Cryptography Notes

Raffaele Castagna

Academic Year 2025-2026

Contents

1 Intro to Cryptography **2**
1.1 Secure Communication . . . 2
1.2 Unconditional Security . . . 2
1.3 Perfect Secrecy . . . 3
1.4 OTP . . . 3
1.5 Proof that the lemmas imply eachother . . . 4
1.6 Message Authentication Codes . . . 5
1.7 Randomness Extraction . . . 6

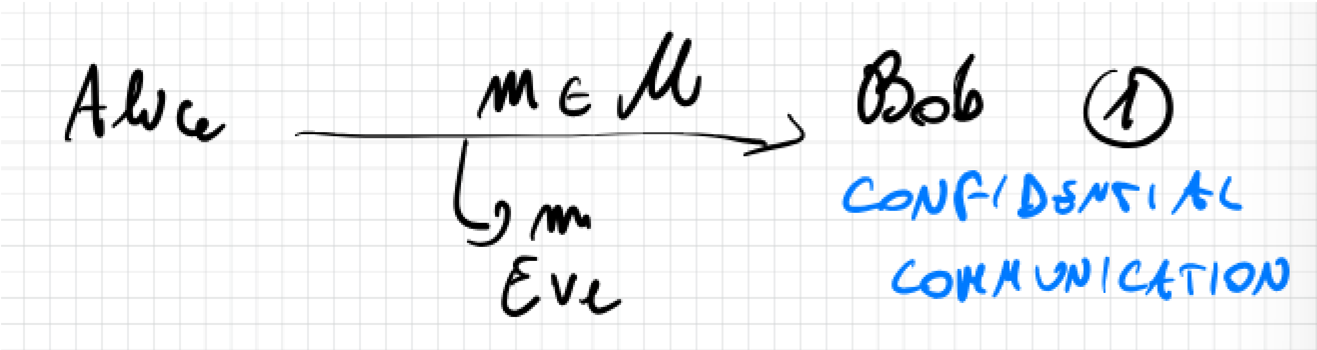
2 Computational Security **9**
2.1 Pseudorandomness . . . 10

1 Intro to Cryptography

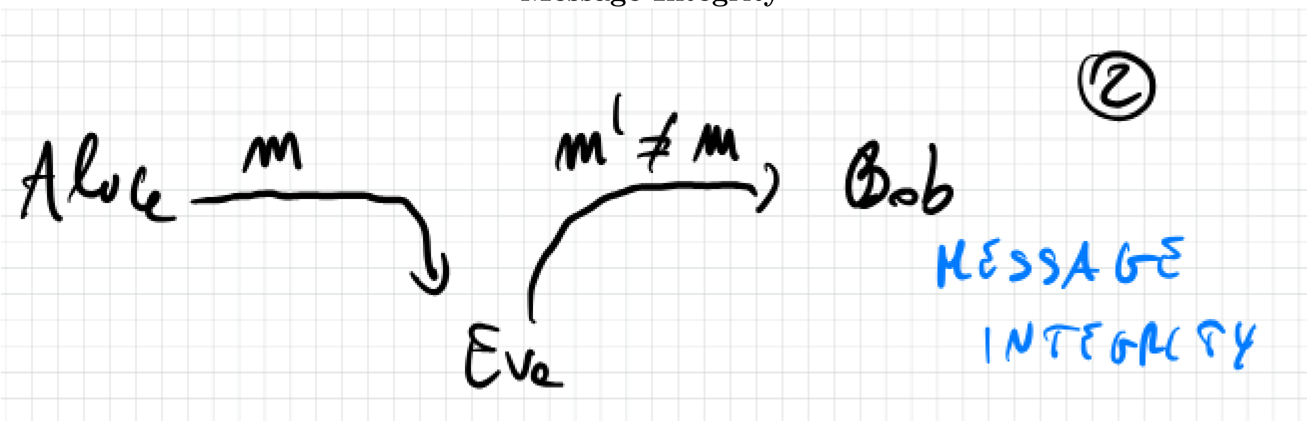
1.1 Secure Communication

We have multiple goals in cryptography, the most important ones being:

Confidential Integrity



Message Integrity



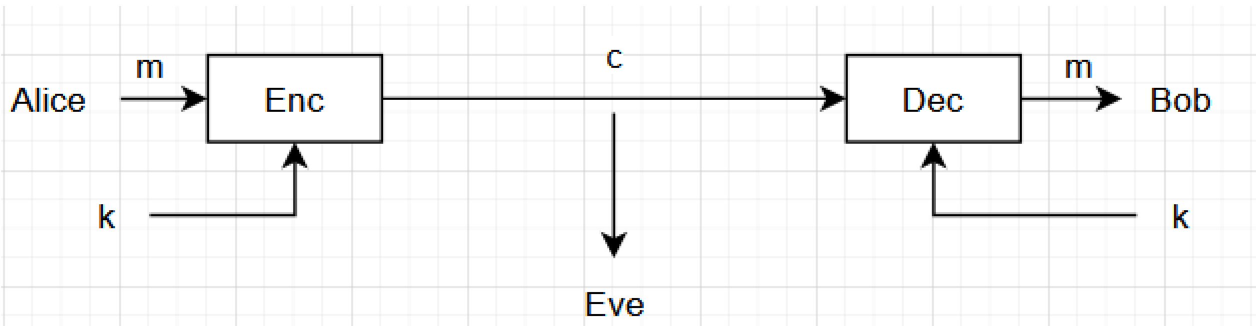
Basically we want our message to be both **confidential**, so no-one except the intended target sees it and we it to be unmodified, so that its **integrity** has not been compromised.

There are many different ways to do this, but in our case we only see two major ways:

- **Symmetric Cryptography**: Where Alice and Bob share a key $k \in \mathscr{K}$,the key is random and unknown to
- **Assymetric Cryptography**: Where Alice and Bob do not share a key, but they have each their own key pair (p_k, s_k) where p_k is the public key and s_k is the secret/private key

1.2 Unconditional Security

To achieve confidential communication, we use symmetric cryptography.



