Cryptography Notes

Raffaele Castagna

Academic Year 2025-2026

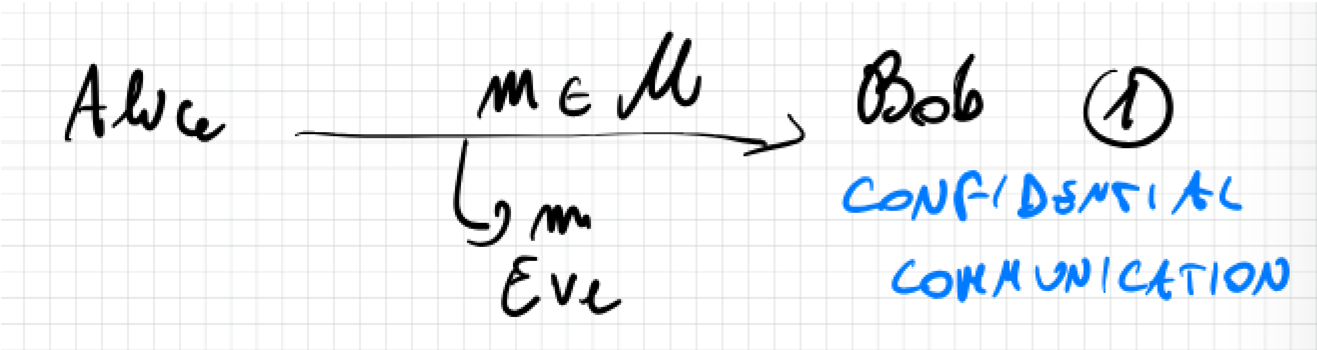
Contents

1 Intro to Cryptography

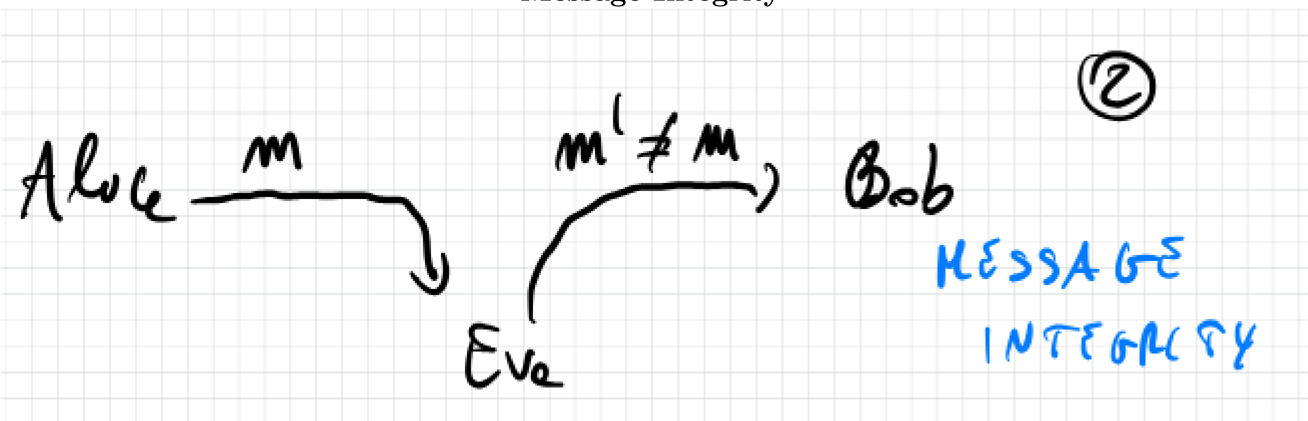
1.1 Secure Communication

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

Confidential Integrity



Message Integrity



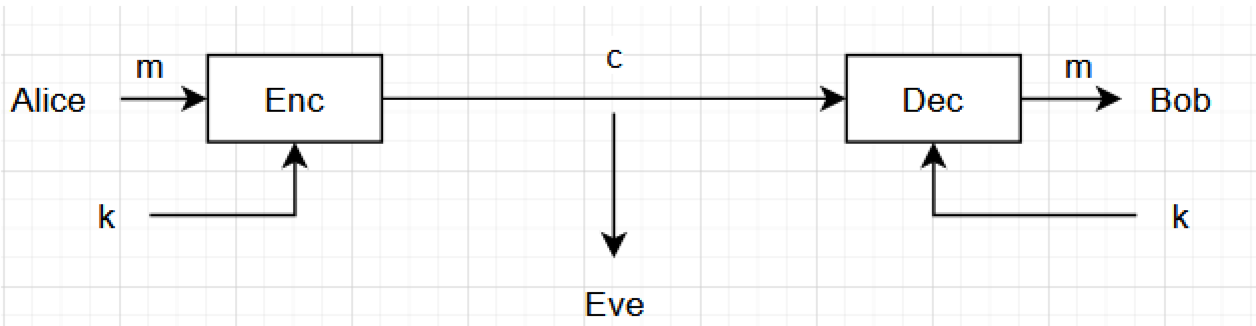
Basically we want our message to be both **confidential**, so no-one except the intended target sees it and we it to be unmodified, so that its **integrity** has not been compromised.

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 : $\mathscr{M} \times \mathscr{K} \rightarrow \mathscr{C}$
- Dec: $\mathscr{C} \times \mathscr{K} \rightarrow \mathscr{C}$
- 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:

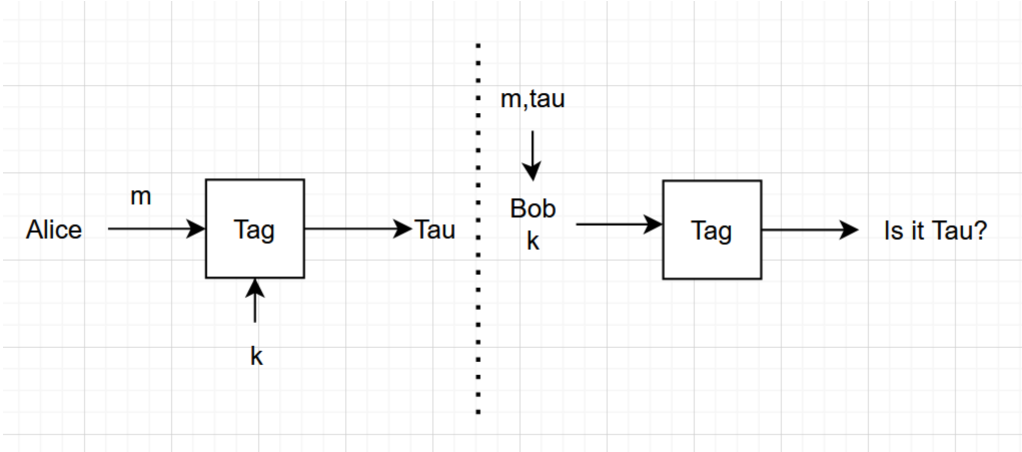
$$Pr[C = c] = \sum_{m'} Pr[C = c \wedge M = m'] = \sum_{m'} Pr[C = c|M = m'] * Pr[M = m'] =$$

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

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

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

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* $\mathscr{H} = \{h_k : \mathscr{M} \rightarrow \mathscr{T}\}_{k \in \mathscr{K}}$ *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 $H_\infty(X) = 0$

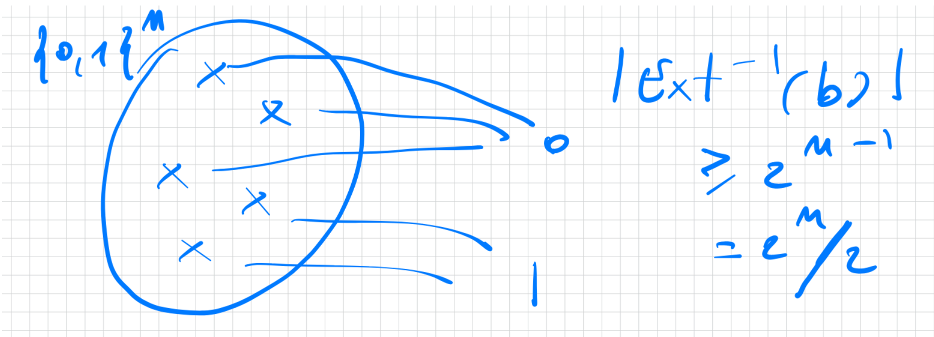
Here's the next best thing:
Design Ext that extracts UNIFORM $Y = Ext(X)$ for every X s.t. $H_\infty(X) \geq k$
But this is also impossible, even if

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

$$k = n - 1$$

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

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

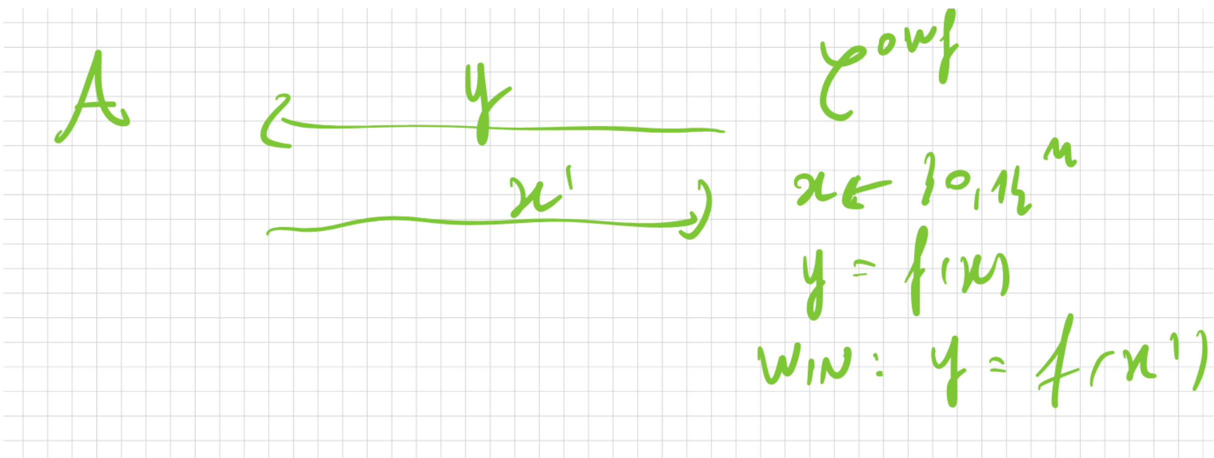
This is our first step towards efficient symmetric crypto. Moreover, pseudorandomness is used in modern computers to simulate real randomness. We will see that OWF are enough for pseudorandomness.

