Cryptography Notes

Raffaele Castagna

Academic Year 2025-2026

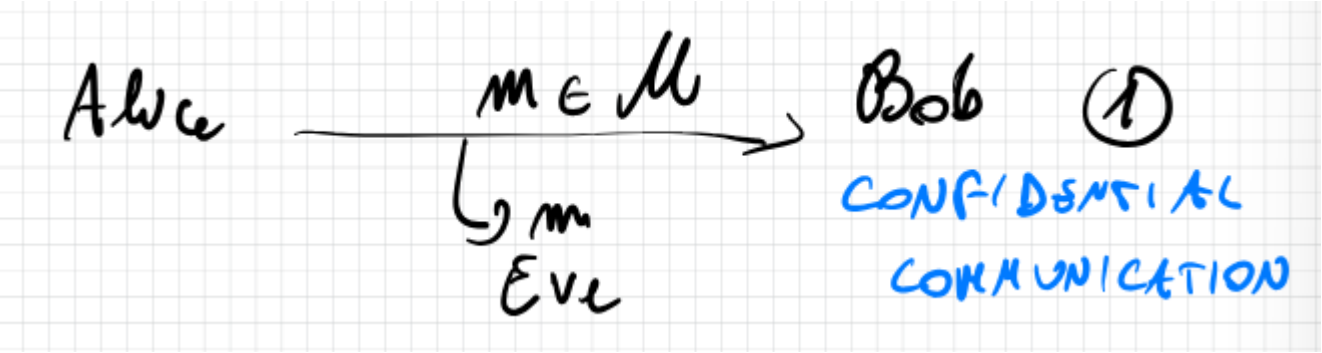
Contents

1 Intro to Cryptography

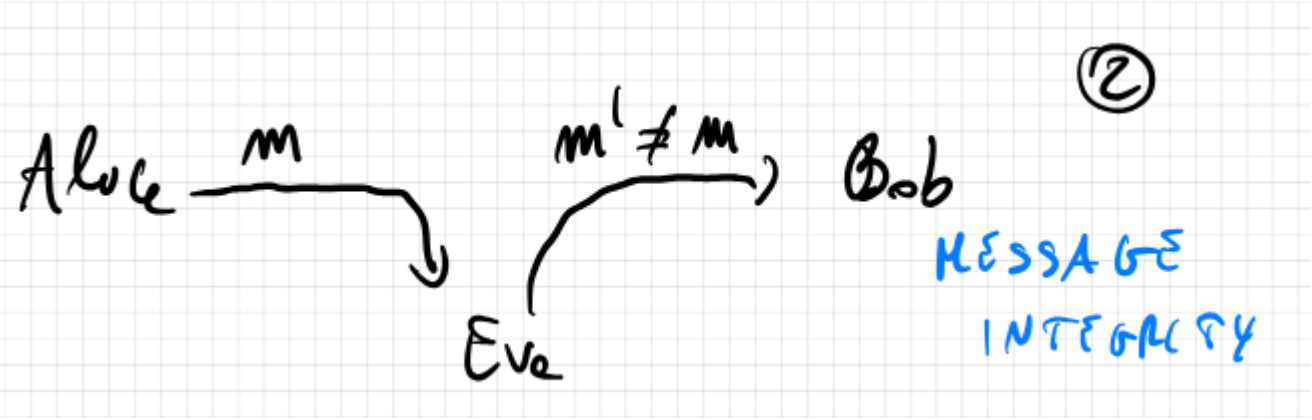
1.1 Secure Communication

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

Confidential Integrity



Message Integrity



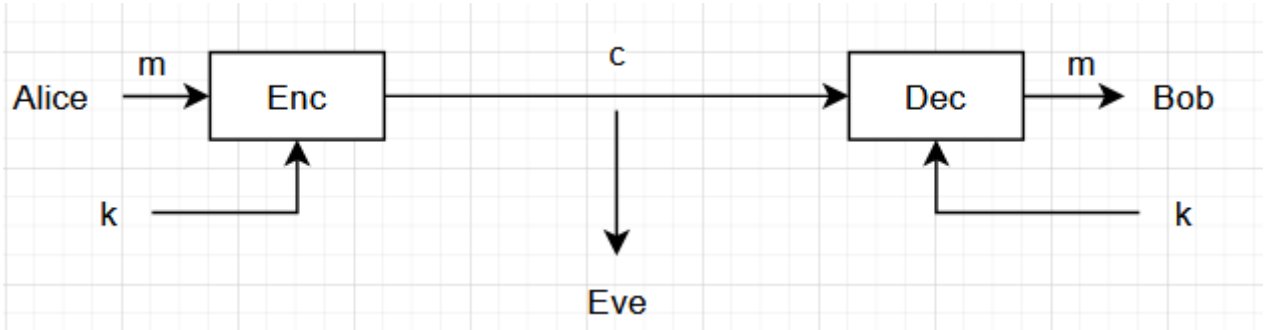
Basically we want our message to be both **confidential**, so no-one except the intended target sees it and we it to be unmodified, so that its **integrity** has not been compromised.

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 : $\mathscr{M} \times \mathscr{K} \to \mathscr{C}$
- 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:

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

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:

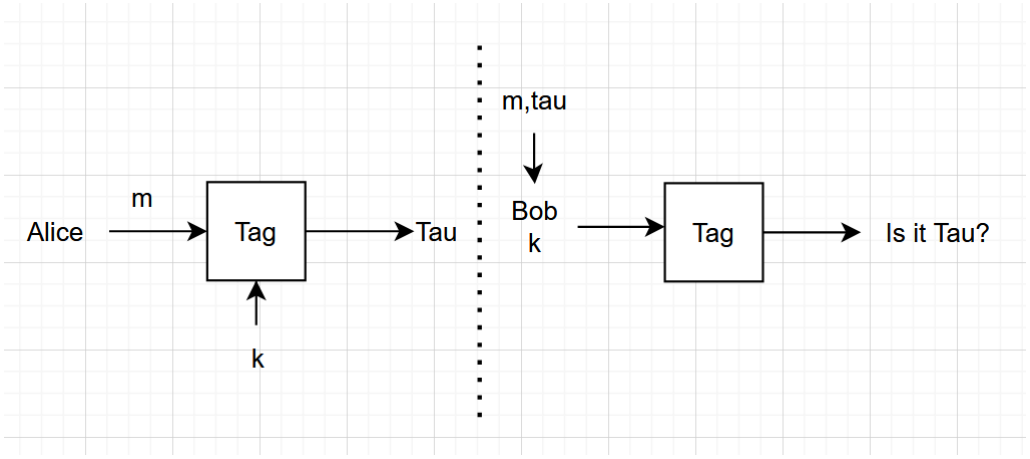
$$Pr[C = c] = \sum_{m'} Pr[C = c \wedge M = m'] = \sum_{m'} Pr[C = c | M = m'] * Pr[M = m'] =$$

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

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

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

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 $H_\infty(X) = 0$

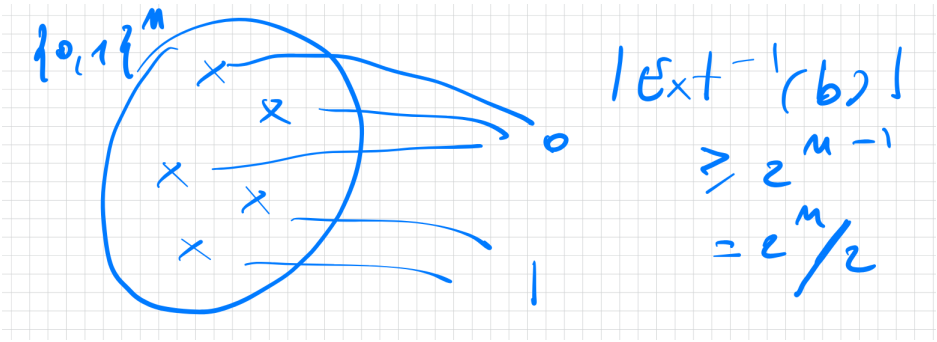
Here's the next best thing:
Design Ext that extracts UNIFORM $Y = Ext(X)$ for every X s.t. $H_\infty(X) \geq k$
But this is also impossible, even if

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

$$k = n - 1$$

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

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.

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

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

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

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

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

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

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

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:

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

□

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

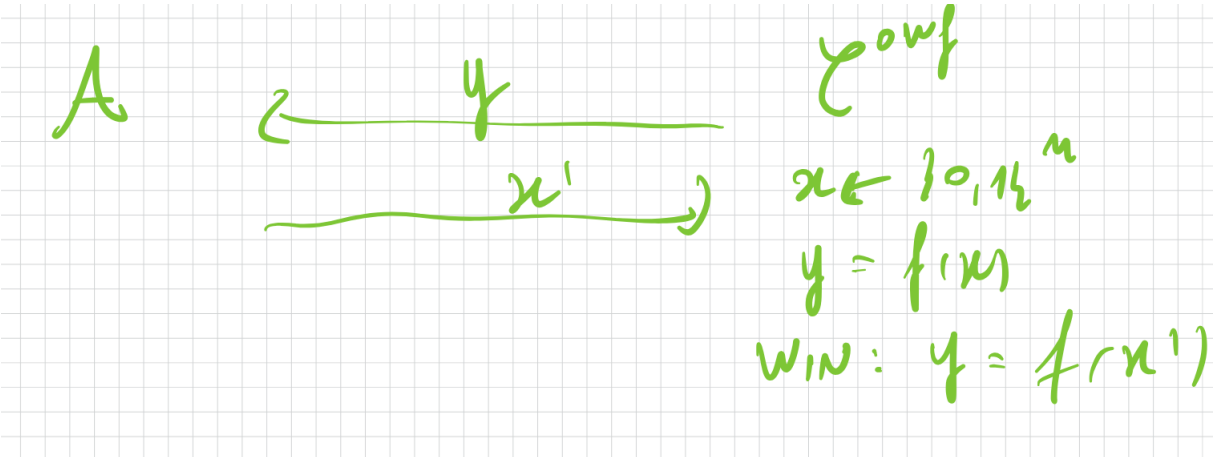
This is our first step towards efficient symmetric crypto. Moreover, pseudorandomness is used in modern computers to simulate real randomness. We will see that OWF are enough for pseudorandomness.

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

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

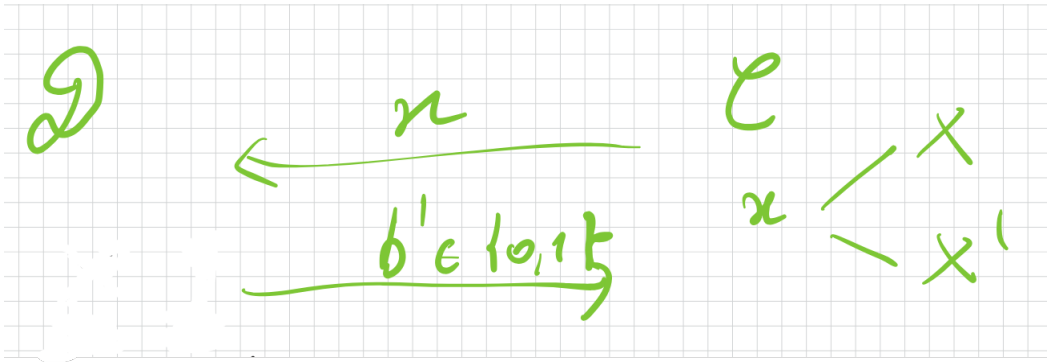
Informally, it goes to zero faster than any inverse of a polynomial function.