With $m \in \mathscr{M}, c \in \mathscr{C}, k \in \mathscr{K}$

In this case we have Alice sending a message m which is then encrypted utilizing a randomly generate key k to generate the cyphertext c, after that to get back to the initial message m, Bob will then need to decrypt it utilizing his own key k on cyphertext c.
In a more formal way we can define Symmetric encryption (SKE) as $\prod = (Enc, Dec)$ such that:

- Enc : $\mathscr{M} \times \mathscr{K} \to \mathscr{C}$
- Dec: $\mathscr{C} \times \mathscr{K} \to \mathscr{C}$
- k is uniform over $\mathscr{K}$ (k is chosen according to some distribution)

An encryption scheme must satisfy the correctness requirement:

Definition 1. *$\forall k \in \mathscr{K}, \forall m \in \mathscr{M}$ it holds that $Dec(k, Enc(k, m)) = m$*

Kerchoff's Principle:

Definition 2. *Security should not depend on the secrecy of the algorithm but on the secrecy of the key.*

1.3 Perfect Secrecy

Definition 3. *Let M be any distribution over $\mathscr{M}$ and K be uniform over $\mathscr{K}$ (Then observe C = Enc(K,M) in a distribution over C), we say that (Enc,Dec) = $\prod$ is **perfectly secret** if $\forall M, \forall m \in \mathscr{M}, \forall c \in \mathscr{C} : Pr[M = m] = Pr(M = m|C = c)$ (The probability that M is m is equal to the probability that M is m knowing that C is c, so by knowing the cyphertext, we dont gain additional information).*

Lemma 1. *The following are equivalent:*

- *Perfect Secrecy*
- *M and C are independant*
- *$\forall m, m' \in \mathscr{M}, \forall c \in \mathscr{C} : Pr[Enc(k, m) = c] = Pr[Enc(k, m') = c]$ with k being uniform over $\mathscr{K}$*

1.4 OTP

Let us see if OTP (*One Time Pad*) is perfectly secret
We know that the OTP uses $\oplus$ to generate and later decypher the cyphertext, we have that $K = M = C = \{0, 1\}^N$ with N being the length of the string, we know that:

- Enc (k,m) $= k \oplus m$
- Dec (k,c) $= c \oplus k = (k \oplus m) \oplus k = m$

To prove that it is perfectly secret let us utilize the third lemma:

$$Pr[C = c|M = m'] = Pr[\text{Enc}_k(m') = c] = Pr[m' \oplus K = c] = Pr[K = m' \oplus c] = 2^{-N}$$

and therefore:

$$Pr[Enc(k, m') = c] = 2^{-N}$$

There seem to be some limitations, the key can only be used once and it must as long as the message,lets assume we encrypt m" and m': $c_1 = k \oplus m_1$ $c_2 = k \oplus m_2$ therefore $c_1 \oplus c_2 = m_1 \oplus m_2$, so if I know a pair (m_1, c_1) then I could compute m_2, therefore we cannot encrypt two messages with the same key.

Theorem 1 (Shannon)**.** *Let $\prod$ be any perfectly secret SKE then we have $|\mathscr{K}| \geq |\mathscr{M}|$.*

Proof. Take $\prod$ to be uniform over $\mathscr{M}$. Take any c s.t. Pr[C=c] > 0.
Consider $\mathscr{M}' = \{Dec(k, c) : k \in \mathscr{K}\}$ and assume $|\mathscr{K}| < |\mathscr{M}|$ by contraddiction, then:

$$|\mathscr{M}|' \leq |\mathscr{K}| < |\mathscr{M}| \rightarrow |\mathscr{M}'| < |\mathscr{M}| \rightarrow \exists m \in \mathscr{M} \setminus \mathscr{M}'$$

Now:

$$Pr[M = m] = |\mathscr{M}|^{-1} \text{ but } Pr[M = m|C = c] = 0$$

□

1.5 Proof that the lemmas imply eachother

Let us prove that 1 $\implies$ 2 $\implies$ 3 $\implies$ 1

Let us start by proving that 1 $\implies$ 2:

Proof. We know that $Pr[M = m] = Pr[M = m|C = c] \rightarrow \frac{Pr[M=m \wedge C=c]}{Pr[C=c]} = Pr[M = m \wedge C = c]$
$= Pr[M = m] * Pr[C = c]$ and therefore we have proved their independence, so $I(M; C) = 0$ □

Let us prove that 2 $\implies$ 3

Proof. Let us fix an m from M and c from C:
$Pr[Enc(K, m) = c] = Pr[Enc(K, M) = c|M = m] \implies Pr[C = c|M = m] = Pr[C = c]$
Remember that Enc(...) is c!

We do the same thing for m' and we get: $Pr[C = c|M = m] = Pr[C = c]$ for both of them.
Therefore: $Pr[Enc(K, m') = c] = Pr[C = c]$ □

And now 3 $\implies$ 1: Take any c from C:
$Pr[C = c] = Pr[C = c|M = m]$ by 2 (we are claiming this)

If the claim is true then:
$Pr[M = m|C = c] * Pr[C = c] = Pr[M = m \wedge C = c] = Pr[C = c|M = m] * Pr[M = m] \implies$

$$\implies Pr[M = m] = \frac{Pr[M=m|C=c] * \cancel{Pr[C=c]}}{\cancel{Pr[C=c|M=m]}}$$

However we still need to prove the claim:

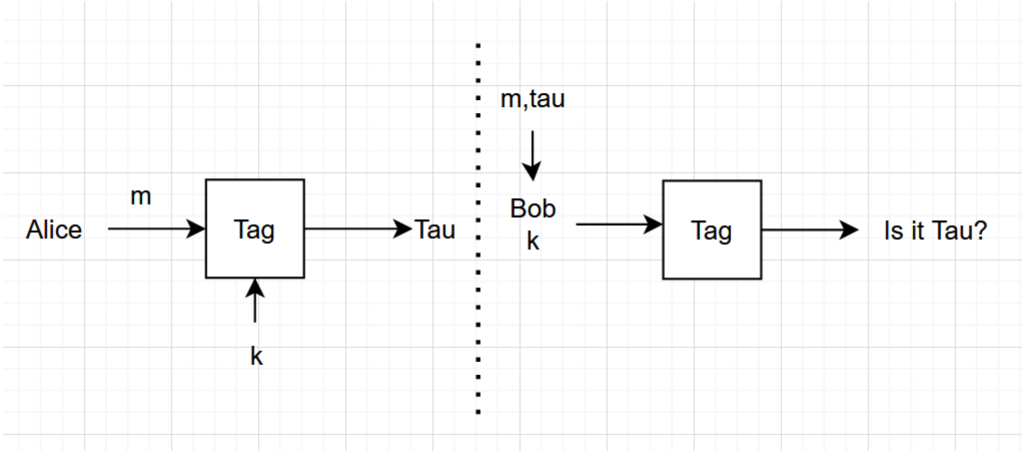
$$Pr[C = c] = \sum_{m'} Pr[C = c \wedge M = m'] = \sum_{m'} Pr[C = c|M = m'] * Pr[M = m'] =$$

$$\sum_{m'} Pr[Enc(K, m') = c|M = m'] * Pr[M = m'] = \sum_{m'} Pr[Enc(K, m') = c] * Pr[M = m']$$

$$\sum_{m'} Pr[Enc(k, m) = c] * Pr[M = m'] = Pr[Enc(k, m) = c] * \sum_{m'} Pr[M = m'] \impliedby 1$$

$$Pr[Enc(k, m) = c] = Pr[Enc(K, M) = c|M = m] \rightarrow Pr[C = c|M = m]$$

1.6 Message Authentication Codes



In case it is τ then we accept it, else no.
There is no need to prove correctness as τ is deterministic, so if we had the same k and m, we should get the same τ

Unforgeability

It should be hard to forge τ' such on msg m' and it should be hard to produce (m,τ) as long as m' $\neq$ m

Definition 4. ***Statistical secure MAC*** *We say that $\prod$ = Tag has ϵ-statistical security (unforgeability) if $\forall m, m' \in \mathscr{M}$ with $m \neq m'$ $\forall \tau, \tau' \in \mathscr{T}$:*

$$Pr[Tag(K, m') = \tau' \mid Tag(K, m) = \tau] \leq \epsilon$$

TLDR: Fix any m,m' with m' $\neq$ m take τ, τ' on the condition that τ is tag of m and given τ', it is always less than or equal to ϵ
Here ϵ is a parameter e.g. 2^{-80}
Exercise Let us prove that it is impossible to get $\epsilon = 0$
Because a random $\tau' \in \mathscr{T}$ has probability $\geq \frac{1}{|\mathscr{T}|}$ to be correct it is impossible.

Note that the definition is valid for One-Time!
We will show:

- The notion is Achievable
- It's inefficient, in fact:

Theorem 2. *Any t-time $2^{-\lambda}$ statistically secure Tag has a key of some (t+1)*λ*

We will now show that any form of hash function with a particular property satisfies the definition.

Definition 5. ***Pairwise independence*** *A family $\mathscr{H} = \{h_k : \mathscr{M} \to \mathscr{T}\}_{k \in \mathscr{K}}$ is pairwise independant if: $\forall m, m' \in \mathscr{M}$ s.t.m $\neq m'$ then: $(h(K, m), h(K, m'))$ is uniform over $\mathscr{T}^2 = \mathscr{T} \times \mathscr{T}$ for K uniform over $\mathscr{K}$*

Theorem 3. *Any family $\mathscr{H}$ of pairwise independent functions directly gives a $\epsilon = \frac{1}{|\mathscr{T}|}$ − statistically secure MAC.*

Proof. Fix any $m \in \mathscr{M}, \tau \in \mathscr{T}$:

$$Pr[Tag(K, m) = \tau] =$$

$$Pr_k[h(K,m) = \tau] = \frac{1}{|\mathscr{T}|} \text{ by pairwise independence}$$

Similiarly, for any m,m' s.t. m $\neq$ m', $\tau, \tau' \in \mathscr{T}$.

$$Pr_k[Tag(K,m) = \tau \wedge Tag(K,m') = \tau'] =$$

$$Pr_k[h(K,m) = \tau \wedge h(K,m') = \tau'] = \frac{1}{|\mathscr{T}|^2}$$

By Bayes:

$$Pr[Tag(K,m') = \tau' | Tag(K,m) = \tau] =$$

$$\frac{Pr[h(K,m;) = \tau' \wedge h(K,m) = \tau]}{Pr[h(K,m) = \tau]} =$$

$$\frac{\frac{1}{|\mathscr{T}|^2}}{\frac{1}{|\mathscr{T}|}} = \frac{1}{|\mathscr{T}|}$$

□

Now we need to instantiate it, here is a construction, Let p be a prime:

$$h_{a,b}(m) = am + b \mod p$$

$$k = (a,b) \in \mathbb{Z}_p^2 = \mathscr{K}$$

$$\mathbb{Z}_p = \mathscr{M} = \mathscr{T}$$

Lemma 2. *The above $\mathscr{H}$ is pairwise independant.*

Proof. For all $m, m' \in \mathbb{Z}_p, \tau, \tau' \in \mathbb{Z}_p$ with $m \neq m'$

$$\Pr_{(a,b)\in\mathbb{Z}_p^2}[h_{a,b}(m) = \tau \wedge h_{a,b}(m') = \tau'] =$$

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

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

$$\frac{1}{p^2} = \frac{1}{|\mathbb{Z}_p|^2} = \frac{1}{|\mathscr{T}|^2}$$

□

1.7 Randomness Extraction

Alice and Bob need a **random** key, how can they generate it?
Randomness is crucial for crypto, and two components are necessary in any RNG (e.g. Fortuna, /dev/rand):

- Randomness extraction: By measuring physical quantities we can get an **unpredictable** sequence of bits (Not necessarily uniform or for cheap!)
  From this we extract a **random** Y which is short (e.g. 256 bits)
- Expand it to any amount (polynomial) using a psedorandom generator (PRG) - but this requires computational assumptions.