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

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

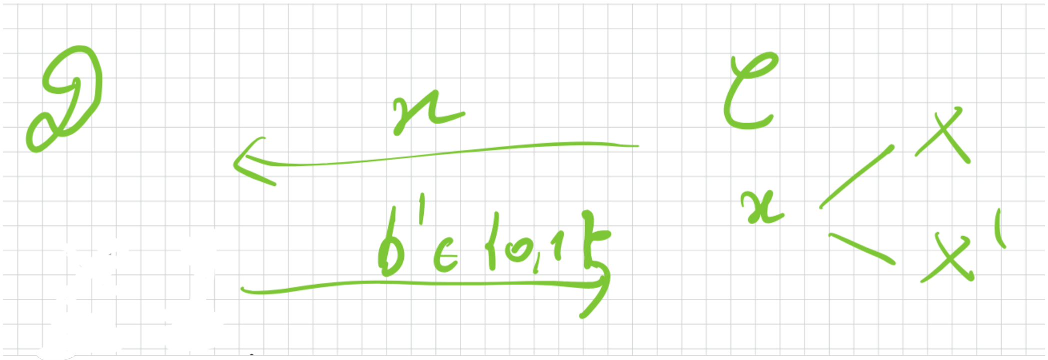
Informally, it goes to zero faster than any inverse of a polynomial function.

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):

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



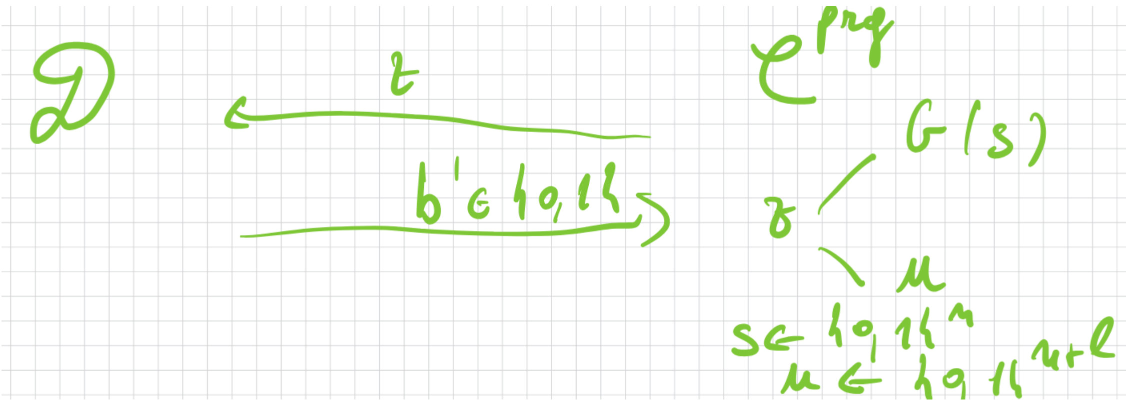
Definition 11. *X (X_n), Y (Y_n) are computationally indistinguishable ($X \approx_c Y$) if $\forall PPT \mathscr{D}$:*

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

With this we can define pseudorandomness:

Definition 12 (Pseudorandom Generator (PRG)). *A function $G : \{0,1\}^n \rightarrow \{0,1\}^{n+l}$ 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:

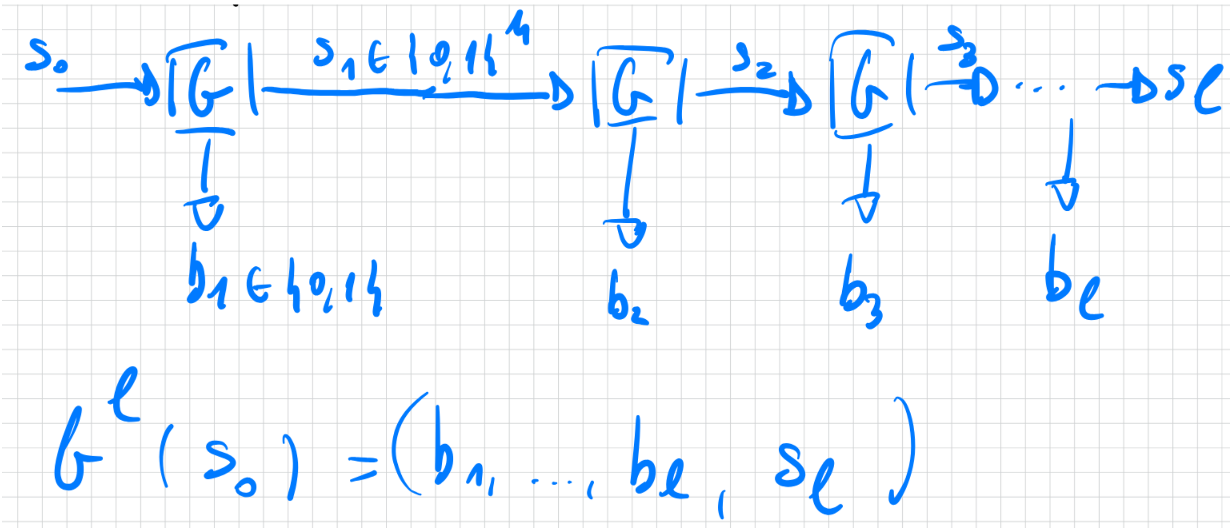
- Use a randomness extractor to get a uniform seed $s \in \{0,,1\}^n$.
- Define a simple PRG $G : \{0,1\}^n \rightarrow \{0,1\}^{n+1}$ with minimal stretch $l = 1$.
- Use G to stretch any $l(n) = poly(n)$.

Theory vs Practice:

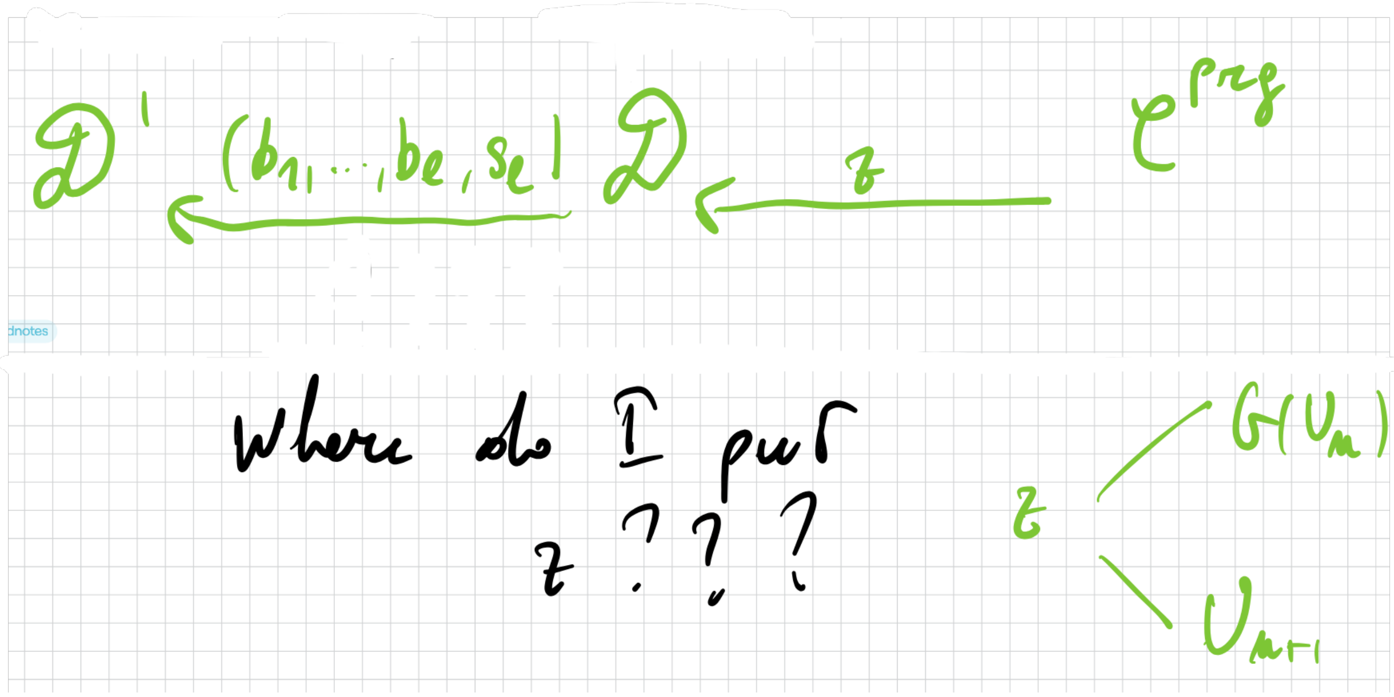
- Randomness extraction is what we already studied. But in practice it is done using Hash Functions.
- Theoretical G can be obtained from any OWF. Practical G is Heuristic