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):

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



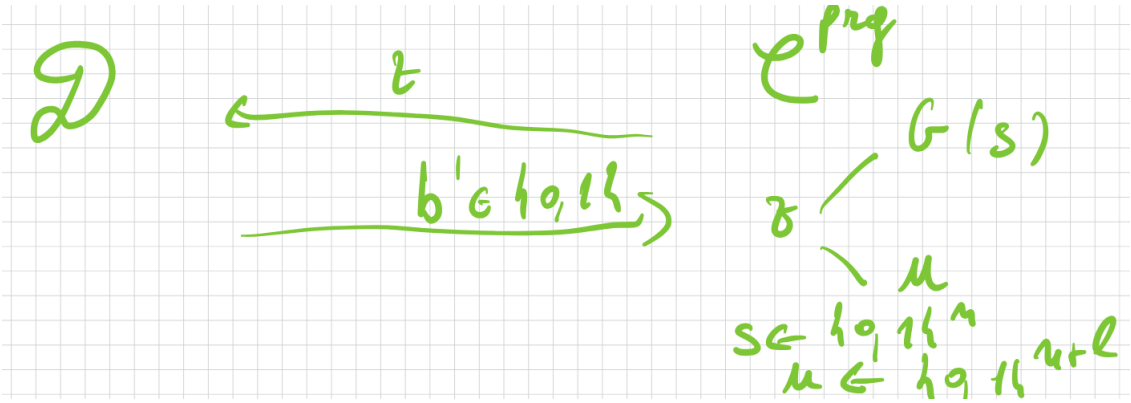
Definition 11. *X (X_n), Y (Y_n) are computationally indistinguishable ($X \approx_c Y$) if $\forall PPT \mathscr{D}$:*

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

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:

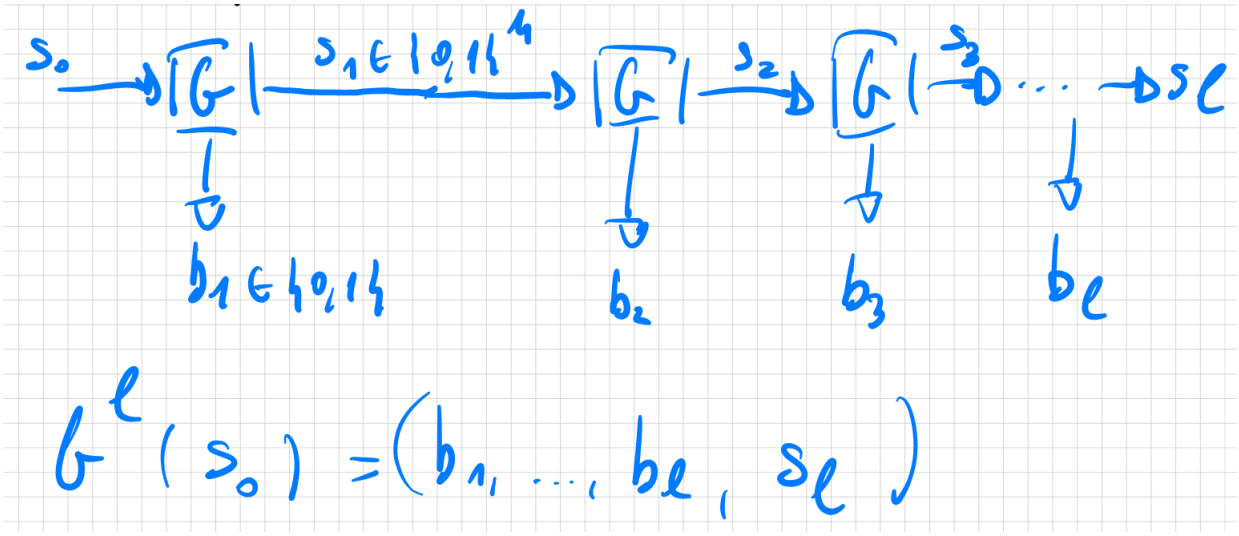
- Use a randomness extractor to get a uniform seed $s \in \{0,,1\}^n$.
- Define a simple PRG $G : \{0,1\}^n \to \{0,1\}^{n+1}$ with minimal stretch $l = 1$.
- Use G to stretch any $l(n) = poly(n)$.

Theory vs Practice:

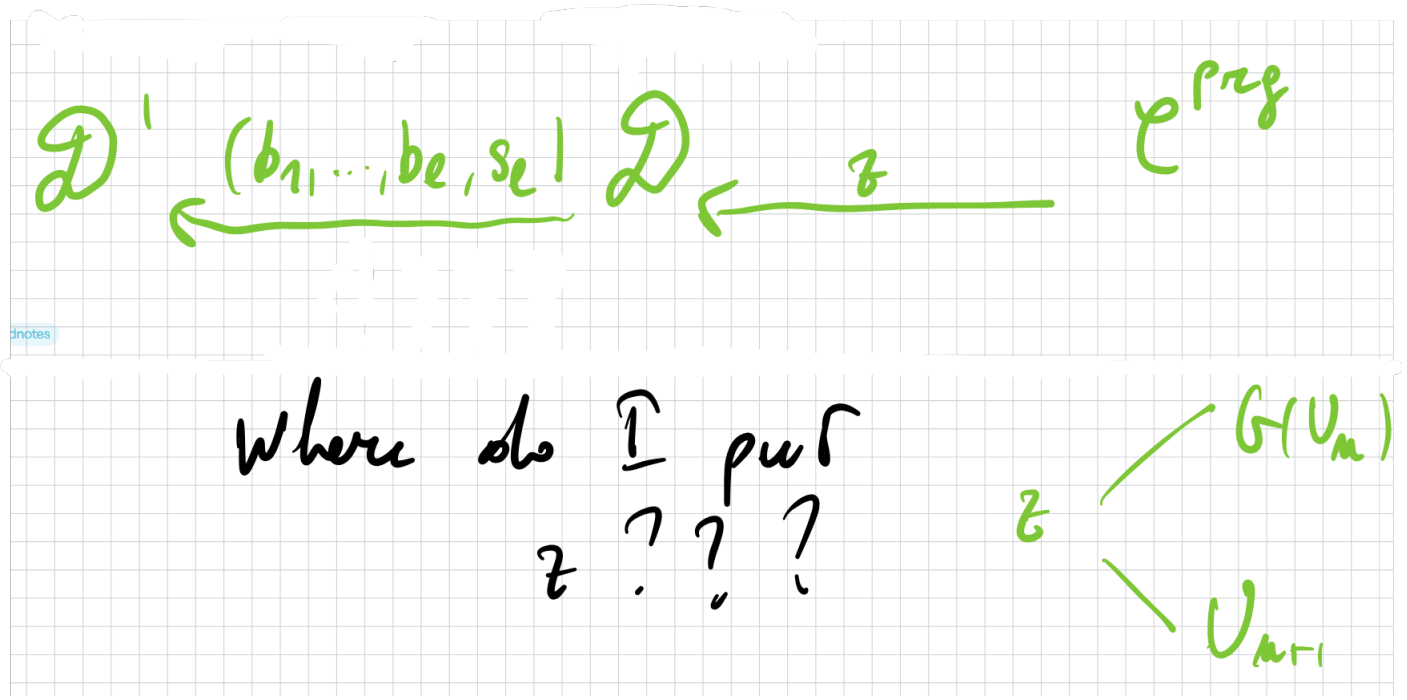
- Randomness extraction is what we already studied. But in practice it is done using Hash Functions.
- Theoretical G can be obtained from any OWF. Practical G is Heuristic

- 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

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

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

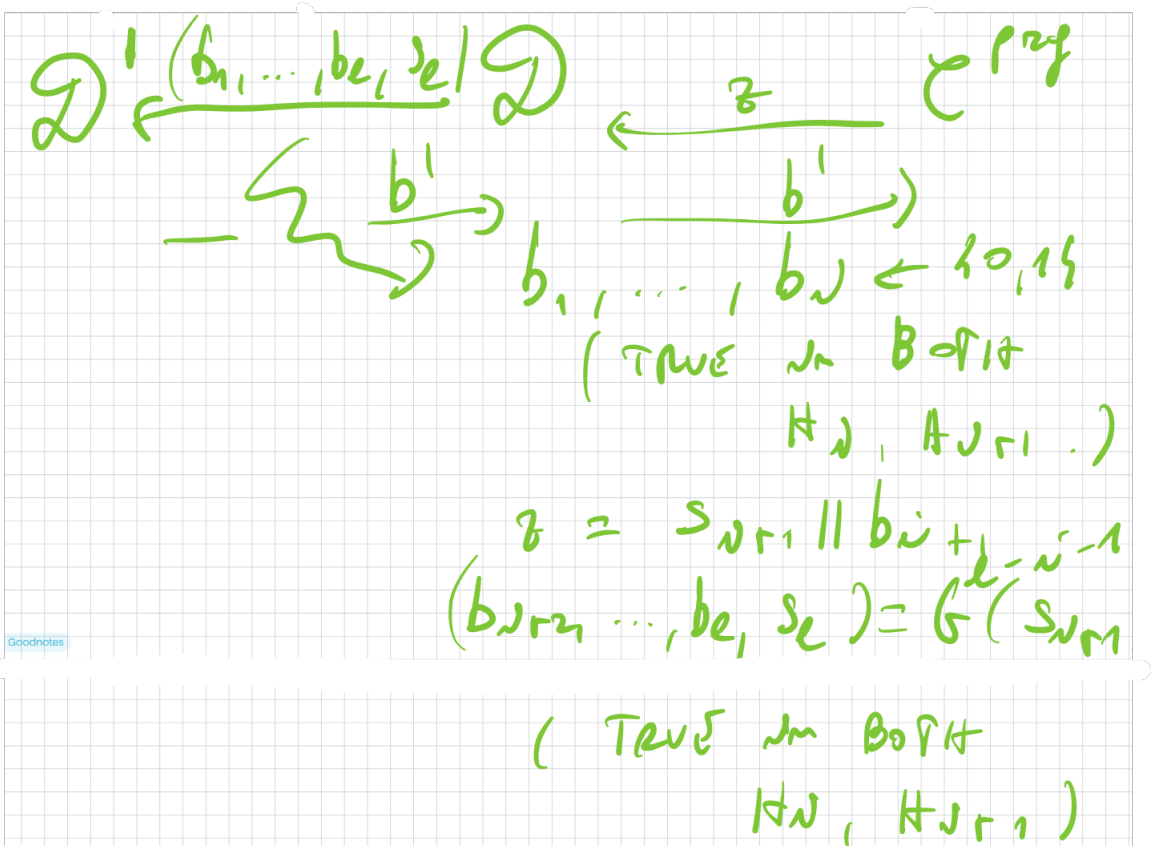
$$H_i(M) \equiv \begin{cases} b_1, \ldots, b_i \leftarrow \{0,1\} \\ s_i \leftarrow \{0,1\}^n \\ (b_{i+1}, \ldots, b_\ell, s_\ell) = G(s_i) \end{cases}$$

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

□

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

Proof. By reduction (as before):



By the above observations:

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

□

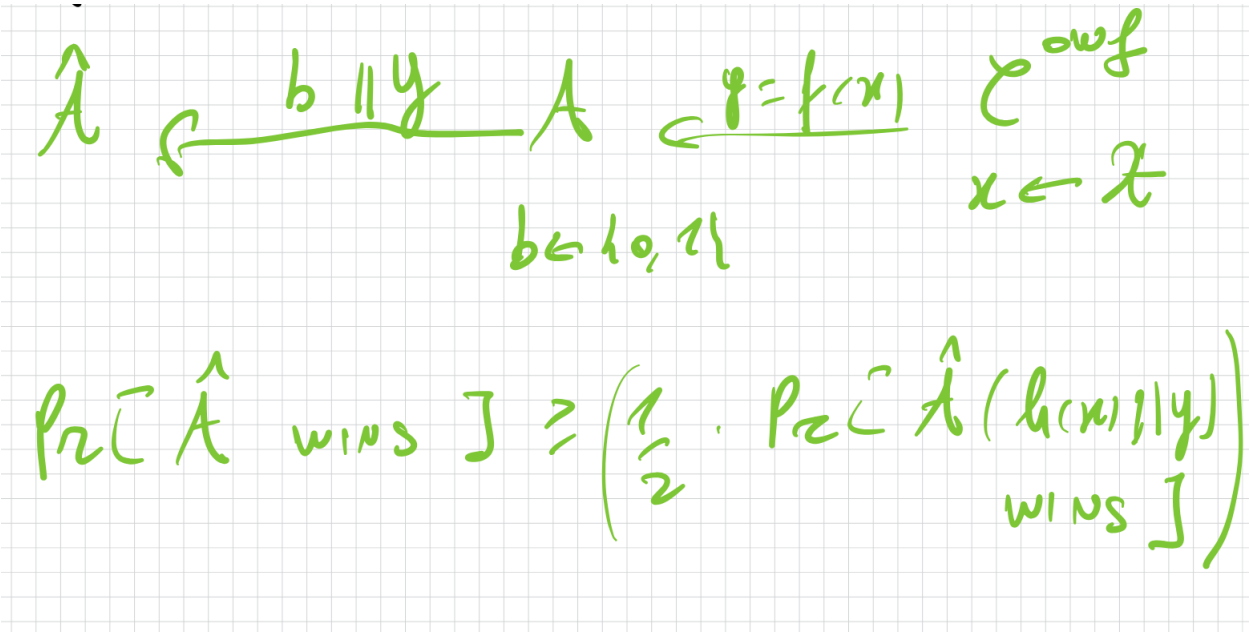
The next question is: How do we build $G : \{0,1\}^n \to \{0,1\}^{n+1}$?

- 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?



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

$$\Pr[\hat{\mathscr{A}} \text{ wins }] = \Pr[\hat{\mathscr{A}}(b,y) \text{ wins } \wedge b = h(x)] + \Pr[\hat{\mathscr{A}}(b,y) \text{ wins } \wedge b \neq h(x)]$$

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

$$\geq \frac{1}{2} * \frac{1}{\text{poly}}$$

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

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

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:*

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

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

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

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

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

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:

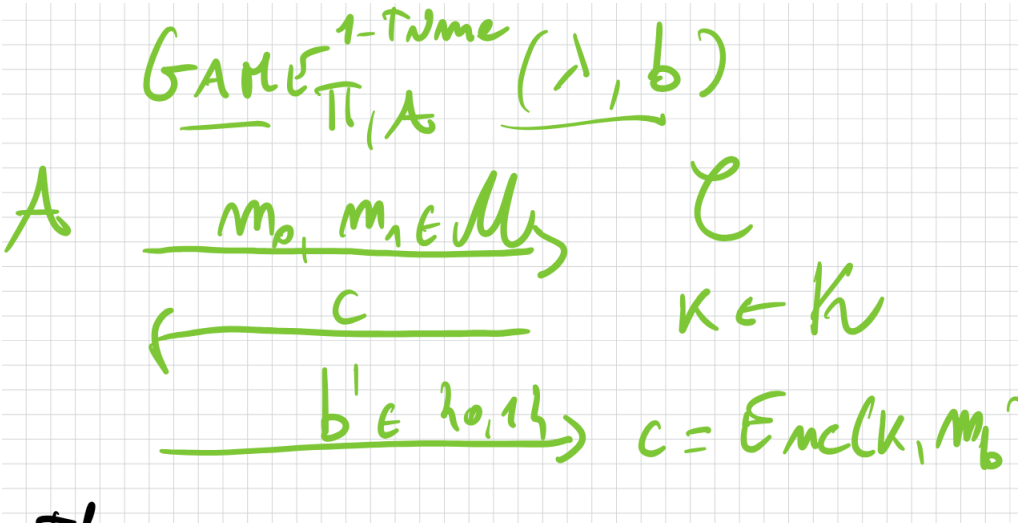
- **Encryption**: $\mathscr{E}nc(k, m) = G(k) \oplus m = c$
- **Decryption**: $\mathscr{D}ec(k, c) = G(k) \oplus c = m$

$k \in \{0,1\}^\lambda$, 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.

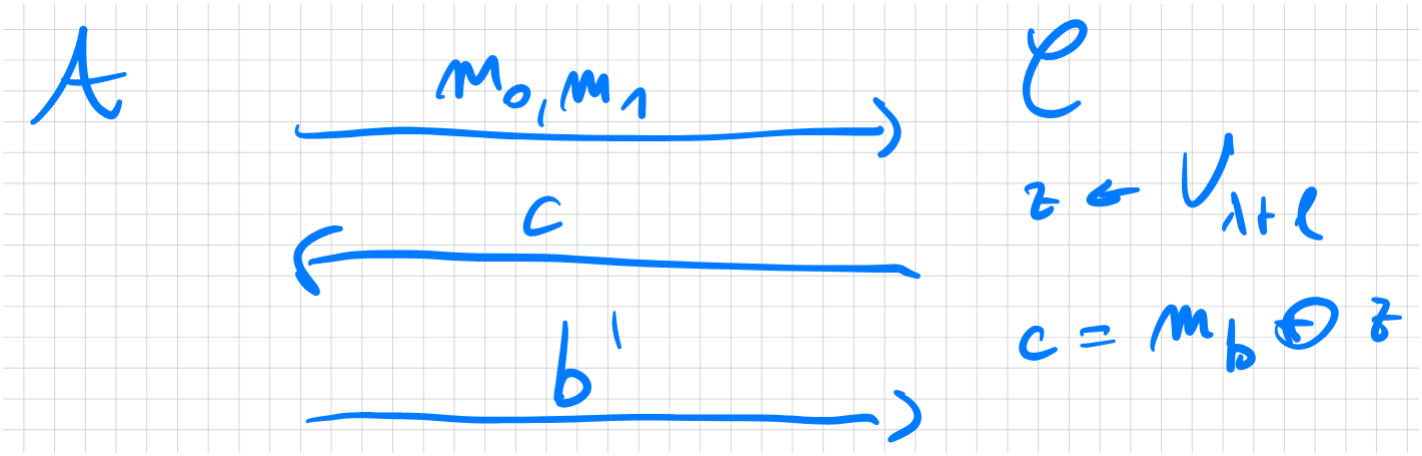
On the negative side, this notion is strictly for a **one-time** scenario (i.e., one key, one message). If the same key is used to encrypt two different messages:

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

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.*

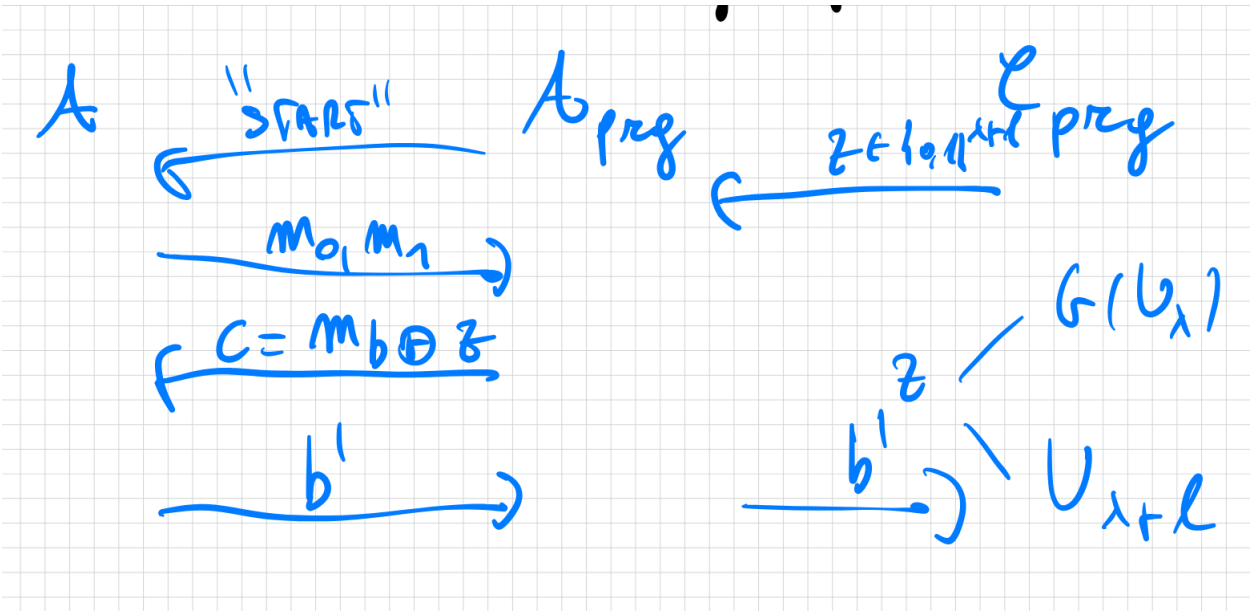
Proof. Starting with the initial experiment $\text{GAME}(\lambda, b) \equiv \text{GAME}^{\text{1-time}}_{\Pi,\mathscr{A}}(\lambda, b)$, we will introduce a hybrid experiment $\text{HYB}(\lambda, b)$ and show that:



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

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

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



By inspection:

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

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

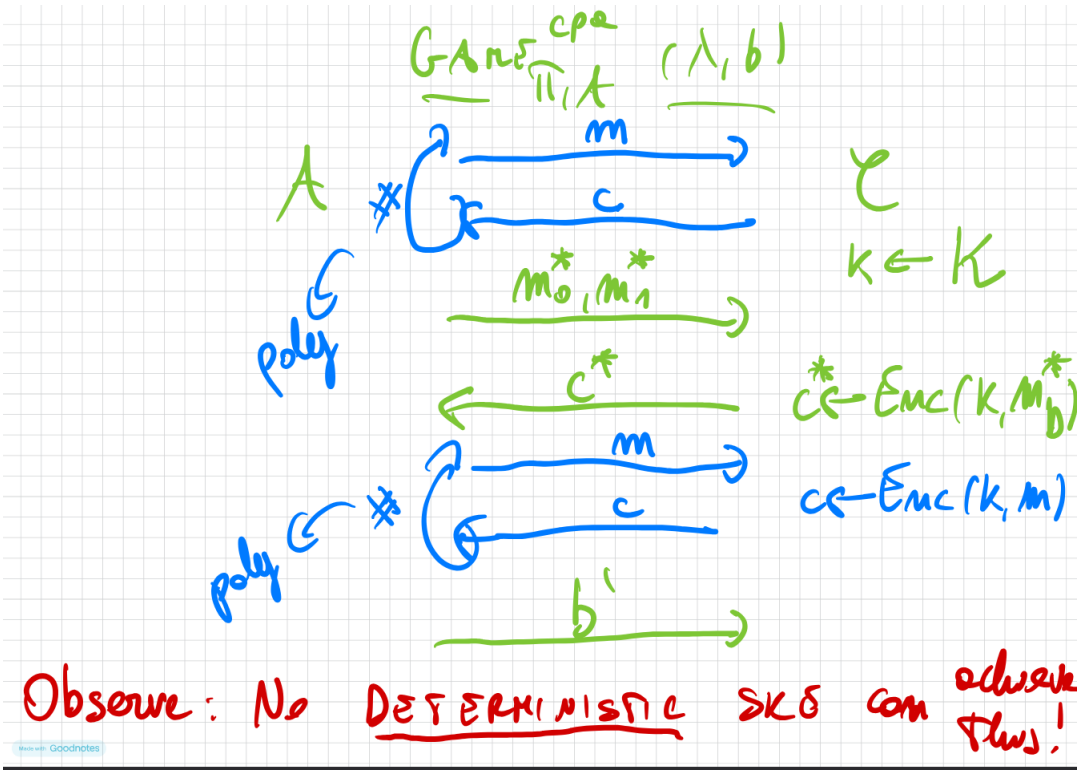
$$\implies \text{GAME}(\lambda, 0) \approx_c \text{GAME}(\lambda, 1)$$