We want to understand how to extract from an unpredictable source X.
Example ***Von Neumann Extractor*** Assume $B \in 0, 1$ s.t. $\Pr[B = 0] = p < \frac{1}{2}$.

- Sample $b_1 \in B, b_2 \in B$
- if $b_1 = b_2$ then Resample
- Else output 1 $\iff b_1 = 0, b_2 = 1$, or 0 if $b_1 = 1, b_2 = 0$

Assuming it outputs something, this will be s.t.

$$\Pr[\text{Output } 0] = Pr[\text{Output } 1] = p * (1 - p)$$

$$\Pr[\text{No output after N tries}] = (1 - 2p(1 - p))^N \text{ which becomes small for large enough N}$$

We want to generalize this question, ideally we want to design a function Ext that takes a random variable X and outputs an uniform Ext(X), but this is impossible as the source must be unpredictable and Ext is deterministic

Definition 6 (Min-Entropy)**.** *The min-entropy of X is:* $H_\infty = -\log_2 \max \Pr[X = x]$

Example: Let $X \equiv U_m$ Uniform over $\{0, 1\}^N$. $H_\infty(X) = N$
If X is a costant we have $H_\infty(X) = 0$

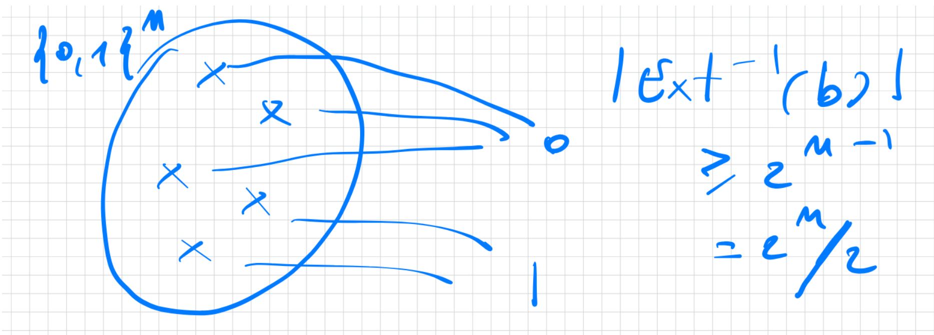
Here's the next best thing:
Design Ext that extracts UNIFORM $Y = Ext(X)$ for every X s.t. $H_\infty(X) \geq k$
But this is also impossible, even if

$$Ext(X) = b \in \{0, 1\}$$

$$k = n - 1$$

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

And here's why: fix any Ext:$\{0, 1\}^n \to 0, 1$ and let $b \in 0, 1$ be the output of maximing $|Ext^{-1}(b)|$



The bad X: Define X to be Uniform over $Ext^{-1}(b)$. Since it is uniform: $H_\infty(X) \geq n-1$ but $Ext(X) = b$ so not uniform.

Solution: Swap the quantifiers.

Definition 7 (Seeded Extractor)**.** *A function* $Ext : \{0, 1\}^n \times \{0, 1\}^d \to \{0, 1\}$ *is a (k,*ϵ*)-seeded extractor if for every X over s.t.* $H_\infty(X) \geq k$*:*

$$(S, Ext(S, X)) \approx_\epsilon (S, U_e)$$

for $S \equiv U_d$ *(uniform over* $\{0, 1\}^d$*). (Note that* $(S, U_e) \equiv U_{d+e}$*).*

What does this mean? There is a standard way to measure distance between distributions:

$$Z \equiv_\epsilon Z' \iff SD(Z, Z') \leq \epsilon$$

$$SD(Z, Z') = \frac{1}{2} \sum_z |Pr[Z = z] - Pr[Z' = z]|$$

This is equivalent: $\forall$ Unbounded adversary A:

$$|Pr[A(z) = 1 : z \in Z] - Pr[A(z) = 1 : z \in Z']| \leq \epsilon$$

Theorem 4 (Leftover Hash Lemma)**.** *Let $\mathscr{H} = \{h_s : \{0,1\}^n \to \{0,1\}^l\}_{s \in \{0,1\}^d}$ be a family of pairwise independent hash functions. Then Ext(x,s) = $h_s(x)$ is a (k,ϵ)-seeded extractor for $k \geq l + 2\log_2(\frac{1}{\epsilon})$-2.*

Lemma 3. *Let Y be a RV over $\mathscr{Y}$. Such that:*

$$Col(Y) = \sum_{y \in \mathscr{Y}} Pr[Y = y]^2 \leq \frac{1}{|\mathscr{Y}|} * (1 + 4\epsilon^2)$$

Then, SD(Y,U) $\leq \epsilon$

Proof.

