{U}_{\lambda+\ell}$. Since $G(\mathcal{U}_\lambda) \approx_c \mathcal{U}_{\lambda+\ell}$ by choice of G, the hybrid game and the original game are expected to also be indistinguishable from each other.

Claim: For each $b \in \{0,1\}$ it holds that $\text{Game}^{\text{1-time}}_{\Pi,A}(\lambda, b) \approx_c \text{Hyb}(\lambda, b)$

Proof of the claim. By way of contradiction, assume that there is an adversary A' such that:

$$\left|\Pr\left[A'(b') = 1 : b' \leftarrow \text{Game}^{\text{1-time}}_{\Pi,A}(\lambda, b)\right] - \Pr\left[A'(b') = 1 : b' \leftarrow \text{Hyb}(\lambda, b)\right]\right| \geq \frac{1}{\lambda^c}$$

for some $c > 0$. We build an adversary D defined as follows:

D = "On input $z \in \{0,1\}^{\lambda+\ell}$.

1. D challenges A' to play $\text{Hyb}(\lambda, b)$ (thus D is the challenger and A' is the adversary). D will use z to compute $m_b \oplus z = c$ instead of choosing a random string.
2. D accepts if A' wins, otherwise D rejects."

Since A' can distinguish the two games, it is also able to distinguish if the string c passed by D was computed with z taken from $G(\mathcal{U}_\lambda)$ or $\mathcal{U}_{\lambda+\ell}$. This implies that:

$$\Pr[D(z) = 1 : z \in_R G(\mathcal{U}_\lambda)] = \Pr\left[A'(b') = 1 : b' \leftarrow \text{Game}^{\text{1-time}}_{\Pi,A'}(\lambda, b)\right]$$

and:

$$\Pr[D(z) = 1 : z \in_R \mathcal{U}_{\lambda+\ell}] = \Pr\left[A'(b') = 1 : b' \leftarrow \text{Hyb}(\lambda, b)\right]$$

concluding that:

$$\begin{aligned}
&|\Pr[D(z) = 1 : z \in_R G(\mathcal{U}_\lambda)] - \Pr[D(z) = 1 : z \in_R \mathcal{U}_{\lambda+\ell}]| \\
=&\left|\Pr\left[A'(b') = 1 : b' \leftarrow \text{Game}^{\text{1-time}}_{\Pi,A'}(\lambda, b)\right] - \Pr\left[A'(b') = 1 : b' \leftarrow \text{Hyb}(\lambda, b)\right]\right| \\
\geq&\frac{1}{\lambda^c}
\end{aligned}$$

and thus contradicting the fact that G is a PRG. □

By definition, it's easy to see that $\text{Hyb}(\lambda, 0) \equiv \text{Hyb}(\lambda, 1)$ since the distribution of c is uniform and independent of b. Therefore, we conclude that:

$$\text{Game}^{\text{1-time}}_{\Pi,A'}(\lambda, 0) \approx_c \text{Hyb}(\lambda, 0) \equiv \text{Hyb}(\lambda, 1) \approx_c \text{Game}^{\text{1-time}}_{\Pi,A'}(\lambda, 1)$$

□

4.2 CPA-security and pseudorandom functions

In the previous section we proved that PRG-OTP is 1-time computationally secure, meaning that it is secure against attackers that know only one message. However, this SKE's weakness is *ciphertext attacks*. Suppose that an attacker knows a ciphertext-message pair (m_1, c_1). If the challenger sends another ciphertext c_2 obtained from a message m_2, the attacker is able to retrieve the contents of m_2 through the SKE itself:

$$c_1 \oplus c_2 \oplus m_1 = (G(K) \oplus m_1) \oplus (G(K) \oplus m_2) \oplus m_1 = m_2$$

The generalization of this type of attack is known as **Known-plaintext Attack (KPA)** and the security counterpart is known as *t-time computational security*. We observe that KPAs require that the attacker only knows the pairs $(m_1, c_1), \ldots, (m_t, c_t)$ but is not able to choose them. When the attacker is able to choose the pairs (e.g. they can choose messages and get their ciphertexts) we talk about **Chosen-plaintext Attack (CPA)**.

In the CPA game the adversary is given access to an encryption **oracle**, i.e. a subprocedures with unlimited power that can be queried in order to get an answer. Each query costs $\Theta(1)$ computational steps. We observe that the oracle queries made by the attacker can be seen as "questions already answered". For instance, suppose that the attacker queries an oracle O capable of returning the ciphertext of a message, i.e. solves the problem $\text{Enc}(K, \cdot)$. If the attacker queries $m_1, \ldots, m_t$, the oracle will answer with $c_1, \ldots, c_t$. This is equivalent to saying that the attacker has chosen the pairs $(m_1, c_1), \ldots, (m_t, c_t)$. The concept of oracle is just a computational medium to express this idea.

Let $\text{Game}^{\text{CPA}}_{\Pi,A}(\lambda, b)$ be the 2-player game defined as follows:

1. The challenger C generates a key $K \in \mathcal{K}$.
2. The adversary A queries $m_1, \ldots, m_t \in \{0,1\}^\lambda$, where $t = \text{poly}(\lambda)$, to an oracle for $\text{Enc}_\Pi(K, \cdot)$. The oracle answers with $c_1, \ldots, c_t$.
3. The adversary sends two messages $m_0^*, m_1^* \in \{0,1\}^\lambda$ to the challenger C
4. C sends $\text{Enc}_\Pi(K, m_b^*) = c^*$ back to A
5. A queries $m_1', \ldots, m_s' \in \{0,1\}^\lambda$, where $s = \text{poly}(\lambda)$, to an oracle for $\text{Enc}_\Pi(K, \cdot)$. The oracle answers with $c_1', \ldots, c_t'$.
6. A tries to guess if it was obtained through m_0 or m_1 by sending a bit $b' \in \{0,1\}$
7. If $b' \neq b$, the challenger wins. Otherwise, the adversary wins.

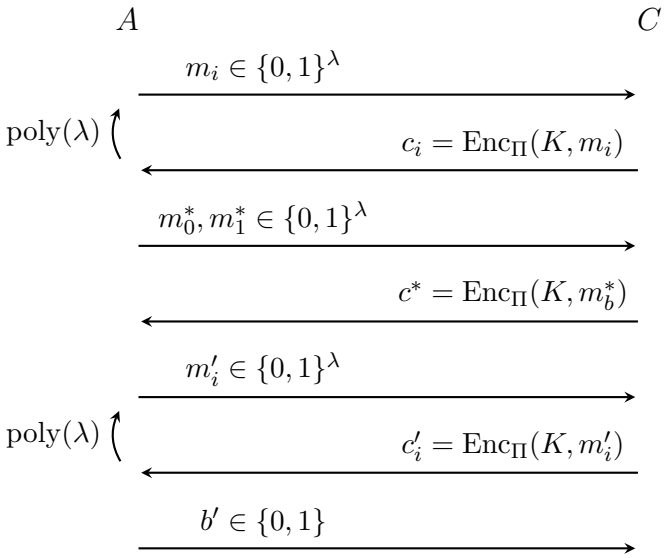

Figure 4.2: Graphical representation of $\mathrm{Game}^{\mathrm{CPA}}_{\Pi,A}(\lambda, b)$. Queries to the oracle can be substituted with queries to C, as if the two players already exchanged some pairs.

Definition 4.2: CPA-security

Let $\Pi = (\mathrm{Enc}, \mathrm{Dec})$ be a SKE. We say that Π is CPA-secure if:

$$\mathrm{Game}^{\mathrm{CPA}}_{\Pi,A}(\lambda, 0) \approx_c \mathrm{Game}^{\mathrm{CPA}}_{\Pi,A}(\lambda, 1)$$

for all PPT adversaries A.

It's easy to see that the PRG-OTP is not CPA-secure since the output of the encryption scheme is *deterministic* (meaning that it will always return the same output when given the same input). Consider the attacker that queries two messages $m_0, m_1 \in \{0,1\}^n$. After obtaining the ciphertexts c_0, c_1 from C, the attacker sends the final queries $m_0^* = m_0$ and $m_1^* = m_1$, obtaining the final ciphertext c^*. Since the scheme is deterministic, we know that either $c^* = c_0$ or $c^* = c_1$ must hold (depending on the value b used by C). Thus, the attacker returns 0 if $c^* = c_0$,, otherwise they returns 1. This allows the attacker to distinguish the two games with probability 1. Clearly, this idea is not restricted to the PRG-OTP: it holds for any deterministic SKE!

Proposition 4.2

No deterministic SKE can be CPA-secure.

In order to make a CPA-secure SKE, we need to substitute pseudorandom generators with **pseudorandom functions**, i.e. functions that are indistinguishable from a randomly selected one (thus they make their internal workings incomprehensible).

Let $\mathcal{R}(s \to t)$ denote the uniform distribution over the set of all functions from strings of s bits to strings of t bits. We define the following 2-player game $\mathrm{Game}^{\mathrm{PRF}}_{\mathcal{F},A}(\lambda, b)$:

1. The challenger C generates a key $K \in \{0,1\}^\lambda$ and sets $J = F_K$ if $b = 0$, otherwise $J \in_R \mathcal{R}(n \to n)$.
2. The adversary A queries $x_1, \ldots, x_t \in \{0,1\}^n$ where $t = \text{poly}(n)$, to an oracle for $J(\cdot)$. The oracle answers with $y_1, \ldots, y_t$.
3. A tries to guess b by sending a bit $b' \in \{0,1\}$
4. If $b \neq b'$, the challenger wins. Otherwise, the adversary wins.

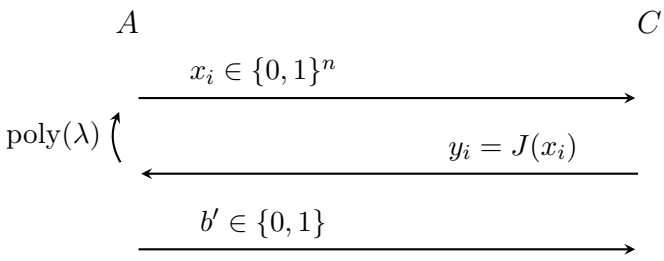

Figure 4.3: Graphical representation of $\text{Game}^{\text{PRF}}_{\mathcal{F},A}(\lambda, b)$. Queries to the oracle can be substituted with queries to C, as if the two players already exchanged some pairs.

Definition 4.3: Pseudorandom functions

Let $\mathcal{F} = \{F_K : \{0,1\}^n \to \{0,1\}^m\}_{K \in \{0,1\}^\lambda}$ be a family of poly-time functions. We say that $\mathcal{F}$ is a pseudorandom function family (PRF) if:

$$\text{Game}^{\text{PRF}}_{\mathcal{F},A}(\lambda, 0) \approx_c \text{Game}^{\text{PRF}}_{\mathcal{F},A}(\lambda, 1)$$

for all PPT adversaries A.

We observe that the above definition requires that $\mathcal{F}$ is a family of efficient deterministic functions: randomness is only used to choose a key that identifies which function of the family must be used. The uniform distribution $\mathcal{R}(n \to n)$, instead, also contains functions that aren't poly-time computable. The idea behind the PRF game is to establish that each function of the family is computationally indistinguishable from a randomly chosen function.

In practical applications, many common encryption schemes (e.g. DES, 3DES, AES, ...) are based on on *pseudorandom permutations (PRP)*, i.e. PRFs that are invertible if they key is known. For now, we'll give the idea behind how PRFs can be used in theory to produce CPA-secure SKEs.

First of all, we observe that we cannot simply swap PRGs with PRFs in order to get a good scheme since each function is deterministic. The trick here is to use a random initial value called **nonce** (short for "number used once"). As their name suggests, these values are used exactly once by the cryptosystem, granting that the same input will always give a different ciphertext.

Given a PRF $\mathcal{F} = \{F_K : \{0,1\}^n \to \{0,1\}^n\}_{K \in \{0,1\}^\lambda}$, let the *PRF One Time Pad (PRF-OTP)* be the scheme defined by the following operations:

1. $\text{Enc}(K, m) = (r, F_K(r) \oplus m)$, where $r \in \{0,1\}^n$ is a nonce
2. $\text{Dec}(K, (r, c)) = F_K(r) \oplus c$

We observe that the encryption operation must also return the used nonce in order for the decryption operation to work. However, this is not an issue for CPA-security: since the nonce will never be used again by the scheme, the adversary gains no information on the key by knowing it.

Proposition 4.3

PRF-OTP is CPA-secure.

Proof. Omitted (an hybrid argument suffices). □

4.2.1 The GGM Tree

After discussing how PRFs can be used to achieve CPA-security, we prove that PRGs can be used to construct PRFs. In particular, we'll give an explicit construction known as the **Goldreich-Goldwasser-Micali Tree (GGM Tree)**.

Definition 4.4: GGM Tree

Given a PRG $G : \{0,1\}^\lambda \to \{0,1\}^{2\lambda}$, let $G_0, G_1 : \{0,1\}^\lambda \to \{0,1\}^\lambda$ be two functions that respectively return the first and last λ bits of the output of G, meaning that $G(k) = G_0(k) \,||\, G_1(k)$ for all $k \in \{0,1\}^\lambda$. The GGM Tree function family is defined as $\mathcal{F} = \{F_K : \{0,1\}^n \to \{0,1\}^\lambda\}_{K \in \{0,1\}^\lambda}$, where:

$$F_K(x_1 \,||\, \ldots \,||\, x_n) = G_{x_n}(G_{x_{n-1}}(\ldots G_{x_2}(G_{x_1}(K))))$$

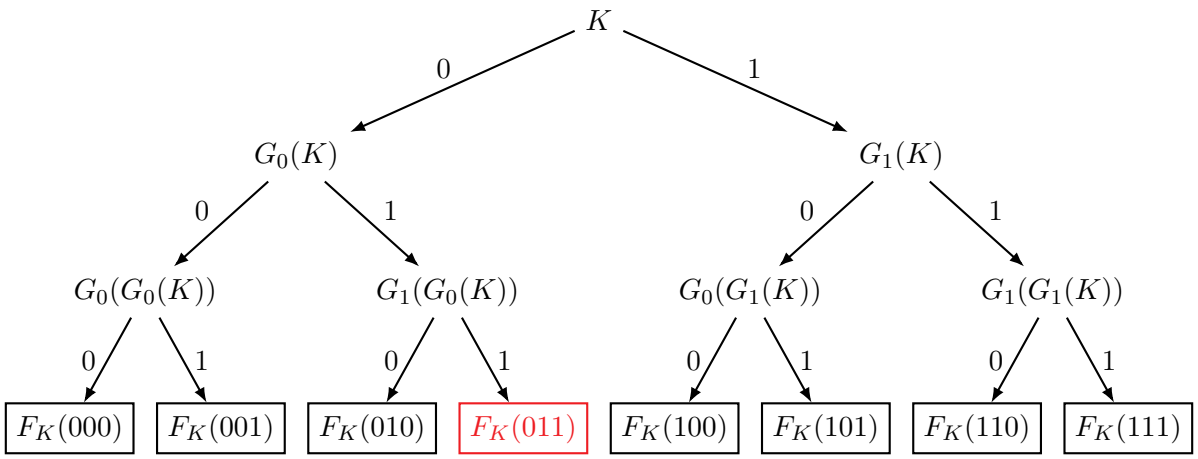

Figure 4.4: The GGM Tree for inputs of length $n = 3$. The path leading to the red rectangle represent the computation $F_K(011) = G_1(G_1(G_0(K)))$.

Lemma 4.1

Let $G : \{0,1\}^\lambda \to \{0,1\}^{2\lambda}$ be a PRG. Given $K_1, \ldots, K_t \in_R \mathcal{U}_\lambda$, with $t = \text{poly}(\lambda)$ and $K_i \neq K_j$ for all $i \neq j$, it holds that:

$$(G(K_1), \ldots, G(K_t)) \approx_c (\mathcal{U}_{2\lambda}, \ldots, \mathcal{U}_{2\lambda})$$

Proof. We define t hybrid distributions $H_0, \ldots, H_t$ such that for each $i \in [t]$ it holds that:

$$H_i = (G(K_1), \ldots, G(K_{t-i}), \underbrace{\mathcal{U}_{2\lambda}, \ldots, \mathcal{U}_{2\lambda}}_{i \text{ times}})$$

We observe that $H_0 \equiv (G(K_1), \ldots, G(K_t))$ and $H_t \equiv (\mathcal{U}_{2\lambda}, \ldots, \mathcal{U}_{2\lambda})$.

Claim: For each $i \in [0, t-1]$ it holds that $H_i \approx_c H_{i+1}$.

Proof of the claim. Fix $i \in [0, t-1]$. We prove the claim by reducing the distinguishability of $G(K_{t-i})$ from $\mathcal{U}_{2\lambda}$ to the distinguishability of H_i from H_{i+1}. By way of contradiction, suppose that $H_i \not\approx_c H_{i+1}$, meaning that there is a distinguisher D_i such that:

$$|\Pr[D_i(z) = 1 : z \in_R H_i] - \Pr[D_i(z) = 1 : z \in_R H_{i+1}]| > \frac{1}{n^c}$$

for some $c > 0$. Let D be the algorithm defined as follows:

D = "On input $z \in \{0,1\}^{2\lambda}$:

1. Extract $t - i$ keys $K_1, \ldots, K_{t-i} \in_R \mathcal{U}_\lambda$
2. Extract $i - 1$ strings $w_1, \ldots, w_{i-1} \in_R \mathcal{U}_{2\lambda}$
3. Return $D_i(G(K_1) \,||\, \ldots \,||\, G(K_{t-i}) \,||\, z \,||\, w_1 \,||\, \ldots \,||\, w_{i-1})$"

We observe that H_i and H_{i+1} differ only on the i-th string of $2n$ bits: the former contains an output of G while the latter contains a truly random strng. This implies that:

$$\begin{aligned} &|\Pr[D(w) = 1 : w \in_R G(K_{t-i})] - \Pr[D(w) = 1 : w \in_R \mathcal{U}_{2\lambda}]| \\ =&|\Pr[D_i(z') = 1 : z' \in_R H_i] - \Pr[D_i(z') = 1 : z' \in_R H_{i+1}]| \\ >&\frac{1}{n^c} \end{aligned}$$

concluding that D is a distinguisher for G, raising a contradiction over G being a PRG. □

By transitivity of $\approx_c$, the claim concludes that:

$$(G(K_1), \ldots, G(K_t)) \equiv H_0 \approx_c \ldots \approx_c H_t \equiv (\mathcal{U}_{2\lambda}, \ldots, \mathcal{U}_{2\lambda})$$

□

Theorem 4.1: The GGM theorem

GGM Tree is a PRF when constructed using a PRG

Proof. Let $G : \{0,1\}^\lambda \to \{0,1\}^{2\lambda}$ be a PRG and let $G_0, G_1 : \{0,1\}^\lambda \to \{0,1\}^\lambda$ be the two functions such that $G(x) = G_0(x) \,||\, G_1(x)$. Let also $\mathcal{F} = \{F_K : \{0,1\}^n \to \{0,1\}^\lambda\}_{K \in \{0,1\}^\lambda}$ be the GGM function family.

Fix a key $K \in \{0,1\}^\lambda$. We proceed by induction on the length n of the input. When $n = 1$ it holds that $F_K(x) = G_x(K)$. Since $\mathcal{U}_{2\lambda} \approx_c G(K) \equiv (G_0(K), G_1(K))$, we know that $G_0(K) \approx_c \mathcal{U}_\lambda$ and $G_1(K) \approx_c \mathcal{U}_\lambda$. This concludes that, independently from the value of x, the output $F_K(x)$ is undistinguishable from a string taken from $\mathcal{U}_\lambda$, making it undistinguishable from a random function taken from $\mathcal{R}(1 \to \lambda)$.

We now assume that the inductive hypothesis holds for $n-1$, i.e. that the GGM family $\mathcal{F} = \{F_K : \{0,1\}^{n-1} \to \{0,1\}^\lambda\}_{K \in \{0,1\}^\lambda}$ is a PRF. First, we observe that for each key $K \in \{0,1\}^\lambda$, for each $x \in \{0,1\}^n$ and for each $y \in \{0,1\}^{n-1}$ it holds that $F_K(x \,||\, y) = G_x(F'_K(y))$.

We define the following three hybrid distributions:

- Hyb_0 is the distribution over the function $F_K(x \,||\, y) = G_x(F'_K(y))$
- Hyb_1 is the distribution over the function $H(x \,||\, y) = G_x(R'(y))$ where $R' \in_R \mathcal{R}(n-1 \to \lambda)$.
- Hyb_2 is the distribution over the function $R \in_R \mathcal{R}(n \to \lambda)$.

To prove that $\mathcal{F}$ is also a PRF, we'll show that $\text{Hyb}_0 \approx_c \text{Hyb}_1 \approx_c \text{Hyb}_2$.

Claim 1: $\text{Hyb}_0 \approx_c \text{Hyb}_1$

Proof of Claim 1. We reduce the distinguishability of $\text{Game}^{\text{PRF}}_{\mathcal{F}',A}(\lambda, 0)$ from $\text{Game}^{\text{PRF}}_{\mathcal{F}',A}(\lambda, 1)$ to the distinguishability of Hyb_0 from Hyb_1. By way of contradiction, suppose that $\text{Hyb}_0 \not\approx_c \text{Hyb}_1$, meaning that there is a PPT adversary D_{01} such that:

$$|\Pr[D_{01}(z) = 1 : z \in_R \text{Hyb}_0] - \Pr[D_{01}(z) = 1 : z \in_R \text{Hyb}_1]| > \frac{1}{n^c}$$

for some $c > 0$. We define the attack strategy A_{01} for $\text{Game}^{\text{PRF}}_{\mathcal{F}',A}(\lambda, b)$ as follows. The idea is for A_{01} to play two games simultaneously.

1. The challenger C generates a key $K \in \{0,1\}^\lambda$ and sets $J = F'_K$ if $b = 0$, otherwise $H \in_R \mathcal{R}(n-1 \to \lambda)$.
2. A_{01} challenges D_{01} to distinguish Hyb_0 from Hyb_1.
3. For each query $x \,||\, y$ – with $x \in \{0,1\}$ and $y \in \{0,1\}^n$ – made by D_{01}, the adversary A_{01} forwards y to the challenger C.
4. For each answer z returned by C, the adversary A_{01} forwards $G_x(z)$ to D_{01}.
5. When D_{01} gives the final bit b', A_{01} forwards the same bit to C.