□

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 $\mathscr{A}$:*

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



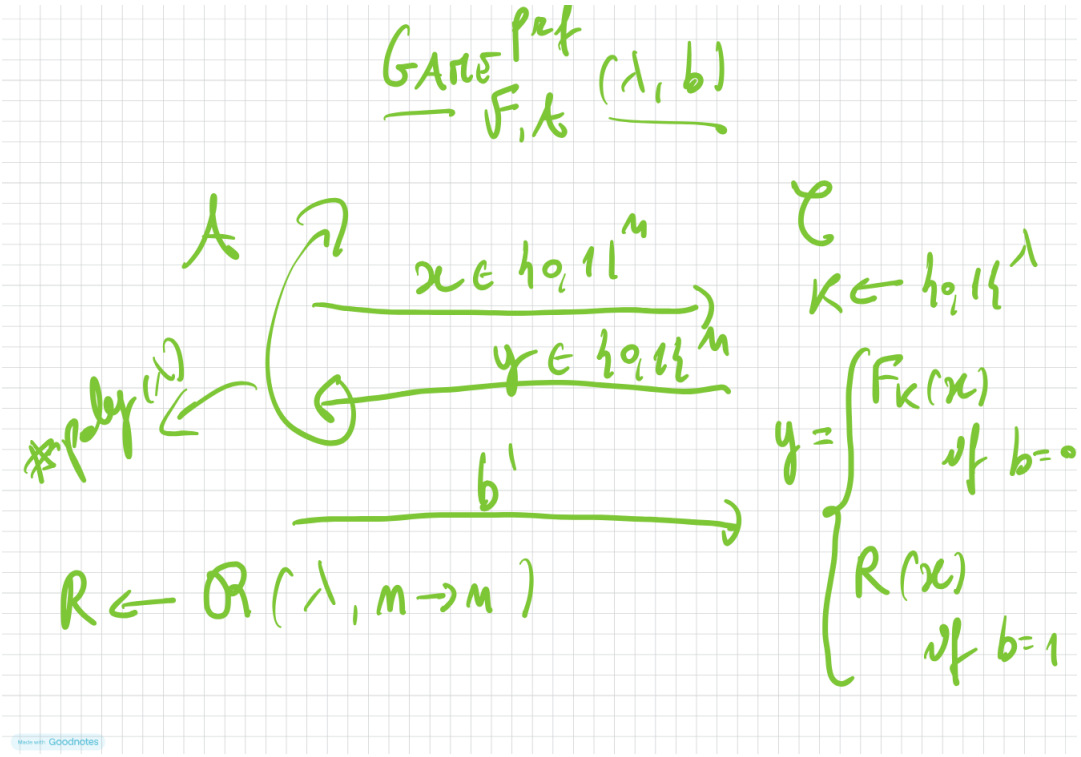
Observation: No deterministic SKE can be CPA-secure. An adversary could query the oracle on m_0^* to get c_0, then submit (m_0^*, m_1^*) as the challenge. If the challenge ciphertext c^* equals c_0, it knows $b = 0$. Therefore, CPA-secure encryption must be randomized or stateful.

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.

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

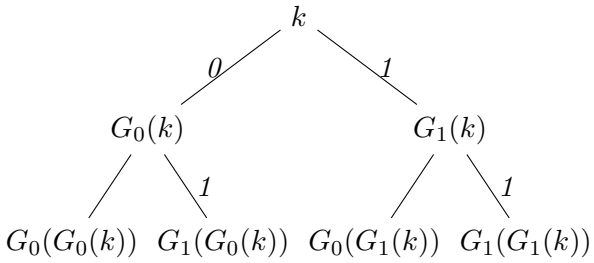
We cover our construction of PRFs.

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:

$$F_k(x_1 x_2 ... x_n) = G_{x_n}(G_{x_{n-1}}(...G_{x_1}(k)...))$$

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 $R(1)$.

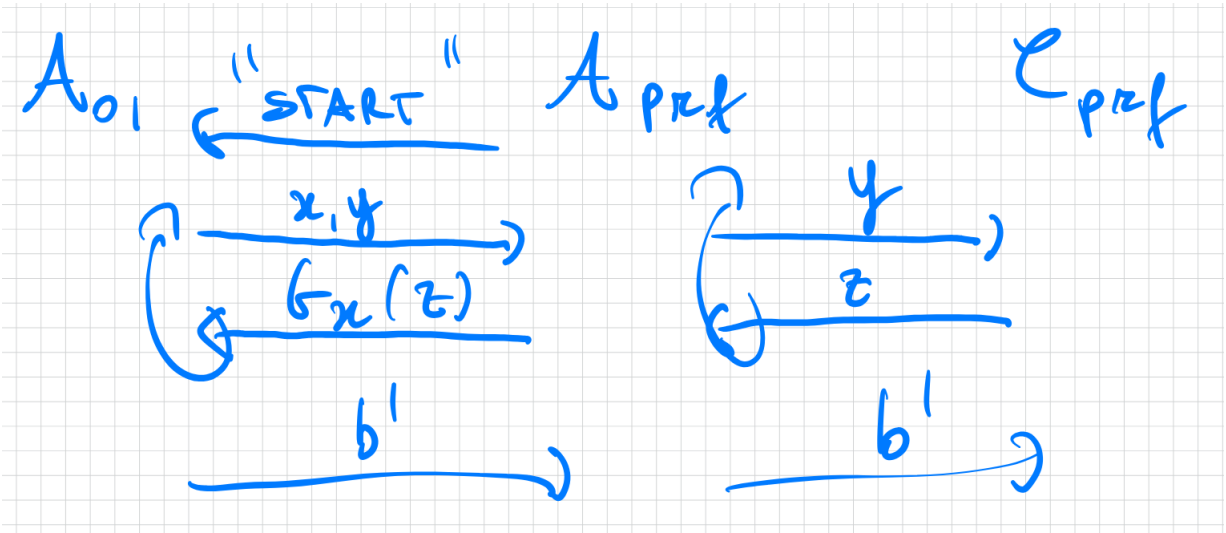
Inductive Step

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.

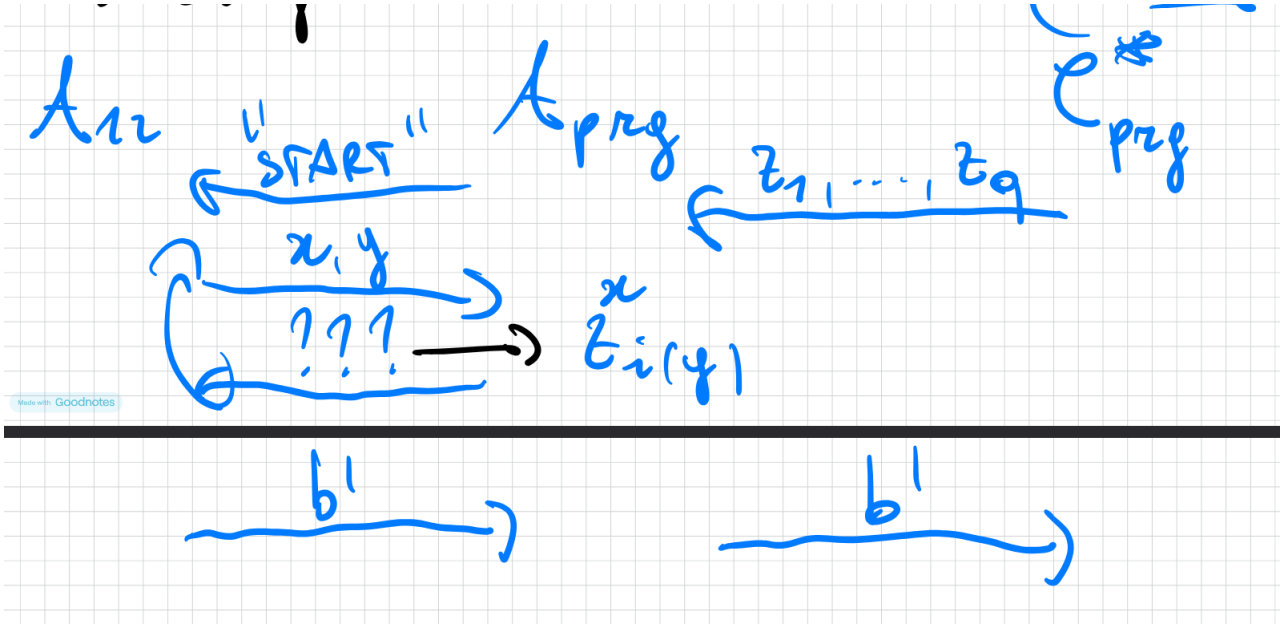
Step 2: $HYB_1 \approx_c HYB_2$ Here, we use the property that G is a PRG.

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).