- 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

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

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

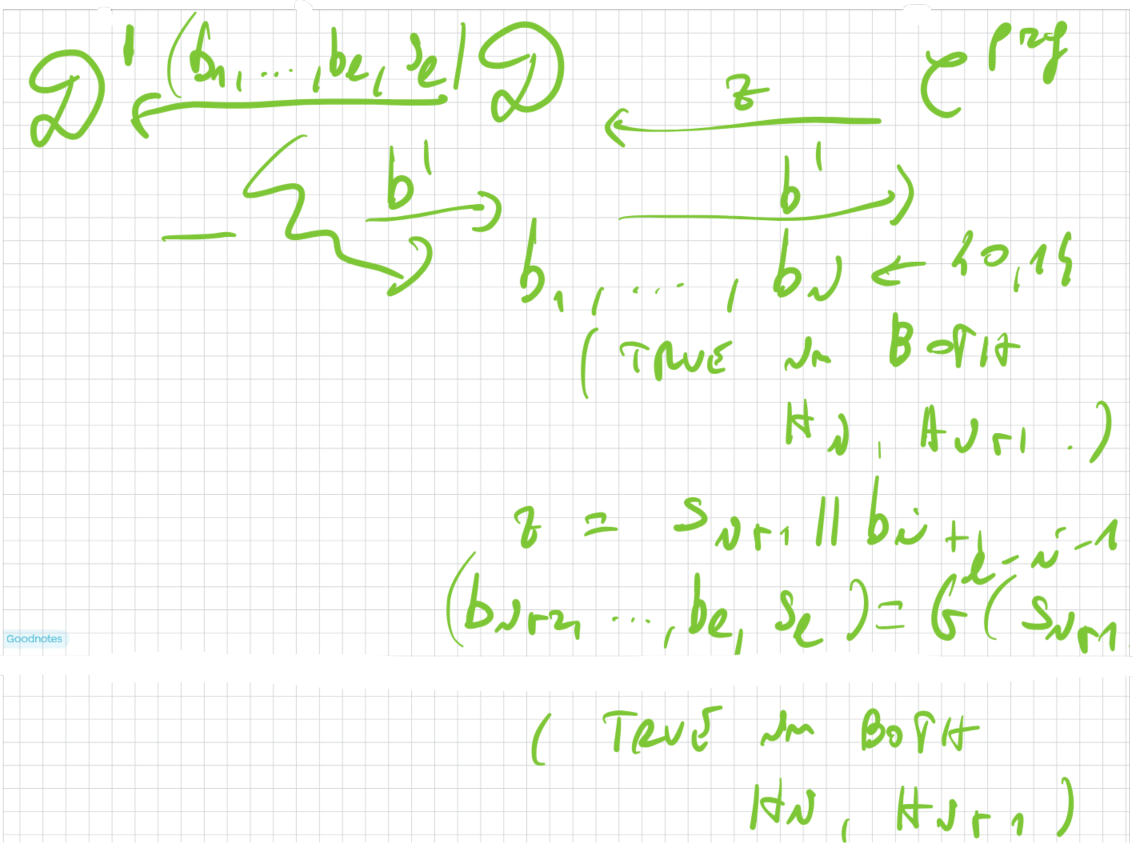
$$H_i(M) \equiv \begin{cases} b_1, \dots, b_i \leftarrow \{0,1\} \\ s_i \leftarrow \{0,1\}^n \\ (b_{i+1}, \dots, b_\ell, s_\ell) = G(s_i) \end{cases}$$

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

$\square$

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

Proof. By reduction (as before):



By the above observations:

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

$\square$

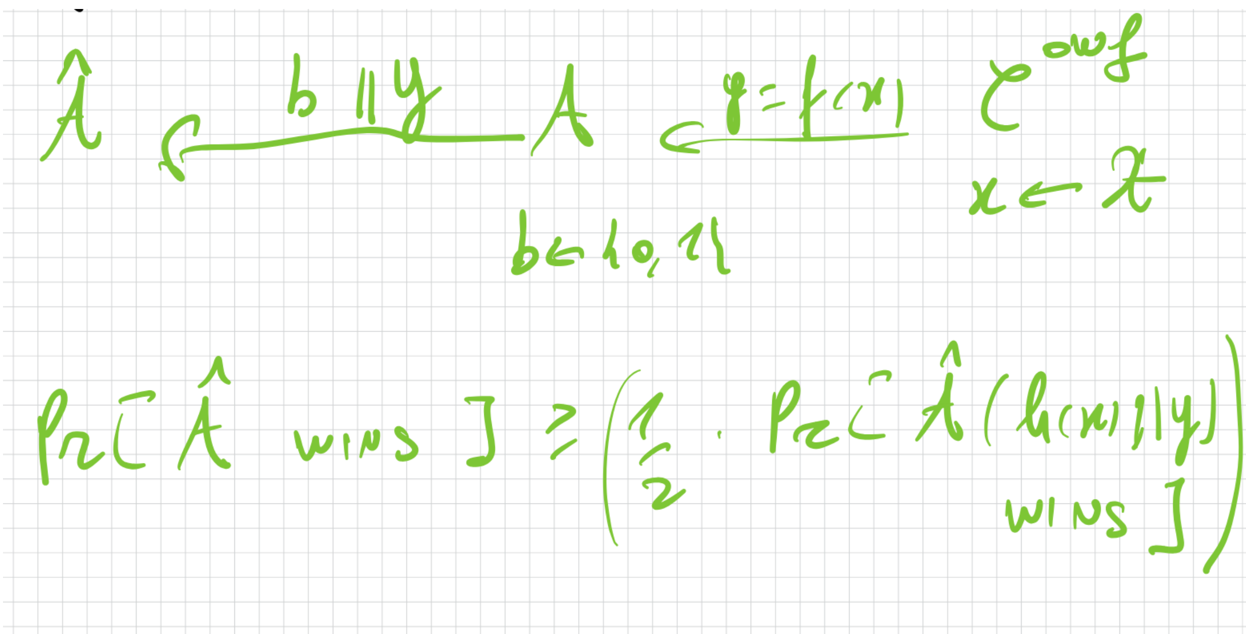
The next question is: How do we build G : $\{0,1\}^n \rightarrow \{0,1\}^{n+1}$?

- Practical: Heuristic construction
- Theoretical: From any OWF

So we need to build a PRG from a OWF. To do so we need to introduce the concept of **Hardcore bits**. They are bits of info about x that are hard to compute given $y = f(x)$. It's a predicate $h(x)$ s.t. $h(x) \in \{0,1\}$ is **hard** to compute given $f(x)$ (w.p. better than $\frac{1}{2}$).
First: Can there be a single h such that h is hardcore for all OWF?
No, because suppose we fix any h; Take f for a OWF, consider:

$$\hat{f}(x) = h(x)||f(x)$$

. h is not hard-core for $\hat{f}$, but is $\hat{f}$ a OWF?



$$\Pr[\hat{\mathscr{A}} \text{ wins }] \geq (\frac{1}{2} * \Pr[\hat{\mathscr{A}}(h(x)||y) \text{ wins }])$$

$$\begin{aligned}
\Pr[\hat{\mathscr{A}} \text{ wins }] &= \Pr[\hat{\mathscr{A}}(b,y) \text{ wins } \wedge b = h(x)] + \Pr[\hat{\mathscr{A}}(b,y) \text{ wins } \wedge b \neq h(x)] \\
&\geq \frac{1}{2} * \Pr[\hat{\mathscr{A}}(h(x),y) \text{ wins }] \\
&\geq \frac{1}{2} * \frac{1}{\text{poly}}
\end{aligned}$$

Solution: swap the quantifiers.

Definition 13. *Let $f : \{0,1\}^n \to \{0,1\}^n$ be a OWF. Then h is hard-core for f if either of the following is true:*

- $\forall$ *PPT* $\mathscr{P}$: $\Pr[\mathscr{P}(y) = h(x) : {}^{x \leftarrow \{0,1\}^n}_{y = f(x)}] \leq \frac{1}{2}$
- $(f(x), h(x)) \approx_c (f(x), b)$ *for* $b \leftarrow \{0,1\}$ *and* $x \leftarrow \{0,1\}^n$

Proof. We show that the following are equivalent for a predicate h and a function f:

1. For all PPT algorithms $\mathscr{P}$,

$$\Pr\left[\mathscr{P}(y) = h(x) : x \leftarrow \{0,1\}^n,\, y = f(x)\right] \leq \frac{1}{2} + \text{negl}(n)$$

2. $(f(x), h(x)) \approx_c (f(x), b)$, where $b \leftarrow \{0,1\}$ is uniform and $x \leftarrow \{0,1\}^n$.

(2) $\implies$ (1):
Suppose $(f(x), h(x)) \approx_c (f(x), b)$. Assume, for contradiction, that there exists a PPT $\mathscr{P}$ such that

$$\Pr[\mathscr{P}(f(x)) = h(x)] \geq \frac{1}{2} + \epsilon$$

for some non-negligible ϵ. Construct a distinguisher $\mathscr{D}$ that, given (y, b'), outputs 1 if $\mathscr{P}(y) = b'$, else 0. Then:

$$|\Pr[\mathscr{D}(f(x), h(x)) = 1] - \Pr[\mathscr{D}(f(x), b) = 1]| = |\Pr[\mathscr{P}(f(x)) = h(x)] - \Pr[\mathscr{P}(f(x)) = b]|$$

But $\Pr[\mathscr{P}(f(x)) = b] = \frac{1}{2}$ since b is uniform and independent. Thus, the advantage is at least ϵ, contradicting computational indistinguishability.

□

It also true that $1 \implies 2$.

Theorem 7. *If one-way permutations exist (OWP), then there exist $f\colon \{0,1\}^n \to \{0,1\}^n$, then $\exists G\colon \{0,1\}^n \to \{0,1\}^{n+1}$ PRG.*

Proof.

$$G(s) = f(s)||h(s) \text{ where h is hard-core for f.}$$

$$G(U_n) \equiv f(U_n)||h(U_n) \approx_c f(U_n)||U_1 \equiv U_{n+1}$$

□

Theorem 8. *If OWF exist, then PRGs with $l(n) = 1$ exist.*

All that is left is to build h for every given f.

Theorem 9 (Goldreich-Levin). *Let $f\colon \{0,1\}^n \to \{0,1\}^n$ be a OWF $g\colon \{0,1\}^{2n} \to \{0,1\}^{2n}$ with hard-core predicate:*

$$h(x, r) = \oplus_{i=1}^{n} x_i * r_i = \quad < \vec{x, r} > \quad \mod 2$$

Proof. Proof ideas: If $\exists$ PPT $\mathscr{P}$ for $h(x, r)$, then $\exists$ PPT $\mathscr{A}$ breaking g, in particular $\mathscr{A}$ can find x.
Simple cases:

Assume $\mathscr{P}$ is super good: $\forall x, r \Pr[\mathscr{P}(y) = h(x, r)] = 1$
Then $\mathscr{A}$ will just run $\mathscr{P}$ on

$$y_1 = (f(x),\, \vec{e}_1)$$

$$y_2 = (f(x),\, \vec{e}_2)$$

$$\vec{e}_i = (0 \cdots 0\, 1\, 0 \cdots) \quad \text{(1 in position } i\text{, 0 elsewhere)}$$

Second idea: Assume $\mathscr{P}$ is very good: $\forall x \in \{0,1\}^n$:

$$\Pr_{r \leftarrow \{0,1\}^n}[\mathscr{P}(f(x), r) = h(x, r)] \geq \frac{3}{4} + \frac{1}{\text{poly}}$$

Run $\mathscr{P}$ on r random and $r \oplus e_i$.

$$x_i =< x, r \oplus e_i > \oplus < x, r >=< x, e_i >$$

Still you can amplify by taking majority of many queries.

