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

October 11, 2025

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 *cyphertext*, i.e. the encrypted message

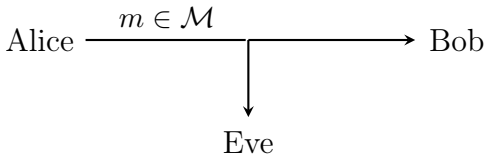

Figure 1.1: Without an encryption scheme, Eve – an evildoer – may be able to read the message that Alice is sending to Bob.

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

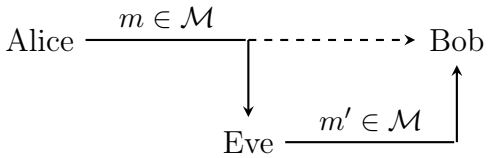

Figure 1.2: Without an encryption scheme, Eve may be able to intercept and alter the message that Alice is sending to Bob.

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 cyphertext space, i.e. the set of all strings of cyphertexts 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 **Secret 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 cyphertext
- A *decryption function* $\text{Dec} : \mathcal{K} \times \mathcal{C} \to \mathcal{M}$ that transforms a cyphertext 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 cyphertext $c \in \mathcal{C}$, 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:

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

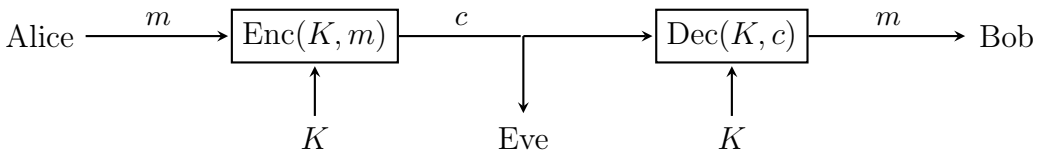

Figure 2.1: Example of SKE with perfect secrecy. Even if Eve intercepts the cyphertext, she cannot obtain the original message since she doesn't know the key.

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 cyphertext c is known. In other words, no cyphertext 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 $\forall c \in \mathcal{C}$ 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 cyphertext 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. $\Longrightarrow$ 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

1. 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:

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

Since every message has the same probability of being correct – without additional information – we know that:

$$Pr_{K \in_R \mathcal{K}}[M = m] = Pr_{K \in_R \mathcal{K}}[M = m']$$

Moreover, through a similar argument to the one above we get that:

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

2. Assume the third statement holds and fix $c \in \mathcal{C}$

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:

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

□

Theorem 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 cyphertext $c \in \mathcal{C}$. 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 cyphertexts that are unreachable by some messages, revealing information on the system.

Theorem 2.2: 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.

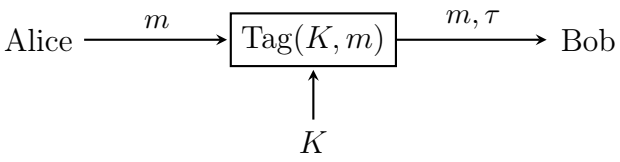

Figure 2.2: Example of an integrity-only MAC.

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 $\forall m, m_1, \ldots, m_t \in \mathcal{M}$ with $m \neq m'$ and $\forall \tau, \tau_1, \ldots, \tau_t \in \mathcal{T}$ 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 2.3

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 2.4

Let $\mathcal{H} = \{h_K : \mathcal{M} \to \mathcal{T}\}_{K \in \mathcal{K}}$ be a family of pairwise indepependent hash functions 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 2.1

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:

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

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 **pseudo-random 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 the quantity is to compare it with the more standard definition of *entropy*, defined as the expectation 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 $X \sim U_n$, where 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 2.2

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 $(S, \text{Ext}(S, X)) \sim_\varepsilon (S, U_\ell)$ when $S \sim U_d$.

We notice that the condition $(S, \text{Ext}(S, X)) \sim_\varepsilon (S, U_\ell)$ 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 2.3

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

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

TODO.

Problem 2.2

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

TODO.

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:

$$\Pr_{x \in_R \{0,1\}^n}[f(x) = y : y = f(x'); x \leftarrow A(y)] \leq \text{negl}(n)$$

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

3.2 Pseudo-random 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 pseudo-random 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: Pseudo-random 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 pseudo-random generator (PRG) with expansion factor ℓ if:

- G can be computed in $\text{poly}(n)$-time
- $G(U_n) \approx_c U_{n+\ell}$.