In the above strategy, A_{01} plays two games simultaneously, using the challenger as a way to compute outputs of either F'_K or a random function R. After obtaining the outputs, A_{01} passes them through G_x before forwarding them to D_{01}. If the challenger used F'_K, the values forwarded from A_{01} will be outputs of F_K. Otherwise, if the challenger used R',

the values forwarded from A_{01} will be outputs of R. Since D_{01} can distinguish between F_K and R, A_{01} will be able to distinguish F'_K from R'. This contradicts the fact that $\mathcal{F}'$ is a PRF, concluding that $\text{Hyb}_0 \approx_c \text{Hyb}_1$ must be true. □

Claim 2: $\text{Hyb}_1 \approx \text{Hyb}_2$

Proof of Claim 2. We reduce the distinguishability of $(G(K_1), \ldots, G(K_t))$ from $(\mathcal{U}_{2\lambda}, \ldots, \mathcal{U}_{2\lambda})$ to the distinguishability of Hyb_1 from Hyb_2. By way of contradiction, suppose that there is an algorithm D_{12} that distinguishes Hyb_1 from Hyb_2 For each string $w \in \{0,1\}^{2\lambda}$, let $w^0, w^1 \in \{0,1\}^\lambda$ denote the first and last λ bits of w, i.e. $w = w^0 \,||\, w^1$. Let A_{12} be the algorithm defined as follows:

A_{12} = "On input $z = z_1 \,||\, \ldots \,||\, z_t$ with $z_i \in \{0,1\}^{2\lambda}$:

1. A_{12} challenges D_{12} to distinguish Hyb_1 from Hyb_2, allowing D_{12} exactly t queries.
2. For each query $x \,||\, y$ – with $x \in \{0,1\}$ and $y \in \{0,1\}^n$ – made by D_{12}, the adversary A_{12} answers with $z^x_{i(y)}$, where $i(y) \in [1,t]$ is either the index that was used by A_{12} if D_{12} already queried y or the next unused index.
3. When D_{01} gives the final bit b', A_{01} returns b'

When $z_1, \ldots, z_t \in \mathcal{U}_{2\lambda}$, all values returned by A_{12} to D_{12} will be truly random, making D_{12} assume that the values are taken from the distribution Hyb_2. When $z_1, \ldots, z_t \in G(\mathcal{U}_\lambda)$, instead, all values will be outputs of G, making D_{12} assume that the values are taken from the distribution Hyb_1. This concludes that A_{12} is a distinguisher for $(G(K_1), \ldots, G(K_t))$ and $(\mathcal{U}_{2\lambda}, \ldots, \mathcal{U}_{2\lambda})$, contradicting Lemma 4.1. □

□

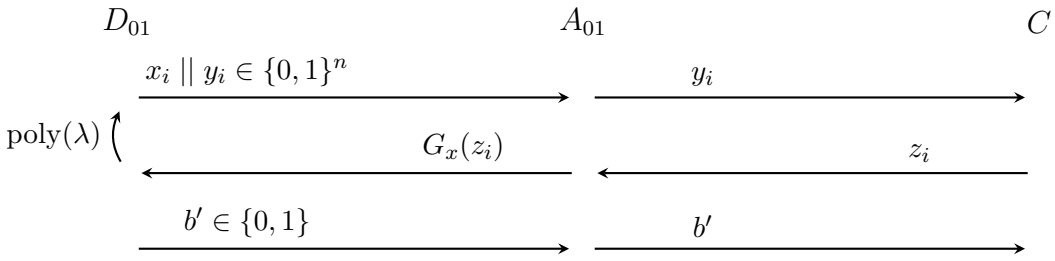

Figure 4.5: Graphical representation of the double-game attack used to prove Claim 1 of the GGM theorem.

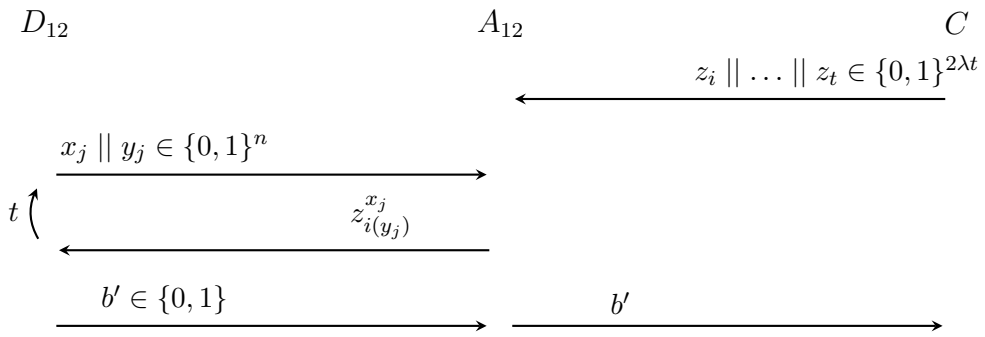

Figure 4.6: Graphical representation of the double-game attack used to prove Claim 2 of the GGM theorem.

4.3 Encryption modes for SKEs

After proving that OWFs can be used to create PRFs through PRGs, we're going to show that PRFs are enough to achieve practical symmetric-key encryption. As we'll see, for some practical applications we strictly require that the pseudorandom functions are invertible, i.e. that they are PRPs. Practical functions with this property are commonly called **block ciphers** (the concept of blockchipers was used way before the introduction of the concept of PRPs). From a theoretical prospective, we'll discuss how PRFs can be used to derive PRPs. From a practical prospective, instead, we'll discuss some blockchipers used in the real world.

First of all, we observe that real-world encryption systems must work with messages of *different lengths*. Formally, we say that these SKEs have **Variable Input Length (VIL)**, meaning that for each message m it holds that $|m| = \text{poly}(\lambda)$, where λ is the key length. In order to achieve security for VIL SKEs, we require a "standardized way" to encrypt messages of variable length. To solve this issue, each message m is split into a variable number of **blocks** of an equal fixed length n (commonly is a power of 2, such as 256). In other words, each message m is considered as $m = (m_1, \ldots, m_d)$ where $d \in \mathbb{N}$ and $m_i \in \{0,1\}^n$.

The simplest encryption type is the **Electronic Code Book (ECB)** mode, where the scheme simply applies the selected PRF to every block of the message.

Definition 4.5: Electronic Code Book (ECB) mode

Let $\mathcal{F} = \{F_K : \{0,1\}^n \to \{0,1\}^n\}_{K \in \{0,1\}^\lambda}$ be a PRF. Let $K \in \{0,1\}^\lambda$ be a chosen key. On input $m = (m_1, \ldots, m_d)$ the ECB encryption mode returns the ciphertext $c = (c_1, \ldots, c_d)$ constructed as:

- For all $i \in [d]$ it holds that $c_i = F_K(m_i)$

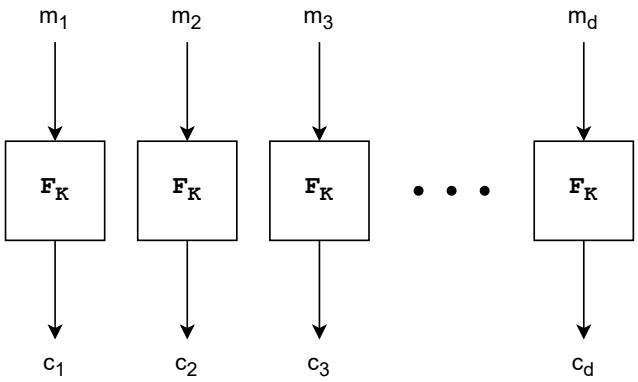

Figure 4.7: Graphical representation of ECB mode.

Clearly, the ECB mode cannot be CPA-secure since it is deterministic. As discussed in the PRF-OTP example, a nonce value is sufficient to make the ecryption non-deterministic. In encryption modes, this nonce value is usually referred to as **Initialization Vector (IV)**. In order to make decryption harder to bruteforce, common encryption modes are based on *sequential encryption*: some value produced during the encryption of block m_i is used for the encryption of block m_{i+1}. This allows us to restrict our interests to the secretness of the key and the IV. Moreover, it also makes decryption unparallelizable, further improving the security of the SKE. The first type of encryption mode based on this idea is the **Cipher Block Chaining (CBC)**.

Definition 4.6: Cipher Block Chaining (CBC) mode

Let $\mathcal{F} = \{F_K : \{0,1\}^n \to \{0,1\}^n\}_{K \in \{0,1\}^\lambda}$ be a PRF. Let $K \in \{0,1\}^\lambda$ be a chosen key. On input $m = (m_1, \ldots, m_d)$ the CBC encryption mode returns the ciphertext $c = (c_1, \ldots, c_d)$ constructed as:

- $c_0 \in_R \mathcal{U}_n$ is a nonce value (IV)
- For all $i \in [d]$ it holds that $c_i = F_K(c_{i-1} \oplus m_i)$

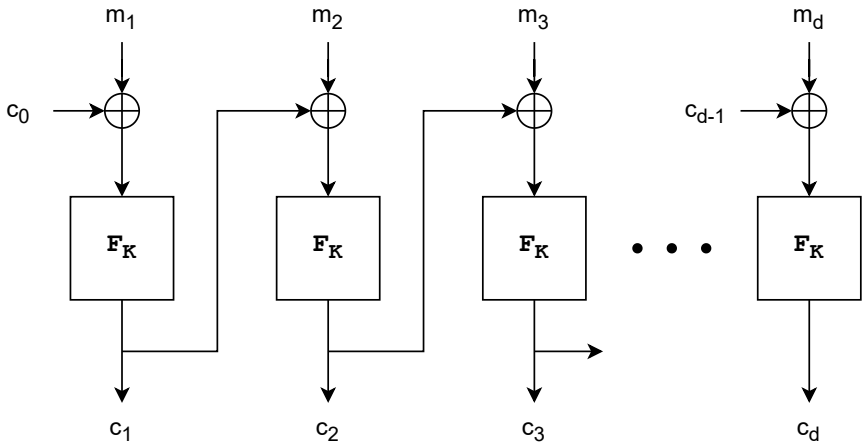

Figure 4.8: Graphical representation of CBC mode.

Other types of encryption mode are pretty much based on ideas similar to CBC. The only real difference lies in the order of operations. For instance, the **Cipher Feedback (CFB)** mode XORes each block after applying the PRF to the nonce value, instead of before. We observe that this was the idea behind our PRF-OTP example, the only difference lies in the chaining of encryptions.

Definition 4.7: Cipher Feedback (CFB) mode

Let $\mathcal{F} = \{F_K : \{0,1\}^n \to \{0,1\}^n\}_{K \in \{0,1\}^\lambda}$ be a PRF. Let $K \in \{0,1\}^\lambda$ be a chosen key. On input $m = (m_1, \ldots, m_d)$ the CFB encryption mode returns the ciphertext $c = (c_1, \ldots, c_d)$ constructed as:

- $c_0 \in_R \mathcal{U}_n$ is a nonce value (IV)
- For all $i \in [d]$ it holds that $c_i = F_K(c_{i-1}) \oplus m_i$

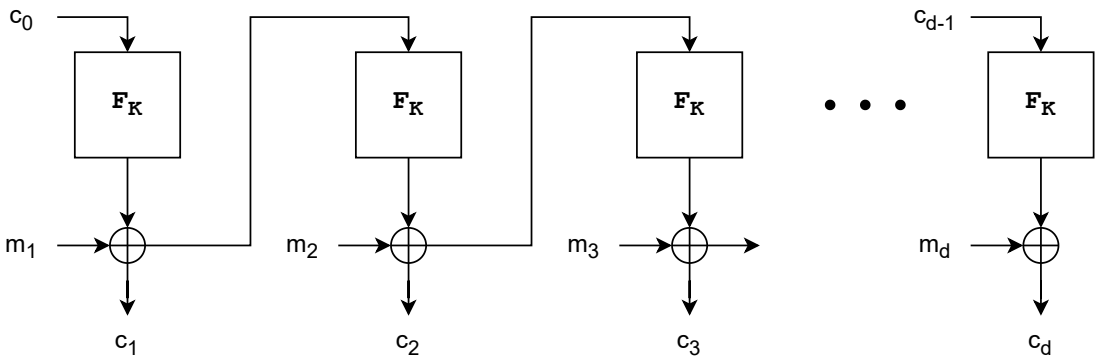

Figure 4.9: Graphical representation of CFB mode.

Another encryption mode extremely similar to CFB is the **Output Feedback (OFB)** mode, where the output of the PRF is directly forwarded instead of being XORed first.

Definition 4.8: Output Feedback (OFB) mode

Let $\mathcal{F} = \{F_K : \{0,1\}^n \to \{0,1\}^n\}_{K \in \{0,1\}^\lambda}$ be a PRF. Let $K \in \{0,1\}^\lambda$ be a chosen key. On input $m = (m_1, \ldots, m_d)$ the OFB encryption mode returns the ciphertext $c = (c_1, \ldots, c_d)$ constructed as:

- $q_0 \in_R \mathcal{U}_n$ is a nonce value (IV)
- For all $i \in [d]$ it holds that $q_i = F_K(q_{i-1})$
- For all $i \in [d]$ it holds that $c_i = q_i \oplus m_i$

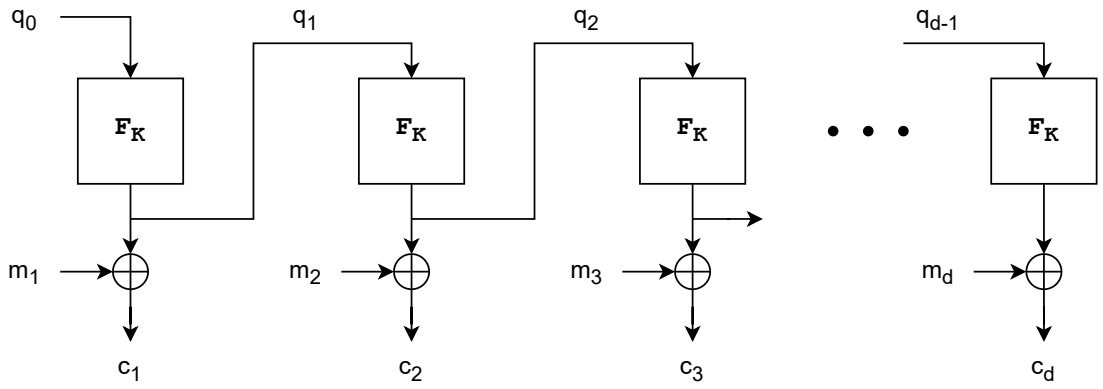

Figure 4.10: Graphical representation of OFB mode.

The last standard encryption mode we discuss is **Counter (CTR)** mode. Instead of using the previous (partial) output as the next input for the PRF, this mode gradually increases the initial nonce value and uses it for the next encryptions.

Definition 4.9: Counter (CTR) mode

Let $\mathcal{F} = \{F_K : \{0,1\}^n \to \{0,1\}^n\}_{K \in \{0,1\}^\lambda}$ be a PRF. Let $K \in \{0,1\}^\lambda$ be a chosen key. On input $m = (m_1, \ldots, m_d)$ the CTR encryption mode returns the ciphertext $c = (c_0, c_1, \ldots, c_d)$ constructed as:

- $r \in_R \mathcal{U}_n$ is a nonce value (IV)
- $c_0 = r$
- For all $i \in [d]$ it holds that $c_i = F_K(r + i - 1) \oplus m_i$

Note: the sum operation is in mod 2.

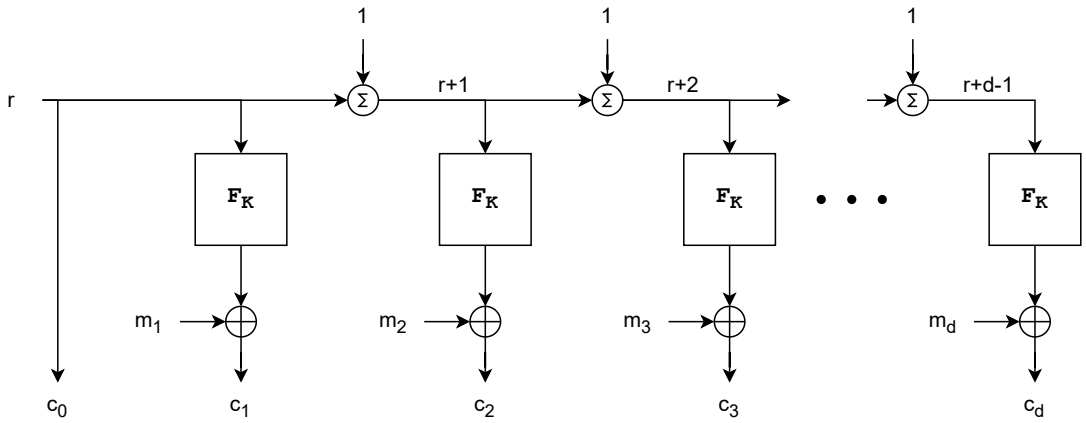

Figure 4.11: Graphical representation of CTR mode.

CBC, CFB, OFB and CTR can all be proven to be CPA-secure for VIL. Instead of proving the security of all of them, we'll give an idea by proving the result only for CTR (which is actually harder to prove).

Theorem 4.2: CPA-security through PRFs

CBC, CFB, OFB and CTR are CPA-secure for VIL when constructed using a PRF

Proof. We prove the theorem only for CTR. Let $G(\lambda, b) := \text{Game}^{\text{CPA}}_{\Pi,A}(\lambda, b)$, where Π is a CTR mode scheme constructed through a PRF $\mathcal{F}$.

Recall that for $G(\lambda, b)$ the challenge is given by two messages m_0^*, m_1^* such that $m_b^* = (m_{b,1}^*, \ldots, m_{b,d^*}^*)$ and an answer $c^* = (c_0^*, c_1^*, \ldots, c_{d^*}^*)$ such that $c_0^* = r^* \in_R \mathcal{U}_n$ and $c_i^* = F_K(r^* + i - 1) \oplus m_{b,i}^*$.

We define an hybrid distribution for both values of b. Let $H(\lambda, b)$ be the distribution obtained by replacing $F_K(\cdot) \in \mathcal{F}$ with $R(\cdot) \in \mathcal{R}(n \to n)$ in the game $G(\lambda, b)$ (in other words, we're using a random function R for the encryption instead of F_K).

Claim 1: $G(\lambda, b) \approx_c H(\lambda, b)$ for $b \in \{0, 1\}$

Proof of Claim 1. We reduce the distringuishability of F_K from R to the distinguishability of $G(\lambda, b)$ from $H(\lambda, b)$. Fix $b \in \{0, 1\}$. By way of contradiction, suppose that $G(\lambda, b) \not\approx_c H(\lambda, b)$, i.e. there is an adversary A that distringuished $G(\lambda, b)$ from $H(\lambda, b)$ with probability at least n^{-c} for some $c > 0$.

Let A' be the adversary for $\mathcal{F}$ defined as follows:

1. The adversary A' starts both games with the challenger C and the other adversary A
2. For $i \in [t]$ iterations, with $t = \text{poly}(\lambda)$, A sends a message $m^{(i)} = (m_1^{(i)}, \ldots, m_{d_i}^{(i)})$ to A'

3. Upon receiving $m^{(i)}$, A' sends $r^{(i)}, r^{(i)}+1, \ldots, r^{(i)}+d-1$ to the challenger C, which answers with $z_1^{(i)}, \ldots, z_d^{(i)}$
4. A' sends $c^{(i)} = (r^{(i)}, c_1^{(i)}, \ldots, c_{d_i}^{(i)})$ back to A, where $c_j^{(i)} = z_j^{(i)} \oplus m_j^{(i)}$
5. After t iterations, the above istructions are repeated again with the challenge messages m_0^*, m_1^*, getting the answer c^*
6. Once A receives c^* it returns a final answer $b' \in \{0,1\}$ and A' forwards b' to C
7. If $b' = b$, A' wins the PRF-game with C, otherwise it loses.

The idea behind the above simultaneous game is to use the challenger C to compute the outputs of either a pseudorandom function or a truly random one (depends on the value of b) and use those outputs to play the game with A.

If $b = 0$, C will use a pseudorandom function to compute the outputs, thus A will recognize that we're playing $G(\lambda, b)$, outputting 0. If $b = 1$, C will use a truly random function to compute the outputs, thus A will recognize that we're playing $H(\lambda, b)$, outputting 1. Therefore, A' distinguishes F_K from R with non-negligible probability, contradicting the fact that $\mathcal{F}$ is a PRF. □

Now, show that both $H(\lambda, 0)$ and $H(\lambda, 1)$ are indistinguishable from $\mathcal{U}_{n(d^*+1)}$, concluding the proof of the theorem through transitivity of $\approx_c$.

Claim 2: $H(\lambda, b) \approx_c \mathcal{U}_{n(d^*+1)}$ for $b \in \{0,1\}$

Proof of Claim 2. Consider the values $R(r^*), R(r^*+1), \ldots, R(r^*+d^*-1)$ computed during the challenge query of the game $H(\lambda, b)$. Similarly, consider the values $R(r^{(i)}), R(r^{(i)}+1), \ldots, R(r^{(i)}+d_i-1)$ computed during the encryption queries.

Let B_i be the binary "bad event" such that $B_i = 1$ if $\exists j \in [0, d_i - 1], j^* \in [0, d^* - 1]$ such that $r^{(i)} + j = r^* + j^*$, otherwise $B_i = 0$. More generally, let $B = B_1 \cup \ldots \cup B_t$

Consider the conditioning over the event $B = 0$. When $B = 0$, the two sequences never overlap over each $i \in [t]$, which means that the challenge ciphertext is distributed as:

- $c_0^* = r^*$
- $c_{j^*}^* = m_{b,j^*}^* \oplus u_{j^*}$ with $u_{j^*} \in_R \mathcal{U}_n$ for $j^* \in [0, d^* - 1]$

which is indistinguishable from $\mathcal{U}_{n(d^*+1)}$. As a consequence of this, we get that:

$$\mathrm{SD}(H(\lambda, b); \mathcal{U}_{n(d+1)}) \leq \Pr[B = 1]$$

We're left with computing $\Pr[B = 1]$. Fix $i \in [t]$. Without loss of generality, assume that $d_i = d^* = t = q$. Since the event $B_i = 1$ holds when $r^*, r^*+1, \ldots, r^*+1-1$ overlaps with $r^{(i)}, r^{(i)}+1, \ldots, r^{(i)}+q-1$, this can happen only when:

$$r^* - q + 1 \leq r^{(i)} \leq r^* + q - 1$$

Therefore, we have that:

$$\Pr[B_i = 1] \leq \frac{1 + (r^* + q - 1) - (r^* - q - 1)}{2^n} = \frac{2q - 1}{2^n}$$

Finally, through the union bound we conclude that:

$$\Pr[B = 1] \leq \sum_{i=1}^{q} \Pr[B_i = 1] = \frac{q(2q-1)}{2^n}$$

thus $H(\lambda, b) \approx_c \mathcal{U}_{n(d^*+1)}$. □

The two claims conclude that $G(\lambda, 0) \approx_c H(\lambda, 0) \approx_c \mathcal{U}_(n(d^* + 1)) \approx H(\lambda, 1) \approx_c G(\lambda, 1)$. □

4.4 UFCMA-security and universal hash families

After showing that the common encryption schemes are CPA-secure, we focus on secure message authentication. Conside any tag function $\text{Tag} : \mathcal{K} \times \mathcal{M} \to \mathcal{T}$. When can this function be considered secure under a computational setting? The idea is to make it hard to forge a pair (m^*, τ^*) such that $\text{Tag}(K, m^*) = \tau^*$ when K is unknown. This concept is formally known as **Unforgeability under Chosen-Message Attacks (UFCMA)**.