$$SD(Y, U) = \frac{1}{2} \sum_{y \in \mathscr{Y}} |Pr[Y = y] - \Pr[U = y]|$$

$$\frac{1}{2} \sum_{y \in \mathscr{Y}} |Pr[Y = y] - \frac{1}{|\mathscr{Y}|}|$$

$$\text{Let } q_y = Pr[Y = y] - \frac{1}{|\mathscr{Y}|}$$

$$\text{Let } s_y = \begin{cases} 1 \text{ if } q_y \geq 0 \\ -1 \text{ else} \end{cases}$$

$$\text{Hence } SD(Y, U) = \frac{1}{2} \sum_{y \in \mathscr{Y}} s_y q_y$$

$$= \frac{1}{2} \langle s, q \rangle \leq \frac{1}{2} \sqrt{< \vec{q}, \vec{q} > * < \vec{s}, \vec{s} >} \text{ by Cauchy-Schwarz}$$

$$= \frac{1}{2} \sqrt{\sum_{y \in \mathscr{Y}} q_y^2 * |\mathscr{Y}|}$$

Now, We analyze the term $\sum_{y \in \mathscr{Y}} q_y^2$:

$$\sum_{y \in \mathscr{Y}} q_y^2 = \sum_{y \in \mathscr{Y}} (Pr[Y = y] - \frac{1}{|\mathscr{Y}|})^2 =$$

$$\sum_{y \in \mathscr{Y}} Pr[Y = y]^2 + \frac{1}{|\mathscr{Y}|^2} - 2\frac{Pr[Y = y]}{|\mathscr{Y}|} =$$

$$\underbrace{\sum_{y \in \mathscr{Y}} \Pr[Y = y]^2}_{Col(Y)} + \frac{1}{|\mathscr{Y}|} - 2\frac{1}{|\mathscr{Y}|} =$$

$$Col(Y) - \frac{1}{|\mathscr{Y}|} \leq \frac{4\epsilon^2}{|\mathscr{Y}|}$$

Then:

$$SD(Y, U) \leq \frac{1}{2} \sqrt{\frac{4\epsilon^2}{\cancel{|\mathscr{Y}|}} * \cancel{|\mathscr{Y}|}} = \epsilon$$

$\square$

Next we apply the lemma to prove the Leftover Hash Lemma:

Proof.

$$Y = (S, Ext(X, S)) = (S, h(S, X))$$

and compute Col(Y):

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

$$= \Pr[S = S' \wedge h(S, X) = h(S', X')]$$

$$= \Pr[S = S' \wedge h(S, X) = h(S, X')]$$

$$= \Pr[S = S'] * \Pr[h(S, X) = h(S, X')]$$

$$= \frac{1}{2^d} * \Pr[h(S, X) = h(S, X')]$$

$$= \frac{1}{2^d} * (\Pr[X = X'] + \Pr[h(S, X) = h(S, X') \wedge X \neq X'])$$

$$\leq \frac{1}{2^d} * (\frac{1}{2^k} + \frac{1}{2^l}) \text{ by pairwise independence and } H_\infty(X) \geq k$$

$$= \frac{1}{2^{d+l}}(1 + 2^{l-k}) \leq \frac{1}{2^{d+l}}(2^{2-2\log_2(\frac{1}{\epsilon})} + 1)$$

$$= \frac{1}{|\mathscr{Y}|} * (1 + 4\epsilon^2)$$

□

2 Computational Security

We know that withouth any assumptions we can do Symmetric crypto and randomness generation, with some strong limitations.

- Privacy: $|msg| = |key|$ and one-time use
- Integrity: same as above.
- Randomness We can't extract more than k from $p_y k$

We want to overcome all these limitations. We'll do so off of the base of some assumptions

- Adversary is Computationally Bounded
- Hard Problems exist

We will make conditional statements:

Theorem 5. *If Problem X is hard (against efficient solvers), Then cryptosystem $\prod$ is secure (against efficient adversaries)*

Consequence: if $\prod$ is insecure, $\exists$ efficient solver for X!
Depending on what X is, the above could be **Groundbreaking**.

Examples:
X = "$P \neq NP$" X = "Factoring is hard"
X = "Discrete Log is hard"

We are not able to just assume $P \neq NP$, we need a stronger assumption: **One-Way Functions**: These are functions that are easy to compute but hard to invert.

Clearly OWF $\implies P \neq NP$, why?
Because if $P = NP$, OWF do not exist as checking if f(x) = y is efficient and this it's in NP=P
We cannot exclude that $P \neq NP$ but still,OWF do not exist.

To better demonstrate this, we can refer to the following worlds created by Russel Impagliazzo:

- Algorithmica: P=NP
- Heuristica: P $\neq$ NP but no "average-hard" problems
- Pessiland: $P \neq NP$ and "average-hard" problems exist, but no OWF
- Minicrypt: OWFs exist
- Cryptomania: OWF exist + Public-key crypto exist

First we must start by fixing a model of computation: Turing Machines
efficient computation = polynomial time TMs.
Let's be generous: Adversaries can use any amount (polynomial) of randomness: Probabilistic Polynomial Time (PPT) TMs.
In what comes next we could define two approaches:

- **Concrete Security** Security hols w.r.t. t-time Tms except w.p. $\leq \epsilon$ (e.g. $t = 2^20$ steps, $\epsilon = 2^{-80}$)
- **Asymptotic Security** Let λ be a security parameter. Adversaries are poly(λ)-time PPT TMs (ϵ = negligeble = negl(λ))

Definition 8 (Negligible). *$\epsilon : \mathbb{N} \to \mathbb{R}$ is negligible if $\forall p(\lambda) = poly(\lambda)\ \exists \lambda_0 \in \mathbb{N}$ s.t. $\forall \lambda > \lambda_0 : \epsilon(\lambda) \leq \frac{1}{p(\lambda)}$*
(In other words, $\epsilon(\lambda) \leq O(\frac{1}{p(\lambda)})\ \forall p(\lambda) = \text{poly}(\lambda)$)