□

2.2 Symmetric Key Encryption

Recap from: G is a Pseudorandom Generator (PRG) with stretch $l(\lambda) = \text{poly}(\lambda)$.

Today, we will apply what we have learned to Symmetric Key Encryption (SKE).

Let us apply what we have learned to SKE, simple idea:

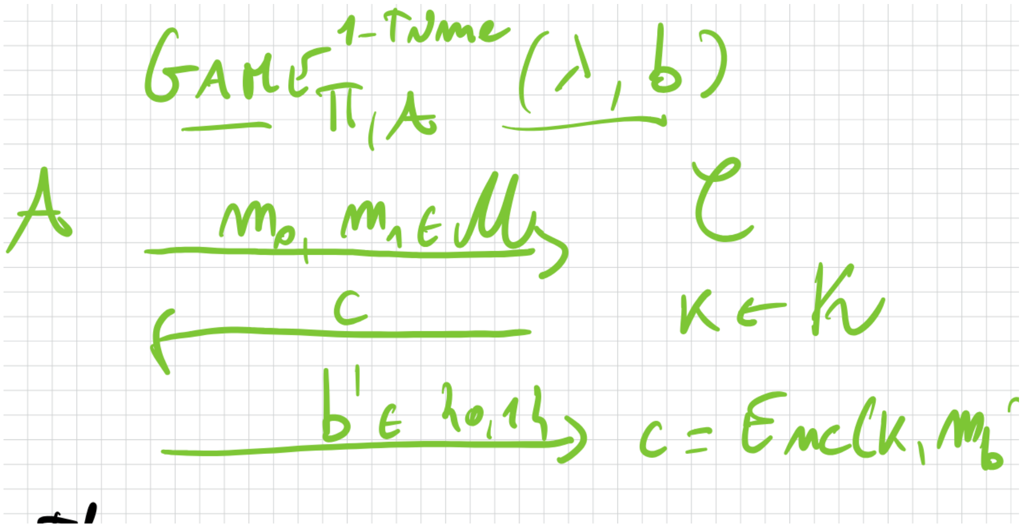
- **Encryption**: $\mathscr{E}nc(k, m) = G(k) \oplus m = c$
- **Decryption**: $\mathscr{D}ec(k, c) = G(k) \oplus c = m$

$k \in \{0,1\}^\lambda$, but $m \in \{0,1\}^{\lambda+l}$ for any l = poly.

What does it mean for the above scheme to be computationally secure? Let's start with a warm-up definition.

Definition 14 (One-Time Computational Security for SKE)**.** *Let* $\Pi = (\mathscr{E}nc, \mathscr{D}ec)$ *be a Symmetric Key Encryption scheme. We say* Π *is* ***one-time computationally secure*** *if:*

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



Recall this means:

$$|\Pr[b' = 1 \colon GAME^{1\text{-}time}_{\Pi,\mathscr{A}}(\lambda, 0)] - \Pr[b' = 1 \colon GAME^{1\text{-}time}_{\Pi,\mathscr{A}}(\lambda, 1)]| \leq negl(\lambda)$$

Why is this definition good? Because it captures natural properties every SKE has:

This definition captures several natural properties that a secure SKE should have:

- It should be hard to compute the secret key.
- It should be hard to compute the entire message.
- It should be hard to compute even the first bit of the message.

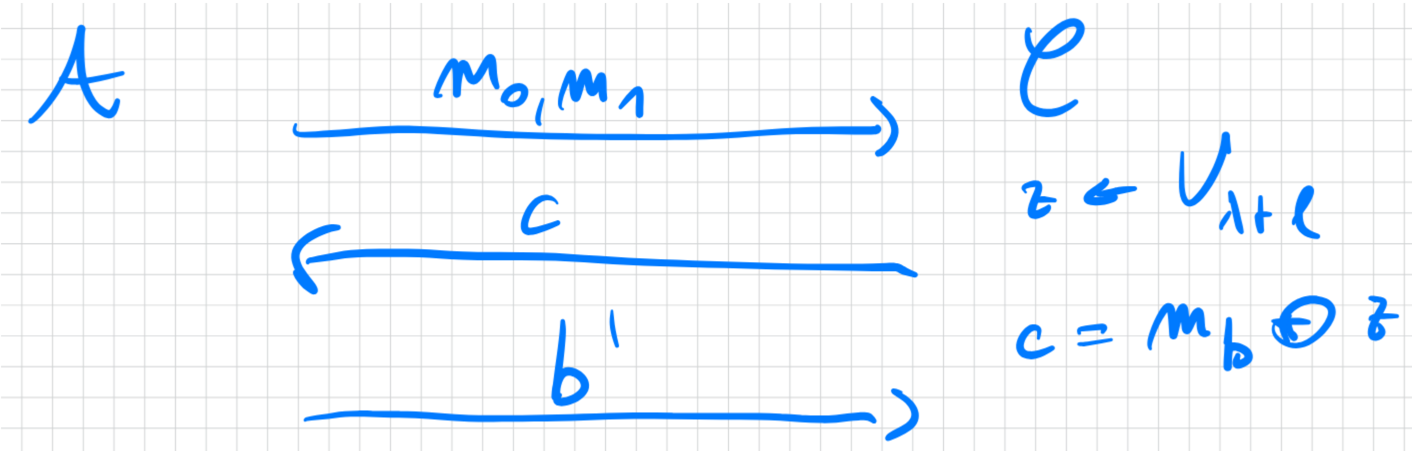
On the negative side, this notion is strictly for a **one-time** scenario (i.e., one key, one message). If the same key is used to encrypt two different messages:

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

Then an adversary can compute $c_1 \oplus c_2 = (G(k) \oplus m_1) \oplus (G(k) \oplus m_2) = m_1 \oplus m_2$. If the adversary knows m_1, they can easily recover m_2.

Theorem 10. *If* G *is a PRG, then the scheme* Π *defined by* $\mathscr{E}nc(k, m) = G(k) \oplus m$ *is one-time computationally secure.*

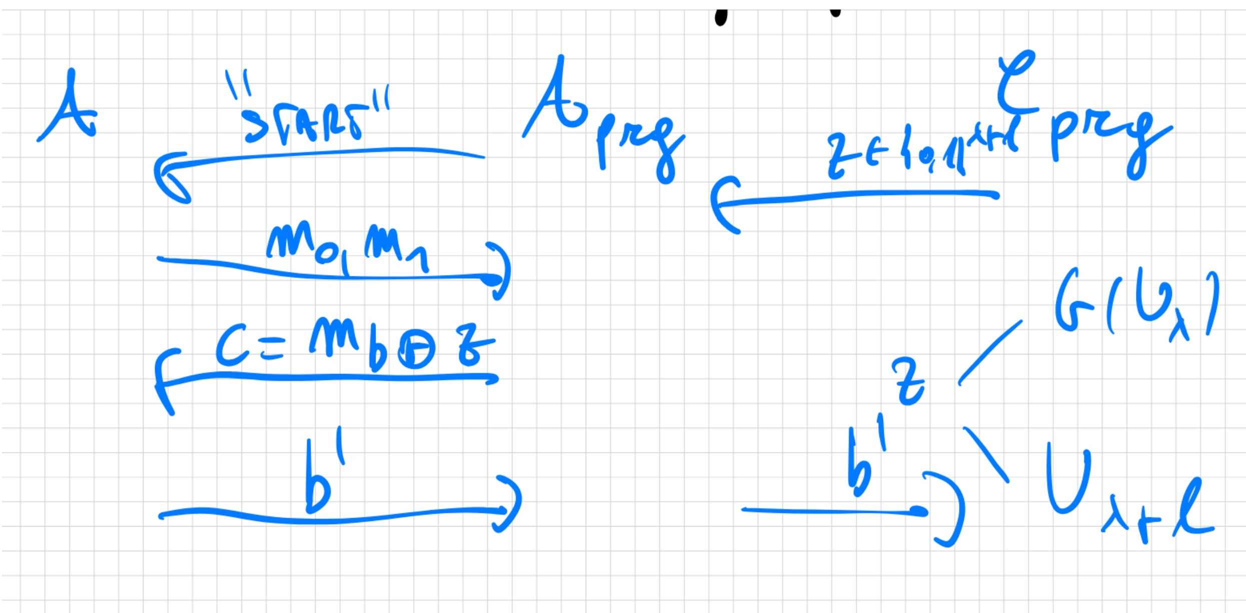
Proof. Starting with the initial experiment $\mathrm{GAME}(\lambda, b) \equiv \mathrm{GAME}^{\text{1-time}}_{\Pi,\mathscr{A}}(\lambda, b)$, we will introduce a hybrid experiment $\mathrm{HYB}(\lambda, b)$ and show that:



Easy to see that $\mathrm{HYB}(\lambda, 0) \equiv \mathrm{HYB}(\lambda, 1)$ (perfect indistinguishability) because the distribution of c is uniform and independant of b.
On the other hand: $\mathrm{GAME}(\lambda, b) \approx_c \mathrm{HYB}(\lambda, b) \forall b \in \{0, 1\}$ (computational indistinguishability). By reduction: assume $\exists PPT \mathscr{A}$ such that:

$$| \Pr[\mathrm{GAME}(\lambda, b = 1)] - \Pr[\mathrm{HYB}(\lambda, b) = 1]| \geq \frac{1}{p(\lambda)}$$