In $\text{Game}^{\text{UFCMA}}_{\Pi,A}(\lambda)$, the adversary is given access to an oracle for the Tag function and it is allowed to make $t = \text{poly}(\lambda)$ tag queries to the oracle. After completing the tag queries, the adversary returns a pair (m^*, τ^*) to the challenger (with $m^* \neq m_i$ for all $i \in [t]$). The challenger computes the tag of m^* and compares it with τ^*. If $\text{Tag}(K, m^*) = \tau^*$ is true, the adversary wins. Otherwise, the challenger wins.

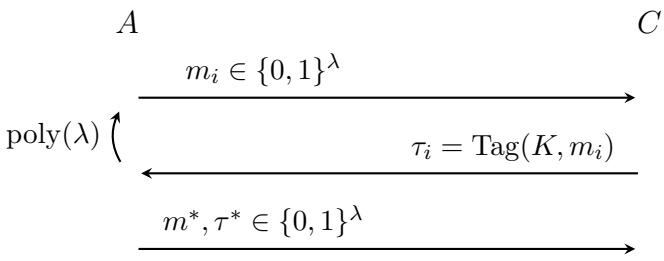

Figure 4.12: Graphical representation of $\text{Game}^{\text{UFCMA}}_{\Pi,A}(\lambda)$

Definition 4.10: UFCMA-security

Let $\Pi = (\text{Tag})$ be a MAC. We say that Π is UFCMA-secure if:

$$\Pr[\text{Game}^{\text{UFCMA}}_{\Pi,A}(\lambda) = 1] \leq \text{negl}(\lambda)$$

for all PPT adversaries A.

Pseudorandom functions are intuitively an easy way to get an UFCMA-secure MAC: since the function is hard to invert, the adversary won't be able to infer informations on the key through message-tag pairs. However, this works only for FIL (Fixed Input Length) schemes.

Theorem 4.3: UFCMA-security through PRFs

If $\mathcal{F} = \{F_K : \{0,1\}^n \to \{0,1\}^n\}_{K\in\{0,1\}^\lambda}$ is a PRF family, $\mathrm{Tag}(K, m) = F_K(m)$ is UFCMA-secure for FIL.

Proof. Let $G(\lambda) := \mathrm{Game}^{\mathrm{UFCMA}}_{\Pi,A}(\lambda)$. Let $H(\lambda)$ be the hybrid distribution obtained by replacing $F_K(\cdot) \in \mathcal{F}$ with $R(\cdot) \in \mathcal{R}(n \to n)$ in the game $G(\lambda)$ (in other words, we're using a randum function R for the tag instead of F_K).

Proceeding in the usual way with a simultaneous game, we can reduce the distringuishability of F_K from R to the distinguishability of $G(\lambda)$ from $H(\lambda)$ (omitted, similar to Claim 1 of Theorem 4.2). This concludes that $G(\lambda) \approx_c H(\lambda)$.

Finally, since $H(\lambda)$ uses a truly random function R, the probability of an adversary being able to guess the correct tag for a message is at most 2^{-n}, concluding that Π is UFCMA-secure. □

But what about VIL tags? What about inputs of longer FIL? Both questions fall into the idea of **domain expansion** for MACs. It's easy to see that simple tricks don't work. For instance, consider the tag function that concatenes the tags of the blocks, i.e. $\mathrm{Tag}(K, m) = \tau$ such that $m = (m_1, \ldots, m_d)$ and $\tau = (\tau_1, \ldots, \tau_d)$. The UFCMA game requires that message in the challenge query is different from the messages in the tag queries. However, a simple trick suffices to "break" this constraint:

1. The adversary queries $m_1 \,||\, m_2$ and gets $\tau_1 \,||\, \tau_2$
2. The adversary then queries $m_3 \,||\, m_4$ and gets $\tau_3 \,||\, \tau_4$
3. The adversary challenges with the pair $(m_1 \,||\, m_3, \tau_1 \,||\, \tau_3)$ and wins with probability 1

Consider now the tag function that XORes the blocks and returns a single tag, i.e. $\mathrm{Tag}(K, \bigoplus_{i=1}^{d} m_i) = \tau$. It's easy to see that this construction still doesn't work:

1. The adversary queries $m_1 \,||\, m_1$ and gets τ_1
2. The adversary challenges with the pair $(m_2 \,||\, m_2, \tau_1)$ and wins with probability 1

It seems that our construction cannot be easily changed to allow VIL or longer FIL inputs. However, we still have a trick up our sleeve: we can hash an input of length nd and reduce it to length n, preserving the UFCMA security of the construction. In other words, we have that $\mathrm{Tag}(K, m) = F_K(h_s(m))$ with $h_s \in \mathcal{H}$, for an hash family $\mathcal{H} = \{h_s : \{0,1\}^{nd} \to \{0,1\}^n\}_{s\in\{0,1\}^\lambda}$.

Does this idea work with every type of hash family? Inuitively, no. The hash family should be such that every hash function hardly collides on different inouts, meanin that it should

be hard to have two messages m, m' with $m \neq m'$ such that $h_s(m) = h_s(m')$. Depending on the degree of hardness, these hash families are called **ϵ-almost universal**.

Definition 4.11: Universal hash families

Let $\mathcal{H} = \{h_s : \{0,1\}^{nd} \to \{0,1\}^n\}_{s \in \{0,1\}^\lambda}$ be a family of hash functions. We say that $\mathcal{H}$ is ε-almost universal if $\forall m, m' \in \mathcal{M}$ with $m \neq m'$ it holds that:

$$\Pr_{s \in_R \{0,1\}^\lambda}[h_s(m) = h_s(m')] \leq \varepsilon$$

When $\varepsilon = \text{negl}(\lambda)$, the family is said to be *almost universal (AU)*. When $\varepsilon = 2^{-n}$, the family is said to be *perfectly universal (PU)*.

As the name suggests, PU hash families are considered *perfect* in a sense that it is practically impossible to get a collision. Nonetheless, for our purposes AU hash families are enough. In fact, it can be proven that composing them with a PRF yields another PRF.

Theorem 4.4: Composition of PRFs with AU hash families

Given a PRF family $\mathcal{F} = \{F_k : \{0,1\}^n \to \{0,1\}^n\}$ and an AU hash family $\mathcal{H} = \{h_s : \{0,1\}^{nd} \to \{0,1\}^n\}_{s \in \{0,1\}^\lambda}$, the family $\mathcal{F}(\mathcal{H})$ defined as:

$$\mathcal{F}(\mathcal{H}) = \{F_K \circ h_s : \{0,1\}^{nd} \to \{0,1\}^n\}_{(K,s) \in \{0,1\}^{2\lambda}}$$

is a PRF family.

Proof. Consider the two games $G(\lambda, b) := \text{Game}^{\text{PRF}}_{\mathcal{F}(\mathcal{H}),A}(\lambda, b)$ with $b \in \{0,1\}$. We recall that $G(\lambda, 0)$ is the PRF game where we use a function $F_K(h_s(\cdot)) \in \mathcal{F}(\mathcal{H})$, while $G(\lambda, 1)$ is the PRF game where we use a truly random function $R \in \mathcal{R}(nd \to n)$.

Ler $H(\lambda)$ be the hybrid distribution that plays the PRF game with the function $R'(h_s(\cdot))$, where $R' \in \mathcal{R}(n \to n)$. It can be proven that $G(\lambda, 0) \approx_c H(\lambda)$ in the standard way by reducing the distringuishability of F_K from R' to the distringuishability of $G(\lambda, 0)$ from $H(\lambda)$.

Therefore, it remains to show that $H(\lambda) \approx_c G(\lambda, 1)$. Let $m_1, \ldots, m_t$ be the encryption queries made by the adversary, where $t = \text{poly}(\lambda)$. Let B be the bad event such that $B = 1$ if $\exists i, j \in [t]$ with $i \neq j$ for which $h_s(m_i) = h_j(m_j)$. When conditioning on $B = 0$, no collisions occurr, thus :

$$\text{SD}(H(\lambda); G(\lambda, 1)) \leq \Pr[B = 1]$$

Through the union bound, we have that:

$$\begin{aligned}\Pr[B=1] &= \Pr_{s\in_R\{0,1\}^R}[\exists i,j\in[t]\ i\neq j \text{ s.t. } h_s(m_i)=h_s(m_j)]\\ &\leq \sum_{i=1}^{t}\sum_{\substack{j=0:\\ j\neq i}}^{t}\Pr_{s\in\{0,1\}^\lambda}[h_s(m_i)=h_s(m_j)]\\ &= \binom{t}{2}\varepsilon\\ &= \text{negl}(\lambda)\end{aligned}$$

Therefore, $H(\lambda)\approx_c G(\lambda,1)$. □

Surprisingly, it is very easy to construct universal hash families, in particular through *Galois fields*. We give two examples:

1. Let $\mathbb{F}=\text{GF}(2^n)$. Let $m\in\mathbb{F}^d$ be the input and let $s\in\mathbb{F}^d$ be the hash seed, where $m_i,s_i\in\mathbb{F}$. We define each $h_s(\cdot)$ as the scalar product between the input and the hash seed:

$$h_s(m)=\langle s,m\rangle=\sum_{i=1}^{d}s_im_i$$

   Then, given $m,m'\in\mathbb{F}^d$ with $m\neq m'$, we have that:

$$\begin{aligned}h_s(m)=h_s(m') &\iff \langle s,m\rangle=\langle s,m'\rangle\\ &\iff \sum_{i=1}^{d}s_im_i=\sum_{i=1}^{d}s_im_i'\\ &\iff \sum_{i=1}^{d}s_i(m_i-m_i')=0\\ &\iff \sum_{i=1}^{d}s_i\delta_i=0\end{aligned}$$

   where $\delta_i=m_i-m_i'$ for each $i\in[d]$. Since $m\neq m'$, there is at least one index $j\in[d]$ such that $m_j\neq m_j'$, therefore $\delta_j\neq 0$:

$$\begin{aligned}h_s(m)=h_s(m') &\iff \sum_{i=1}^{d}s_i\delta_i=0\\ &\iff s_j=-\frac{1}{\delta_j}\sum_{\substack{i=1:\\ j\neq i}}^{d}s_i\delta_i\end{aligned}$$

   which is true with probability at most 2^{-d}. This concludes that $\mathcal{H}$ is a perfectly universal hash family.

2. Let $\mathbb{F} = \mathrm{GF}(2^n)$. Let $m \in \mathbb{F}^d$ be the input and let $s \in \mathbb{F}$ be the hash seed, where $m_i, s_i \in \mathbb{F}$. We define each $h_s(\cdot)$ as the d-degree polynomial $Q_m(s) \in \mathbb{F}[s]_d$ with coefficients $m_1, \ldots, m_d$ and variable s:

$$h_s(m) = Q_m(s) = \sum_{i=1}^{d} m_i s^{i-1}$$

Then, given $m, m' \in \mathbb{F}^d$ with $m \neq m'$, we have that:

$$\begin{aligned} h_s(m) = h_s(m') &\iff Q_m(s) = Q_{m'}(s) \\ &\iff \sum_{i=1}^{d} m_i s^{i-1} = \sum_{i=1}^{d} m'_i s^{i-1} \\ &\iff \sum_{i=1}^{d} (m_i - m'_i) s^{i-1} = 0 \end{aligned}$$

which is true with probability at most $(d-1)2^{-n}$. This concludes that $\mathcal{H}$ is an almost universal hash family.

A more complex type of almost universal hash family can be obtained through encryption modes. For instance, the CBC encryption mode can be modified to define an AU hash family usually referred to as **CBC-MAC**.

Definition 4.12: CBC-MAC

Let $\mathcal{F} = \{F_k : \{0,1\}^n \to \{0,1\}^n\}_{k \in \{0,1\}^\lambda}$ be a PRF family. We define the CBC-MAC family as $\mathcal{H}_{\mathrm{CBC}} = \{h_s : \{0,1\}^n \to \{0,1\}^{nd}\}_{s \in \{0,1\}^\lambda}$, where:

$$h_s(m_1, \ldots, m_d) = F_s(m_d \oplus F_s(m_{d-1} \oplus F_s(\ldots m_2 \oplus F_s(m_1)) \ldots))$$

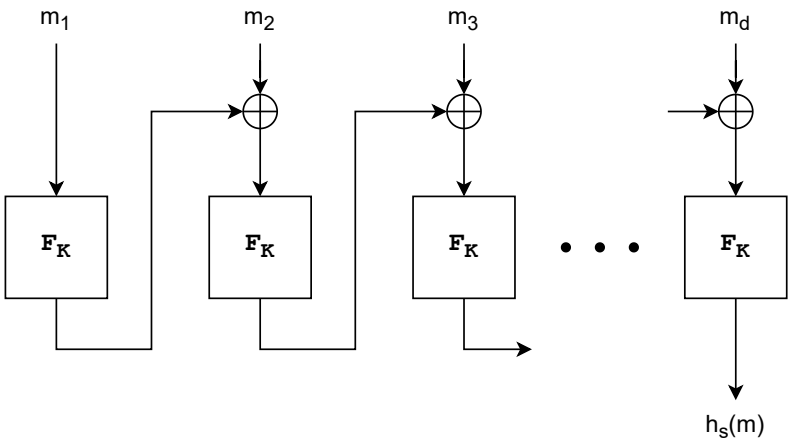

Figure 4.13: Graphical representation of CBC-MAC.

The CBC-MAC family has many properties. First of all, it can be easily proven that $\mathcal{H}_{\text{CBC}}$ is also a PRF family through a reduction argument. In a similar way, it can be proven that it is also UFCMA-secure, but only for FIL. Nonetheless, Theorem 4.3 and Theorem 4.4 still imply that $\mathcal{F}(\mathcal{H}_{\text{CBC}})$ is an UFCMA-secure MAC scheme.

Theorem 4.5

Let $\mathcal{F}$ be a PRF family and let $\mathcal{H}_{\text{CBC}}$ be the corresponding CBC-MAC hash family. Then, it holds that:

- $\mathcal{H}_{\text{CBC}}$ is a PRF
- $\mathcal{H}_{\text{CBC}}$ is AU for FIL
- $\mathcal{F}(\mathcal{H}_{\text{CBC}})$ is a PRF
- $\mathcal{F}(\mathcal{H}_{\text{CBC}})$ is UFCMA-secure

Proof. Omitted. □

4.5 CCA-security and non-malleability

In the previous sections we gave the idea behind computationally secure encryption and computationally secure message authentication. Now, we're ready to combine the two concepts. We say that an encryption scheme is **malleable** if it is possible to transform a ciphertext $c \in \mathcal{C}$ – of which we don't know the original plaintext – into another ciphertext $\widetilde{c}$ which decrypts to another plaintext related to the original one. Formally, given an encryption c of a plaintext m, it is possible to generate another cyphertext $\widetilde{c}$ of another plaintext $\widetilde{m}$ that is in some way related to m.

For instance, consider again the PRG-OTP scheme, where $\text{Enc}(K, m) = G(K) \oplus$ for some PRG G. Suppose that we know a ciphertext $c \in \mathcal{C}$, without knowing the original message m. Then, for any value $t \in \{0,1\}^n$ we can construct the ciphertext $\widetilde{c}$ of the message $\widetilde{m} = m \oplus t$ by XORing c with t:

$$\widetilde{c} = c \oplus t = \text{Enc}(K, m) \oplus t = (G(K) \oplus m) \oplus t = G(K) \oplus (m \oplus t) = \text{Enc}(K, \widetilde{m})$$

Malleability is often an undesirable property in a general-purpose cryptosystem (e.g. digital auctions), since it allows an attacker to modify the contents of a message: Eve could steal the ciphertext c from Alice, compute $\widetilde{c}$ and send it to Bob, which will decrypt it as $\widetilde{m}$. It's easy to see that even CPA-secure SKEs may be malleable. For instance, consider again the PRF-OTP scheme, where:

1. $\text{Enc}(K, m) = (r, F_K(r) \oplus m)$, where $r \in \{0,1\}^n$ is a nonce
2. $\text{Dec}(K, (r, c)) = F_K(r) \oplus c$

for a PRF family $\mathcal{F} = \{F_K : \{0,1\}^n \to \{0,1\}^n\}_{K \in \{0,1\}^\lambda}$. We already discussed how this scheme is CPA-secure. Consider now the ciphertext $(r, c) \in \{0,1\}^n \times \mathcal{C}$ of an unknown

message m, but the same trick used for PRG-OTP works even in this case. Let $\tilde{c} = c \oplus t$ for any $t \in \{0,1\}^n$. Then, we have that

$$(r, \tilde{c}) = (r, c \oplus t) = (r, (F_k(r) \oplus m) \oplus t) = (r, F_k(r) \oplus (m \oplus t)) = \text{Enc}(K, \widetilde{m})$$

where $\widetilde{m} = m \oplus t$.

In the computational security context, *malleability attacks* are known as **Chosen-ciphertext Attack (CCA)**. Formally, CCA-security is defined through the game $\text{Game}^{\text{CCA}}_{\Pi,A}(\lambda, b)$, similar to the one for CPA-security with the addition of *decryption queries* after each series of encryption queries (both before and after the sending the challenge messages). Obviously, the ciphertext sent in each decryption query must be strictly different from the challenge ciphertext, otherwise the attacker could win the game with probability 1.

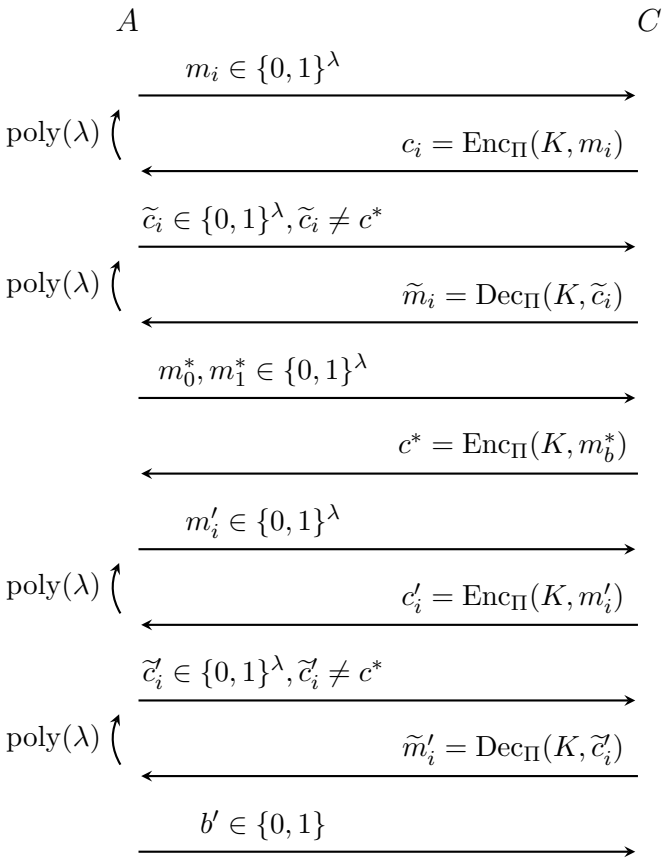

Figure 4.14: Graphical representation of $\text{Game}^{\text{CCA}}_{\Pi,A}(\lambda, b)$.

Definition 4.13: CCA-security

Let $\Pi = (\text{Enc}, \text{Dec})$ be a SKE. We say that Π is CCA-secure if:

$$\text{Game}^{\text{CCA}}_{\Pi,A}(\lambda, 0) \approx_c \text{Game}^{\text{CCA}}_{\Pi,A}(\lambda, 1)$$

for all PPT adversaries A.

Since the CCA-game is way more complex than the CPA-game due to the presence of decryption queries, we need a simpler procedure to prove that a scheme is CCA-secure. In particular, we'll prove that CPA-security together with an additional property implies CCA-security. This additional property is the computational security equivalent of **Integrity (INT)**: it must be hard for any adversary PPT to generate a *valid* chipertext without knowing the key chosen by the scheme, otherwise they could use this ability to alter the original message and forge a new chipertext that decrypts into the altered message.

In $\text{Game}^{\text{INT}}_{\Pi,A}(\lambda)$, the adversary is given access to an oracle for the Enc function and it is allowed to make $t = \text{poly}(\lambda)$ encryption queries to the oracle. After completing the encryption queries, the adversary returns a chipertext c^* to the challenger (with $c^* \neq c_i$ for all $i \in [t]$). The challenger then computes $\text{Dec}(K, c^*)$ and compares it with a special value $\perp$, which can be thought of as a "fake message". If $\text{Dec}(K, c^*) \neq \perp$ is true, the adversary wins. Otherwise, the challenger wins.

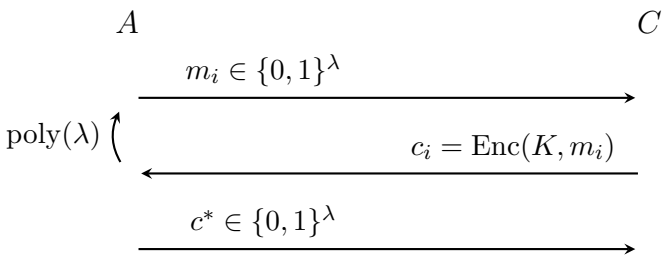

Figure 4.15: Graphical representation of $\text{Game}^{\text{INT}}_{\Pi,A}(\lambda)$

Let's solidify the idea behind the INT game: if the adversary is able to forge a ciphertext that decrypts into a message m that is different from the fake message $\perp$, we know that they are clearly capable of forging a valid chipertext even if they don't know the key, breaking the integrity of the scheme.

Definition 4.14: Integrity (in computational security)

Let $\Pi = (\text{Enc}, \text{Dec})$ be a SKE. We say that Π satisfies integrity if:

$$\Pr[\text{Game}^{\text{INT}}_{\Pi,A}(\lambda) = 1] \leq \text{negl}(\lambda)$$

for all PPT adversaries A.

Theorem 4.6: CPA+INT implies CCA

Let $\Pi = (\text{Enc}, \text{Dec})$ be a SKE. Then, Π is CCA-secure if it is CPA-secure and satisfies integrity.

Proof (sketch). The full proof requires to be extremely formal, hence we'll only give a sketch. The idea is simple: we can reduce both the distinguishability of $\text{Game}^{\text{CPA}}_{\Pi,A}(\lambda, b)$

and a win of $\text{Game}^{\text{INT}}_{\Pi,A}(\lambda)$ to the distinguishability of $\text{Game}^{\text{CCA}}_{\Pi,A}(\lambda, b)$.