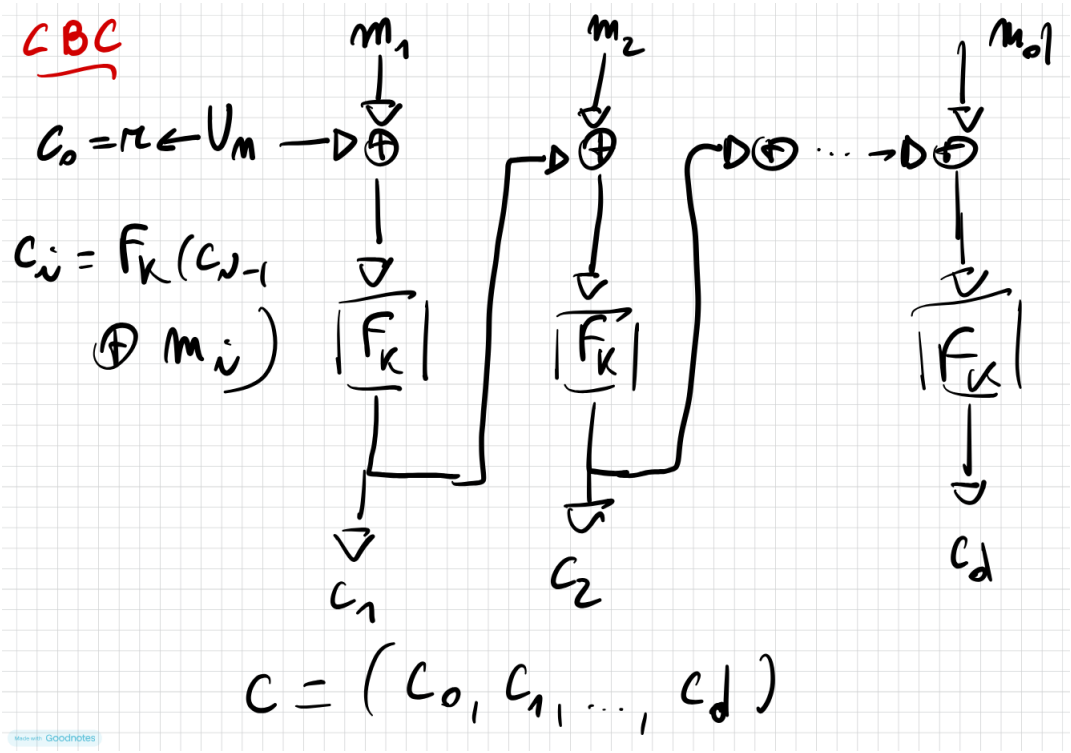
Mode of Operation

A mode of operation is a standardized way to encrypt messages $m = (m_1, \ldots, m_d)$, 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 $c = (F_k(m_1), \ldots, F_k(m_d))$. 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 $i = 1, \ldots, d$
- Output: $c = (c_0, c_1, \ldots, c_d)$

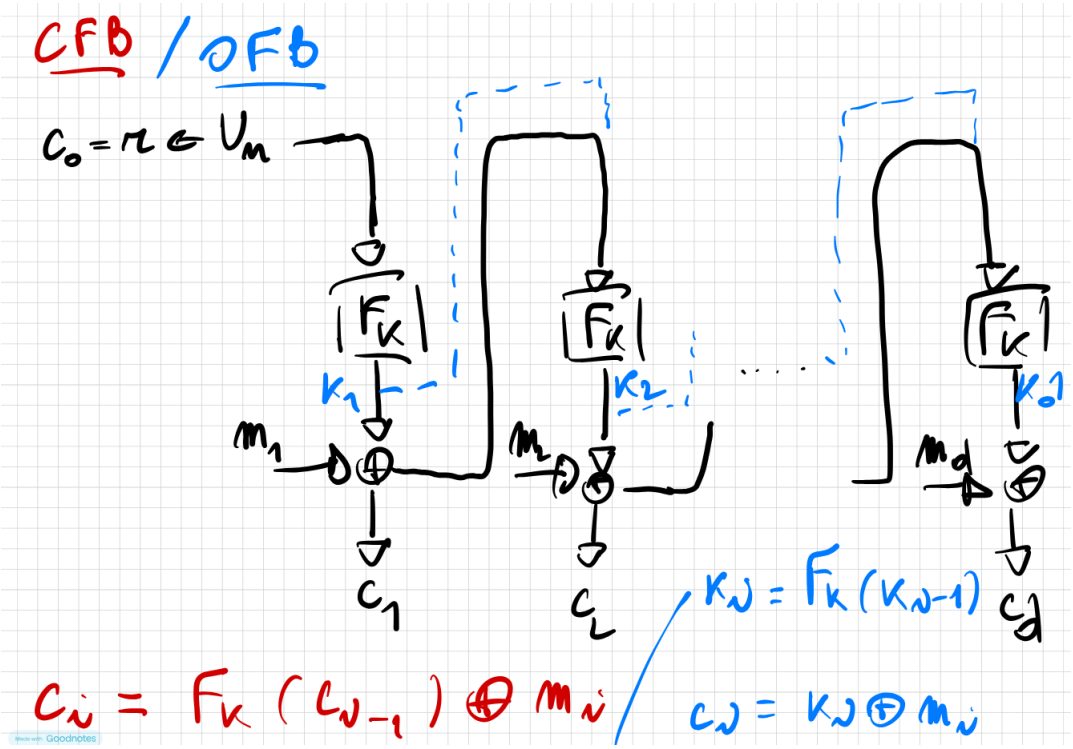


Decryption requires F_k^{-1}, so F must be a PRP. Encryption is sequential.

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

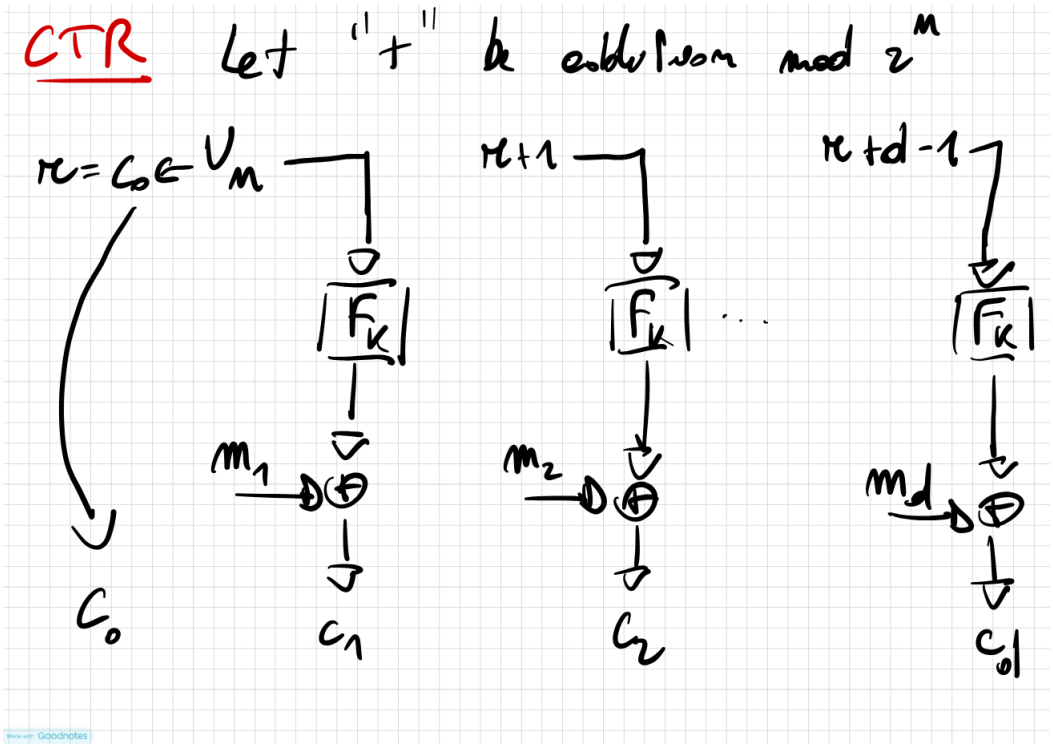
OFB (Output Feedback) Mode

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



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.



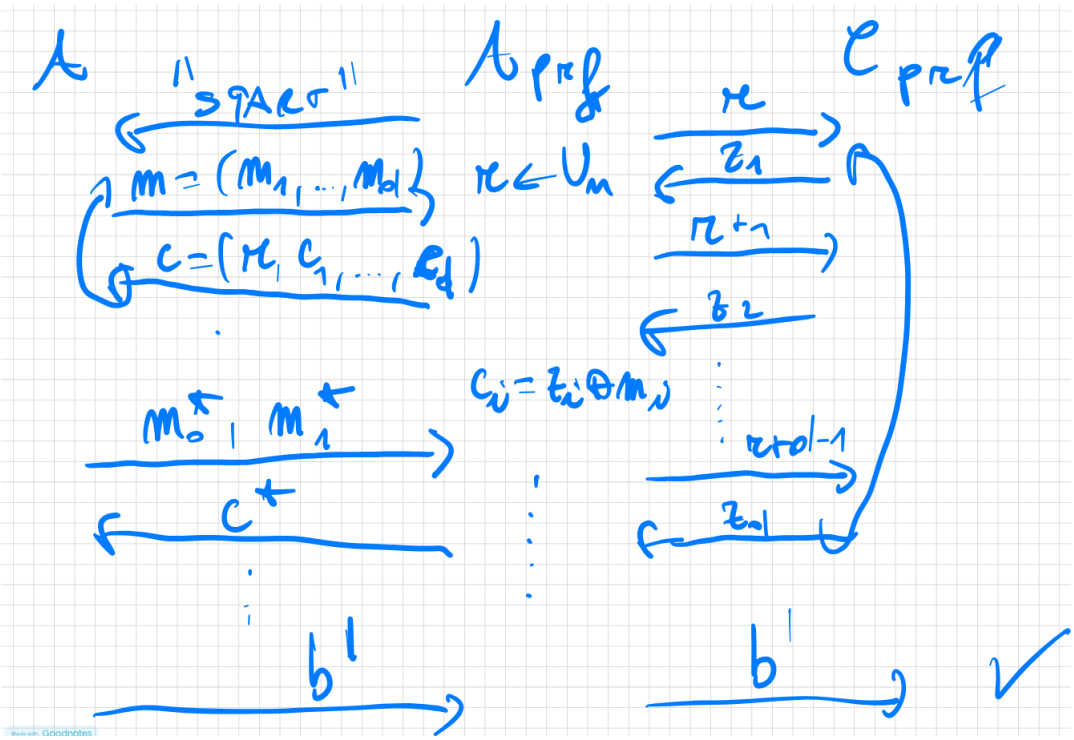
Theorem 13. *Assuming F is a PRF, CTR mode is a CPA-secure SKE for VIL.*

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

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

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

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

Proof of Theorem 14. We prove this by a standard reduction to the security of the PRF $\mathscr{F}$. We will show that if there exists a PPT adversary $\mathscr{A}$ that can win the UF-CMA game with non-negligible probability $\epsilon(\lambda)$, then we can build a PPT distinguisher $\mathscr{D}$ that can break the PRF security of $\mathscr{F}$ with the same non-negligible advantage.

Recall the PRF security game [cite: 626]: The distinguisher $\mathscr{D}$ is given an oracle $\mathscr{O}$, which is either the real PRF $F_k(\cdot)$ (for a random k) [cite: 639] or a truly random function $R(\cdot)$[cite: 640]. $\mathscr{D}$ must output a bit b' guessing which oracle it has. Its advantage is $|Pr[\mathscr{D} \text{ wins} | \mathscr{O} = F_k] - Pr[\mathscr{D} \text{ wins} | \mathscr{O} = R]|$.

Our distinguisher $\mathscr{D}$ will work as follows, using $\mathscr{A}$ as a subroutine:

1. $\mathscr{D}$ receives its oracle $\mathscr{O}$ (which is either F_k or R).
2. $\mathscr{D}$ runs the adversary $\mathscr{A}$.
3. When $\mathscr{A}$ makes a MAC query for a message m_i[cite: 865], $\mathscr{D}$ queries its own oracle $\mathscr{O}$ to get $\tau_i = \mathscr{O}(m_i)$. $\mathscr{D}$ then returns τ_i to $\mathscr{A}$.
4. $\mathscr{A}$ makes $q = poly(\lambda)$ such queries.
5. Eventually, $\mathscr{A}$ outputs its forgery attempt (m^*, τ^*)[cite: 866]. By the game rules, $m^* \notin \{m_1, \ldots, m_q\}$[cite: 867].
6. $\mathscr{D}$ must now output its guess. $\mathscr{D}$ queries its oracle $\mathscr{O}$ on m^* to get the "correct" tag $\tau'_* = \mathscr{O}(m^*)$.
7. **If** $\tau^* = \tau'_*$ (meaning $\mathscr{A}$'s forgery was successful), $\mathscr{D}$ outputs 1 (guessing it has the real PRF).
8. **Else** (if the forgery failed), $\mathscr{D}$ outputs 0 (guessing it has the random function).