Then build PPT $\mathscr{A}_{prg}$ against G:



By inspection:

$$\Pr\left[b' = 1 : z \leftarrow G(U_\lambda)\right] = \Pr\left[b' = 1 : \mathrm{GAME}(\lambda, b)\right]$$
$$\Pr\left[b' = 1 : z \leftarrow U_{\lambda+l}\right] = \Pr\left[b' = 1 : \mathrm{HYB}(\lambda, b)\right]$$
$$\implies \left|\Pr\left[b' = 1 : z \leftarrow G(U_\lambda)\right] - \Pr\left[b' = 1 : z \leftarrow U_{\lambda+l}\right]\right| \geq \frac{1}{p(\lambda)}$$

$$\begin{aligned} \implies \quad & \mathrm{GAME}(\lambda, 0) \approx_c \mathrm{HYB}(\lambda, 0) \\ \equiv \quad & \mathrm{HYB}(\lambda, 1) \\ \approx_c \quad & \mathrm{GAME}(\lambda, 1) \end{aligned}$$

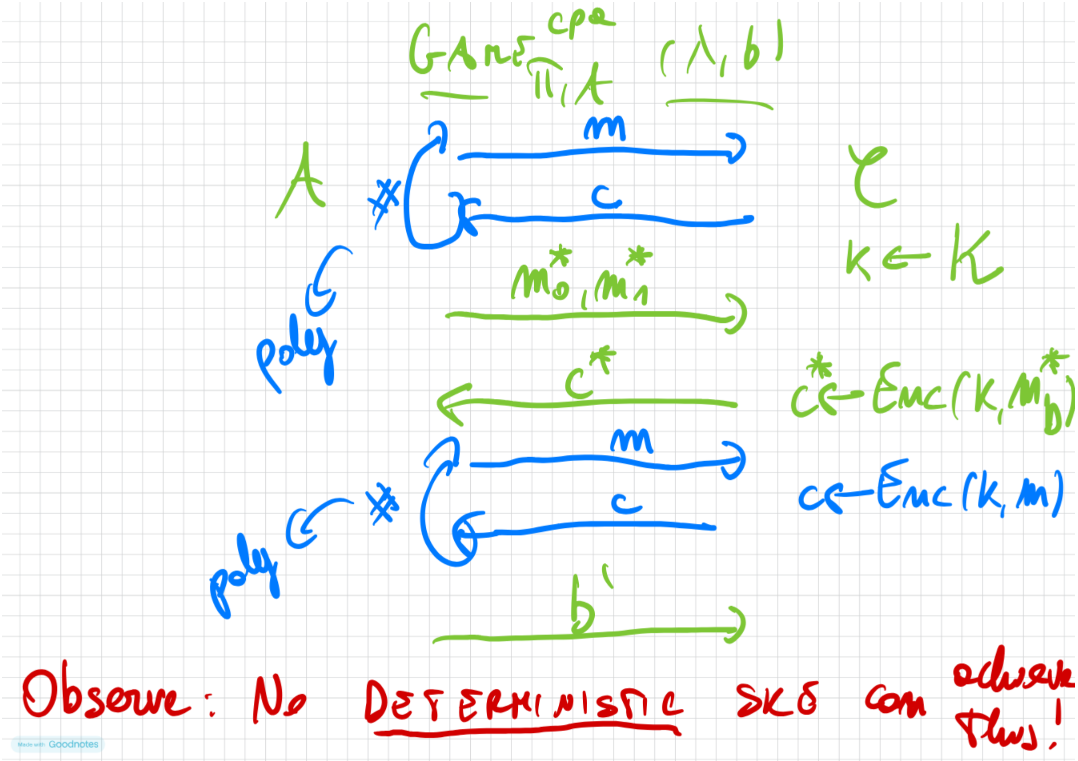
$$\implies \text{GAME}(\lambda, 0) \approx_c \text{GAME}(\lambda, 1)$$

□

Our Next goal: Chosen-Plaintext Attack (CPA) Security.

Definition 15. *Let $\Pi = (\mathscr{E}nc, \mathscr{D}ec)$ be an SKE scheme. We say Π is **CPA-secure** (secure against chosen-plaintext attacks) if for any PPT adversary $\mathscr{A}$:*

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



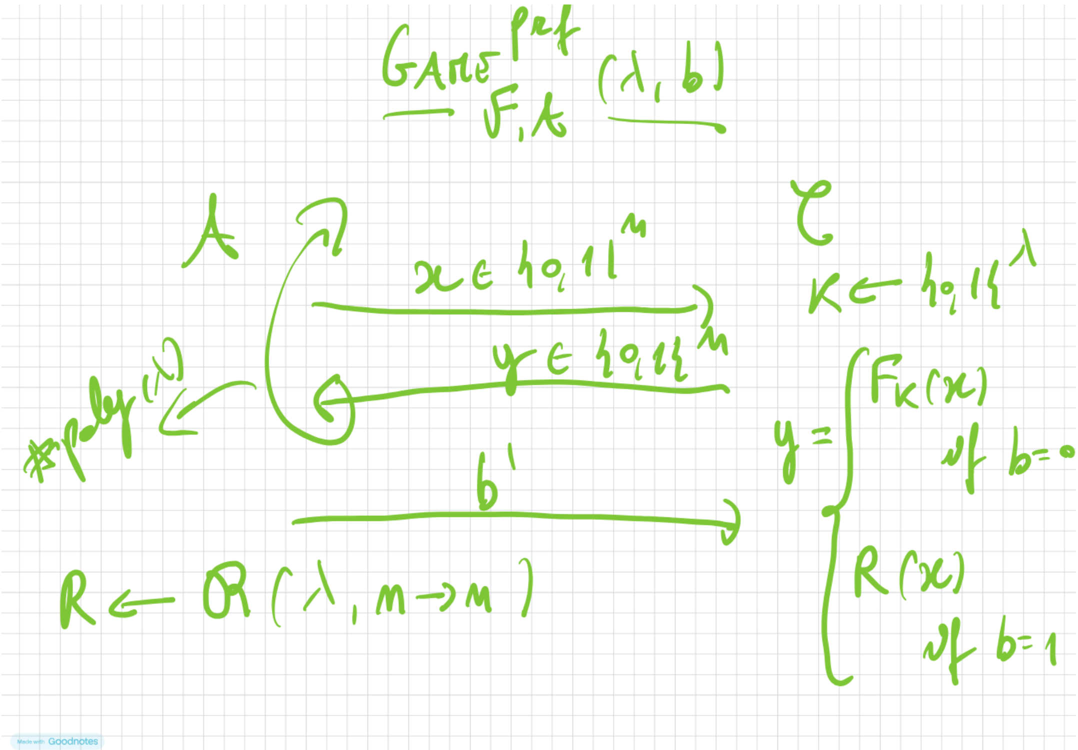
Observation: No deterministic SKE can be CPA-secure. An adversary could query the oracle on m_0^* to get c_0, then submit (m_0^*, m_1^*) as the challenge. If the challenge ciphertext c^* equals c_0, it knows $b = 0$. Therefore, CPA-secure encryption must be randomized or stateful.

The previous one-time scheme is not CPA-secure because it is deterministic. We need a new tool.

Definition 16 (Pseudorandom Function (PRF))**.**

A function family $\mathscr{F} = \{F_k : \{0,1\}^n \to \{0,1\}^n\}_{k \in \{0,1\}^\lambda}$ is a PRF if:

$$\text{GAME}^{\text{prf}}_{\mathscr{F},\mathscr{A}}(\lambda, 0) \approx_c \text{GAME}^{\text{prf}}_{\mathscr{F},\mathscr{A}}(\lambda, 1)$$



Note: R is not efficiently computable as it takes exponential space to store it. $F(k, x)$ instead is efficiently computable for all k,x.
Plan:

1. Build a PRF.
2. Use it to get CPA secure SKE and more!

How to build a PRF?

1. Practice: many examples like DES, AES (more accurately, PRP, pseudorandom permutations, which are invertible PRFs).
2. Theory: The existence of OWF implies the existence of PRG, which in turn implies the existence of PRF.

$$OWF \implies PRG \implies PRF \implies PRP$$

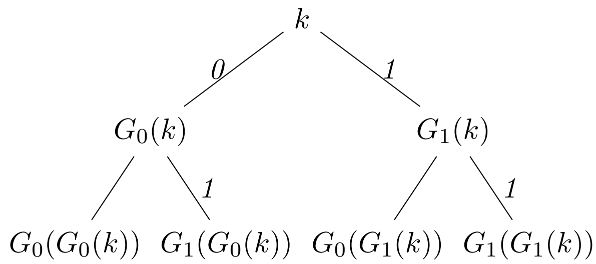
We cover our construction of PRFs.

Definition 17 (The Goldreich-Goldwasser-Micali (GGM) Construction)**.** *We will show one construction that proves PRGs imply PRFs:*
The GGM tree, basically, its a proof that PRG $\implies$ *PRF. Let* $G : \{0,1\}^n \to \{0,1\}^{2n}$ *be a PRG. We can split its output into two halves:* $G(s) = (G_0(s), G_1(s))$*, where* $|G_0(s)| = |G_1(s)| = n$*.*

In other words:

$$F_k(x_1x_2...x_n) = G_{x_n}(G_{x_{n-1}}(...G_{x_1}(k)...))$$

Think of G *as* $F(k, x)$ *for* $x \in \{0,1\}$*.*



In general:

$$F_k(x_1x_2...x_n) = G_{x_n}(G_{x_{n-1}}(...G_{x_1}(k)...))$$

Theorem 11. *If G is a secure PRG, then the GGM construction F is a secure PRF. $\mathscr{F} = \{f_k\}$ is a PRF.*

The proof relies on a hybrid argument and the following lemmas.

- **Lemma 1**: If $G : \{0,1\}^n \to \{0,1\}^{2n}$ is a PRG, then for any polynomial $t(\lambda)$, the following two ensembles are computationally indistinguishable:

$$\{(G(k_1), ..., G(k_t))\} \approx_c \{(U_{2n}, ..., U_{2n})\}$$

$$k_1, ..., k_t \leftarrow U_n$$

Next, given F'_k: $\{0,1\}^{n-1} \to \{0,1\}^n$ a PRF, then define:

$$F_k(x, y) = G_x(F'_k(y)) \text{ with } x \in \{0,1\}, y \in \{0,1\}^{n-1}$$

- **Lemma 2**: If F'_k is a secure PRF, then F_k is also a secure PRF.