We start with the CPA-game. Let $H(\lambda, b)$ be the game identical to the CCA-game where decryption queries are substituted by the following logic: if the attacker queries a ciphertext $\widetilde{c}_i$ that was the answer to a previous encryption query m_i, the challenger returns m_i, otherwise it returns the special message $\perp$.

Claim 1: $H(\lambda, 0) \approx_c H(\lambda, 1)$

Proof of Claim 1. By way of contradiction, suppose that there is an adversary A that is able to distinguish the CCA-game with non-negligible probability. Let B be the adversary defined as follows:

1. B starts the hybrid game with A and the CPA game with C
2. A sends a series of encryption queries to B. Each query m_i gets forwarded to the challenger C, which will answer with a chipertext c_i. The ciphertext is then forwarded from B to A.
3. A sends a series of decryption queries to B. For each query $\widetilde{c}_i$, B returns m_i if $(m_i, \widetilde{c}_i)$ is a query-answer pair of the previous encryption queries, otherwise it returns $\perp$
4. A sends the challenge messages m_0^*, m_1^* to B, which then forwards them to C. Afterwards, C replies with c^*, which gets forwarded back to A by B.
5. A and B play another series of encryption queries followed by decryption queries, using the same rules as the previous ones.
6. A makes a guess by sending a bit $b' \in \{0, 1\}$ to B, which forwards it to C. If $b' = b$, B wins the game

We observe that $H(\lambda, b)$ is basically the CCA game where we "fix" the decryption queries, "collapsing" back to the CPA game. Therefore, the above reduction concludes that $H(\lambda, 0) \approx_c H(\lambda, 1)$, otherwise we could distinguish the CPA game. □

We're now left with proving that $\text{Game}^{\text{CCA}}_{\Pi,A}(\lambda, b) \approx_c H(\lambda, b)$.

Claim 2: $\text{Game}^{\text{CCA}}_{\Pi,A}(\lambda, b) \approx_c H(\lambda, b)$ for $b \in \{0, 1\}$

Proof of Claim 2. We define a reduction identical to one of Claim 1 but with a small difference: instead of forwarding both challenge messages m_0^*, m_1^*, we'll forward only m_b^*. By way of contradiction, suppose that there is an adversary A' that is able to distinguish $\text{Game}^{\text{CCA}}_{\Pi,A}(\lambda, b)$ from $H(\lambda, b)$ with non-negligible probability. A' returns 0 when it recognizes the CCA game, otherwise it returns 1 when it recognizes the hybrid game.

We define an adversary B' for the INT game:

1. B' starts both games with A' and C'
2. A' sends a series of encryption queries to B'. Each query m_i gets forwarded to the challenger C', which will answer with a chipertext c_i. The ciphertext is then forwarded from B' to A'.

3. A' sends a series of decryption queries to B'. For each query $\widetilde{c}_i$, B' returns m_i if $(m_i, \widetilde{c}_i)$ is a query-answer pair of the previous encryption queries, otherwise it returns $\perp$
4. A' sends the challenge messages m_0^*, m_1^* to B', which then forwards them to C'. Afterwards, C' replies with c^*, which gets forwarded back to A' by B'.
5. A' and B' play another series of encryption queries followed by decryption queries, using the same rules as the previous ones.
6. A' makes a guess by sending a bit $b' \in \{0, 1\}$ to B', which forwards it to C'. If $b' = b$, B' wins the game

The above forms a reduction from a win of $\text{Game}^{\text{INT}}_{\Pi,A'}(\lambda)$ to the distinguishability of $\text{Game}^{\text{CCA}}_{\Pi,A'}(\lambda, b)$ from $H(\lambda, b)$. In particular, let E be the event such that $E = 1$ if A' makes a decryption query $\widetilde{c}$ that is both different from all the answers to the encryption queries and such that $\text{Dec}(K, \widetilde{c}) \neq \perp$. When conditioning on $E = 0$, we get that $H(\lambda, b) \equiv \text{Game}^{\text{CCA}}_{\Pi,A}(\lambda, b)$, therefore for A' it holds that:

$$\Pr[\text{Game}^{\text{IND}}_{\Pi,A'}(\lambda) = 1] \geq \Pr[E = 0] \Pr[A' \text{ causes } E = 0 \mid E = 0] \geq \frac{1}{s} \cdot \frac{1}{n^c}$$

where s is the number of decryption queries, contradicting the fact that Π satisfies integrity. □

□

The very next natural thing to inquire is a SKE that satisfies both CPA-security and integrity, in order to obtain a CCA-secure scheme. Not so surprisingly, combining a CPA-secure scheme with a secure MAC satisfies our needs. First, let's see in which ways we could combine these two ideas. Let $\Pi_1 = (\text{Enc}, \text{Dec})$ be an SKE with key space $\mathcal{K}_1$ and let $\Pi_2 = (\text{Tag})$ be a MAC with key space $\mathcal{K}_2$. We define the following ways to combine the two schemes:

1. **Encrypt-and-MAC (E&M)**: we define a new SKE $\Pi' = (\text{Enc}', \text{Dec}')$ as:

$$\text{Enc}'((K_1, K_2), m) = (\text{Enc}(K_1, m), \text{Tag}(K_2, m))$$

$$\text{Dec}'((K_1, K_2), (c, \tau)) = \begin{cases} \perp & \text{if } \text{Tag}(K_2, \text{Dec}(K_1, m)) \neq \tau \\ \text{Dec}(K_1, m) & \text{otherwise} \end{cases}$$

   where $(K_1, K_2) \in \text{K}_1 \times \text{K}_2$.

2. **Encrypt-then-MAC (EtM)**: we define a new SKE $\Pi' = (\text{Enc}', \text{Dec}')$ as:

$$\text{Enc}'((K_1, K_2), m) = (\text{Enc}(K_1, m), \text{Tag}(K_2, \text{Enc}(K_1, m)))$$

$$\text{Dec}'((K_1, K_2), (c, \tau)) = \begin{cases} \perp & \text{if } \text{Tag}(K_2, c) \neq \tau \\ \text{Dec}(K_1, c) & \text{otherwise} \end{cases}$$

   where $(K_1, K_2) \in \text{K}_1 \times \text{K}_2$.

3. **MAC-then-Encrypt (MtE)**: we define a new SKE $\Pi' = (\text{Enc}', \text{Dec}')$ as:

$$\text{Enc}'((K_1, K_2), m) = \text{Enc}(K_1, m \,||\, \text{Tag}(K_2, m))$$

$$\text{Dec}'((K_1, K_2), c) = \begin{cases} \perp & \text{if } \text{Tag}(K_2, m) \neq \tau \\ m & \text{otherwise} \end{cases}$$

where $(K_1, K_2) \in \text{K}_1 \times \text{K}_2$ and $\text{Dec}(K_1, c) = m \,||\, \tau$.

In real-world implementations, Encrypt-and-MAC was used by old versions of the *SSL/TLS protocol*, while new versions use Encrypt-then-MAC. Mac-then-Encrypt is used by the *SSH protocol*. Out of all the three constructions, only EtM guarantees that the resulting scheme satisfies both CPA-security and integrity when Π_1 is CPA-secure and Π_2 is **Strong UFCMA (SUFCMA)**. The SUFCMA game is defined in the same way as the UFCMA game, with the additional constraint imposing that $\tau^* \neq \tau_i$ for all $i \in [t]$ (we recall that the UFCMA game only imposes that $m^* \neq m_i$)

Definition 4.15: SUFCMA-security

Let $\Pi = (\text{Tag})$ be a MAC. We say that Π is SUFCMA-secure if:

$$\Pr[\text{Game}^{\text{SUFCMA}}_{\Pi,A}(\lambda) = 1] \leq \text{negl}(\lambda)$$

for all PPT adversaries A.

Theorem 4.7

Let $\Pi' = (\text{Enc}', \text{Dec}')$ be an EtM SKE constructed from the SKE $\Pi_1 = (\text{Enc}, \text{Dec})$ and the MAC $\Pi_2 = (\text{Tag})$. Then, Π' satisfies both CPA-security and integrity if Π_1 is CPA-secure and Π_2 is SUFCMA-secure.

Proof. As usual, we proceed with two reductions: one that goes from one CPA-security to the other and one that goes from integrity to SUFCMA-security.

Claim 1: Π' is CPA-secure

Proof of Claim 1. By way of contradiction, suppose that there is an adversary A' such that $\text{Game}^{\text{CPA}}_{\Pi',A'}(\lambda, 0) \not\approx_c \text{Game}^{\text{CPA}}_{\Pi',A'}(\lambda, 1)$. We define a new adversary A_1 to distinguish $\text{Game}^{\text{CPA}}_{\Pi_1,A_1}(\lambda, 0)$ from $\text{Game}^{\text{CPA}}_{\Pi_1,A_1}(\lambda, 1)$:

1. A_1 picks a key $K_2 \in_R \mathcal{K}_\in$. This key will be used to compute tags (observe that the challenger C_1 also picks a key $K_1 \in_R \mathcal{K}_\infty$, which will be used for encryptions)
2. A_1 starts the CPA game both with A' and C_1
3. Each encryption query m_i from A' to A_1 is forwarded to C_1, which replies with the ciphertext c_i
4. Upon receiving c_i, A_1 computes $\text{Tag}(K_2, c_i) = \tau_i$ and returns (c_i, τ_i) to A'

5. After a polynomial amount of encryption queries, A' sends two challenge messages m_0^*, m_1^* to A_1, who will forward them to C_1. The challenger then replies with the chipertext c^*.
6. Upon receiving c^*, A_1 computes $\text{Tag}(K_2, c^*) = \tau^*$ and returns (c^*, τ^*) to A'
7. After another exchange of a polynomial amount of encryption queries, A' guesses the value of b by sending $b' \in \{0, 1\}$. This bit is then forwarded by A_1 to C_1
8. If $b' = b$, A_1 wins the CPA game against C_1, otherwise C_1 wins.

The above reduction has no particular instruction that must be discusses: A_1 just acts as an intermediate between A' and C_1, while also tagging the chipertexts sent by C_1. Therefore, A_1 is able to distinguish the CPA game for Π_1 if and only if A' is able to do so for Π', raising a contradiction on the CPA-security of Π_1. □

Claim 2: Π' satisfies integrity

Proof of Claim 2. By way of contradiction, suppose that there is an adversary A'' such that $\Pr[\text{Game}^{\text{INT}}_{\Pi', A''}(\lambda) = 1] \geq \lambda^{-c}$ for some $c > 0$. We define a new adversary A_2 such that $\Pr[\text{Game}^{\text{SUFCMA}}_{\Pi_2, A_2}(\lambda) = 1] \geq \lambda^{-c}$:

1. A_2 picks a key $K_1 \in_R \mathcal{K}_\infty$. This key will be used to compute encryptions (observe that the challenger C_2 also picks a key $K_2 \in_R \mathcal{K}_\in$, which will be used for taggging)
2. A_2 starts the INT game with A'' and the SUFCMA game with C_2
3. Each encryption query m_i from A'' to A_2 is forwarded to C_2, which replies with the tag τ_i.
4. Upon receiving τ_i, A_2 computes $\text{Enc}(K_1, m_1) = c_i$ and returns (c_i, τ_i) to A''
5. After a polynomial amount of encryption queries, A'' sends the challenge pair (c^*, τ^*) to A_2, who will forward it to C_2.
6. If $\text{Tag}(K_2, c^*) = \tau^*$ and $c^* \neq c_i$ for all c_i, A_2 wins the SUFCMA game against C_2, otherwise C_2 wins

Unlike the previous claim, this reduction requires some discussion. First of all, we observe that in order to play a valid INT game the adversary A'' must pick a challenge pair (c^*, τ^*) such that $(c^*, \tau^*) \neq (c_i, \tau_i)$ for every encryption query m_i. However, this can hold true both when $c^* \neq c_i$ and when $\tau^* \neq \tau_i$ (or both). This is where SUFCMA-security plays a crucial role: we know that each tag τ_i returned by the challenger C_2 is unique.

Therefore, when (c^*, τ^*) is such that $\text{Dec}'((K_1, K_2), (c^*, \tau^*)) \neq \perp$ we know that both $\text{Tag}(K_2, c^*) = \tau^*$ (guaranteed since the decryption works) and $c^* \neq c_i$ for all c_i hold (guaranteed by the uniqueness of tags), declaring the adversary A_2 as winner. Otherwise, if $\text{Dec}'((K_1, K_2), (c^*, \tau^*)) = \perp$ then $\text{Tag}(K_2, c^*) \neq \tau^*$ must hold, declaring the challenger C_2 as winner. □

□

Corollary 4.1

Let $\Pi' = (\text{Enc}', \text{Dec}')$ be an EtM SKE constructed from the SKE $\Pi_1 = (\text{Enc}, \text{Dec})$ and the MAC $\Pi_2 = (\text{Tag})$. Then, Π' is CCA-secure if Π_1 is CPA-secure and Π_2 is SUFCMA-secure.

Proof. Follows from the two previous theorems. □

4.6 Block ciphers and Feistel networks

We saw how to get CPA-secure SKEs and UFCMA-secure MACs through PRFs. To get SUFCMA-secure MACS, small changes to the given construction suffice (e.g. using PRFs with unique outputs, such as PRPs). By combining these two constructions in an Encrypt-then-Mac fashion, we finally get our CCA-secure construction, proving that excellent cryptography symmetric-key encryption is theoretically possible. However, we're still missing a link: we need to show that PRP can be constructed.

PRPs are useful not only for their output uniqueness, but also for their invertibility. In particular, real-world cryptosystems are based on encryption schemes whose decryption function is exacly the inverse of the encryption function. Since the encryption must be efficient both in theory and practice, decryptions based on PRPs also achieve this property.

In practical applications, PRPs are referred to as **block ciphers**. The two most common block ciphers are known as *Data Encryption Standard (DES)* and *Advances Encryption Standard (AES)*. We remark that neither of these block ciphers can be really proven to be secure under standard assumptions (recall that we don't know if PRFs really do exist in practice). However, their design is high inspired by standard results in provable security. The DES block cipher is based on the theoretical notion of **Feistel networks**, a way to construct a PRP family from any PRF family.

Definition 4.16: Feistel function

Given a function $f : \{0,1\}^n \to \{0,1\}^n$, the Feistel function over f is the function $\psi_f : \{0,1\}^{2n} \to \{0,1\}^{2n}$ such that:

$$\psi_f(x, y) = (y, x \oplus f(y))$$

with $x, y \in \{0,1\}^n$

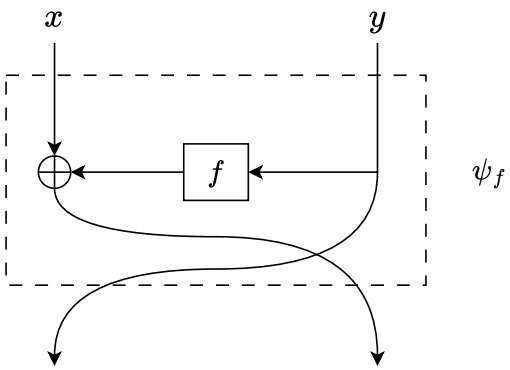

Figure 4.16: A feistel function.

By definition, the Feistel function is easily invertible. By setting $\psi_f^{-1}(x', y') = (y' \oplus f(x'), x')$, we get that:

$$\psi_f^{-1}(\psi_F(x, y)) = \psi_f^{-1}(y, x \oplus f(y)) = (x \oplus f(y) \oplus f(y), y) = (x, y)$$

Nonetheless, the Feistel function is not pseudorandom by itself even when $f = F_K$ for some PRF family $\mathcal{F} = \{F_k : \{0,1\}^n \to \{0,1\}\}_{K \in \{0,1\}^n}$: since the second half of the input is equal to the first half of the output, an attacker can easily distinguish it from a truly random function. The strength of the Feistel function comes from repeated applications, using different pseudorandom functions. These repeated applications are called **Feistel networks**.

Definition 4.17: Feistel network

Given t functions $f_1, \ldots, f_t : \{0,1\}^n \to \{0,1\}^n$, the Feistel network over $f_1, \ldots, f_t$ is the function $\psi_{f_1,\ldots,f_t} : \{0,1\}^{2n} \to \{0,1\}^{2n}$ such that:

$$\psi_{f_1,\ldots,f_t}(x, y) = \psi_{f_t}(\psi_{f_{t-1}}(\ldots \psi_{f_1}(x, y)))$$

with $x, y \in \{0,1\}^n$

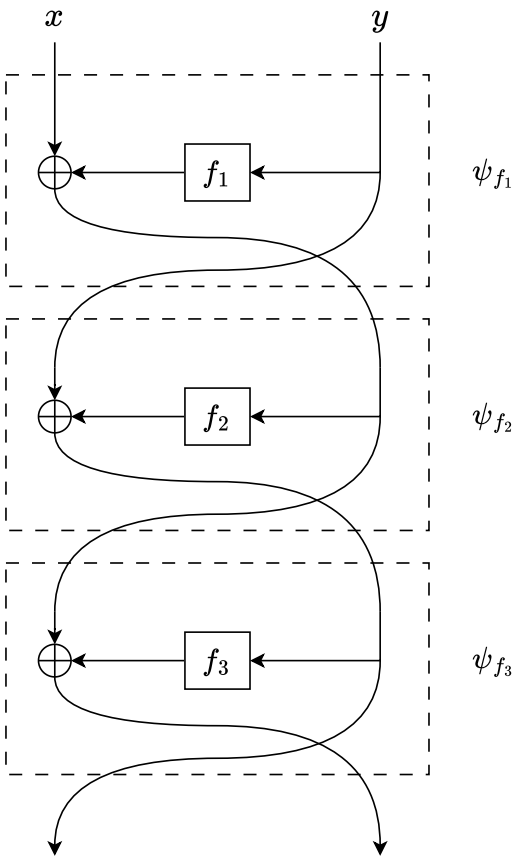

Figure 4.17: A 3-layer Feistel network.

Is a 2-layer Feistel network enough over independent PRFs to get a PRP? The answer is still no, but they almost suffice. Consider two keys $K_1, K_2 \in \{0,1\}^\lambda$. Then, for any two pairs $(x, y), (x', y') \in \{0,1\}^{2n}$ we have that:

$$\begin{aligned}
&\psi_{F_{K_2},F_{K_1}}(x,y) \oplus \psi_{F_{K_2},F_{K_1}}(x',y') \\
=&\psi_{F_{K_2}}(y, x \oplus F_{K_1}(y)) \oplus \psi_{F_{K_2}}(y', x' \oplus F_{K_1}(y')) \\
=&(x \oplus F_{K_1}(y), y \oplus F_{K_2}(x \oplus F_{K_1}(y))) \oplus (x' \oplus F_{K_1}(y'), y' \oplus F_{K_2}(x' \oplus F_{K_1}(y'))) \\
=&(x \oplus x', y \oplus y' \oplus F_{K_2}(x \oplus F_{K_1}(y)) \oplus F_{K_2}(x' \oplus F_{K_1}(y')))
\end{aligned}$$

implying that an attacker can just query $(x, y), (x', y')$ and then check if the first half of the output is $x \oplus x'$ to easily distinguish $\psi_{F_{K_2},F_{K_1}}$ from a truly random function. Hence, we strictly require *at least* 3 applications: after three applications, both halves of the output will be XORed with a PRF, making them hard to distinguish (recall that the adversary doesn't have access to the key $(K_1, K_2, K_3) \in \{0,1\}^{3\lambda}$ used by the challenger). To formally prove this result, the following lemma is required, for which we omit the proof.

Lemma 4.2

Let H and H' be the two distributions defined as follows:

- H is given by $\psi_{R_1,R_2}(\cdot,\cdot)$ where $R_1, R_2 \in_R \mathcal{R}(n \to n)$
- H' is given by $R(\cdot,\cdot)$ where $R \in_R \mathcal{R}(2n \to 2n)$

Assuming all queries $(x_1, y_1), \ldots, (x_q, y_q)$, with $q = \text{poly}(\lambda)$, are "y-unique", i.e. $y_i \neq y_j$ for all $i \neq j$, for every unbounded attacker A the distributions S and R are computationally indistinguishable.

Proof. Omitted. □

Theorem 4.8: Luby-Rackoff theorem

Let $\mathcal{F} = \{F_k : \{0,1\}^n \to \{0,1\}^n\}_{K \in \{0,1\}^\lambda}$ be a PRF family. Then, for any three independent keys $K_1, K_2, K_3 \in_R \{0,1\}^\lambda$ the 3-layer Feistel network $\psi_{F_{K_1},F_{K_2},F_{K_3}}$ is a PRP.

Proof. Let H_1, H_2, H_3, H_4 be the four hybrid distribution defined as follows:

- H_1 is given by $\psi_{F_{K_1},F_{K_2},F_{K_3}}(\cdot,\cdot)$ with $K_1, K_2, K_3 \in_R \{0,1\}^\lambda$
- H_2 is given by $\psi_{R_1,R_2,R_3}(\cdot,\cdot)$ with $R_1, R_2, R_3 \in_R \mathcal{R}(n \to n)$
- H_3 is given by $R(\cdot,\cdot)$ with $R \in_R \mathcal{R}(2n \to 2n)$
- H_4 is given by $P(\cdot,\cdot)$ with $P \in_R \mathcal{P}(2n \to 2n)$, where $\mathcal{P}$ is the set of truly random permutations on $\{0,1\}^{2n}$

By definition, $\text{Game}^{\text{PRP}}_{\psi_{F_{K_1},F_{K_2},F_{K_3}},A}(\lambda, 0) \equiv H_1$ and $\text{Game}^{\text{PRP}}_{\psi_{F_{K_1},F_{K_2},F_{K_3}},A}(\lambda, 1) \equiv H_4$.

Claim 1: $H_1 \approx_c H_2$

Proof of Claim 1. Omitted (standard triple reduction to distinguishibility of PRF) □

Claim 2: $H_3 \approx_c H_4$.

Proof of Claim 2. Let B be the event such that $B = 1$ if $\exists i, j \in [q]$ with $i \neq j$ such that $R(x_i, y_i) = R(x_j, y_j)$. When conditioning over $B = 0$, we have that $H_3 \equiv H_4$ since R becomes bijective. Therefore:

$$\text{SD}(H_3; H_4) \leq \Pr[B = 1] = \sum_{i=1}^{q} \sum_{\substack{j=1: \\ j \neq i}}^{q} \Pr[R(x_i, y_i) = R(x_j, y_j)] \leq \binom{q}{2} \frac{1}{2^{2n}} \leq \text{negl}(n)$$

□

Claim 3: $H_2 \approx_c H_3$

Proof of Claim 3. We'll use the previous lemma to prove this claim. For the following argument, it is more convinient to look at H_4 as given by $\psi_{R_1,R_2}(\psi_{R_3}(\cdot,\cdot))$. Consider the queries $(x_1, y_1), \ldots, (x_q, y_q)$, where $(x_i, y_i) \neq (x_j, y_j)$.Let B' be the event such that $B = 1$ if the outputs $\psi_{R_3}(x_i, y_i) = (x'_i, y'_i)$ are not "y'-unique", i.e. there are at least two outputs whose second halves are the same.