2.1 Pseudorandomness

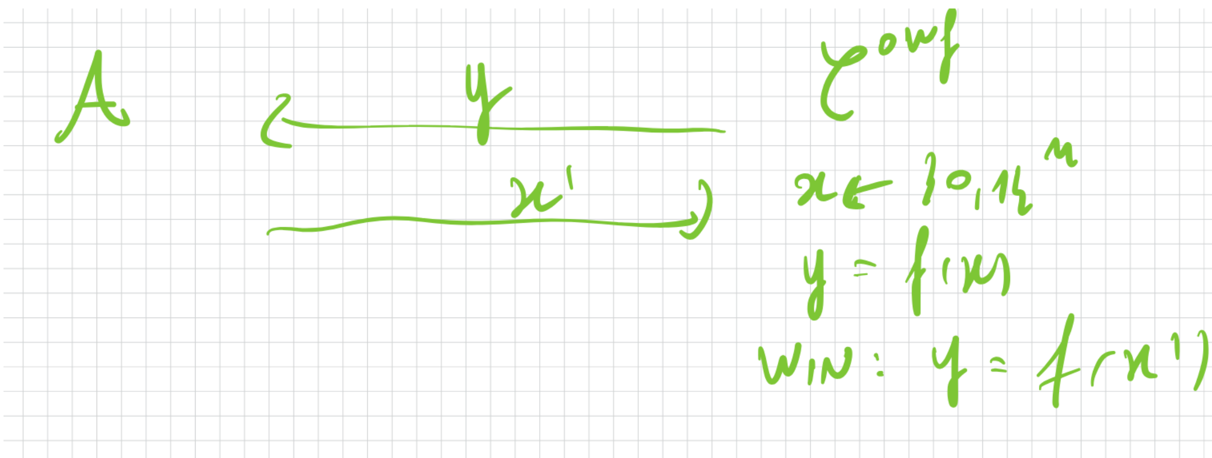
This is our first step towards efficient symmetric crypto. Moreover, pseudorandomness is used in modern computers to simulate real randomness. We will see that OWF are enough for pseudorandomness.

Definition 9 (OWF). *A function $f : \{0,1\}^n \to \{0,1\}^n$ is One-Way, if: $\forall PPT \mathscr{A}$:*

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

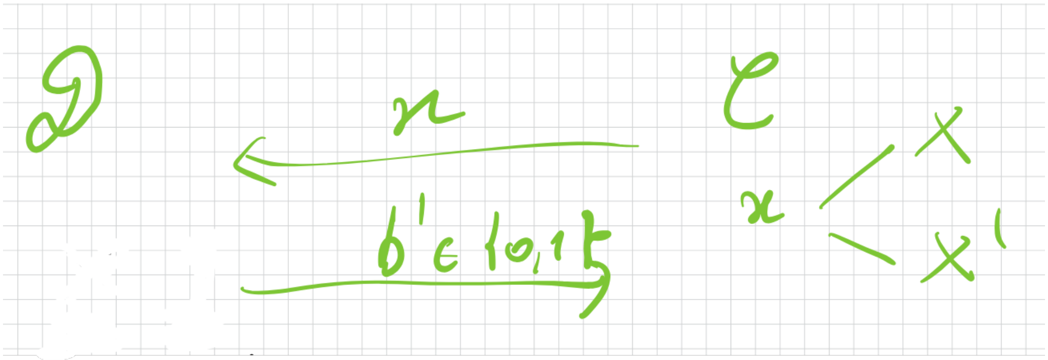
Informally, it goes to zero faster than any inverse of a polynomial function.

Example of negl(n) is 2^{-n}
An alternative way to think about it:



Definition 10 (Pseudorandomness)**.** *Pseudorandomness is a sequence of bits that are not random, but look random. We capture this requirement using* ***Indistinguishability (computational)****.*
We have already seen something like this in SD. Given X,X' RVs over some domain, SD(X,X') $\leq \epsilon$ *is equivalent to:* $\forall \mathscr{D}$ *(adversary):*

$$|Pr[\mathscr{D}(x) = 1 : x \leftarrow X] - Pr[\mathscr{D}(y) = 1 : y \leftarrow X']| \leq \epsilon$$



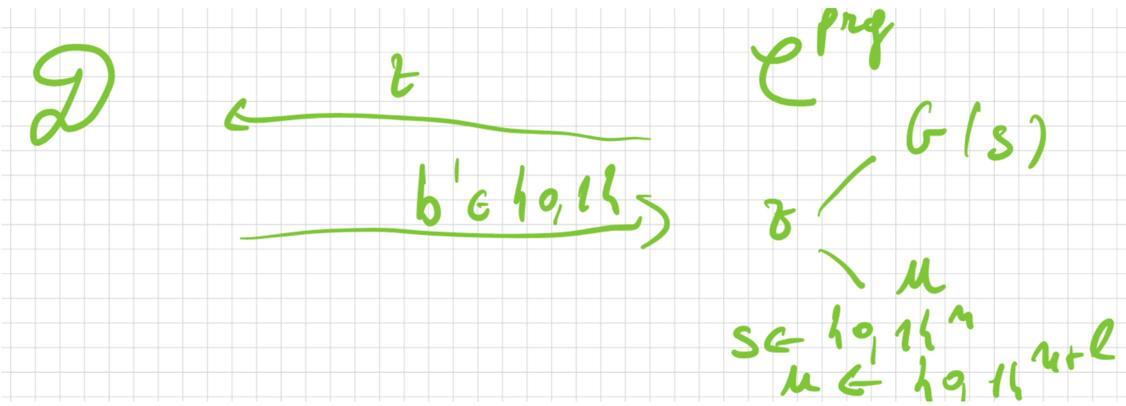
Definition 11. *X* (X_n)*, Y* (Y_n) *are computationally indistinguishable* $(X \approx_c Y)$ *if* $\forall PPT \mathscr{D}$*:*

$$|Pr[\mathscr{D}(z) = 1 : z \leftarrow X_n] - Pr[\mathscr{D}(z) = 1 : x' \leftarrow Y_n]| \leq negl(n)$$