Explanation of the Reduction: We are trying to see if $\mathscr{A}$'s success rate changes depending on the oracle $\mathscr{D}$ is given.

- **Case 1: $\mathscr{D}$ has the random function $R(\cdot)$ (Game 1 / H(1) [cite: 9]).** In this case, $\mathscr{A}$ is interacting with a truly random function. It queries $m_1, \ldots, m_q$ and receives q random values $\tau_1, \ldots, \tau_q$. It then outputs (m^*, τ^*) for a *new* message m^*. Since m^* has never been queried before, $R(m^*)$ is a fresh, uniformly random value from $\{0,1\}^n$, completely independent of all previous queries. The probability that $\mathscr{A}$'s guess τ^* is equal to this random value is exactly $1/2^n$. Therefore, $Pr[\mathscr{D} \text{ outputs } 1 | \mathscr{O} = R] = Pr[\mathscr{A} \text{ wins}|\text{Ideal}] \leq \frac{1}{2^n} = negl(\lambda)$.
- **Case 2: $\mathscr{D}$ has the real PRF $F_k(\cdot)$ (Game 0 / G(0) [cite: 5]).** In this case, $\mathscr{A}$ is interacting with the real MAC. The probability that $\mathscr{D}$ outputs 1 is exactly the probability that $\mathscr{A}$ wins the real UF-CMA game. $Pr[\mathscr{D} \text{ outputs } 1 | \mathscr{O} = F_k] = Pr[\mathscr{A} \text{ wins}|\text{Real}]$. By assumption, $\mathscr{A}$ wins with non-negligible probability $\epsilon(\lambda)$. So, $Pr[\mathscr{D} \text{ outputs } 1 | \mathscr{O} = F_k] = \epsilon(\lambda)$.

Conclusion: The advantage of our distinguisher $\mathscr{D}$ is:

$$|Pr[\mathscr{D} \text{ outputs } 1 | \mathscr{O} = F_k] - Pr[\mathscr{D} \text{ outputs } 1 | \mathscr{O} = R]|$$
$$= |\epsilon(\lambda) - negl(\lambda)|$$

This advantage is non-negligible. This contradicts the assumption that $\mathscr{F}$ is a secure PRF. Therefore, no such adversary $\mathscr{A}$ can exist, and the MAC is UF-CMA secure. □

3 Extending MACs to Longer Messages

[cite: 18] The previous theorem proves security for fixed-length messages, where the message length is the input length of the PRF. But what about variable-length or long messages? [cite: 19]

3.1 Warm-up: Insecure Constructions

[cite: 21] Let's analyze a few simple (and bad) ideas for extending a fixed-length MAC $Tag(k, \cdot) = F_k(\cdot)$ to handle a long message $m = (m_1, m_2, \ldots, m_d)$.

3.1.1 Construction 1: Block-by-Block MAC

Define $Tag(k, m) = (\tau_1, \ldots, \tau_d)$ where $\tau_i = F_k(m_i)$[cite: 23].

- **Attack (Splicing / Mix-and-Match):** [cite: 24-25] This construction is not secure. An attacker can perform a "mix-and-match" attack.
  1. Attacker queries for the tag on $m = (m_1, m_2)$[cite: 26]. It receives $\tau = (\tau_1, \tau_2)$.
  2. Attacker queries for the tag on $m' = (m_3, m_4)$[cite: 31]. It receives $\tau' = (\tau_3, \tau_4)$.
  3. Attacker can now forge the tag for a new message $m_{new} = (m_1, m_4)$. The forged tag is $\tau_{new} = (\tau_1, \tau_4)$[cite: 31]. This is a valid tag for a message the attacker never queried.

3.1.2 Construction 2: Simple XOR Sum

Define $Tag(k, m) = \bigoplus_{i=1}^{d} F_k(m_i)$[cite: 32].

- **Attack (Reordering / Forgery):** [cite: 37] This is also insecure.
  1. Attacker queries for the tag on a one-block message $m = (m_1)$. It receives $\tau = F_k(m_1)$.
  2. Attacker can now forge the tag for the two-block message $m^* = (m_1, m_1)$[cite: 45].
  3. The valid tag would be $\tau^* = F_k(m_1) \oplus F_k(m_1) = 0$.
  4. The attacker outputs $(m^*, 0)$. This is a valid forgery for a new message.

3.2 A Secure Construction: Hash-then-MAC

The secure idea is to first hash the long message m into a short, fixed-length string, and then apply our secure fixed-length MAC to that string[cite: 51].

Let $\mathscr{H} = \{h_s : \{0,1\}^n d \to \{0,1\}^n\}_{s \in \{0,1\}^\lambda}$ be a family of hash functions, where s is a key for the hash function. Let $\mathscr{F} = \{F_k : \{0,1\}^n \to \{0,1\}^n\}_{k \in \mathscr{K}}$ be a PRF.

The Hash-then-MAC construction is:

$$Tag_{(k,s)}(m) = F_k(h_s(m))$$

[cite: 51] The full key is the pair (k, s).

What property do we need from $\mathscr{H}$? [cite: 51] If an attacker can find $m \neq m'$ such that $h_s(m) = h_s(m')$ (a collision) [cite: 54-55], they can break the MAC. They would ask for the tag $\tau = Tag(m) = F_k(h_s(m))$. They would then know that τ is also the tag for m', a message they never queried.

So, we need $\mathscr{H}$ to be **Collision-Resistant**[cite: 56]. However, this is a strong requirement and can be computationally expensive. We can get away with a weaker, statistical property.

Definition 19 (ϵ-Almost Universal (AU) Hash Family)**.** *[cite: 57-58] A family of hash functions $\mathscr{H} = \{h_s : \mathscr{M} \to \mathscr{T}\}_{s \in \mathscr{S}}$ is **ϵ-Almost Universal** if for all distinct messages $m, m' \in \mathscr{M}$ (with $m \neq m'$)[cite: 59], we have:*

$$Pr_{s \leftarrow \mathscr{S}}[h_s(m) = h_s(m')] \leq \epsilon$$

*[cite: 60-61] If $\epsilon = 1/|\mathscr{T}|$ (e.g., 2^{-n}) [cite: 62], the family is called **Perfectly Universal**[cite: 65]. We only require ϵ to be a negligible function, $\epsilon = negl(\lambda)$[cite: 67].*

Theorem 15. *If $\mathscr{F}$ is a secure PRF and $\mathscr{H}$ is an ϵ-AU hash family (with $\epsilon = negl(\lambda)$), then the Hash-then-MAC construction $Tag_{(k,s)}(m) = F_k(h_s(m))$ is a UF-CMA secure MAC. [cite: 71]*

Proof. We use a hybrid argument to prove this. Let $\mathscr{A}$ be a PPT adversary making q queries.

- **Game 0 (H_0):** The real game[cite: 76]. The challenger picks random $k \in \mathscr{K}$ and $s \in \mathscr{S}$. $\mathscr{A}$ interacts with an oracle $\mathscr{O}(m) = F_k(h_s(m))$.
- **Game 1 (H_1):** The "Hashed Random Function" game[cite: 78]. The challenger picks a random $s \in \mathscr{S}$ and a truly random function $R : \{0,1\}^n \to \{0,1\}^n$. $\mathscr{A}$ interacts with $\mathscr{O}(m) = R(h_s(m))$.
- **Game 2 (H_2):** The "Ideal MAC" game[cite: 81]. The challenger picks a truly random function $R' : \{0,1\}^* \to \{0,1\}^n$. $\mathscr{A}$ interacts with $\mathscr{O}(m) = R'(m)$.

We will show that $Pr[\mathscr{A}$ wins in $H_0] \approx Pr[\mathscr{A}$ wins in $H_1] \approx Pr[\mathscr{A}$ wins in $H_2]$, and that $Pr[\mathscr{A}$ wins in $H_2]$ is negligible.

Step 1: $H_0 \approx_c H_1$ [cite: 82] This step relies on the security of the PRF $\mathscr{F}$. We can build a distinguisher $\mathscr{D}$ for the PRF game:

1. $\mathscr{D}$ receives its oracle $\mathscr{O}'$ (which is either F_k or R).
2. $\mathscr{D}$ simulates H_0 or H_1 for $\mathscr{A}$. It picks a hash key $s \in \mathscr{S}$ itself.

3. When $\mathscr{A}$ queries m_i, $\mathscr{D}$ computes the digest $d_i = h_s(m_i)$.

4. $\mathscr{D}$ queries its own oracle $\mathscr{O}'$ to get $\tau_i = \mathscr{O}'(d_i)$ and returns τ_i to $\mathscr{A}$.

5. When $\mathscr{A}$ outputs a forgery (m^*, τ^*), $\mathscr{D}$ computes $d^* = h_s(m^*)$ and checks if $\mathscr{O}'(d^*) = \tau^*$ and m^* is new.

6. If $\mathscr{A}$ wins, $\mathscr{D}$ outputs 1, else 0.

From $\mathscr{A}$'s perspective, this simulation is perfect. $Pr[\mathscr{D}$ outputs $1|\mathscr{O}' = F_k] = Pr[\mathscr{A}$ wins in $H_0]$. $Pr[\mathscr{D}$ outputs $1|\mathscr{O}' = R] = Pr[\mathscr{A}$ wins in $H_1]$. By the PRF security, $|Pr[\mathscr{A}$ wins in $H_0] - Pr[\mathscr{A}$ wins in $H_1]| \leq negl(\lambda)$.

Step 2: $H_1 \approx H_2$ [cite: 94] This step relies on the ϵ-AU property of $\mathscr{H}$. We compare H_1 (oracle $R(h_s(\cdot))$) and H_2 (oracle $R'(\cdot)$). In H_2, every new message m_i maps to a fresh, independent random value $R'(m_i)$. In H_1, this is also true... *unless* a hash collision occurs.

Let **BAD** be the event that a collision occurs among any of the q queries $\mathscr{A}$ makes, or its final forgery m^* (for a total of $q+1$ messages)[cite: 97, 100].

$$\text{BAD} \equiv \exists i \neq j \in \{1, \ldots, q, *\} \text{ s.t. } m_i \neq m_j \text{ and } h_s(m_i) = h_s(m_j)$$

- **If ¬BAD occurs (no collision):** Then every distinct message m_i maps to a distinct digest $d_i = h_s(m_i)$. Since R is a random function, $R(d_i)$ is a fresh random value for each i. This is *identical* to the behavior of H_2. So, $Pr[\mathscr{A}$ wins $|\neg\text{BAD}, H_1] = Pr[\mathscr{A}$ wins $|\neg\text{BAD}, H_2]$.

- The only difference between the games is when BAD occurs. By the "Difference Lemma", the statistical distance between the games is bounded by the probability of this event: $SD(H_1, H_2) \leq Pr[\text{BAD}]$[cite: 105].