By conditioning over $B = 0$, H_2 and H_3 are indistinguishable thanks to the previous lemma. Therefore:

$$\mathrm{SD}(H_2; H_3) \leq \Pr[B = 1]$$

Fix two indices $i, j \in [q]$ such that $i \neq j$. We may assume that $y_i \neq y_j$ since otherwise if $y_i = y_j$ then $x_i \neq x_j$ since $(x_i, y_i) \neq (x_j, y_j)$ holds. Then, we have that:

$$y'_i = x_i \oplus R_3(y_i) \neq x_j \oplus R_3(y_i) = x_j \oplus R_3(y_j) = y'_j$$

thus these queries don't affect $\Pr[B = 1]$. Now, we observe that:

$$\begin{aligned} y'_i = y'_j &\iff x_i \oplus R_3(y_i) = x_j \oplus R_3(y_j) \\ &\iff x_i \oplus x_i \oplus R_3(y_i) \oplus R_3(y_j) = x_i \oplus x_j \oplus R_3(y_j) \oplus R_3(y_j) \\ &\iff R_3 \oplus R_3(y_j) = x_i \oplus x_j \end{aligned}$$

Therefore, we conclude that:

$$\Pr[B = 1] = \sum_{i=1}^{q} \sum_{\substack{j=1: \\ j \neq i}}^{q} \Pr[y'_i = y'_j] \leq \binom{q}{2} \frac{1}{2^n} \leq \mathrm{negl}(n)$$

□

□

In practical implementations, the DES encryption scheme uses a 18-layer Fiestel network over an *heuristic* family instead of a PRF family (no PRF family has been proven to exist!), where the keys $K_1, \ldots, K_{18}$ are generated from a single key K. Even thought 18-layers may seem a lot, the DES scheme is actually very insecure due to the usage a key that is too short (56 bits), making it very easy to bruteforce by any modern computer. The easiest way to fix this issue is through multiple applications of DES (using independent keys): we're basically doubling the key length without changing the internal structure.

Nonetheless, 2-DES can also be broken through a meet-in-the-middle attack (finding the two keys by simultaneously encrypting known plaintext and decrypting the corresponding ciphertext to find matching intermediate values). Intuitively, the meet-in-the-middle attack works doesn't work for k-DES with $k > 2$ since the intermediate keys cannot be bruteforced.

The 3-DES encryption scheme is the most used out of all the variants of DES since 168 bits for the key length are enough for modern computers. However, 3-DES is very inefficient since it is basically a 54-layer Fiestel network. For this reason, the AES standard was created. This protocol solves the issues of DES by using a longer key (128, 192 or 256 bits) and a different internal network called *Substitution Permutation Network (SPN)*, which is basically the repeated application of XORs and publicly known permutations.

4.7 Collision-resistant hash families

When we discussed domain expansion for secure MACs, we saw how a PRF family $\mathcal{F}$ can be combined with an hash family $\mathcal{H}$ to get a new PRF family $\mathcal{F}(\mathcal{H})$ with an input length that can be way larger than the output length. The idea was to use hash families with inputs longer than the outputs in order to "compress" the input into a string whose size is compatibile with $\mathcal{F}$.

Naturally, an hash family with such property is subject to **collisions**: by the pidgeonhole principle, a function $h_s : \{0,1\}^{\ell(n)} \to \{0,1\}^n$ with $\ell(n) >> n$ is guaranteed inputs with the same output (actually, a lot of them). Collisions can be used by an attacker to infer information on the structure of the seed and the hash function itself. In our $\mathcal{F}(\mathcal{H})$ construction, this is not an issue since the resulting hashes are passed to a pseudorandom function, making collisions hard to find even for *unbounded* adversaries.

In many cases, however, we strictly require to use compressing hash families without passing the output through pseudorandom functions. These families are clearly insecure against unbounded adversaries: even when the seed is hidden, we can just test every input with every seed until we find collisions. Therefore, we require that our compressing hash families are **collision-resistant (CRH)**, meaning that collisions are hard to find by a *bounded* adversary. To ensure security, this hardness should depend only on the structure of the hash family and not on the secretness of the seed (in some applications, the seed is required to be public).

Let $\text{Game}^{\text{CRH}}_{\mathcal{H},A}(\lambda)$ be the CRH game, defined as follows:

1. The challenger sends a seed $s \in_R \mathcal{U}_\lambda$
2. The adversary returns two inputs $x, x' \in \mathcal{M}$ with $x \neq x'$
3. If $h_s(x) = h_s(x')$, the adversary wins. Otherwise, the challenger wins.

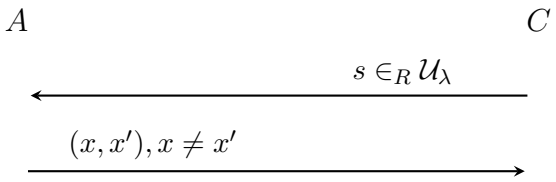

Figure 4.18: Graphical representation of $\text{Game}^{\text{CRH}}_{\mathcal{H},A}(\lambda)$

Definition 4.18: Collision-resistant hash

Let $\mathcal{H} = \{h_s : \{0,1\}^{\ell(n)} \to \{0,1\}^n\}_{s \in \{0,1\}^\lambda}$ be an hash family, where $\ell(n) >> n$. We say that $\mathcal{H}$ is a collision resistant hash when:

$$\Pr[\text{Game}^{\text{CRH}}_{\mathcal{H},A}(\lambda) = 1] \leq \text{negl}(\lambda)$$

for all PPT adversaries A.

In the theoretical context, assuming a CRH exists, the domain extension for hash families can be achieve. In particular, we study the **Merkle-Damgård transform**. The idea is to start with a CRH family and repeatedly apply it to obtain a CRH family with arbitrary domain size, preserving efficient computability.

Definition 4.19: Merkle-Damgård transform

Let $\mathcal{H} = \{h_s : \{0,1\}^{2n} \to \{0,1\}^n\}_{s\in\{0,1\}^\lambda}$ be an hash family. We define the Merkle-Damgård transform over $\mathcal{H}$ as the hash family $\mathcal{H}_{MD} : \{H_s : \{0,1\}^{dn} \to \{0,1\}^n\}_{s\in\{0,1\}^\lambda}$, for any chosen $d > 2$, such that H_s is defined by:

- The input is $x = (x_1, \ldots, x_d)$, with $m_i \in \{0,1\}^\lambda$
- $y_0, \ldots, y_d$ are partial computations, where $y_0 = 0^n$
- For all $i \in [d]$ it holds that $y_i = h_s(x_i \,||\, y_{i-1})$
- The output is $y = y_d$

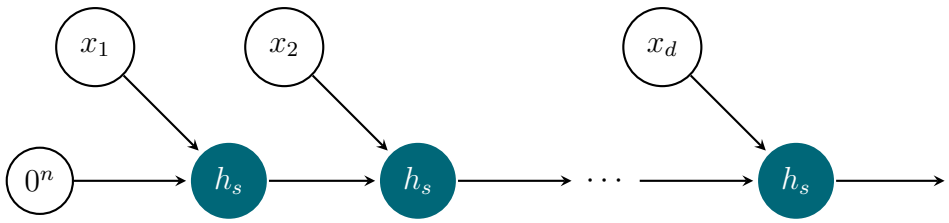

Figure 4.19: The Merkle-Damgård transform

Proposition 4.4

Let $\mathcal{H} = \{h_s : \{0,1\}^{2n} \to \{0,1\}^n\}_{s\in\{0,1\}^\lambda}$ be an hash family. If $\mathcal{H}$ is collision-resistant, $\mathcal{H}_{MD}$ is also collision-resistant for any $d > 2$.

Proof. The idea is to basically reduce a collision for $\mathcal{H}_{MD}$ to a collision for $\mathcal{H}$. Fix a seed $s \in_R \mathcal{U}_\lambda$. Assume that $x = (x_1, \ldots, x_d)$ and $x' = (x'_1, \ldots, x'_d)$ are two inputs such that $x \neq x'$ and $H_s(x) = H_s(x')$.

Given the partial computations $y_1, \ldots, y_d$ and $y'_1, \ldots, y'_d$, consider the largest index $i \in [d]$ such that $(x_i, y_{i-1}) \neq (x', y_{i-1})$ and $h_s(x_i, y_{i-1}) = h_s(x'_i, y'_i)$. Notice that such index always exists since $x \neq x'$ and $H_s(x) = H_s(x')$. Then, index the i-th application of h_s gives a collision for $x_i \,||\, y_{i-1} \neq x' \,||\, y_{i-1}$ since $h_s(x_i \,||\, y_{i-1}) = h_s(x'_i \,||\, y'_i)$. Therefore, we can define a distinguisher for $\mathcal{H}$ using a distinguisher for $\mathcal{H}_{MD}$ □

Observe that the above proof only works when $d > 2$ is FIL. However, it doesn't work for VIL. The flaw in the proov is given by the fact that we can't rule out that $h_s(0^{2n}) = 0^n$ may be a possibility since we don't know the structure of h_s. When this happens, by construction we have that:

$$H_s(x) = H_s(0^n \,||\, x) = H_s(0^{2n} \,||\, x) = H_s(0^{3n} \,||\, x) = \ldots$$

If the construction has FIL, the number of colliding inputs is finite, thus negligible. When the construction has VIL, instead, this the number of colliding inputs becomes infinite, thus non-negligible.

Nonetheless, the Merkle-Damgård transform can be modified to allow VIL. To avoid the previous issue, we encode each input x in a way such that it isn't the suffix of another input, i.e. there is no other input x' such that x is a suffix of x' (making the strings $0^n \,||\, x$ and $0^{2n} \,||\, x'$ illegal). To achieve this, it suffices to add a final round that hashes the standard output with an encoding of the number d of blocks in the input, written as $\langle d \rangle$.

Definition 4.20: Strenghtened Merkle-Damgård transform

Let $\mathcal{H} = \{h_s : \{0,1\}^{2n} \to \{0,1\}^n\}_{s \in \{0,1\}^\lambda}$ be an hash family. We define the strenghtened Merkle-Damgård transform over $\mathcal{H}$ as the hash family $\mathcal{H}^*_{MD} : \{H^*_s : \{0,1\}^* \to \{0,1\}^n\}_{s \in \{0,1\}^\lambda}$ such that H^*_s is defined by:

- The input is $x = (x_1, \ldots, x_d)$, with $m_i \in \{0,1\}^\lambda$ and with d being any value
- $y_0, \ldots, y_d$ are partial computations, where $y_0 = 0^n$
- For all $i \in [d]$ it holds that $y_i = h_s(x_i \,||\, y_{i-1})$
- The output is $y = h_s(\langle d \rangle \,||\, y_d)$, where $\langle d \rangle$ is an encoding of d using n bits

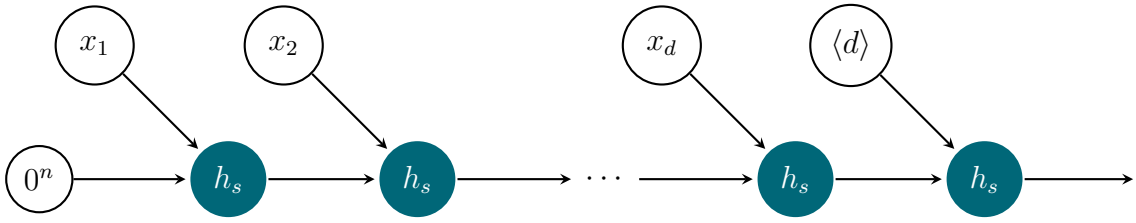

Figure 4.20: The Strenghtened Merkle-Damgård transform

Theorem 4.9

Let $\mathcal{H} = \{h_s : \{0,1\}^{2n} \to \{0,1\}^n\}_{s \in \{0,1\}^\lambda}$ be an hash family. If $\mathcal{H}$ is collision-resistant, $\mathcal{H}^*_{MD}$ is also collision-resistant for VIL.

Proof. The proof is almost identical to the previous proposition. Assume that $x = (x_1, \ldots, x_d)$ and $x' = (x'_1, \ldots, x'_{d'})$ are two inputs such that $x \neq x'$ and $H^*_s(x) = H^*_s(x')$. We have two cases:

- If $d = d'$, we can proceed as in the previous proposition since the lengths are equal
- If $d \neq d'$ then the collision happens in the final round since $\langle d \rangle \,||\, y_d \neq \langle d' \rangle \,||\, y_{d'}$ and $h_s(\langle d \rangle \,||\, y_d) = H^*_s(x) = H^*_s(x') = h_s(\langle d' \rangle \,||\, y_{d'})$.

□

A good eye may notice a small problem with our last construction: since $\langle d \rangle \in \{0,1\}^n$, we can have at most 2^n blocks, but this is a huge number for real values of n (e.g. 128, 256), making it enough.

There are many other constructions for achieving CRH families with VIL, such as **Merkle trees**. The idea behind this construction is simple: instead of applying repeated hashes in a "linearl-like fashion" as done in the Merkel-Damgård transform, we apply them in a "tree-like fashion".

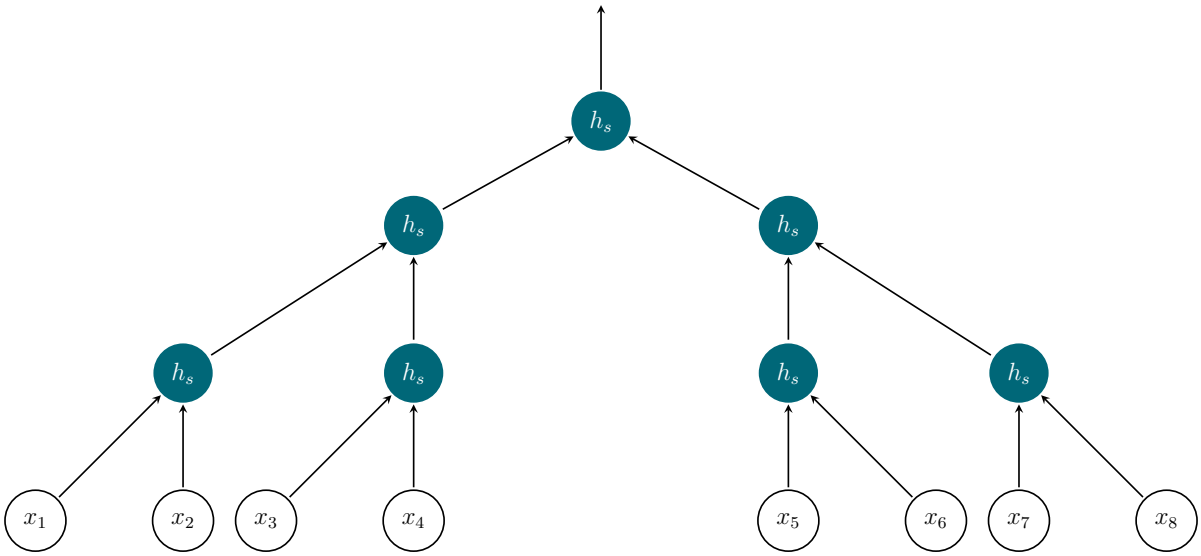

Figure 4.21: A Merkle tree with 4 levels.

In practical applications (e.g. MD5, SHA-1, SHA-2), the Merkel-Damgård transform is adapted by replacing the initial CRH family H with an heuristic equivalent (similar to what is done in DES, 3-DES and AES). In the theoretical setting, instead, the initial CRH family can indeed be constructed, but it requires additional *number theoretic assumptions* (e.g. the hardness of factoring, discrete log) or *post-quantum assumptions*. A middle-ground solution is obtained by using AES, where $h_s(x_1, x_2) = \text{AES}(x_1, x_2) \oplus x_2$. The security of this last construction can be proven only by assuming that AES is an *ideal cipher*, i.e. a truly random permutation for every choice of the key.

Hash functions are also used to build MACs. The HMAC scheme uses $\text{Tag}(K, m) = H(K \mid\mid m)$, with H being hash function. Assuming that $H(\cdot) \in \mathcal{R}(\ell(n) \to n)$ and that it can be computed only by querying an oracle, the HMAC scheme is secure. If H is constructed through the Merkle-Damgård transform, the scheme can be broken. The study of cryptosystems under the usage of an ideal cipher is called **Random Oracle Model (ROM)**.

Assumption 4.1: Random Oracle Model

Given an hash function H, in the Random Oracle Model (ROM) we assume that H is a truly random function and that it can be computed only through access to an oracle.

4.8 Solved exercises

Problem 4.1

Let $\mathcal{F} = \{F_k : \{0,1\}^n \to \{0,1\}^n\}_{K\in\{0,1\}^\lambda}$ be a PRF family and let $\mathcal{F}' = \{F_k : \{0,1\}^{n-1} \to \{0,1\}^{2n}\}_{K\in\{0,1\}^\lambda}$ be a family such that:

$$F'_K(m) = F_K(0 \,||\, m) \,||\, F_K(m \,||\, 1)$$

Prove or disprove that $\mathcal{F}'$ is a PRF family.

Proof. The claim is false. To disprove it, we define an adversary A as follows:

A = "On input $w \in \{0,1\}^{2n}$:

1. Query $m_1 = 0^{n-1}$ and $m_2 = 0^{n-2} \,||\, 1$ to C, which will answer with $y_1 = J(m_1)$ and $y_2 = J(m_2)$
2. Let s_i, t_i be the strings $s, t \in \{0,1\}^n$ such that $y_i = s_i \,||\, t_i$
3. If $t_1 = s_2$ return 1, otherwise 0

In $\text{Game}^{\text{PRF}}_{\mathcal{F}',A}(\lambda, 0)$, the challenger will use $J = F'_K$ for some $K \in_R \mathcal{K}$, thus we get that:

$$F'_K(m_1) = F'_K(0^{n-1}) = F_K(0^n) \,||\, F_K(0^{n-1} \,||\, 1)$$

$$F'_K(m_2) = F'_K(0^{n-2} \,||\, 1) = F_K(0^{n-1} \,||\, 1) \,||\, F_K(0^{n-2} \,||\, 1^2)$$

making the attacker return 1 with probability 1. In $\text{Game}^{\text{PRF}}_{\mathcal{F}',A}(\lambda, 1)$, instead, the challenger will use $J \in_R \mathcal{R}(n-1 \to 2n)$, giving no extra information to the attacker, making them return 1 with probability at most $2^{n-1}/2^{2n} = 2^{-n-1}$. Therefore, we conclude that:

$$|\Pr[A(w) = 1 : b = 0] - \Pr[A(w) = 1 : b = 1]| \geq 1 - \frac{1}{2^{n+1}} \geq \frac{1}{n}$$

thus $\mathcal{F}'$ is not a PRF family. □

Problem 4.2

Let $\mathcal{F} = \{F_K : \{0,1\}^n \to \{0,1\}^n\}_{K\in\{0,1\}^\lambda}$ be a PRF family and let $G : \{0,1\}^n \to \{0,1\}^{n+1}$ be a PRG. For each of the following SKE with key space $\mathcal{K} = \{0,1\}^\lambda$, prove or disprove whether the scheme is CPA-secure or not:

A Π_a : given a plaintext $m \in \{0,1\}^{n+1}$ and pick nonce $r \in_R \{0,1\}^n$, then output $(r, G(r) \oplus m)$

B Π_b : given a plaintext $m \in \{0,1\}^n$, pick a key $K \in_R \{0,1\}^\lambda$, then output $F_K(0^n) \oplus m$

C Π_c : given a plaintext $m \,||\, \widehat{m} \in \{0,1\}^{2n}$ with $m, \widehat{m} \in \{0,1\}^n$, pick a nonce $r \in_R \{0,1\}^n$ and a key $K \in_R \{0,1\}^\lambda$, then output $(r, F_K(r) \oplus m, F_K(r+1) \oplus \widehat{m})$, where $+$ is addition modulo 2^n

Solution:

A The first scheme is not CPA-secure. In fact, it can be proven that it isn't even 1-time secure. Consider the adversary $A^{\text{1-time}}_{\Pi_a}$ defined as follows:

- The adversary $A^{\text{1-time}}_{\Pi_a}$ sends m^*_0, m^*_1 and the challenger $C^{\text{1-time}}_{\Pi_a}$ returns (r^*, c^*)
- If $c^* = G(r^*) \oplus m^*_0$, the adversary returns 0. Otherwise, they return 1.

The above adversary is able to win the 1-time game with probability 1 since they can directly decrypt the challenge ciphertext, concluding that Π_a is not 1-time secure and thus not CPA-secure

B The second scheme is not CPA-secure. Consider the adversary $A^{\text{CPA}}_{\Pi_c}$ defined as follows:

- The adversary $A^{\text{CPA}}_{\Pi_a}$ queries m and the challenger $C^{\text{CPA}}_{\Pi_a}$ returns c.
- The adversary sends m^*_0, m^*_1 and the challenger returns (r^*, c^*)
- The adversary computes $\gamma = c \oplus m$.
- If $c^* = m^*_0 \oplus \gamma$, the adversary returns 0. Otherwise, they return 1.

The first encryption query allows the adversary to compute $\gamma = c \oplus m = (F_K(0^n) \oplus m) \oplus m = F_K(0^n)$, enabling them to directly decrypt the challenge chiphertext. Therefore, the above adversary is able to win the CPA game with probability 1, concluding that Π_b is not CPA-secure.

C The third scheme is CPA-secure. In fact, Π_c corresponds exactly to the CTR mode SKE with FIL set to 2. The proof of CPA-security for CTR mode can be easily adapted to show that Π_c is CPA-secure.

Problem 4.3

Let $\Pi = (\text{Enc}, \text{Dec})$ be a SKE scheme over key space $\mathcal{K}$, message space $\mathcal{M}$ and ciphertext space $\mathcal{C}$, such that $\mathcal{K} \subseteq \mathcal{M}$. We say that Π is circularly secure if the standard notion of CPA-security holds even when the adversary is given $\text{Enc}(K, K)$, where K is the key of the scheme. Give a formal definition of circular security. Prove or dispove: CPA-security implies circular security.

Solution:

A formal definition of circular security can be achieved by adding an intial message sent by the challenger containing $\text{Enc}(K, K)$ to the standard game $\text{Game}^{\text{CPA}}_{\Pi,A}(\lambda, b)$. We refer to this modified game as the CIRC game and we say that Π is circularly secure when $\text{Game}^{\text{CIRC}}_{\Pi,A}(\lambda, b) \approx_c \text{Game}^{\text{CIRC}}_{\Pi,A}(\lambda, b)$. By definition, circular security implies CPA-security, but the converse doesn't always hold.