With this we can define pseudorandomness:

Definition 12 (Pseudorandom Generator (PRG))**.** *A function* $G : \{0,1\}^n \to \{0,1\}^{n+l}$ *with* $l \geq 1$ *(The Stretch) is secure if:*

$$G(U_n) \approx_c U_{n+l}$$
$$U_n \equiv \text{ uniform over } \{0,1\}^n$$
$$U_{n+l} \equiv \text{ uniform over } \{0,1\}^{n+l}$$



Let's understand how to build PRGs:

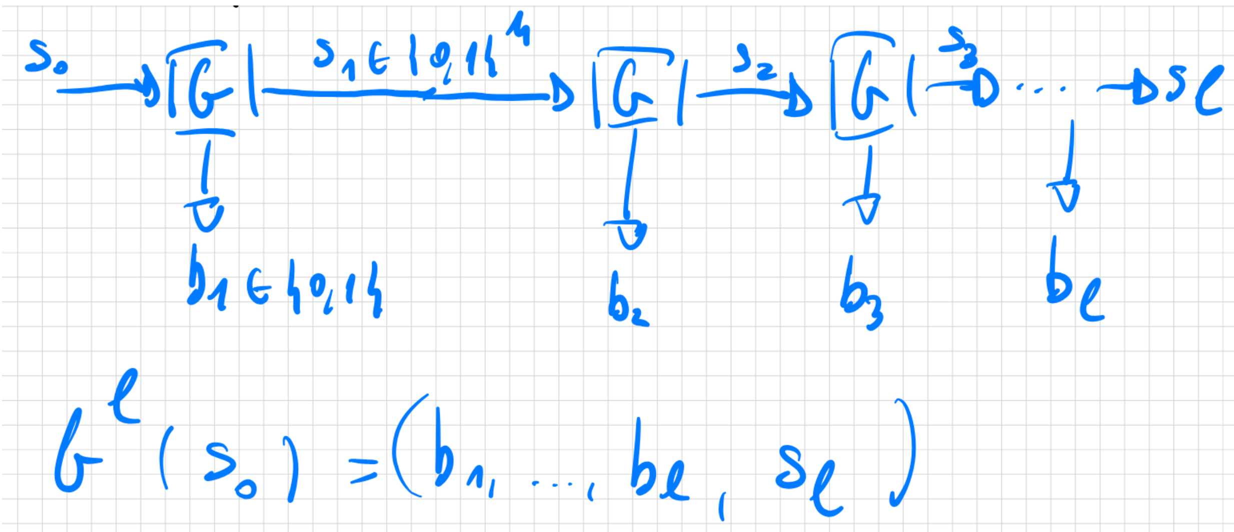
- Use a randomness extractor to get a uniform seed $s \in \{0, , 1\}^n$.
- Define a simple PRG $G : \{0,1\}^n \to \{0,1\}^{n+1}$ with minimal stretch $l = 1$.
- Use G to stretch any $l(n) = poly(n)$.

Theory vs Practice:

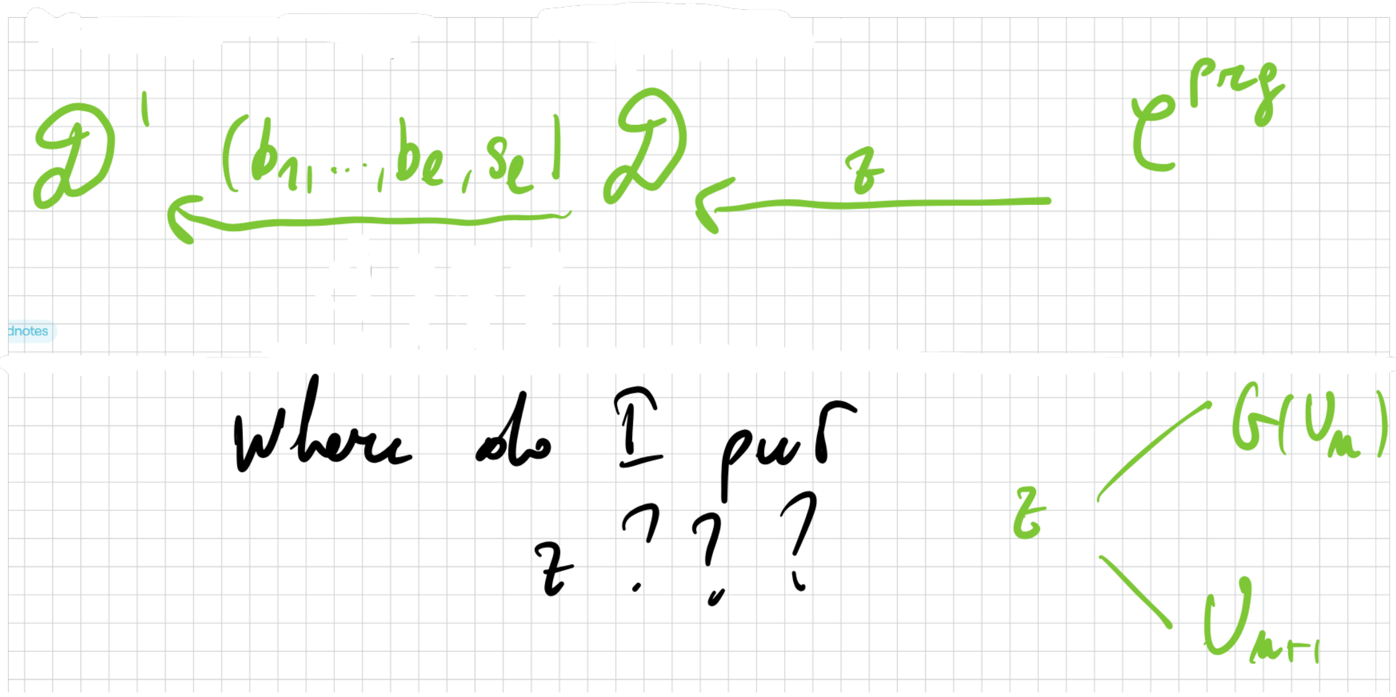
- Randomness extraction is what we already studied. But in practice it is done using Hash Functions.
- Theoretical G can be obtained from any OWF. Practical G is Heuristic