Recall the GGM (Goldreich-Goldwasser-Micali) construction:

$$G : \{0,1\}^\lambda \to \{0,1\}^{2\lambda}$$

$$G(k) = (G_0(k), G_1(k))$$

We build $\mathscr{F} = \{F_k : \{0,1\}^{n(\lambda)} \to \{0,1\}^\lambda\}$ with $k \in \{0,1\}^\lambda$ such that

$$F_k(x) = G_{x_n}(G_{x_{n-1}}(...G_{x_1}(k)...))$$

where $x = x_1x_2...x_n \in \{0,1\}^n$.

Proof of Security (by Induction)

For the proof, let $n(\lambda) = \text{poly}(\lambda)$. We use induction on n.

Let $F_k^{'} : \{0,1\}^{n-1} \to \{0,1\}^\lambda$ be the GGM construction for inputs of length $n-1$. We can write F_k for n-bit inputs as:

$$F_k(x, y) = G_x(F_k^{'}(y))$$

where $x \in \{0,1\}$ and $y \in \{0,1\}^{n-1}$.

Lemma 5. *If $\{F_k^{'}\}$ (GGM on $n-1$ inputs) is a PRF, then $\{F_k\}$ (GGM on n inputs) is also a PRF family.*

We can use this lemma to prove the security of GGM by induction.

Base Case $(n = 1)$

For $n = 1$, the GGM construction is:

$$F_k(x) = G_x(k), \quad x \in \{0, 1\}$$

This is:

$$F_k(x) = \begin{cases} G_0(k) & \text{if } x = 0 \\ G_1(k) & \text{if } x = 1 \end{cases}$$

This is a PRF because G is a PRG, so $(G_0(k), G_1(k)) \approx_c U_{2\lambda}$. An adversary querying $F_k(0)$ and $F_k(1)$ just gets the output of the PRG, which is indistinguishable from two random λ-bit strings $R(0)$ and $R(1)$.

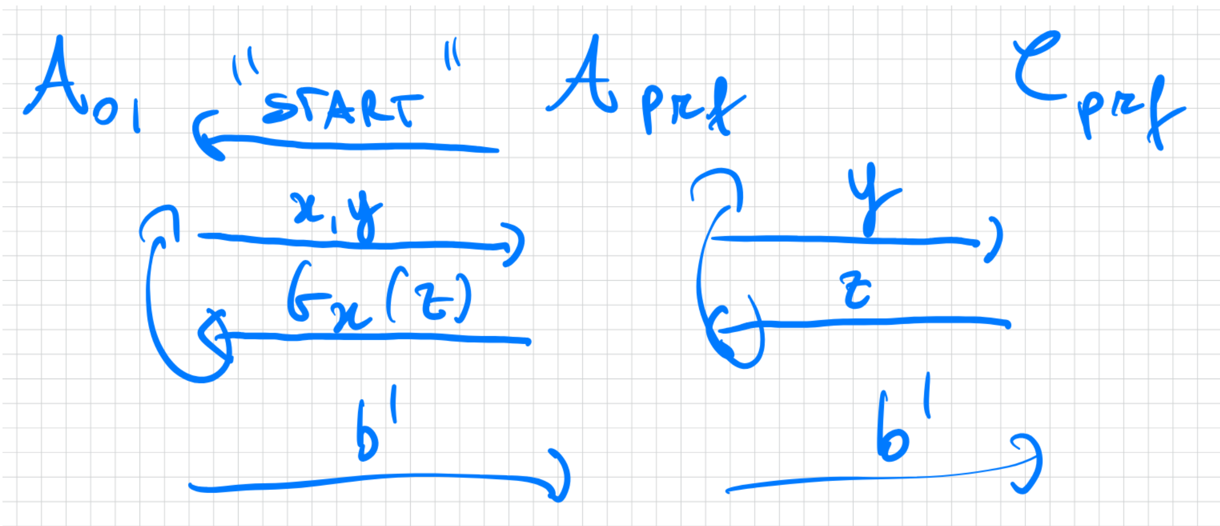
Inductive Step

Assume $\{F_k^{'}\}$ (GGM on $n-1$ inputs) is a PRF. We want to prove $\{F_k\}$ (GGM on n inputs) is a PRF. This is exactly what the Lemma states.

Proof (of Lemma). We use a hybrid argument. Let $\mathscr{A}$ be a PPT adversary.

- **HYB 0**: The real world.
$$z = F_k(x, y) = G_x(F_k^{'}(y))$$
where $k \leftarrow U_\lambda$.
- **HYB 1**:
$$z = G_x(R^{'}(y))$$
where $R^{'} :\leftarrow \mathscr{R}(\lambda, n-1 \to \lambda)$ is a truly random function from.
- **HYB 2**: The ideal world.
$$z = R(x, y)$$
where $R :\leftarrow \mathscr{R}(\lambda, n \to \lambda)$ is a truly random function.

Step 1: $HYB_0 \approx_c HYB_1$ We show $HYB_0(\lambda) \approx_c HYB_1(\lambda)$ by reduction. Assume a PPT $\mathscr{A}_{01}$ distinguishes HYB_0 and HYB_1 with non-negligible probability. We build a reduction $\mathscr{A}_{prf}$ that breaks the PRF security of $\{F_k^{'}\}$.



If z $= F_k^{'}(y)$, then $G_x(z)$ is identical to what $\mathscr{A}_{prf}$ receives in HYB_0. On the other hand, if z $= R^{'}(y)$, then $G_x(z)$ is identical to what $\mathscr{A}_{prf}$ receives in HYB_1.

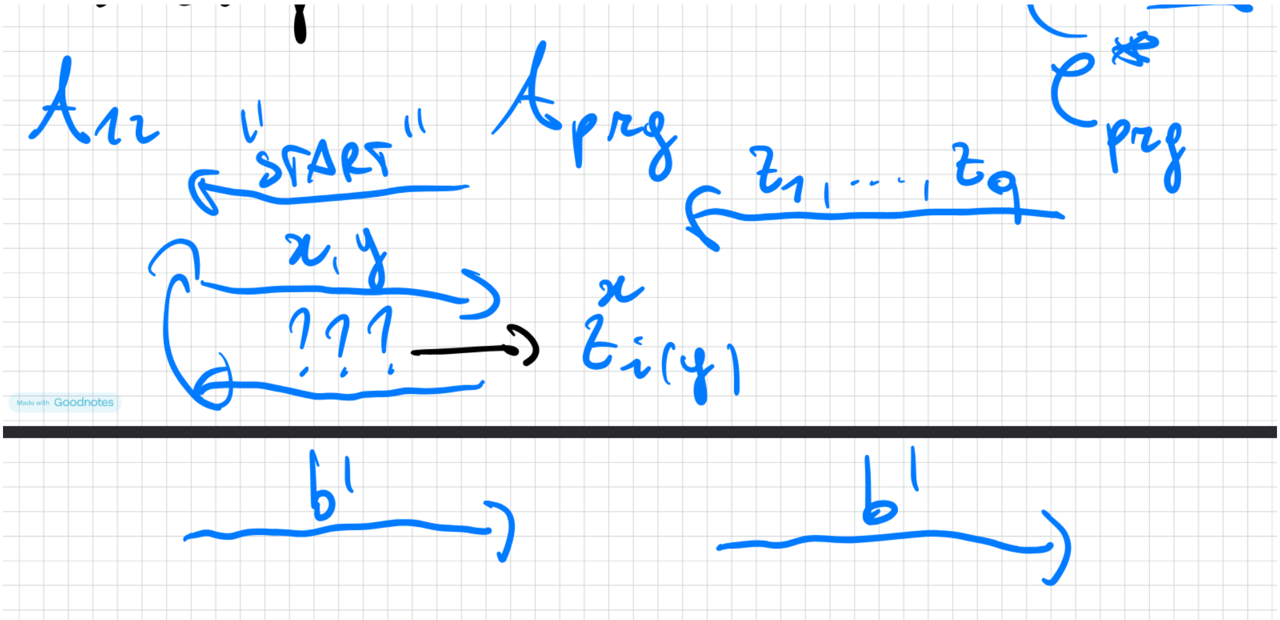
Step 2: $HYB_1 \approx_c HYB_2$ Here, we use the property that G is a PRG.

Lemma 6 (PRG Expansion). *If $G : \{0,1\}^\lambda \to \{0,1\}^{2\lambda}$ is a PRG, then for any $t = poly(\lambda)$:*

$$(G(k_1), \ldots, G(k_t)) \approx_c (U_{2\lambda}, \ldots, U_{2\lambda}) \equiv U_{2\lambda \cdot t}$$

where $k_1, \ldots, k_t \leftarrow U_\lambda$ are chosen independently.

Now, assume a PPT $\mathscr{A}_{12}$ distinguishes HYB_1 and HYB_2. We build a PPT $\mathscr{A}_{prg}$:breaking the above claim.



Let $t(\lambda) = q(\lambda)$ - the number of queries made by $\mathscr{A}_{12}$. Each of $z_i \in \{0,1\}^{2\lambda}$ and we can think of it as:

$$z_i = (z_i^0, z_i^1)$$

With each corresponding to λ bits.

In the above reduction, $i(y)$ is the index of the sample z_i that was used when $\mathscr{A}_{12}$ asked already for x,y. If it never asked use the next available z_i.

□

In the next few lectures, we'll see PRFs are enough to do practical symmetric crypto:

- CPA-Secure SKE (Symmetric Key Encryption) for messages of variable length (VIL).
- MACs (Message Authentication Codes) for messages of VIL.
- Non-malleable SKE (a.k.a. CCA-Secure SKE), which is equivalent to combining message privacy and message authentication.

It will be important that our PRF F is a **PRP (Pseudorandom Permutation)**, namely it is an efficient, length-preserving permutation for a given key. In practice, we call it a **BLOCKCIPHER**. We will show that $PRPs \approx PRFs$. This will also explain the real-world design of some blockciphers (e.g., DES, AES).