To prove this, we give a counterexample. Let $\Pi = (\text{Enc}, \text{Dec})$ be a SKE such that:

$$\text{Enc}(K, m) = \begin{cases} K & \text{if } m = K \\ \text{Enc}'(K, m) & \text{if } m \neq K \end{cases} \qquad \text{Dec}(K, c) = \begin{cases} K & \text{if } c = K \\ \text{Dec}'(K, c) & \text{if } c \neq K \end{cases}$$

where $\Pi' = (\text{Enc}', \text{Dec}')$ is a CPA-secure SKE. Let $m_1, \ldots, m_q$ be encryption queries, with $q = \text{poly}(\lambda)$, and consider the event B such that $B = 1$ when $\exists i \in [q]$ with $m_i = K$. When conditioning on $B = 0$, we have that $\text{Game}^{\text{CPA}}_{\Pi,A}(\lambda, b) \equiv \text{Game}^{\text{CPA}}_{\Pi',A}(\lambda, b)$, therefore:

$$\text{SD}(\text{Game}^{\text{CPA}}_{\Pi,A}(\lambda, b); \text{Game}^{\text{CPA}}_{\Pi',A}(\lambda, b)) \leq \Pr[B = 1] \leq \frac{q}{2^n}$$

concluding that $\text{Game}^{\text{CPA}}_{\Pi,A}(\lambda, b) \approx_c \text{Game}^{\text{CPA}}_{\Pi',A}(\lambda, b)$ and thus that Π is also CPA-secure. However, Π can be easily proven to not be circularly secure after receiving $y_0 = \text{Enc}(K, K)$, we have full access to the key since $y_0 = K$.

Problem 4.4

Suppose we are given a pseudorandom permutation $P : \mathcal{K} \times \mathcal{X} \to \mathcal{X}$. We want to construct a pseudorandom permutation $P' : \mathcal{K} \times \mathcal{X}' \to \mathcal{X}'$ that operates on a smaller domain $\mathcal{X}' \subseteq \mathcal{X}'$. Given $K \in \mathcal{K}$ and $x \in \mathcal{X}$, we compute $P'(K, x)$ as follows:

1. Set $y = P(K, x)$
2. Until $y \notin \mathcal{X}'$, set $y = P(K, y)$
3. Output y

To invert a given $y \in \mathcal{X}'$ one simply computes the inverse permutation until the result lies in $\mathcal{X}'$. Answer the following questions:

- Let $t = |\mathcal{X}| / |\mathcal{X}'|$. How many evaluations of P are needed in expectation to evaluate $P'(K, x)$ as a function of t? When is the evaluation of P' efficient?
- Using induction on $|\mathcal{X}| - |\mathcal{X}'|$ show that if P is a PRP with domain $\mathcal{X}$ then P' is a PRP with domain $\mathcal{X}'$.

Solution:

Let I be the random variable denoting the number of times P is evaluated to get an output. Since I has geometric distribution, we have that:

$$\mathbb{E}[I] = \sum_{i=1}^{+\infty} i \Pr[I = i] = \sum_{i=1}^{+\infty} i(1-t)^{i-1}t = \frac{1}{t}$$

The evaluation of P' can therefore be considered efficient when $\frac{1}{t}$ is polynomial in the length $n = \log |\mathcal{X}|$ of an element of $\mathcal{X}$, i.e. when:

$$\frac{|\mathcal{X}|}{|\mathcal{X}'|} = \frac{1}{t} \leq \log^c |\mathcal{X}| \implies |X'| \geq \frac{|\mathcal{X}|}{\log^c |\mathcal{X}|} = \frac{2^n}{n^c}$$

for some $c > 0$.

For the second request, we get a trivial base case when $|\mathcal{X}| - |\mathcal{X}'| = 0$ this is true only when $\mathcal{X} = \mathcal{X}'$. Now, assume that $|\mathcal{X}| - |\mathcal{X}'| > 0$. Fix $\widehat{x} \in \mathcal{X} - \mathcal{X}'$ and let $\widehat{P}$ the equivalent of P' but defined on $\widehat{X} = \mathcal{X}' \cup \{\widehat{x}\}$. By inductive hypothesis, we know that $\widehat{P}$ is a PRP.

By way of contradiction, suppose that there is an adversary A^{PRP} for the PRP game of P'. We notice that A^{PRP} is also a distinguisher for the PRP game of $\widehat{P}$:

$$\begin{aligned}
&\Pr_{x \in_R \widehat{\mathcal{X}}'}[\widehat{P}(x') = y : y = \widehat{P}(x); x' \leftarrow A^{\text{PRP}}(y)] \\
&= \Pr_{x \in_R \widehat{\mathcal{X}}}[\widehat{P}(x') = y : y = \widehat{P}(x'); x' \leftarrow A^{\text{PRP}}(y); x \neq \widehat{x}] \Pr_{x \in_r \widehat{\mathcal{X}}}[x \neq \widehat{x}] \\
&= \frac{\left|\widehat{\mathcal{X}}\right| - 1}{\left|\widehat{\mathcal{X}}\right|} \Pr_{x \in_R \mathcal{X}'}[P'(x') = y : x = P'(y); x' \leftarrow A^{\text{PRP}}(y)] \\
&\geq \text{negl}(\log |\mathcal{X}|)
\end{aligned}$$

contradicting the fact that $\widehat{P}$ is a PRP, thus P' must be a PRP.

5

Number theory in cryptography

5.1 Brush-up on number theory

We briefly discussed how number theoretic assumptions are required to build the theoretical tools that we have seen in the previous chapter. In this chapter, we'll introduce some of these assumptions (e.g. the hardness of factoring, discrete log and learning with erros) and use them to build valid tools.

Number theory is the field of algebraic mathematics that studies prime numbers and modular arithmetic, i.e. the sets $\mathbb{Z}_n = \{0, 1, \dots, n-1\}$ with $n \in \mathbb{N}$. In particular, the field studies algebraic structures such as groups (e.g. $(\mathbb{Z}_n, \cdot)$), rings (e.g. $(\mathbb{Z}_n, +, \cdot)$) and fields (e.g. $(\mathbb{Z}_p, +, \cdot)$ with $p \in \mathbb{P}$). We recall that:

- $(\mathbb{G}, +)$ is a group if $+$ satisfies aassociativity, the existence of a neutral element and the existence of an inverse element
- $(\mathbb{G}, +, \cdot)$ is a ring if $(\mathbb{G}, +)$ is a group and $\cdot$ satisfies aassociativity and the existence of a neutral element
- $(\mathbb{G}, +, \cdot)$ is a field if $(\mathbb{G}, +)$ is a group and $\cdot$ satisfies aassociativity, the existence of a neutral element and the existence of an inverse element (except for the neutral element of $+$).

where $+ : \mathbb{G} \times \mathbb{G} \to \mathbb{G}$ and $\cdot : \mathbb{G} \times \mathbb{G} \to \mathbb{G}$ are two binary operations (here we'll always assume that they are the standard addition and multiplication). We also recall that in modular arithmetic we have that $a \equiv b \pmod{n}$ when $a = b + kn$ for some $k \in \mathbb{Z}$. In this section we'll state and proof a list of basic number theory results that will be used in the following sections.

Let Z_n^* denote the set of invertible elements of $\mathbb{Z}_n$, i.e. the elements $a \in \mathbb{Z}_n$ for which $\exists b \in \mathbb{Z}_n$ such that $ab = 1$. It can be easily proven that an element of $\mathbb{Z}_n$ is invertible if and only if its GCD with n is exactly 1.

Proposition 5.1

Given $a \in \mathbb{Z}_n$, it holds that $a \in \mathbb{Z}_n^*$ if and only if $\gcd(a, n) = 1$

Proof. Let $d = \gcd(a, n)$. Suppose that a is invertible, that is $\exists b \in \mathbb{Z}_n$ such that $ab \equiv 1 \pmod{n}$. Then, we have that $ab = 1 + kn$ for some $k \in \mathbb{Z}$, thus $ab - kn = 1$. By definition of GCD, we have that d must divide $ab - kn$, but the only possible natural number that divides 1 is 1 itself, concluding that $d = 1$. Vice versa, suppose that $d = 1$. Then, $\exists k_1, k_2$ such that $1 = d = k_1 a + k_2 n$, implying that $k_1 a = 1 - k_2 n$. Therefore, we conclude that $ka \equiv 1 \pmod{n}$ and thus that k_1 is the inverse of a. □

The above proposition immediately concludes that $\mathbb{Z}_n^* = \{a \in \mathbb{Z}_n \mid \gcd(a, n) = 1\}$ and that $|\mathbb{Z}_n^*| = \varphi(n)$, where $\varphi(n)$ is **Euler's totient function**, the function that returns the number of coprimes of n. In the special case of $\mathbb{Z}_p^*$ with $p \in \mathbb{P}$, we have that every number $\mathbb{Z}_p$ except zero is coprime with p, therefore $\mathbb{Z}_p^* = \{1, \dots, p-1\}$ and $\phi(p) = p - 1$. We'll see ways to compute the totient function for general $n \in \mathbb{N}$ in future sections.

Lemma 5.1

Given $a, b \in \mathbb{N}$ such that $a \geq b > 0$, it holds that:

$$\gcd(a, b) = \gcd(b, a \pmod{b})$$

Proof. Let $a \equiv x \pmod{b}$ with x being as small as possible. Let also $d = \gcd(a, b)$ and let $d' = \gcd(b, x)$. Since $a = x + kb$ for some $k \in \mathbb{Z}$, by definition of GCD we know that d must divide x since $x = a - kb$, implying that $x = dq$ for some $q \in \mathbb{N}$. Similarly, by definition of GCD we have that d' divides a since $a = x + kb$, implying that $a = d'q'$ for some $q \in \mathbb{N}$. By transitivity, we conclude that both d and d' divide each other, which is possible only if $d = d'$. □

The above lemma can be used to define an algorithm that iteratively computes the GCD between two numbers by reducing them little by little up until $\gcd(x, 0)$ is reached, outputting x. This is known as the **Euclidean algorithm**.

Algorithm 5.1: Euclidean algorithm

Given $a, b \in \mathbb{Z}$ with $0 < a \leq b$ as input, the Euclidean algorithm is defined as follows and it outputs $\gcd(a, b)$ in $O(|a| + |b|)$:

1. Set $r_0 := b$ and $r_1 := a$. Initialize $i = 0$
2. While $r_i \neq 0$, set $r_{i+1} := r_{i-1} \pmod{r_i}$ (i.e. $r_{i-1} = r_i q_i + r_{i+1}$ with r_{i+1} as small as possible), then increment i
3. Return r_{i-1}

Proof. Omitted. □

We give an example of computation made by the algorithm. Given $a = 14$ and $b = 10$, we have that:

1. Set $r_0 = 10$ and $r_1 = 14$
2. $r_1 = r_0 \cdot 1 + 4$, thus $r_2 = 4$
3. $r_2 = r_1 \cdot 2 + 2$, thus $r_3 = 2$
4. $r_3 = r_2 \cdot 2 + 0$, thus $r_4 = 0$
5. The output is $r_3 = 2$

We observe that this algorithm can also be extended to express the GCD as a *Bézout identity*, meaning that $\gcd(a, b) = ak + bh$ for some $k, h \in \mathbb{Z}$. This can be achieved by simply inverting each iteration made by the algorithm. For instance, in our previous example we have that:

1. $r_2 = r_1 \cdot 2 + r_3$, thus $r_3 = r_2 - r_1 \cdot 2$
2. $r_1 = r_0 \cdot 2 + r_2$, thus $r_2 = r_1 - r_0 \cdot 2$
3. Combining all the found relationships we get that:

$$r_3 = r_2 - r_1 \cdot 2 = (r_1 - r_0 \cdot 2) - r_0 \cdot 2 = r_1 - 4r_0 = (1)a + (-4)b$$

Observe that when $\gcd(a, n) = 1$, the extended Euclinean algorithm finds two valus $k, h \in \mathbb{Z}$ such that $1 = \gcd(a, n) = ak + hn$, meaning that $ak = 1 + (-h)n$ and thus that k is the inverse of a in $\mathbb{Z}_n$.

After proving that we have a tool to efficiently compute the GCD of two numbers, we require to prove that exponentiation modulo n can also be efficiently computed. Given two numbers $a, b \in \mathbb{N}$, let $b_t b_{t-1} \dots b_0$ be the binary representation of b. Then, we have that:

$$a^b \equiv a^{\sum_{i=0}^{t} b_i 2^i} \equiv \prod_{i=0}^{t} (a^{2^i})^{b_i} \pmod{n}$$

Observe that the exponents $b_0, b_1, \dots, b_t$ acts as a "toggle", meaning that we skip the inner exponentiations with $b_i = 0$ and compute only those with $b_j = 1$. The values $a, a^2, a^4, \dots, a^{2^t} \pmod{n}$ can be easily computed through a recursive approach (e.g. $a^8 \equiv (a^4)^2 \equiv ((a^2)^2)^2$, thus we can compute $((a^2 \pmod{n})^2 \pmod{n})^2 \pmod{n})$ with $\log 2^t = t$ recursion levels. This concludes that exponentiation can also be computed in polynomial time with respect to $|b|$.

Now, we need one last tool: finding prime numbers. The **prime number theorem** states that the number $\pi(x)$ of primes up to $x > 0$ is bounded by:

$$\pi(x) \geq \frac{x}{3 \log x} \approx \frac{x}{\log x}$$

Assuming that we can test if a number is prime or not, we can efficiently generate λ-bit primes through a random process that is guaranteed to eventually terminate. We sample

$n \in_R [2^\lambda - 1]$ and test if it is prime. If it's prime, we output it. Otherwise, we repeat the process. Notice that:

$$\Pr[\text{no output after } 3\lambda^2 \text{ samples}] \leq \left(1 - \frac{1}{3\log(2^\lambda - 1)}\right)^{3\lambda^2} \leq \left(1 - \frac{1}{3\lambda}\right)^{3\lambda^2} \leq e^{-\lambda}$$

Therefore, we're left with proving that there is an algorithm that can test prime numbers. Almost all the currently known algorithms for primality testing are based on **Fermat's little theorem**, which states that:

$$a^{p-1} \equiv 1 \pmod{p} \quad \forall p \in \mathbb{P}, a \neq 0$$

Fermat's little theorem is a particular case of Euler's theorem, which states that:

$$a^{\varphi(n)} \equiv 1 \pmod{n} \quad \forall n \in \mathbb{N}, a \in \mathbb{Z}_n^*$$

and even more generally that:

$$a^b \equiv a^{b \pmod{\varphi(n)}} \pmod{n} \quad \forall b, n \in \mathbb{N}, a \in \mathbb{Z}_n^*$$

We start by considering the **Fermat primality test**. On input n, sample $a \in_r [2, n-2]$ and test if $a^{n-1} \equiv 1 \pmod{n}$. If it's true, we output "maybe prime". Otherwise, we output "not prime". It's easy to see that this test can only distinguish non prime numbers: if the test fails, the input cannot be a prime number since otherwise it would satisfy Fermat's little theorem. When the test succeeds, we cannot ensure that the input is prime: a prime number definitely passes the test, but some non-prime numbers may also pass it for some values of a. These inputs are called *Fermat's liars* for a.

An easy fix could be to iteratively apply the test until we find a nice value of a. However, there are some specific values of n that pass the test for any possible value of a. These inputs are called *Carmicheal numbers*.

Nonetheless, effective primality tests do exists. The Miller-Robin primality test uses an efficient random process, based on advanced algebra and Fermat's little theorem, that is guaranteed to eventually terminate. The Agrawal-Kayal-Saxena primality test, instead, uses a process similar to that of Miller-Robin without randomness, at the cost of some efficiency. Therefore, generating prime numbers is effectively possible!

5.2 The DL, CDH and DDH assumptions

We formally introduce our first number theory hardness assumptions. Consider the equation $a^x = b$ over a continuous setting such as $\mathbb{R}$. With common mathematical tools, this quation can easily by solved as $x = \log_a b$. Consider now the equivalent version of this problem over a discrete set such as $\mathbb{Z}_p$, i.e. $a^x \equiv b \pmod{p}$. In the continuous context, the problem has an unique solution. In the discrete context, instead, we have infinite solutions due $\mathbb{Z}_p$ being a *cyclic group*. For instance, for the equation $2^x \equiv 4 \pmod{5}$ we have that:

$$4 \equiv 2^2 \equiv 2^6 \equiv 2^{10} \equiv 2^{14} \equiv \ldots \pmod{5}$$

Therefore, it's very hard to find the exact value of x that was originally used. This problem is known as the **Discrete Logarithm (DL)** problem. For centuries, many mathematicians tried to find an efficient way to solve this problem (observe that it can be trivially solved in exponential time through bruteforce). To this day, the best algorithms achieve only a sub-exponential time, which is still worse than polynoamial time. Over time, researchers started to assume that the problem is too hard to be solved in polynomial time even through randomness, i.e. that no PPT algorithm can solve the problem efficiently, even thought this has not yet been proven to be true. This is known as the *DL assumption*.

Assumption 5.1: DL assumption

Given a prime $p \in \mathbb{P}$, the DL assumption states that:

$$\Pr[A(g, p, y) = x : x \in_R \mathbb{Z}_p, y \equiv g^x \pmod{p}] \leq \text{negl}(\lambda)$$

for all PPT adversaries A and for a chosen value $\lambda > 0$

It's easy to see that, by definition, the DL assumption automatically implies the existence of an OWF (the discrete log itself). Diffie and Hellman used the DL assumption to start the public key revolution of cryptography. They proposed a **key exchange protocol** whose security depends only on this assumption.

Algorithm 5.2: Diffie-Hellman key exchange

The Diffie-Hellman key exchange allows two users Alice and Bob to generate a sharede key over a public channel without allowing third parties to infer the key:

1. Alice samples a prime $p \in_R \mathbb{P}$ and computes a generator g of $\mathbb{Z}_p$, sharing the pair (g, p) with Bob
2. Alice samples a value from $x \in_R \mathbb{Z}_{p-1}$ and sends $X = g^x \pmod{p}$ to Bob
3. Bob samples a value from $y \in_R \mathbb{Z}_{p-1}$ and sends $Y = g^y \pmod{p}$ to Alice
4. Alice computes $K = Y^x = g^{xy} \pmod{p}$, while Bob computes $K = X^y = g^{xy} \pmod{p}$
5. Both parties will use K as the key

We recall that a *generator* g for a cyclic group $\mathbb{G}$ is a value $g \in \mathbb{G}$ such that $\mathbb{G} = \{g^0, g^1, g^2, \ldots, g^{o(g)}\}$, where $o(g)$ is the *order* of g, i.e. the smallest value such that $g^{o(g)} = 1$ in $\mathbb{G}$.

The protocol clearly requires some hardness assumptions to hold, otherwise an attacker could intercept the messages $(g, p), X, Y$ and derive the values of x, y from them, computing the key $K = g^{xy}$. This is usually referred to as a **passive attack** since the thid party eavesdrops without changing the messages. The protocol may also suffer from another type of attack, called **active attack**, where the third party also changes the messages

(also referred to as *man-in-the-middle attack*).

We start by analyzing passive security. To "fix" this issue, we have to assume that the key is hard to compute without knowing the values of x and y. Consider the following generalization of the DH key exchange to any cyclic group:

1. Alice chooses a cyclic group $\mathbb{G}$, computes a generator g of $\mathbb{G}$ and the order q of g over $\mathbb{G}$, sharing the triple $(\mathbb{G}, g, q)$ with Bob
2. Alice samples a value from $x \in_R \mathbb{Z}_q$ and sends $X = g^x \pmod p$ to Bob
3. Bob samples a value from $y \in_R \mathbb{Z}_q$ and sends $Y = g^y \pmod p$ to Alice
4. Alice computes $K = Y^x = g^{xy}$ in $\mathbb{G}$, while Bob computes $K = X^y = g^{xy}$ in $\mathbb{G}$
5. Both parties will use K as the key

Currently, it is not known if the DL assumption suffices to make it hard to compute the key for passive attackers. This motivates a new assumption, the **Computatational Diffie-Hellman (CDH)** assumption.

Assumption 5.2: CDH assumption

Given a triple $(\mathbb{G}, g, q)$ where $\mathbb{G}$ is a cyclig group, $g \in \mathbb{G}$ is a generator and q is the order of g in $\mathbb{G}$, the CDH assumption states that:

$$\Pr[A(\mathbb{G}, g, q, g^x, g^y) = g^{xy} : x, y \in_R \mathbb{Z}_q] \leq \text{negl}(\lambda)$$

for all PPT adversaries A and for a chosen value $\lambda > 0$

Clearly, the CDH assumption implies that the DL assumption is true: if one can solve the DL problem, it can also solve the CDH problem by finding x, y. As already stated, the converse implication is currently not known to hold. There is a lot of evidence for the CDH assumption to hold in $\mathbb{Z}_p$ with $p \in \mathbb{P}$, but it currenlty hasn't been problem.

We observe that the CDH assumption is not enough to fix the issue of passive security: an attacker may still infer information about the key without inverting X and Y but simply by analyzing them. What we really need is for the key to be indistinguishable from a value chosen uniformly-at-random over $\mathbb{G}$. This is called the **Decisional Diffie-Hellman (DDH)** assumption.

Assumption 5.3: DDH assumption

Given a triple $(\mathbb{G}, g, q)$ where $\mathbb{G}$ is a cyclig group, $g \in \mathbb{G}$ is a generator and q is the order of g in $\mathbb{G}$, the DDH assumption states that:

$$(\mathbb{G}, g, q, g^x, g^y, g^{xy}) \approx_c (\mathbb{G}, g, q, g^x, g^y, g^z)$$

with $x, y, z \in_R \mathbb{Z}_q$, for all PPT adversaries A and for a chosen value $\lambda > 0$

The DDH assumption implies the CDH assumption, but the converse is proven to not hold for a general triple $(\mathbb{G}, g, q)$. Moreover, the DDH assumption is proven to not hold for $\mathbb{G} = \mathbb{Z}_p$.

Proposition 5.2

The DDH assumption doesn't hold for $\mathbb{G} = \mathbb{Z}_p$.