- Stretch is the same
- In practice the seed is refreshed periodically collecting new entropy

Theorem 6. *If there exists a PRG $G : \{0,1\}^n \to \{0,1\}^{n+1}$, then there exists a PRG $G^l : \{0,1\}^n \to \{0,1\}^{n+l}$ for any $l(n) = poly(n)$*



Proof. Assume G^l not secure, $\exists$ PPT $\mathscr{D}^l$ that can distinguish $G^l(U_n)$ from U_{n+l} with probability $\geq \frac{1}{p(n)}$ for some polynomial. We want to buildt PPT $\mathscr{D}$ that can distinguish $G(U_n)$ from U_{n+1} with probability $\frac{1}{p(n)}$. ($\mathscr{D}$ is called a reduction)



Hybrid argument

$$H_0(n) \equiv G^n(U_m)$$

$$b_1, \dots, b_\ell \leftarrow \{0,1\}^\ell$$

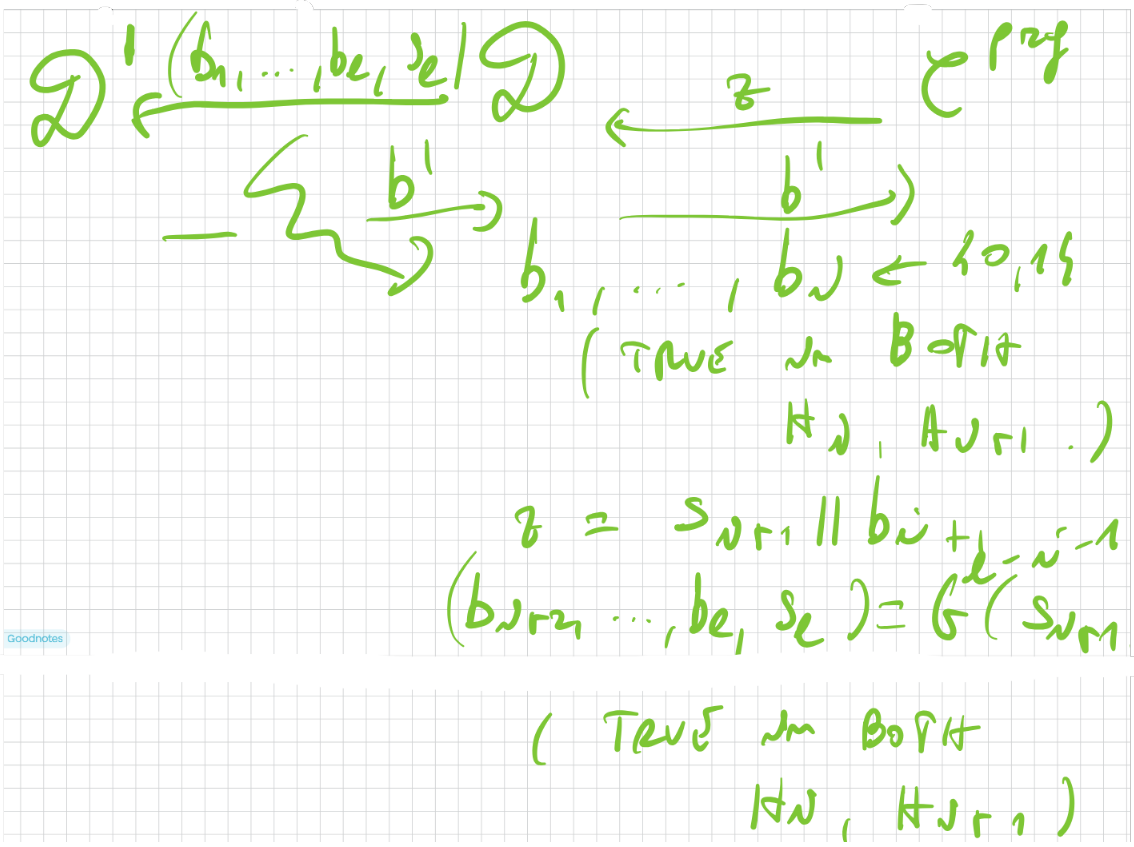
$$H_i(M) \equiv \begin{cases} b_1, \ldots, b_i \leftarrow \{0,1\} \\ s_i \leftarrow \{0,1\}^n \\ (b_{i+1}, \ldots, b_\ell, s_\ell) = G(s_i) \end{cases}$$

$$H_\ell(n) \equiv U_{\ell+m}$$

□

Lemma 4. $\forall i : H_i \approx_c H_{i+1}$.

Proof. By reduction (as before):



By the above observations:

$$\Pr[\mathscr{D}(z) = 1 : z = G(s); s \in \{0,1\}^n]$$
$$= \Pr[\mathscr{D}'(b1, \ldots, b_\ell, s_ell) = 1 : (b1, \ldots, b_\ell, s_ell) \in H_i(n)]$$
$$\Pr[\mathscr{D}(z) = 1 : z \leftarrow U_{n+1}] = \Pr[\mathscr{D}'(b1, \ldots, b_\ell, s_ell) = 1 : (b1, \ldots, b_\ell, s_ell) \in H_{n+1}(n)] \implies$$
$$|\Pr[\mathscr{D}(z) = 1 : z = G(U_n)] - \Pr[\mathscr{D}(z) = 1 : z \in U_n + 1]| \geq \frac{1}{p'(n)}.$$
$$\implies H_i \approx_c H_i + 1$$

□