CPA-Secure SKE for Variable Length Messages

Let's start with encryption (CPA-security). Recall the CPA indistinguishability game (GAME^{CPA}): The adversary $\mathscr{A}$ submits two messages m_0, m_1 of the same length ($|m_0| = |m_1|$) to a challenger. The challenger picks $k \leftarrow \mathscr{K}$, computes $c \leftarrow \mathscr{E}nc(k, m_b)$ for a random bit b, and sends c to $\mathscr{A}$. $\mathscr{A}$ wins if it guesses b. We need this to work for messages of any polynomial length (VIL).

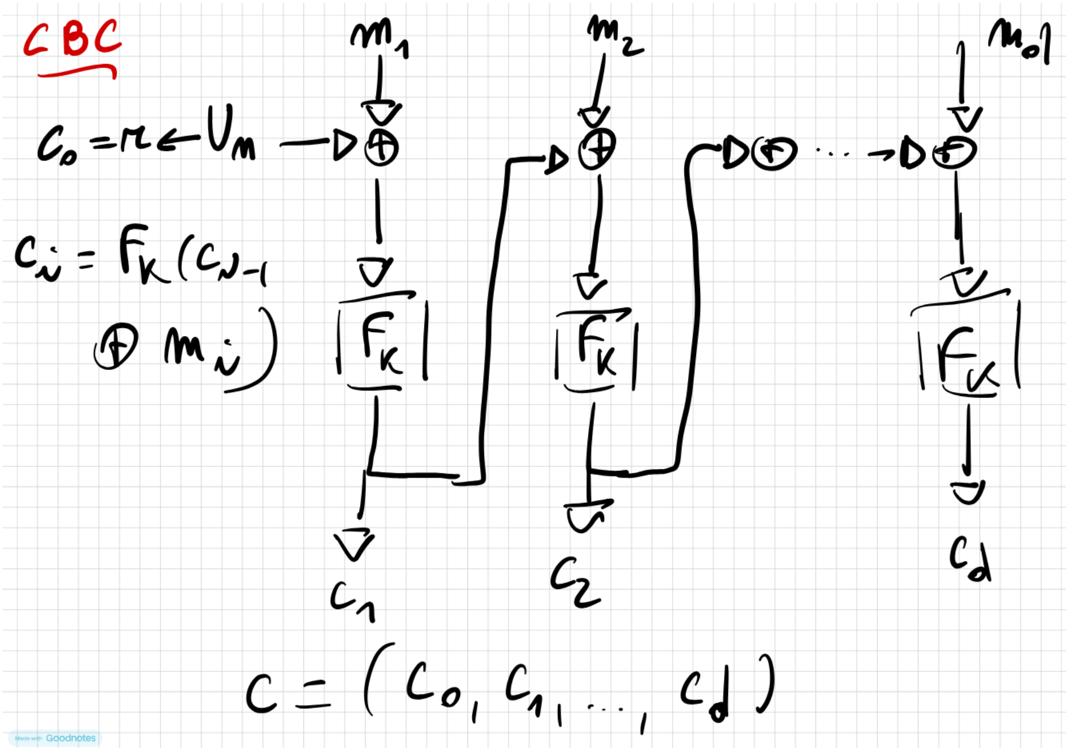
Mode of Operation

A mode of operation is a standardized way to encrypt messages $m = (m_1, \dots, m_d)$, where $m_i \in \{0,1\}^n$, using a PRF $\mathscr{F} = \{F_k : \{0,1\}^n \to \{0,1\}^n\}$.

Remark 1. *We can't just use $c = (F_k(m_1), \dots, F_k(m_d))$. This is Electronic Codebook (ECB) mode, and it's not secure (it leaks equalities between blocks). This is true even if F is a PRP.*

CBC (Cipher Block Chaining) Mode

- $c_0 = IV \in U_n$ (Initialization Vector)
- $c_i = F_k(c_{i-1} \oplus m_i)$ for $i = 1, \dots, d$
- Output: $c = (c_0, c_1, \dots, c_d)$

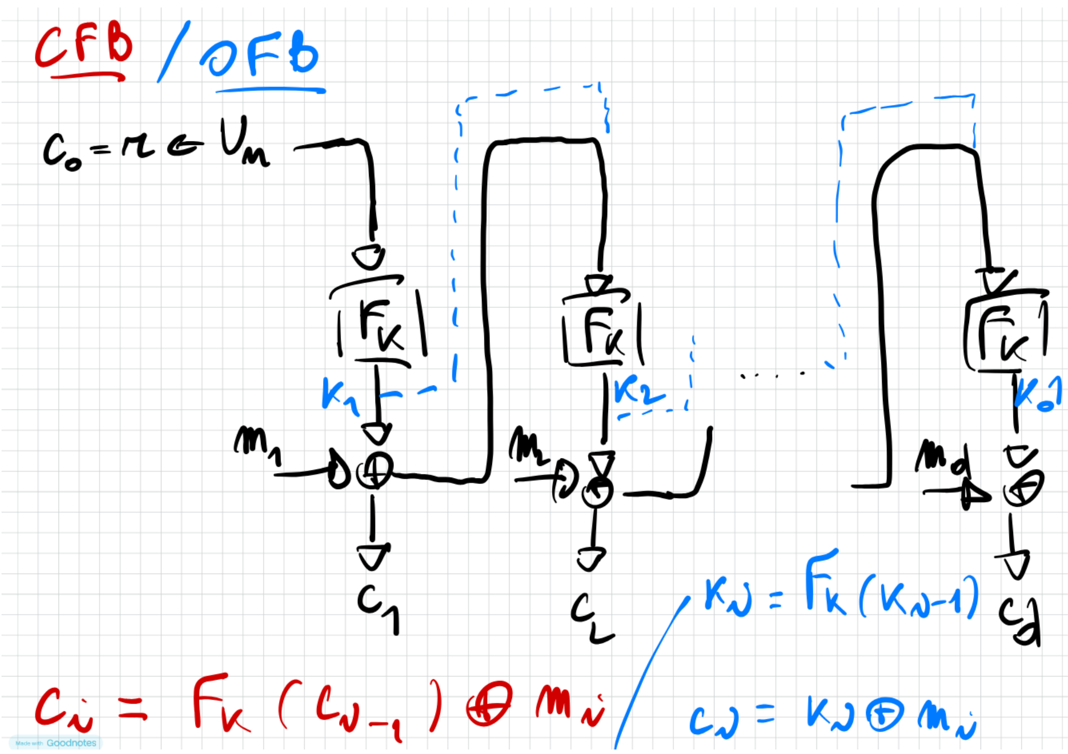


Decryption requires F_k^{-1}, so F must be a PRP. Encryption is sequential.

Theorem 12. *If F is a PRP, then CBC-Mode is CPA-secure for VIL.*

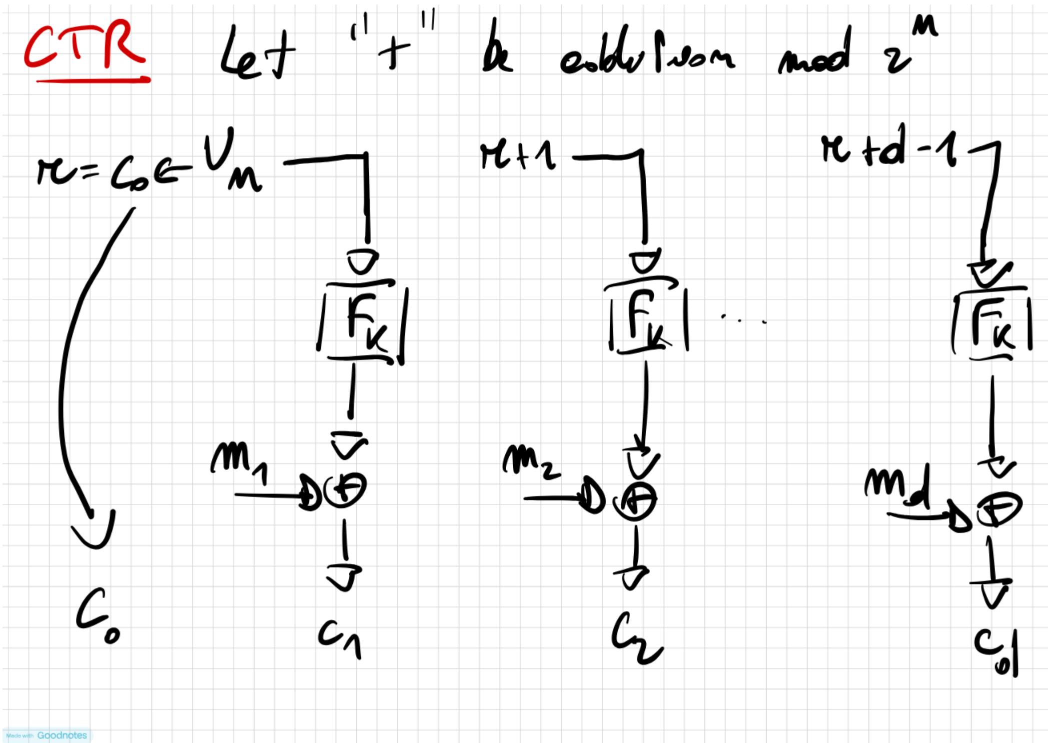
OFB (Output Feedback) Mode

- $c_0 = r \in U_n$
- $k_0 = r$
- $k_i = F_k(k_{i-1})$ for $i = 1, \dots, d$
- $c_i = k_i \oplus m_i$ for $i = 1, \dots, d$
- Output: $c = (c_0, c_1, \dots, c_d)$



CTR (Counter) Mode

- $c_0 = r \in U_n$ (where r is a counter)
- $c_i = F_k(r + i - 1) \oplus m_i$ for $i = 1, \ldots, d$
- Output: $c = (c_0, c_1, \ldots, c_d)$
- Note: "+" can be modulo 2^n arithmetic.