Proof. Consider the group $\mathbb{QR}_p$ of quadratic residues over p, defined as $\mathbb{QR}_p = \{y \in \mathbb{Z}_p \mid \exists x \in \mathbb{Z}_p \; y \equiv x^2 \pmod{p}\}$. Equivalently, we have that $\mathbb{QR}_p = \{y \in \mathbb{Z}_p \mid \exists k \in \mathbb{Z} \; y \equiv g^{2k} \pmod{p}\}$

Claim: $y \in \mathbb{QP}_p$ if and only if $y^{\frac{p-1}{2}} \equiv 1 \pmod{p}$

Proof. Suppose that $y \in \mathbb{QP}_p$, meaning that $y \equiv g^{2k} \pmod{p}$ for some $k \in \mathbb{Z}$. Then, we have that:

$$y^{\frac{p-1}{2}} \equiv (g^{2k})^{\frac{p-1}{2}} \equiv (g^{p-1})^k \equiv 1 \pmod{p}$$

Vice versa, suppose that $y \notin \mathbb{QP}_p$, meaning that $y \equiv g^{2k'+1} \pmod{p}$ for some $k' \in \mathbb{Z}$. Then, we have that:

$$y^{\frac{p-1}{2}} \equiv (g^{2k'+1})^{\frac{p-1}{2}} \equiv (g^{p-1})^k g^{\frac{p-1}{2}} \equiv g^{\frac{p-1}{2}} \not\equiv 1 \pmod{p}$$

since $p-1$ is the order of g in $\mathbb{Z}_p$. □

The above claim easily gives us a distinguisher for the DDH condition. When $Z = g^z$ with $z \in_R \mathbb{Z}_p$ then $Z \in \mathbb{QR}_p$ with probability $1/2$. When $Z = g^{xy}$ with $x, y \in_R \mathbb{Z}_p$, instead, we have that $Z \in \mathbb{QR}_p$ with probability $3/4$: either $g^x \in \mathbb{QR}_p$ or $g^y \in \mathbb{QR}_p$ or both. Therefore, we can build an attacker that distinguishes the DDH condition with probability $3/4 - 1/2 = 1/4$. □

The above implies that we require a new type of cyclic group for which the DDH assumption is believed to hold. Luckily, there are many groups for which this is true. The easiest fix is to set $\mathbb{G} = \mathbb{QP}_p$ where $p = 2q + 1$ and $p, q \in \mathbb{P}$ to break the previous proposition. Another very popular choice is *elliptic curve groups*. An elliptic curve is composed of solutions to an equation $y = x^3 + ax^2 + bx + c$, for some coefficients $a, b, c \in \mathbb{R}$. The elliptic curve group modulo $p \in \mathbb{P}$ is a group whose elements are the points of an elliptic curve with coordinates modulo p.

What about active security? It's easy to see that even the DDH assumption is not enough. A third party Eve could intercept the message $X = g^x$ sent by Alice and replace it with $X' = g^{x'}$, sending it to Bob. Then, Eve intercepts $Y = g^y$ from Bob and sends $Y' = g^{y'}$ to Alice. Now, Alice will use the key $K_{AE} = Y'^x = g^{xy'}$ and Bob will use the key $K_{EB} = X'^y = g^{x'y}$, while Eve can compute both keys.

This issue cannot be fixed under any assumption by itself. Nonetheless, it can be fixed by strengthening the protocol through the use of a *master key* (a key that is already shared between Alice and Bob but never used for communications) to authenticate the messages

sent by Alice and Bob, using either MACs or digital signatures. We'll return to this topic when we'll extensively discuss public-key encryption.

5.3 Building crypto-tools through number theory

To close our discussion on the DL, CDH and DDH assumptions, let's talk about other ways in which they can be used for cryptography. We already saw that DDH $\rightarrow$ CDH $\rightarrow$ DL $\rightarrow$ OWF, from which we can then prolong our chain to get OWF $\rightarrow$ PRG $\rightarrow$ PRF $\rightarrow$ PRP $\rightarrow$ CRH, concluding that the DDH assumption is effectively enough to get secure symmetric encryption. Actually, we don't need to apply each construction of the full chain to get every single tool. In fact, each tool can be derived directly through the DDH assumption (or even the DL assumption).

Let's start by building a PRG. Consider any triple $(\mathbb{G}, g, q)$ for which the DDH assumption holds. We define a function $G_{\mathbb{G},g,q} : \mathbb{Z}_q \rightarrow \mathbb{G}^3$ as $G_{\mathbb{G},g,q}(x, y) = (g^x, g^y, g^{xy})$. Under the DDH assumption, we have that $G_{\mathbb{G},g,q}(\mathcal{U}_{\mathbb{Z}_q^2}) \approx_c (\mathcal{U}_{\mathbb{G}}, \mathcal{U}_{\mathbb{G}}, \mathcal{U}_{\mathbb{G}}) \approx_c \mathcal{U}_{\mathbb{G}^3}$. When $\mathbb{G} = \mathbb{Z}_q$, $G_{\mathbb{G},g,q}$ becomes a PRG with strech 1. This construction can be further improved to a PRG $\mathbb{G}^t_{\mathbb{G},g,q} : \mathbb{Z}_q^{t+1} \rightarrow \mathbb{G}^{2t+1}$ with stretch t, where:

$$\mathbb{G}^t_{\mathbb{G},g,q}(x, y_1, \ldots, y_t) = (g^x, g^{y_1}, g^{xy_1}, g^{y_2}, g^{xy_2}, \ldots, g^{y_t}, g^{xy_t})$$

Now, let's build a PRF. We already saw that PRFs can be constructed through the GGM tree using a PRG whose output can be nicely split in half. Copying this idea, we can construct a nice PRF with a closed form (thus we don't have to recursively compute the two halves of the PRG).

Consider the following PRG $G_a : \mathbb{G} \rightarrow \mathbb{G}^2$ where $a \in \mathbb{Z}_q$ and $G_a(g^b) = (g^b, g^{ab})$ (proof of pseudorandomness through the DL assumption). By setting $G_{a,0}(g^b) = g^b$ and $G_{a,0}(g^b) = g^{ab}$, we get that $G_a(g^b) = G_{a,0}(g^b) \mid\mid G_{a,1}(g^b)$. Let $\mathcal{F} = \{F_{\vec{a}} : \{0,1\}^n \rightarrow \mathbb{G}\}_{\vec{a} \in \mathbb{Z}_q^{n+1}}$ where:

$$F_{\vec{a}}(x_1, \ldots, x_n) = g^{a_0 \prod_{i=1}^n a_i^{x_i}}$$

After some thought, we can convince ourselves that $\mathcal{F}$ is basically the GGM tree of G, proving that it is a PRF. After obtaining a PRF, we can easily construct a PRP through a 3-layer Feistel network.

We're left with constructing a valid CRH hash family. Consider the hash function $H_{g_1,g_2}(x_1, x_2) = g_1^{x_1} g_2^{x_2}$ where $g_1, g_2 \in \mathbb{G}$ and $x_1, x_2 \in \mathbb{Z}_q$, with q being the order of $\mathbb{G}$. Suppose that there is an adversary that finds two colliding inputs $(x_1, x_2), (x_1', x_2')$ with $(x_1, x_2) \neq (x_1', x_2')$. Given a DL input triple (g, q, y), if we randomly select an index $i \in_R \{0, 1\}$ and set $g_i = y$ and $g_j = g^b$, with $j \neq i$ and $b \in_R \mathbb{Z}_q$, we have that:

$$g_1^{x_1} g_2^{x_2} = g_1^{x_1'} g_2^{x_2} \iff y^{x_i - x_i'} = g^{b(x_j - x_j')} \iff y = g^{b(x_i - x_j)^{-1}(x_j - x_j')}$$

We observe that this holds only if $x_i - x_i' \neq 0$, which is possible with $1/2$ probability since i is randomly selected and $(x_1, x_2) \neq (x_1', x_2')$. Since we reduced the DL game of $\mathbb{G}$ to the CRH game of H_{g_1,g_2}, the hash function is collision resistant under the DL assumption.

5.4 Solved exercises

Problem 5.1

Let $\mathbb{G}$ be a cyclic group of prime order q generated by $g \in \mathbb{G}$. Let n be a polynomially bounded parameter. Consider the hash family $\mathcal{H}_{\mathbb{G},n} : \{H_{g_1,\ldots,g_n} : \mathbb{Z}_q^n \to G\}$ parametrized by the group $\mathbb{G}$ and by n randomly chosen group elements $g_1, \ldots, g_n \in \mathbb{G}$ such that for $x_1, \ldots, x_n \in \mathbb{Z}_q$ it holds that:

$$H_{g_1,\ldots,g_n}(x_1, \ldots, x_n) = g_1^{x_1} \cdot \ldots \cdot g_n^{x_n}$$

Prove that $\mathcal{H}_{\mathbb{G},n}$ is collision resistant assuming the DL assumption for $\mathbb{G}$.

Proof. By way of contradiction, suppose that there is an adversary $A^{\text{CRH}}_{\mathcal{H}_{\mathbb{G},n}}$ that finds a collision with in $\mathcal{H}_{\mathbb{G},n}$ with probability at least n^{-c} for some $c > 0$. We define a new adversary $A^{\text{DL}}_{\mathbb{G}}$ as follows:

1. The challenger $C^{\text{DL}}_{\mathbb{G}}$ sends the input (g, p, y) to $A^{\text{DL}}_{\mathbb{G}}$.
2. $A^{\text{DL}}_{\mathbb{G}}$ samples $i \in_R \mathcal{U}_n$ and sets $g_i = y$
3. For each $j \in [n] - \{i\}$, $A^{\text{DL}}_{\mathbb{G}}$ samples $r_j \in_R \mathbb{Z}_q$ and sets $g_j = g^{r_j}$
4. $A^{\text{DL}}_{\mathbb{G}}$ sends $(g_1, \ldots, g_n)$ to $A^{\text{CRH}}_{\mathcal{H}}$, which replies with $(x_1, \ldots, x_n)$ and $(x'_1, \ldots, x'_n)$ such that $(x_1, \ldots, x_n) \neq (x'_1, \ldots, x'_n)$
5. $A^{\text{DL}}_{\mathbb{G}}$ returns $(x_i - x'_i)^{-1} \sum_{j=1:j\neq i}^{n} r_j(x_j - x'_j)$ to $C^{\text{DL}}_{\mathbb{G}}$

Suppose that $A^{\text{CRH}}_{\mathcal{H}}$ finds a collision for the seed $(g_1, \ldots, g_n)$, meaning that $g_1^{x_1} \cdot \ldots \cdot g_n^{x_n} = g_1^{x'_1} \cdot \ldots \cdot g_n^{x'_n}$ with $(x_1, \ldots, x_n) \neq (x'_1, \ldots, x'_n)$. By rearranging the equation, we get that:

$$\begin{aligned}
\prod_{k=1}^{n} g_k^{x_k} = \prod_{k=1}^{n} g_k^{x'_k} &\iff g_i^{x_i - x'_i} = \prod_{\substack{j=1: \\ j\neq i}}^{n} g_n^{x_j - x'_j} \\
&\iff y^{x_i - x'_i} = \prod_{\substack{j=1: \\ j\neq i}}^{n} g^{r_j(x_j - x'_j)} \\
&\iff y^{x_i - x'_i} = g^{\sum_{j=1:j\neq i}^{n} r_j(x_j - x'_j)}
\end{aligned}$$

Assuming that $x_i - x'_i \neq 0$, we get that:

$$y = g^{(x_i - x'_i)^{-1} \sum_{j=1:j\neq i}^{n} r_j(x_j - x'_j)}$$

Observe that the probability of this assumption to be true is at least $1/n$ since $i \in_R \mathcal{U}_n$ and $(x_1, \ldots, x_n) \neq (x'_1, \ldots, x'_n)$. Therefore, we conclude that:

$$\Pr[A^{\text{DL}}_{\mathbb{G}}(g, p, y) = \alpha : \alpha \in_R \mathbb{Z}_q, y = g^{\alpha}] \geq \frac{1}{n}\frac{1}{n^c}$$

contradicting the fact that the DL assumption holds in $\mathbb{G}$. □

6

Public-key encryption

6.1 The aymmetric key paradigm

In the previous chapter we introduced key exchange, a basic concept of public-key cryptography. Differently from symmetric-key encryption, in **public-key encryption (PKE)** the two parties use a pair of keys (pk, sk), where pk is the *public key* and it is used for encryption, while sk is the *secret key* (or *private key*) and it is used for decryption.

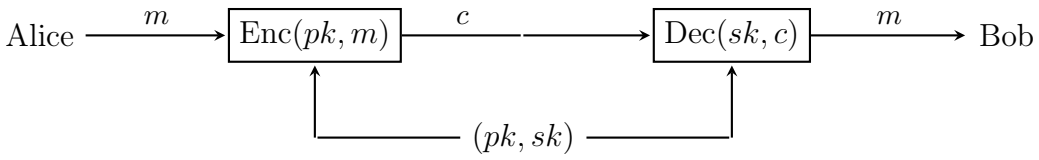

Figure 6.1: Example of public-key encryption.

Since we're working with two keys and one must act as the tool to invert the other, in PKE schemes the **keygen** (*key generation*) algorithm must be explicitily stated (we can't just say "let (pk, sk) be a public-secret key pair"). Therefore, PKE schemes are formally considered as a triple $\Pi = (\mathrm{KGen}, \mathrm{Enc}, \mathrm{Dec})$.

As the names suggest, the public key must be shared with the other party in order to achieve encryption, meaning that we have to find a way to securely send Bob's public key to Alice without it being subject of a man-in-the-middle attack, similar to what we have seen for the Diffie-Hellman protocol. To achieve this, we'll have to introduce the concept of digital signature and public key infrastructure.

For now, we start by translating the security concepts that we have already discussed from the symmetric setting to the asymmetric one (CPA-security, CCA-security, ...). The main idea is to use the same identical games to show the security of PKEs, with the addition of an initial message sent by the challenger. This initial message contains the public key generated together with the secret key at the start of the game. We observe that

sending the public key allows the adversary to encrypt the queries by themselves, without having to send them to the challenger. This implies that we can remove encryption queries from both the CPA game and the CCA game. In particular, we recall that the encryption algorithm is assumed to be randomized, meaning that the adversary cannot trick the challenger by encrypting the challenge messages m_0^*, m_1^* by themselves and get c^* before the challenger sends it.

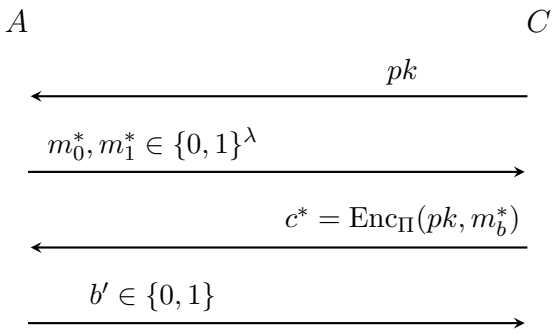

Figure 6.2: Graphical representation of $\mathrm{Game}^{\mathrm{CPA}}_{\Pi,A}(\lambda, b)$ for public-key encryption.

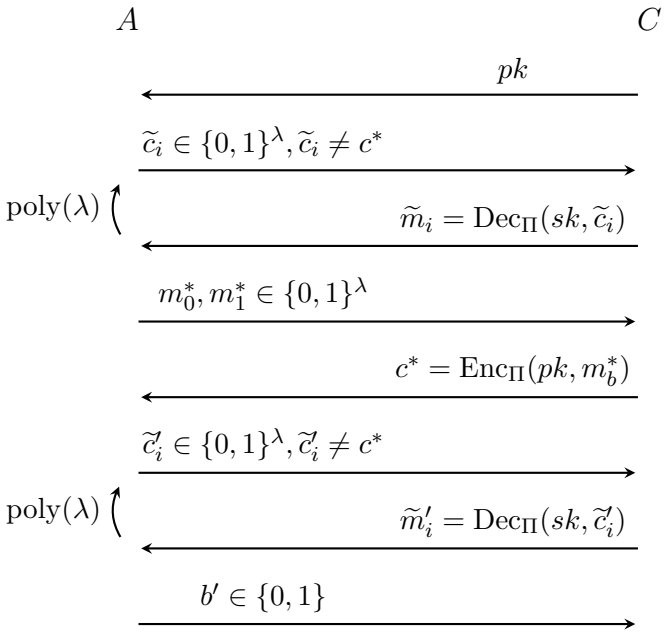

Figure 6.3: Graphical representation of $\mathrm{Game}^{\mathrm{CCA}}_{\Pi,A}(\lambda, b)$ for public-key encryption.

Thanks to our theoretical background with SKEs, we can jump straight away into constructing PKE schemes using the above two games. Our first PKE scheme is the **ElGamal scheme**.

Algorithm 6.1: ElGamal PKE scheme

Given a triple $(\mathbb{G}, g, q)$ where $\mathbb{G}$ is a cyclig group, $g \in \mathbb{G}$ is a generator and q is the order of g in $\mathbb{G}$, the ElGamal PKE scheme $\Pi = (\text{KGen}, \text{Enc}, \text{Dec})$ is defined as follows:

- $\text{KGen}(g, q) = (pk, sk)$ where $pk = g^x$ and $sk = x$, for some $x \in_R \mathbb{Z}_q$
- $\text{Enc}(pk, m) = (c_1, c_2)$ where $c_1 = g^r$ and $c_2 = pk^r \cdot m$, for some $r \in_R \mathbb{Z}_q$
- $\text{Dec}(sk, (c_1, c_2)) = \frac{c_2}{c_1^s k}$

The correctness of the scheme easily follows by simple substitution:

$$\text{Dec}(sk, (c_1, c_2)) = \frac{c_2}{c_1^s k} = \frac{pk^r \cdot m}{(g^r)^{sk}} = \frac{g^{xr} \cdot m}{g^{rx}} = m$$

Theorem 6.1

The ElGamal PKE scheme is CPA-secure under the DDH assumption.

Proof. Let $G(\lambda, b) = \text{Game}^{\text{CPA}}_{\Pi,A}(\lambda, b)$. We define an hybrid $H(\lambda, b)$ where pk is replaced by g^z for some $z \in_R \mathbb{Z}_q$. Assume that the DDH assumption holds for $\mathbb{G}$ and fix $b \in \{0, 1\}$. By way of contradiction, suppose that there is an adversary A that distinguishes $G(\lambda, b)$ from $H(\lambda, b)$ with non-negligible probability. We define a new adversary A' for the DDH assumption:

1. The challenger C sends $(\mathbb{G}, g, q, g^x, g^y, Z)$ to A', with Z being either g^{xy} or g^z for some $z \in_R \mathbb{Z}_q$
2. A' sends $pk = g^x$ to A, which replies with m_0^*, m_1^*
3. A' then sends $(g^y, Z \cdot m_b)$ to A
4. A replies with a guessing bit b', which gets forwarded to C

Since A' is able to distinguish $G(\lambda, b)$ from $H(\lambda, b)$ with non-negligible probability, they can distinguish $Z = g^{xy}$ from $Z = g^z$, meaning that A' wins breaks the DDH assumption on $\mathbb{G}$. Now, by definition we have $H(\lambda, 0) \equiv H(\lambda, 1)$ since we don't care about which of the two messages actually gets encrypted. This concludes that $G(\lambda, 0) \approx_c G(\lambda, 1)$. □

The simplicity of the ElGamal scheme comes at a price since its CCA game can be easily broken: after receiving the challenge ciphertext (c_1^*, c_2^*), we can query $\text{Dec}(sk, (c_1^*, c_2^* \cdot \widehat{m}'))$ with a fresh message m' and check if the answer is $m_0^* \widehat{m}'$ or $m_1^* \widehat{m}'$.

6.2 RSA encryption and trapdoor permutations

We study the computational security of the ever-famous Rivest-Shamir-Adleman (RSA) public-key encryption scheme. The original PKE construction was based on the hardness of factoring a number n composed of two primes, i.e. finding the primes $p, q \in \mathbb{P}$ such

that $n = pq$. As for the discrete-log problem, we state the hardness of factoring through a computational assumption, the **FACTORING assumption**.

Assumption 6.1: FACTORING assumption

The FACTORING assumption states that:

$$\Pr[A(n) = (p, q) : p, q \in_R \mathbb{P}, n = pq] \leq \text{negl}(\lambda)$$

for all PPT adversaries A and for a chosen value $\lambda > 0$

Algorithm 6.2: RSA PKE scheme

The RSA PKE scheme $\Pi = (\text{KGen}, \text{Enc}, \text{Dec})$ is defined as follows:

- $(pk, sk) \in_R \text{KGen}(1^\lambda)$ where $pk = (n, e)$ and $sk = (n, d)$, with $n = pq$ for two primes $p, q \in \mathbb{P}$ of approximately λ bits and e, d being each other's inverse in $\mathbb{Z}_{\varphi(n)}$, i.e. $ed \equiv 1 \pmod{\varphi(n)}$
- $\text{Enc}(pk, m) = m^e \pmod{n}$
- $\text{Dec}(sk, c) = c^d \pmod{n}$

The correctness of the RSA scheme comes from Euler's theorem, which we already discussed in the previous chapter. First, we observe that $\phi(n) = (p-1)(q-1)$. Second, we observe that $ed \equiv 1 \pmod{\phi(n)}$ implies that $ed = 1 + k\phi(n)$ for some $k \in \mathbb{Z}$. Therefore, assuming $c = \text{Enc}(pk, m)$, we have that:

$$\text{Dec}(sk, c) = c^d \equiv (m^{ed}) \equiv m^{k\phi(n)+1} \equiv m(m^{\phi(n)})^k \equiv m \pmod{n}$$

It's easy to see that the scheme is not CPA-secure since the scheme is deterministic. To fix this, we encode the message m into a string $\widehat{m}$ in a randomized – but invertible – fashion. The Public-key Cryptography Standard (PKCS) #1.5 sets $\widehat{m} = m \,||\, r$ with randomly chosen string r. Clearly, r must have strictly more that $\Omega(\log \lambda)$ bits, otherwise we could just find it through bruteforce. Unfortunately, for real-world values of r we can't prove the CPA-security of RSA even with PKCS #1.5, except when $m \in \{0, 1\}$. Nonetheless, this last result requires a stronger computational assumption about the RSA scheme itself.

The RSA assumption states that RSA is a **trapdoor permutation (TDP)**. A trapdoor permutation is formed by two functions f and f^{-1} that work, respectively, with a public and a secret key.The first function is assumed to be a *one-way function* even when the public key is known. The second function is an inverse of the first one, but it is must be efficiently computable only when the secret key is known (hence the name *trapdoor*).

Definition 6.1: Trapdoor permutation

We say that a triple (KGen, f, f^{-1}) with parameter λ is a trapdoor permutation when:

- $(pk, sk) \in_R \text{KGen}(1^\lambda)$, with $pk, sk \in \{0,1\}^\lambda$
- $f : \{0,1\}^\lambda \times \{0,1\}^n \to \{0,1\}^n$ is a OWF
- $f^{-1} : \{0,1\}^\lambda \times \{0,1\}^n \to \{0,1\}^n$ is such that $f^{-1}(sk, f(pk, x)) = x$ for all $x \in \{0,1\}^n$
- $f^{-1}(K, x)$ is efficiently computable only when $K = sk$

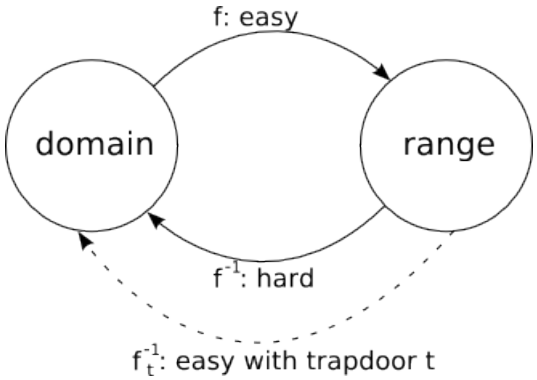

Figure 6.4: Graphical representation of a trapdoor permutation.