Bounding $Pr[$BAD$]$: [cite: 109] We have $q+1$ messages in total. The number of pairs is $\binom{q+1}{2}$. We can use a Union Bound over all pairs:

$$Pr[\text{BAD}] \leq \sum_{1 \leq i < j \leq q+1} Pr[h_s(m_i) = h_s(m_j)]$$

[cite: 112] By the ϵ-AU property, each term is $\leq \epsilon$[cite: 60].

$$Pr[\text{BAD}] \leq \binom{q+1}{2} \cdot \epsilon$$

[cite: 112] Since $q = poly(\lambda)$ and $\epsilon = negl(\lambda)$, $Pr[$BAD$]$ is negligible. Thus, $Pr[\mathscr{A}$ wins in $H_1] \leq Pr[\mathscr{A}$ wins in $H_2] + Pr[\text{BAD}] \approx Pr[\mathscr{A}$ wins in $H_2]$.

Step 3: Bounding the Win Probability in H_2 In H_2, $\mathscr{A}$ interacts with a truly random function R'. It makes q queries and gets q random tags. It then outputs (m^*, τ^*) for a new message m^*. The probability that τ^* equals the fresh random value $R'(m^*)$ is $1/2^n$. $Pr[\mathscr{A}$ wins in $H_2] \leq \frac{1}{2^n} = negl(\lambda)$.

Conclusion: Putting it all together: $Pr[\mathscr{A}$ wins in $H_0] \approx Pr[\mathscr{A}$ wins in $H_1] \approx Pr[\mathscr{A}$ wins in $H_2] \leq negl(\lambda)$. Therefore, the probability of winning the real game is negligible. □

4 Examples of AU Hash Families

[cite: 113] To use the Hash-then-MAC construction, we need concrete AU hash families.

4.1 Example 1: Vector Dot Product

[cite: 116]

- **Field:** Let $\mathbb{F} = GF(2^n)$[cite: 114].

- **Messages:** $m = (m_1, \ldots, m_d)$ where each $m_i \in \mathbb{F}$.

- **Keys:** $s = (s_1, \ldots, s_d)$ where each $s_i \in \mathbb{F}$. The key space $\mathscr{S} = \mathbb{F}^d$[cite: 116].
- **Hash Function:** $h_s(m) = \sum_{i=1}^{d} s_i \cdot m_i = \langle \vec{s}, \vec{m} \rangle$ [cite: 116] (This is the dot product in $\mathbb{F}$).

Proof of AU Property: We want to calculate $Pr_{s \in \mathbb{F}^d}[h_s(m) = h_s(m')]$ for $m \neq m'$[cite: 121, 123].

$$h_s(m) = h_s(m') \implies \sum_{i=1}^{d} s_i m_i = \sum_{i=1}^{d} s_i m_i'$$

$$\implies \sum_{i=1}^{d} s_i (m_i - m_i') = 0$$

[cite: 125] Let $\delta_i = m_i - m_i'$[cite: 119]. Since $m \neq m'$, at least one $\delta_j \neq 0$. Without loss of generality, assume $\delta_1 \neq 0$[cite: 122]. We can now solve for s_1:

$$s_1 \delta_1 = -\sum_{i=2}^{d} s_i \delta_i$$

Since $\delta_1 \neq 0$, it has a multiplicative inverse in $\mathbb{F}$.

$$s_1 = (\delta_1)^{-1} \cdot \left(-\sum_{i=2}^{d} s_i \delta_i \right)$$

[cite: 126] **Explanation:** This equation shows that for any possible choice of $s_2, \ldots, s_d$ (of which there are $|\mathbb{F}|^{d-1}$ choices), the value of s_1 is *uniquely determined* for a collision to occur. The total number of keys $s \in \mathscr{S}$ is $|\mathbb{F}|^d$. The number of keys causing a collision for this fixed m, m' is $|\mathbb{F}|^{d-1}$ (since $s_2, \ldots, s_d$ can be anything, but s_1 is then fixed).

$$Pr[\text{collision}] = \frac{\#\text{ collision keys}}{\#\text{ total keys}} = \frac{|\mathbb{F}|^{d-1}}{|\mathbb{F}|^d} = \frac{1}{|\mathbb{F}|} = \frac{1}{2^n}$$

[cite: 127] This family is ϵ-AU with $\epsilon = 2^{-n}$. It is perfectly universal.

4.2 Example 2: Polynomial Evaluation

[cite: 130]

- **Field:** Let $\mathbb{F} = GF(2^n)$[cite: 128].
- **Messages:** $m = (m_1, \ldots, m_d)$ where each $m_i \in \mathbb{F}$.
- **Keys:** $s \in \mathbb{F}$. The key s is just a single element from the field[cite: 128].
- **Hash Function:** We treat the message blocks m_i as coefficients of a polynomial of degree $d-1$:

$$q_m(x) = \sum_{j=1}^{d} m_j x^{j-1}$$

[cite: 130] The hash is the evaluation of this polynomial at the key s:

$$h_s(m) = q_m(s)$$

[cite: 130]

Proof of AU Property: We want $Pr_{s\in\mathbb{F}}[h_s(m) = h_s(m')]$ for $m \neq m'$[cite: 131].

$$h_s(m) = h_s(m') \implies q_m(s) = q_{m'}(s) \implies q_m(s) - q_{m'}(s) = 0$$

[cite: 133] This is equivalent to:

$$q_{m-m'}(s) = \sum_{j=1}^{d}(m_j - m'_j)s^{j-1} = 0$$

[cite: 134-135] Let $q_\delta(x) = q_{m-m'}(x)$. Since $m \neq m'$, the vector of coefficients $(\delta_1, \ldots, \delta_d)$ is not all-zero, so $q_\delta(x)$ is a non-zero polynomial. The degree of $q_\delta(x)$ is at most $d-1$[cite: 136]. By the **Schwartz-Zippel Lemma** (or the fact that a non-zero polynomial of degree $d-1$ over a field can have at most $d-1$ roots), there are at most $d-1$ values of $s \in \mathbb{F}$ that can be a root of $q_\delta(x)$.

$$Pr[\text{collision}] = \frac{\#\text{ of roots}}{\#\text{ of possible keys}} \leq \frac{d-1}{|\mathbb{F}|} = \frac{d-1}{2^n}$$

[cite: 136] This family is ϵ-AU with $\epsilon = (d-1)/2^n$. This is negligible as long as the number of blocks d is polynomial.

4.3 Example 3: CBC-MAC as a Hash Function

[cite: 137] The CBC-MAC construction itself can be viewed as a way to construct an AU hash family[cite: 137].

- **Construction:** Let $\mathscr{F} = \{F_s : \{0,1\}^n \to \{0,1\}^n\}$ be a PRF family. The hash key is s, the PRF key.
- Let $m = (m_1, \ldots, m_d)$. The CBC-MAC construction is [cite: 141-147]:
  1. $t_0 = 0^n$ (a fixed IV)
  2. $t_i = F_s(t_{i-1} \oplus m_i)$ for $i = 1, \ldots, d$
- **Hash Function:** $h_s(m) = t_d$.

Theorem 16 (Informal)**.** *If $\mathscr{F}$ is a secure PRF, then the CBC-MAC construction (for fixed-length messages m of d blocks) defines a **computationally ϵ-AU** hash family[cite: 153].*

Explanation: This means that for any $m \neq m'$, the probability $Pr_s[h_s(m) = h_s(m')]$ is negligible. This is a *computational* bound (it relies on the adversary's inability to break the PRF) rather than a purely *statistical* one like the previous two examples. Proving this is more involved but shows that CBC-MAC is a secure method for hashing a long message into a short, pseudorandom tag.

5 Authenticated Encryption and CCA Security

So far, we have looked at two goals of symmetric cryptography in isolation:

- **Privacy:** Ensuring messages are confidential (CPA-Secure SKE).
- **Integrity:** Ensuring messages are authentic (UF-CMA MACs).

What's left is to combine them. This leads to the strongest notion of security for symmetric key encryption: non-malleability, also known as security against Chosen-Ciphertext Attacks (CCA).

5.1 Malleability and Chosen-Ciphertext Attacks (CCA)

Definition 20 (Malleability)**.** *Malleability is the property of an encryption scheme where an adversary, upon seeing a ciphertext c, can modify it to create a new ciphertext c' such that the corresponding plaintext m' is meaningfully related to the original plaintext m, even without knowing m.*

This is a problem in many applications, such as sealed-bid auctions. If an adversary sees Alice's encrypted bid $c = Enc(k, "100")$ and can change it to $c' = Enc(k, "99")$ (a related plaintext), they can win the auction unfairly.

Unfortunately, **CPA-security does not prevent malleability**. Consider the following CPA-secure SKE (a variant of CTR mode):

- $Enc(k, m) = (r, F_k(r) \oplus m)$, where $r \leftarrow U_n$.

An adversary can intercept a ciphertext $c = (c_0, c_1) = (r, F_k(r) \oplus m)$. They can then create a new, "mangled" ciphertext:

$$\tilde{c} = (c_0, c_1 \oplus \vec{v})$$

where $\vec{v} = (1, 0, \ldots, 0)$ is a vector. The decryption of $\tilde{c}$ will be:

$$Dec(k, \tilde{c}) = F_k(c_0) \oplus (c_1 \oplus \vec{v}) = F_k(r) \oplus (F_k(r) \oplus m \oplus \vec{v}) = m \oplus \vec{v}$$

The adversary successfully flipped the first bit of the plaintext without ever knowing what it was. This is a classic malleability attack.

To prevent this, we introduce the formal notion of **Non-Malleability (NM)**, which is defined using the **Chosen-Ciphertext Attack (CCA)** security game.

Definition 21 (CCA Security)**.** *A Symmetric Key Encryption scheme $\Pi = (Enc, Dec)$ is **secure against Chosen-Ciphertext Attacks (CCA-secure)** if for any PPT adversary $\mathscr{A}$, the advantage in the following game is negligible:*

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

The $GAME^{CCA}$ proceeds as follows:

1. A key $k \leftarrow \mathscr{K}$ is generated.
2. The adversary $\mathscr{A}$ is given oracle access to $Enc(k, \cdot)$ and $Dec(k, \cdot)$.
3. $\mathscr{A}$ makes any number of encryption and decryption queries.
4. $\mathscr{A}$ submits two challenge messages m_0^*, m_1^* (of equal length).
5. The challenger picks $b \leftarrow \{0, 1\}$, computes the challenge ciphertext $c^* \leftarrow Enc(k, m_b^*)$, and sends c^* to $\mathscr{A}$.
6. $\mathscr{A}$ continues to make encryption and decryption queries, with one crucial restriction: **it cannot ask to decrypt c^* itself**.
7. $\mathscr{A}$ outputs a guess b'. $\mathscr{A}$ wins if $b' = b$.

This is a very strong definition. The adversary gets to see the decryptions of any ciphertext it wants, *except* the one it is trying to break.

5.2 A General Recipe: CPA + AUTH $\implies$ CCA

Proving CCA security directly is complex. A simpler "recipe" is to combine two properties: CPA security (which we know) and a new property, **Authenticity**.

Definition 22 (Authenticity (AUTH))**.** *A SKE scheme $\Pi = (Enc, Dec)$ has **ciphertext authenticity** if it is computationally infeasible for an adversary to produce a new, valid ciphertext that it did not see before. A ciphertext $\tilde{c}$ is **valid** if $Dec(k, \tilde{c}) \neq \perp$, where $\perp$ is a special error symbol indicating decryption failed.*

We formalize this with the $GAME^{auth}$:

Definition 23 (AUTH Security)**.** *A SKE Π is **AUTH-secure** if for any PPT adversary $\mathscr{A}$, $Pr[GAME^{auth}_{\Pi,\mathscr{A}}(\lambda) = 1] \leq negl(\lambda)$.*

The $GAME^{auth}$ proceeds as follows:

1. A key $k \leftarrow \mathscr{K}$ is generated.
2. $\mathscr{A}$ is given oracle access to $Enc(k, \cdot)$. Let $Q = \{(m_i, c_i)\}$ be the set of query/response pairs.
3. $\mathscr{A}$ outputs a new ciphertext $\tilde{c}$.
4. $\mathscr{A}$ wins if $\tilde{c} \notin \{c_1, \ldots, c_q\}$ (it's a new ciphertext) AND $Dec(k, \tilde{c}) \neq \perp$ (it's valid).