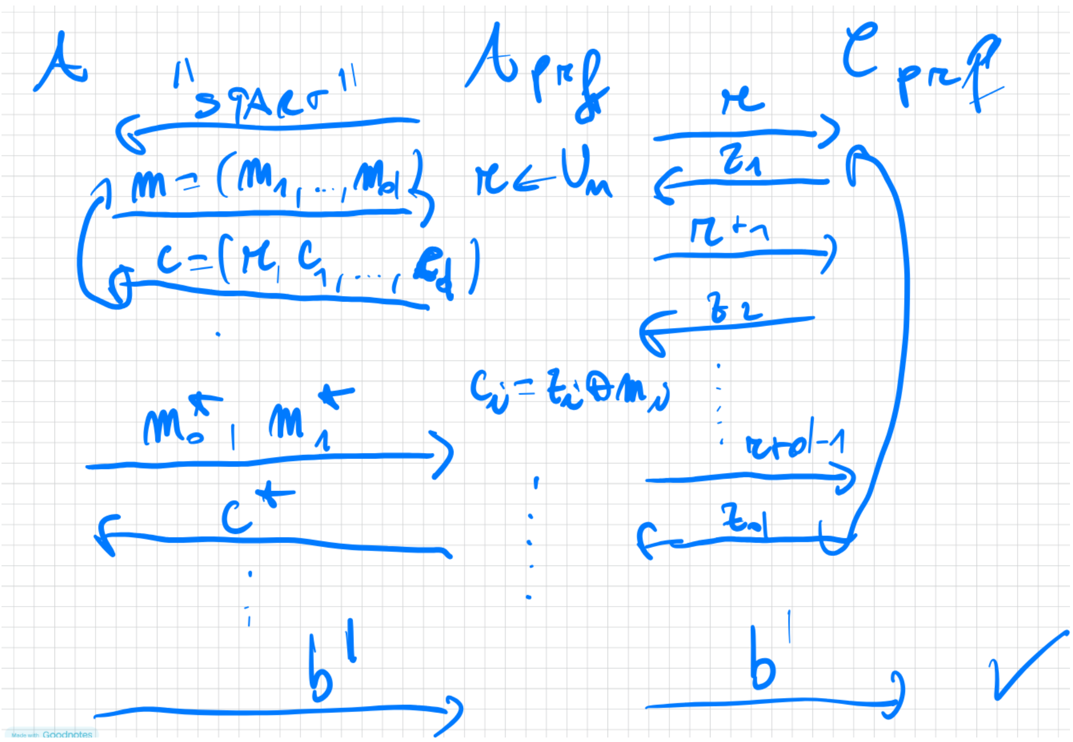
Theorem 13. *Assuming F is a PRF, CTR mode is a CPA-secure SKE for VIL.*

Proof. We use a hybrid argument. Let $G(\lambda, b) \equiv \text{GAME}_{\Pi}^{cpa}(\lambda, b)$ be the CPA game where Π is CTR mode using $\mathscr{F}$. We want to show $G(\lambda, 0) \approx_c G(\lambda, 1)$. Recall that in $G(\lambda, 0)$:
Upon input an encryption query $m = (m_1, \ldots, m_d)$, we return $c = (c_1, \ldots, c_d)$ such that $c_0 = r \in U_n$ and $c_i = F_k(r + i - 1) \oplus m_i$.
For the challenge $m_b^* = (m_{b,1}^*, \ldots, m_{b,d^*}^*)(d^* \in \mathbb{N} is the dimension)$, we return $c^* = (c_0^*, \ldots, c_{d^*}^*)$ such that $c_0^* = r^* \in U_n$ and $c_i^* = F_k(r^* + i - 1) \oplus m_{b,i}^*$.

- **Game** $G(\lambda, b)$: Real game.
  - Encryption query $m = (m_1, \ldots, m_d)$: return $c = (c_0, \ldots, c_d)$ where $c_0 = r \in U_n$ and $c_i = F_k(r + i - 1) \oplus m_i$.
  - Challenge query m_b^*: return $c^* = (c_0^*, \ldots, c_{d^*}^*)$ where $c_0^* = r^* \in U_n$ and $c_i^* = F_k(r^* + i - 1) \oplus m_{b,i}^*$.
- **Hybrid** $H_1(\lambda, b)$: Same as $G(\lambda, b)$, but replace $F_k(\cdot)$ with a truly random function $R(\cdot)$.
- **Hybrid** $H_2(\lambda)$: The challenge ciphertext c^* is uniform and independent of b. (i.e., $c_0^* = r^*$ and $c_i^* = u_i \oplus m_{b,i}^*$ where u_i are fresh uniform strings. This is equivalent to c^* being c_0^* and d^* fresh uniform strings, which is independent of b.)

Lemma 7. *$G(\lambda, b) \approx_c H_1(\lambda, b)$ for all $b \in \{0, 1\}$.*

Proof. Standard reduction. Fix b. Assume a PPT $\mathscr{A}$ distinguishes $G(\lambda, b)$ and $H_1(\lambda, b)$. We build a PPT adversary $\mathscr{A}_prf$ against the PRF F.



∎

Lemma 8. *$H_1(\lambda, b) \approx_c H_2(\lambda)$, as long as the number of encryption queries $q(\lambda) = poly(\lambda)$.*

Proof. In $H_1(\lambda, b)$, the challenge ciphertext is created using the values $R(r^*), R(r^*+1), \ldots, R(r^*+d^*-1)$. An encryption query j uses values $R(r_j), R(r_j + 1), \ldots, R(r_j + d_j - 1)$.

Let **BAD** be the event that any counter value used for the challenge overlaps with any counter value used for any encryption query.

$$\text{BAD} = \exists j, i, i' \text{ s.t. } r^* + i' = r_j + i$$

$(i' \in [0, d^* - 1]$, $i \in [0, d_j - 1])$

Conditioned on $\neg$BAD, all values $r^* + i'$ are "fresh" inputs to the random function R. This means all outputs $R(r^* + i')$ are independent and uniformly random. In this case, $c_i^* = R(r^* + i - 1) \oplus m_{b,i}^*$ is a one-time pad encryption, and the ciphertext c^* is uniform and independent of b. This is exactly $H_2(\lambda)$.

By the properties of statistical distance:

$$SD(H_1(\lambda, b); H_2(\lambda)) \leq \Pr[\text{BAD}]$$

We just need to bound Pr[BAD]. Let $q = q(\lambda)$ be the number of queries. Let BAD_j be the event that the challenge overlaps with query j. $\Pr[\text{BAD}] = \Pr[\cup_{j=1}^q \text{BAD}_j] \leq \sum_{j=1}^q \Pr[\text{BAD}_j]$ (by Union Bound).

Let's bound $\Pr[\text{BAD}_j]$. WLOG, assume all message lengths are at most q. Overlap BAD_j occurs if $\{r^*, \ldots, r^* + q - 1\}$ overlaps with $\{r_j, \ldots, r_j + q - 1\}$. r^* and r_j are chosen uniformly from $\{0, \ldots, 2^n - 1\}$. Overlap happens if $r_j \in [r^* - q + 1, r^* + q - 1]$. The size of this interval is $(r^* + q - 1) - (r^* - q + 1) + 1 = 2q - 1$. So, $\Pr[\text{BAD}_j] = \frac{2q-1}{2^n}$.

$$\Pr[\text{BAD}] \leq \sum_{j=1}^{q} \frac{2q-1}{2^n} = q \cdot \frac{2q-1}{2^n} \leq \frac{2q^2}{2^n} = \text{negl}(\lambda)$$

Since $q = \text{poly}(\lambda)$ and n (the block size) is related to λ (e.g., $n = \lambda$), 2^n is exponential in λ. □

Conclusion: $G(\lambda, 0) \approx_c H_1(\lambda, 0) \approx_c H_2(\lambda) \approx_c H_1(\lambda, 1) \approx_c G(\lambda, 1)$. The advantages $G(\lambda, 0) \approx_c H_1(\lambda, 0)$ and $H_1(\lambda, 1) \approx_c G(\lambda, 1)$ are negligible by Lemma 1. The advantages $H_1(\lambda, 0) \approx_c H_2(\lambda)$ and $H_2(\lambda) \approx_c H_1(\lambda, 1)$ are negligible by Lemma 2. Thus, $G(\lambda, 0) \approx_c G(\lambda, 1)$ by the triangle inequality, and CTR mode is CPA-secure. □

2.3 Message Authentication Codes (MACs)

We now switch to the problem of message authentication. We need a security guarantee that it is computationally hard to forge a message/tag pair (m^*, τ^*) such that $\text{Vrfy}(k, m^*) = \text{accept}$ (or $\text{Tag}(k, m^*) = \tau^*$) without knowing the secret key k.

Definition 18 (UF-CMA)**.** *We say a MAC scheme* $\Pi = (\mathit{Gen}, \mathit{Tag}, \mathit{Vrfy})$ *is* ***Unforgeable Under a Chosen-Message Attack (UF-CMA)*** *if for all PPT adversaries* $\mathscr{A}$*,* $\Pr[\mathit{GAME}_{\Pi,\mathscr{A}}^{ufcma}(\lambda) = 1] \leq \mathit{negl}(\lambda)$*.*

The game $\mathit{GAME}_{\Pi,\mathscr{A}}^{ufcma}(\lambda)$ *proceeds as follows:*

1. $k \leftarrow \mathit{Gen}(\lambda)$
2. $\mathscr{A}$ *gets oracle access to* $\mathit{Tag}(k, \cdot)$*.* $\mathscr{A}$ *makes queries* $m_1, \ldots, m_q$ *and receives* $\tau_i = \mathit{Tag}(k, m_i)$*.*
3. $\mathscr{A}$ *outputs a pair* (m^*, τ^*)*.*
4. $\mathscr{A}$ *wins if* $\mathit{Vrfy}(k, m^*) = \mathit{accept}$ *(i.e.,* $\mathit{Tag}(k, m^*) = \tau^*$*) AND* $m^* \notin \{m_1, \ldots, m_q\}$*.*

Theorem 14. *If* $\mathscr{F} = \{F_k\}$ *is a PRF, then the MAC scheme* $\mathit{Tag}(k, m) = F_k(m)$ *is UF-CMA for fixed-length messages.*