Assumption 6.2: RSA assumption

The RSA PKE scheme is a trapdoor permutation.

It's easy to see that the RSA assumption implies the FACTORING assumption: if we could easily solve factoring, we could also easily break RSA by finding the two primes p, q forming n. From trapdoor permutations we can easily construct CPA-secure PKE schemes using the hard-core predicate of the encrypting function.

Theorem 6.2

Let (KGen, f, f^{-1}) be a TDP. Given the hard-core predicate h of f, let $\Pi = (\text{KGen}, \text{Enc}, \text{Dec})$ be the PKE defined by:

$$\text{Enc}(pk, m) = (f(pk, r), h(pk, r) \oplus m)$$

$$\text{Dec}(sk, (c_0, c_1)) = c_1 \oplus h(pk, f^{-1}(sk, c_0))$$

with $m \in \{0,1\}$. Then, Π is CPA-secure.

Proof. Omitted. □

We also observe that PKE CPA-secure schemes for one bit can be extended to a polynomial amount of bits. Therefore, the RSA assumption and the above TDP construction implies that we can achieve CPA-secure public-key encryption. But what about CCA-security? The PKCS #2.0 uses a different encoding to achieve this in **Optimal Asymmetric Encryption Padding (OAEP)**. The idea is very similar to a 2-layer Feistel network using two different hash functions H, H' on $m \,||\, r$, with r is chosen uniform-at-random. The CCA-security of this scheme can be proven only under the RSA assumption and in the *Random Oracle Model (ROM)*: H, H' are assumed to be truly random hash functions and they can be computed only by querying an oracle.

Algorithm 6.3: OAEP PKE scheme

Let $\Pi_{\text{RSA}} = (\text{KGen}', \text{Enc}', \text{Dec}')$ be the RSA PKE and H, H' be two hash functions. The OAEP PKE scheme $\Pi_{\text{OAEP}} = (\text{KGen}, \text{Enc}, \text{Dec})$ is defined as follows:

- Fix $\lambda_1, \lambda_2 \in \mathbb{N}$ and sample $r \in_R \{0,1\}^{\lambda}$
- Let $s = m \,||\, 0^{\lambda_1} \oplus H(r)$ and $t = r \,||\, H'(s)$
- $\text{Enc}(pk, m) = (s \,||\, t)^e$, with $pk = (n, e)$ being the RSA public key

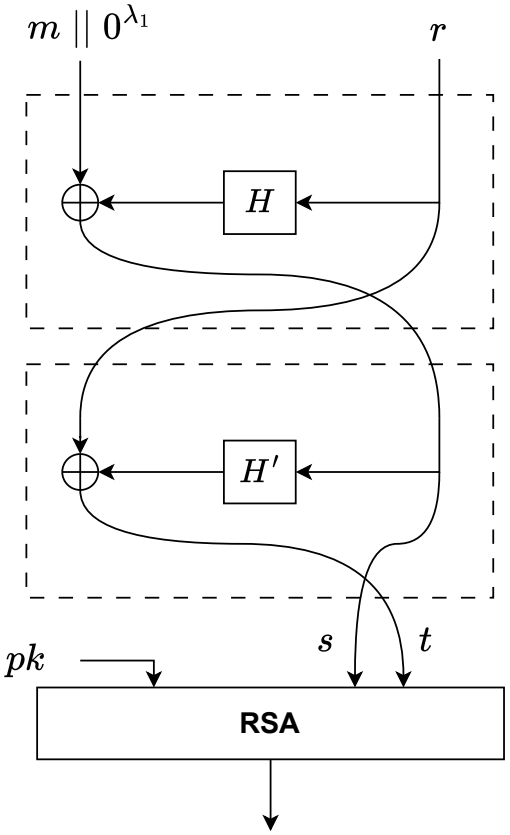

Figure 6.5: The OAEP scheme.

Theorem 6.3

The OAEP PKE scheme is CCA-secure under the RSA assumption in the ROM.

Proof. Omitted. □

6.3 Digital signatures

After proving that CCA-security is achievable also for public-key cryptography, we're left with constructing the equivalent of MACs for PKEs. In the context of asymmetric encryption, we talk about **Digital Signatures (DS)** schemes. The critical difference between MACs and DSs is that the signature is publicly verifiable: we don't need to know the key that was used for tagging. This gives birth to a some-what mirror concept of public-key encryption were we use the secret key to sign the message and then use the public key to verify it, instead of using the public key for encryption and the secret key for decryption.

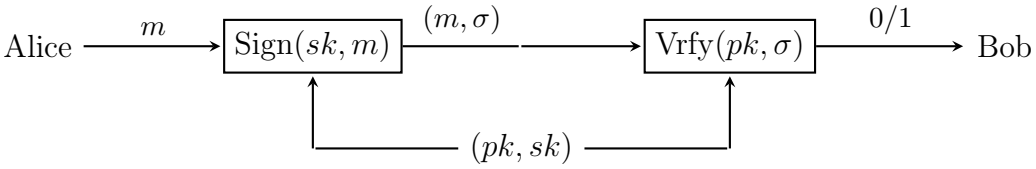

Figure 6.6: Example of digital signature.

Just like MACs, digital signatures are used to authenticate a message to guarantee its integrity. Moreover, the asymmetric paradigm allows us to form some sort of hierarchy of authentications. If Alice wants to know Bob's public key without being subject of a man-in-the-middle attack, they can ask the key to the *Certificate Authority (CA)* to which Bob previously sent his key in an authenticated manner. The idea here is that the CA is also authenticated by another CA of "higher grade", which is also authenticated by a CA of higher grade and so on and so forth, until we reach a final CA with the highest grade of authority (usually, a physical entity subject to very strict regulations). This is known as the Public Key Infrastructure (PKI).

The simplest digital signature scheme can be constructed through RSA itself. The idea is to use the decryption function as the signing function and the encryption function as the verifying function. This process simple idea works thanks to how the public and secret key are inverse of each other: we don't care about the order of application.

Algorithm 6.4: RSA DS scheme

The RSA DS scheme $\Pi = (\text{KGen}, \text{Sign}, \text{Vrfy})$ is defined as follows:

- $(pk, sk) \in_R \text{KGen}(1^\lambda)$ where $pk = (n, e)$ and $sk = (n, d)$, with $n = pq$ for two primes $p, q \in \mathbb{P}$ of approximately λ bits and e, d being each other's inverse in $\mathbb{Z}_{\varphi(n)}$, i.e. $ed \equiv 1 \pmod{\varphi(n)}$
- $\text{Sign}(sk, m) = (m, m^d \pmod{n})$
- $\text{Vrfy}(pk, (m, \sigma)) = 1$ if $m \equiv \sigma^e \pmod{n}$ and 0 otherwise.

For the computational security of digital signatures, we still use the concept of UFCMA-security. Similarly to what we did with CPA-security and CCA-security, the UFCMA game for PKEs is identical to the one for SKEs, with the addition of an initial message sent by the challenger containing the public key.

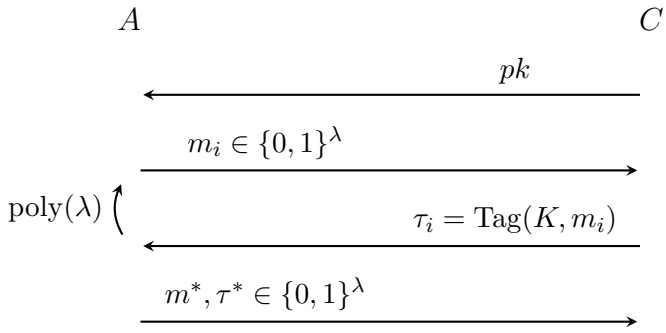

Figure 6.7: Graphical representation of $\text{Game}^{\text{UFCMA}}_{\Pi,A}(\lambda)$ for PKEs

It's easy to see that the standard RSA digital signature is not UFCMA-secure. Given two messages m_1 and m_2, let σ_1 and σ_2 be their signatures under RSA. Now, observe that:

$$(\sigma_1\sigma_2)^e \equiv \sigma_1^e\sigma_2^e \equiv m_1m_2 \pmod{n}$$

Therefore, an adversary can just query m_1, m_2 and then send the challenge pair $(m_1m_2, \sigma_1\sigma_2)$ to win with probability 1. Nonetheless, we can fix this issue with a simple trick: we can hash each message before signing it. Even more generally, we can use a trapdoor permutation instead of the RSA scheme. This is known as the **Full-Domain Hash (FDH)** signature scheme.

Algorithm 6.5: FDH DS scheme

Let (KGen, f, f^{-1}) be a TDP and let H be an hash function. The Full-Domain Hash (FDH) DS scheme $\Pi_{\text{FDH}} = (\text{KGen}, \text{Enc}, \text{Dec})$ is defined as follows:

- $(pk, sk) \in_R \text{KGen}(1^\lambda)$ where $pk = (n, e)$ and $sk = (n, d)$, with $n = pq$ for two primes $p, q \in \mathbb{P}$ of approximately λ bits and e, d being each other's inverse in $\mathbb{Z}_{\varphi(n)}$, i.e. $ed \equiv 1 \pmod{\varphi(n)}$
- $\text{Sign}(sk, m) = f^{-1}(sk, H(s))$
- $\text{Vrfy}(pk, (m, \sigma)) = 1$ if $H(m) = f(pk, \sigma)$ and 0 otherwise.

Theorem 6.4

The FDH signature scheme is UFCMA-secure in the ROM when constructed through a TDP.

Proof. This will be our first proof in the Random Oracle Model. We recall that in the ROM the hash function H is assumed to be truly random and that an attacker may compute it only through queries to an oracle.

Let $\Pi = (\text{KGen}, \text{Sign}, \text{Vrfy})$ be the FDH scheme constructed through a TDP (KGen, f, f^{-1}). We build a reduction from the distinguishability of the UFCMA game of Π to the OWF security of f. By way of contradiction, suppose that there is an adversary A^H that distinguishes the UFCMA game of Π with probability at least n^{-c} for some $c > 0$, where A^H also makes RO queries for H. We build a new adversary A as follows:

1. A will substitute the oracle for H, answering A^H RO queries.
2. The challenger C sends the pair (pk, y) to A, who then forwards pk to A
3. A^H tells A that they will make $q = \text{poly}(\lambda)$ RO queries.
4. A samples $j \in_R \mathcal{U}_{q_h}$ and sets H' as the function such that $H'(m_i) = y$ when $i = j$ and $H(m_i) = f(pk, x_i)$, for $x_i \in_R \mathcal{U}_n$, when $i \neq j$
5. A^H starts the RO queries. For each query m_i, A will answer with $y_i = H'(m_i)$. One of these RO queries will be for the future challenge message m^*, meaning that $m^* = m_{i^*}$ for some $i^* \in [q]$, but A doesn't know which one
6. A^H starts the sign queries. For each previously queried m_i such that $m_i \neq m^*$, A^H queries the signature of m_i to A, which replies with x_i if $i \neq j$ and with ABORT when $i = j$.
7. After receiving the ABORT message, A^H sends the challenge pair (m^*, σ^*) to A, who then forwards σ^* to C

We observe that if the signature σ^* sent by A^H is valid then σ^* is a pre-image of $y^* = H'(m^*)$. Therefore, the main trick of the reduction is to enforce with good enough probability that $y^* = y$. This is true if and only if $j = i^*$, which happens with probability $1/q$. If j is guessed correcly by A, A^H will find y with non-negligible probability.

Therefore, we conclude that:

$$\Pr[A \text{ wins}] = \Pr[j = i^*] \Pr[A^H \text{ wins } \mid j = i^*] > \frac{1}{qn^c}$$

□

7

Identification schemes

7.1 Σ-protocols and zero-knowledge proofs

An **identification scheme** is a protocol whereby Peggy the Prover proves to Victor the Verifier that she is indeed who she says she is. In practice, Peggy's identity is encoded in a private key a and a public key y. The protocol takes the form of Peggy proving to Victor that she has knowledge of the private key a. For instance, suppose that private key is a value $a \in \mathbb{Z}_{p-1}$ and that the public key y is such that $y \equiv x^a \pmod{p}$ for some $x \in \mathbb{Z}_p$ with $p \in \mathbb{P}$. Peggy can prove her to Victor by demonstrating that she knows the discrete logarithm of y to the base x.

In order for an identification scheme to be a valid method of identification, we require that Peggy must be able to always convince the Victor of her rightfulness (**perfect correctness**), even when they act through random choices – meaning that they are PPT algorithms. This requirement translates to making Victor accept with probability 1 for each possible exchange of messages, namely a *transcript*. We denote the set of all transcripts between a prover P and a verifier V with $P(pk, sk) \rightleftarrows V(pk)$.

Definition 7.1: Identification scheme

We say that a triple $\Pi = (\text{KGen}, P, V)$ is an identification scheme (IDS) when:

- $(pk, sk) \in_R \text{KGen}(1^\lambda)$, with $pk, sk \in \{0,1\}^\lambda$
- P and V are two PPT algorithms, called prover and verifier, such that $P(pk, sk)$ tries to prove to $V(pk)$ that it knows sk
- $P(pk, sk) \rightleftarrows V(pk)$ denotes the set of all transcripts, i.e. sequences of messages, between $P(pk, sk)$ and $V(pk)$
- $\text{out}(P(pk, sk) \rightleftarrows V(pk))$ is a r.v. such that $\text{out}(P(pk, sk) \rightleftarrows V(pk))(\tau) = 1$ when τ is a transcript that convinces V and 0 otherwise
- *Perfect correctness*: for all $\lambda \in \mathbb{N}$ and for all (pk, sk) it holds that:

$$\Pr[\text{out}(P(pk, sk) \rightleftarrows V(pk)) = 1] = 1$$

  meaning that the prover always convinces the verifier with probability 1

We observe that Peggy cannot simply tell Victor that the value is a to prove her identity in a secure way since a is her private key, meaning that he could impersonate her in the future. A viable identification scheme must prevent this from happening; we require that Victor can't impersonate Peggy even if she proves her identity to him polynomially many times (**passive security**). Differently from the other constraint, this one can be enforced through a game. Let $\text{Game}^{\text{ID}}_{\Pi,A}(\lambda)$ be the 2-player game defined as follows:

1. The adversary $A = (A_1, A_2)$ is composed of two separate algorithms.
2. The challenger C generates a key pair (pk, sk) and sends pk to the adversary A
3. A makes $q = \text{poly}(\lambda)$ transcript queries to C, which replies with $\tau_1, \ldots, \tau_q \in_R P(pk, sk) \rightleftarrows V(pk)$
4. A computes $A_1(\tau_1, \ldots, \tau_q) = s$, where s is a "state"
5. A samples $\tau^* \in_R A_2(s, pk) \rightleftarrows V(pk)$
6. A wins if $\text{out}(P(pk, sk) \rightleftarrows V(pk))(\tau^*) = 1$, otherwise C wins

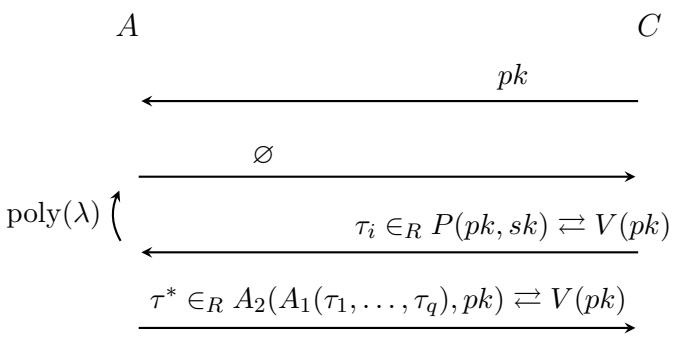

Figure 7.1: Graphical representation of $\text{Game}^{\text{ID}}_{\Pi,A}(\lambda)$.

Definition 7.2: Passive security for IDSs

Let $\Pi = (\text{KGen}, P, V)$ be an IDS. We say that Π has passive security if:

$$\Pr[\text{Game}^{\text{ID}}_{\Pi,A}(\lambda) = 1] \leq \text{negl}(\lambda)$$

for all PPT adversaries A

By definition of the game itself, proving that an IDS has passive security is really hard. Similarly to what we did for CCA-security, we'll prove a result that allows us to get passive security from simpler properties.

We will work with a particular type of identification schemes, called **Σ-protocols**. In these IDS we add a constraint on the length of the transcripts: only 3 messages are allowed before V declares the output (2 from the prover and 1 from the verifier). Each message is randomized for both parties (*3-move public coins protocol*). Moreover, we also want the first message of the prover to be hard to guess to avoid trivial protocols.

Definition 7.3: Σ-protocols

We say that a triple $\Pi = (\text{KGen}, P, V)$ is a Σ-protocol when:

- Π is an identification scheme where $P = (P_1, P_2)$ and $V = (V_1, V_2)$
- The transcriptions have three randomized messages α, β, γ:
  1. The prover sends $\alpha \in_R P_1(pk, sk)$
  2. The verifier answers with $\beta \in_R V_1(pk, \alpha)$
  3. The prover replies with $\gamma \in_R P_2(pk, sk, \alpha, \beta)$
  4. The verifier outputs $V_2(pk, sk, \alpha, \beta, \gamma)$
- *Non-triviality*: it's hard to predict the first message:

$$\forall \widehat{\alpha}\ \forall (pk, sk)\ \Pr[\alpha = \widehat{\alpha}] \leq \text{negl}(\lambda)$$

We give an example of a standard Σ-protocol, that is **Schnorr's protocol**.

Algorithm 7.1: Schnorr's Σ-protocol

The Schnorr Σ-protocol $\Sigma_{\text{Schorr}} = (\text{KGen}, \text{Sign}, \text{Vrfy})$ is defined as follows:

- $(\mathbb{G}, g, q)$ are the protocol parameters, where $\mathbb{G}$ is a cyclic group generated by $g \in \mathbb{G}$ with order q
- $(pk, sk) \in_R \text{KGen}(1^\lambda)$ where $pk = g^x$ and $s_k = x$, with $x \in_R \mathbb{Z}_q$
- The transcripts are given by:
  1. $P_1(pk, sk) = \alpha$ such that $\alpha = g^a$ for some $a \in_R \mathbb{Z}_q$
  2. $V_1(pk, \alpha) = \beta$ such that $\beta \in_R \mathbb{Z}_q$
  3. $P_2(pk, sk, \alpha, \beta) = \gamma$ such that $\gamma = \beta x + \alpha$
  4. $V_2(pk, \alpha, \beta, \gamma) = 1$ if $g^\gamma = \alpha(pk)^\beta$ and 0 otherwise

Through some algebraic manipulation, we can show that this protocol has perfect correctness. First, we observe that:

$$g^\gamma = g^{\beta x + \alpha} = \alpha(pk)^\beta$$

Then, we observe the fact that γ can be correctly computed if and only if the prover P really knows the value x of the secret key. After some tought, we can also convince ourselves of the protocol's non-leakage of the secret key. In fact, Schnorr's protocol also satisfies passive security and non-triviality, meaning that it is in fact a Σ-protocol.

To show the passive security of this protocol, we have to introduce two stronger properties that when summed up give passive security as a natural byproduct: **Honest-verifier Zero-knowledge (HVZK)** and **Special Soundness (SS)**. The first property tells us that real transcripts computed by the protocol are computationally indistinguishable from *simulated transcripts* produced by an simulation algorithm. In other words, the transcripts don't give any information to the verifier.

Definition 7.4: Honest-verifier Zero-knowledge

Let $\Pi = (\text{KGen}, P, V)$ be an IDS. We say that Π has Honest-verifier Zero-knowledge (HVZK) if:

$$(pk, sk, \text{Sim}(pk)) \approx_c (pk, skP(pk, sk) \rightleftarrows V(pk))$$

for some PPT simulator Sim

Proposition 7.1

Schnorr's protocol has HVZK.

Proof. Consider $\Sigma_{\text{Schnorr}} = (\text{KGen}, P, V)$ and let Sim be the such that $\text{Sim}(pk) = (\alpha', \beta', \gamma')$, where $\beta', \gamma' \in_R \mathbb{Z}_q$ and $\alpha' = g^{\gamma'}(pk)^{-\beta'}$.Given the value $a' = \gamma' - \beta' x$, we observe that $\alpha' = g^{\gamma' - \beta' x} = g^{a'}$.

Consider now a transcript $(\alpha, \beta, \gamma) \in_R P(pk, sk) \rightleftarrows V(pk)$. Since $\gamma = \beta x + \alpha$ and $a \in_R \mathbb{Z}_q$, we get that $(\alpha', \beta', \gamma')$ and (α, β, γ) have the same probability since the first and third element of both distributions constrain other in the same way. □

The second property, instead, is restricted only to Σ-protocols and it describes the impossibility of an adversary to forge valid transcripts starting from the same initial message α sent by P_1. Let $\text{Game}^{\text{SS}}_{\Pi,A}(\lambda)$ be the 2-player game defined as follows:

1. The challenger C generates (pk, sk) and sends pk to the adversary A
2. A returns two transcripts $\tau = (\alpha, \beta, \gamma)$ and $\tau' = (\alpha, \beta', \gamma')$
3. A wins if $\beta \neq \beta'$ and $V_2(p_k, \alpha, \beta, \gamma) = V_2(p_k, \alpha, \beta', \gamma') = 1$

Definition 7.5: Special Soundness

Let $\Sigma = (\text{KGen}, P, V)$ be a Σ-protocol. We say that Σ has special soundness if:

$$\Pr[\text{Game}^{\text{SS}}_{\Pi,A}(\lambda) = 1] \leq \text{negl}(\lambda)$$

for all PPT adversaries A

Proposition 7.2

Schnorr's protocol has special soundness under the DL assumption.

Proof. Consider $\Sigma_{\text{Schnorr}} = (\text{KGen}, P, V)$. By way of contradiction, suppose that there is a distinguisher A for the SS of Σ_{Schnorr}. Suppose that A finds two transcripts $\tau = (\alpha, \beta, \gamma)$ and $\tau' = (\alpha, \beta', \gamma')$ such that $\beta \neq \beta'$ and $V_2(p_k, \alpha, \beta, \gamma) = V_2(p_k, \alpha, \beta', \gamma') = 1$. Then, we have that:

$$g^{\gamma-\gamma'} = \alpha(pk)^{\beta}\alpha^{-1}(pk)^{-\beta'} \iff g^{\gamma-\gamma'} = (g^x)^{\beta-\beta} \iff g^x = g^{(\gamma-\gamma')(\beta-\beta')^{-1}}$$

Therefore, we can build a new distinguisher A' through A that computes x from g^x with non-negligible probability. □

Theorem 7.1: HVZK+SS implies passive security

Let $\Sigma = (\text{KGen}, P, V)$ be a Σ-protocol. Then, Σ has passive security if has HVZK and special soundness.

Proof. To do. □