The above definition requires a bit of discussion in order to be fully understood. First, we observe that G is deterministic, making $G(U_n) \approx_c U_{n+\ell}$ a very strong property. Then, we observe that the idea 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 $G : \{0,1\}^n \to \{0,1\}^{n+1}$ up to any desired 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:

- $H_0 \approx_c G^\ell(U_n)$ and $H_\ell \approx_c U_{n+\ell}$
- $H_i \approx_c H_{i+1}$ for each $i \in [0, \ell - 1]$
- Each H_i is an hybrid of truly random and pseudo-random bits, where the number of truly random bits increases as i grows, getting replacing the pseudo-random 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 $f_i(s) = b_1 \ldots b_\ell s_\ell$ 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. $b_j \in_R U_1$
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 $b_j \in \{0,1\}$."

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 $\ell = \text{poly}(n)$. Let $H_0, \ldots, H_\ell$ be the distributions over the outputs of the functions $f_0, \ldots, f_\ell$. We observe that $H_\ell = U_{n+\ell}$ since every bit is truly random in f_ℓ. Hence, by letting $G^\ell = f_0$ we get that $H_0 = G^\ell(U_n)$ and $H_\ell = U_{n+\ell}$.

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 $G(U_n)$ from 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 $z = sb$, where $s \in \{0,1\}^n ame$ and $b \in \{0,1\}$.
2. Compute $f(s) = b_1 \ldots b_\ell s_\ell$.

3. Set $z' = b_1 \dots b_{i-1} b b_{i+1} \dots b_\ell s_\ell$.
4. Return $D_i(z')$.

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 $w \in G(U_n)$, 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 $w \in_R U_{n+1}$, 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:

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

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 $G^\ell(U_n) = H_0 \approx_c \dots \approx_c H_\ell = U_{n+\ell}$, meaning that G^ℓ is a PRG. □

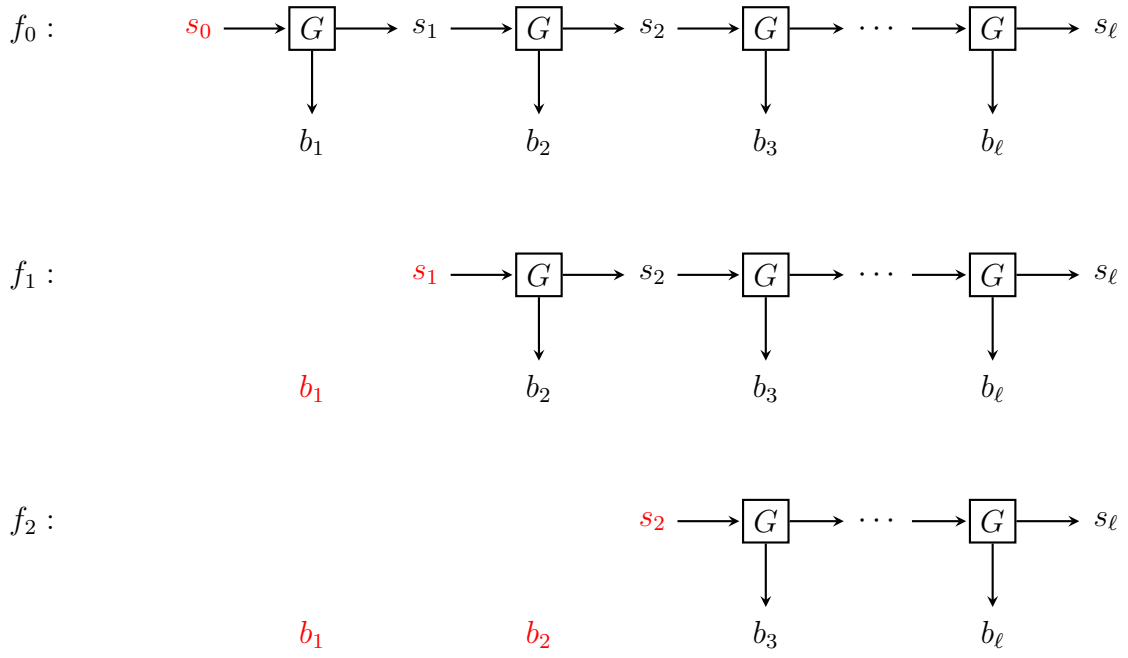

Figure 3.1: The idea behind the construction of the functions $f_0, \dots, f_\ell$. Red strings represent truly random bits.