Theorem 17. *If a SKE scheme Π is both **CPA-secure** and **AUTH-secure**, then it is **CCA-secure**.*

Proof Sketch. We prove this by reduction. Assume, for contradiction, that an adversary $\mathscr{A}_{CCA}$ can break the CCA security of Π. We will show that this implies Π is either not CPA-secure or not $AUTH$-secure.

We construct a CPA adversary $\mathscr{A}_{CPA}$ that uses $\mathscr{A}_{CCA}$ as a subroutine. $\mathscr{A}_{CPA}$ will simulate the CCA game for $\mathscr{A}_{CCA}$.

- **Challenger:** $\mathscr{C}_{CPA}$ (the CPA challenger).
- **Reducer:** $\mathscr{A}_{CPA}$ (our reduction).
- **Adversary:** $\mathscr{A}_{CCA}$ (the adversary we assume exists).

The Simulation: $\mathscr{A}_{CPA}$ simulates the $GAME^{CCA}$ for $\mathscr{A}_{CCA}$:

1. **Encryption Queries:** When $\mathscr{A}_{CCA}$ asks to encrypt m_i, $\mathscr{A}_{CPA}$ passes m_i to its own CPA oracle, $\mathscr{C}_{CPA}$, and gets back c_i. $\mathscr{A}_{CPA}$ returns c_i to $\mathscr{A}_{CCA}$. $\mathscr{A}_{CPA}$ stores the pair (m_i, c_i).
2. **Challenge Query:** When $\mathscr{A}_{CCA}$ submits m_0^*, m_1^*, $\mathscr{A}_{CPA}$ passes them to $\mathscr{C}_{CPA}$ and gets back the challenge c^*. $\mathscr{A}_{CPA}$ gives c^* to $\mathscr{A}_{CCA}$.
3. **Decryption Queries:** When $\mathscr{A}_{CCA}$ asks to decrypt $\tilde{c}$, $\mathscr{A}_{CPA}$ must answer. It does not have a real decryption oracle.
   - **Rule 1:** If $\tilde{c} = c^*$, $\mathscr{A}_{CPA}$ returns $\perp$ (as per $GAME^{CCA}$ rules).
   - **Rule 2:** If $\tilde{c}$ is on its list of known ciphertexts (i.e., $\tilde{c} = c_i$ for some i), $\mathscr{A}_{CPA}$ returns the corresponding m_i.
   - **Rule 3:** If $\tilde{c}$ is new (not c^* and not c_i), $\mathscr{A}_{CPA}$ returns $\perp$.
4. **Guess:** $\mathscr{A}_{CCA}$ outputs a guess b'. $\mathscr{A}_{CPA}$ outputs the same b'.

Analysis of the Reduction: Let G_{CCA} be the real CCA game, and G' be the simulated game G' run by $\mathscr{A}_{CPA}$. The simulation in G' is perfect, *unless* Rule 3 fails. Rule 3 fails if $\mathscr{A}_{CCA}$ submits a new ciphertext $\tilde{c}$ for which the *real* answer is $Dec(k, \tilde{c}) = m' \neq \perp$. Let's call this the **BAD** event.

$$\text{BAD} \equiv \mathscr{A}_{CCA} \text{ queries } \tilde{c} \notin \{c_1, \ldots, c_q\} \text{ s.t. } Dec(k, \tilde{c}) \neq \perp$$

This is *exactly* the winning condition for the $GAME^{auth}$! Because Π is $AUTH$-secure, we know $Pr[BAD] \leq negl(\lambda)$.

By the Difference Lemma, the adversary's advantage can only differ by at most $Pr[BAD]$:

$$|Adv_{G_{CCA}}(\mathscr{A}_{CCA}) - Adv_{G'}(\mathscr{A}_{CCA})| \leq Pr[BAD] \leq negl(\lambda)$$

So, $Adv_{G_{CCA}}(\mathscr{A}_{CCA}) \approx Adv_{G'}(\mathscr{A}_{CCA})$.

Now, look at the game G'. In G', the decryption oracle is useless: it either returns $\perp$ or information $\mathscr{A}_{CCA}$ already knew (the m_i for a c_i). It provides no new information. Therefore, the game G' is effectively just the CPA game.

$$Adv_{G'}(\mathscr{A}_{CCA}) = Adv_{CPA}(\mathscr{A}_{CPA})$$

Because Π is CPA-secure, we know $Adv_{CPA}(\mathscr{A}_{CPA}) \leq negl(\lambda)$.

By the triangle inequality:

$$Adv_{G_{CCA}}(\mathscr{A}_{CCA}) \approx Adv_{G'}(\mathscr{A}_{CCA}) = Adv_{CPA}(\mathscr{A}_{CPA}) \leq negl(\lambda)$$

Therefore, the advantage of $\mathscr{A}_{CCA}$ in the real CCA game must be negligible. □

5.3 Constructions for Authenticated Encryption

We now know that $CCA \iff CPA + AUTH$. How do we build a scheme that is both? The idea is to combine a CPA-secure SKE (let's call it Enc) and a UF-CMA MAC (let's call it Tag). We need two independent keys, k_e for Enc and k_m for Tag.

There are three main ways to combine them:

5.3.1 Method 1: Encrypt-and-MAC (E&M)

We encrypt and MAC the *plaintext* in parallel.

$$c' = (c, \tau) \quad \text{where} \quad c \leftarrow Enc(k_e, m) \quad \text{and} \quad \tau \leftarrow Tag(k_m, m)$$

This is not secure. It fails to provide CPA security. **Why?** The Tag function is applied to m. If the Tag function is deterministic, an adversary can check if two plaintexts are equal. Enc is randomized to prevent this, but τ leaks the information. For example, if $\mathscr{A}$ encrypts m_0 and m_1 and sees (c_0, τ_0) and (c_1, τ_1), if $\tau_0 = \tau_1$, it knows $m_0 = m_1$. This breaks the CPA game.

5.3.2 Method 2: MAC-then-Encrypt (M&E)

We MAC the plaintext, then encrypt *both*.

$$c' \leftarrow Enc(k_e, m||\tau) \quad \text{where} \quad \tau \leftarrow Tag(k_m, m)$$

This is not secure. This construction can fail $AUTH$. **Why?** The Enc scheme only guarantees privacy (CPA), not integrity. An SKE scheme might be malleable. For example, consider a bad SKE: $Enc(k, m||\tau) = (Enc(k, m), \tau)$. This is CPA-secure. But the Dec function, $Dec(k, (c, \tau'))$, might just decrypt c and ignore τ'. An attacker could flip bits in τ', and the ciphertext would still decrypt to a valid m, failing the $AUTH$ property.

5.3.3 Method 3: Encrypt-then-MAC (EtM)

We encrypt the plaintext first, then MAC the *ciphertext*. This is the secure method.

- $Enc'((k_e, k_m), m)$:
  1. $c \leftarrow Enc(k_e, m)$
  2. $\tau \leftarrow Tag(k_m, c)$
  3. Output $c' = (c, \tau)$
- $Dec'((k_e, k_m), c' = (c, \tau))$:
  1. If $Vrfy(k_m, c, \tau) \neq$ accept, output $\perp$.
  2. Else, $m \leftarrow Dec(k_e, c)$.
  3. Output m.

Theorem 18. *The **Encrypt-then-MAC (EtM)** construction is CCA-secure, provided that Π_{SKE} is CPA-secure and Π_{MAC} is a $UF-CMA$ MAC (with unique/deterministic tags).*

Proof Sketch. We prove $CPA + AUTH$, which implies CCA.

1. **Proof of CPA Security:** We build a reduction $\mathscr{A}_{CPA-SKE}$ that breaks the CPA security of
   - $\mathscr{A}_{CPA-SKE}$ (the reducer) gets a CPA oracle from its challenger.
   - To simulate the EtM game, $\mathscr{A}_{CPA-SKE}$ needs to produce tags. It picks its own MAC key k_m.
   - **Encryption Query m:** $\mathscr{A}_{CPA-SKE}$ gets c from its oracle, computes $\tau = Tag(k_m, c)$ itself, and returns (c, τ).
   - **Challenge Query m_0^*, m_1^*:** $\mathscr{A}_{CPA-SKE}$ passes this up to its oracle, gets back c^*, computes $\tau^* = Tag(k_m, c^*)$, and returns (c^*, τ^*).
   - **Guess:** $\mathscr{A}_{CPA-SKE}$ outputs whatever $\mathscr{A}_{CPA-EtM}$ outputs.
   - **Analysis:** This simulation is perfect. The tag τ is computed on an already-encrypted ciphertext c, so it leaks no information about m that c didn't already leak. The advantage is identical. $Adv_{CPA-EtM} = Adv_{CPA-SKE} \leq negl(\lambda)$.
2. **Proof of $AUTH$ Security:** We build a reduction $\mathscr{A}_{UF-CMA}$ that breaks the $UF-CMA$ security of Tag if an adversary $\mathscr{A}_{AUTH}$ breaks EtM.
   - $\mathscr{A}_{UF-CMA}$ (the reducer) gets a Tag oracle from its challenger.
   - To simulate the EtM game, $\mathscr{A}_{UF-CMA}$ needs to encrypt. It picks its own SKE key k_e.
   - **Encryption Query m:** $\mathscr{A}_{UF-CMA}$ computes $c = Enc(k_e, m)$ itself. It then queries its Tag oracle $\mathscr{O}_{Tag}(c)$ to get τ. It returns (c, τ).
   - **Adversary Forgery:** $\mathscr{A}_{AUTH}$ outputs a forgery $(\tilde{c}, \tilde{\tau})$.
   - **Reducer Forgery:** $\mathscr{A}_{UF-CMA}$ outputs $(\tilde{c}, \tilde{\tau})$ as its own forgery.
   - **Analysis:**
     - $\mathscr{A}_{AUTH}$ wins if $(\tilde{c}, \tilde{\tau})$ is new AND $Dec(k_e, \tilde{c}) \neq \perp$.
     - $\mathscr{A}_{UF-CMA}$ wins if $Vrfy(k_m, \tilde{c}) = \tilde{\tau}$ AND $\tilde{c}$ is new (assuming standard UF-CMA, since $\tilde{c}$ is the "message" for the MAC).
   - The Dec' algorithm for EtM *first* checks the tag. $Dec(k_e, \tilde{c}) \neq \perp$ can only happen if $Vrfy(k_m, \tilde{c}, \tilde{\tau}) =$ accept.
   - We just need to check that the "newness" conditions align.
   - $\mathscr{A}_{AUTH}$ queries $m_i \rightarrow (c_i, \tau_i)$. $\mathscr{A}_{UF-CMA}$ queries $c_i \rightarrow \tau_i$.
   - $\mathscr{A}_{AUTH}$ outputs $(\tilde{c}, \tilde{\tau}) \notin \{(c_i, \tau_i)\}$.

- If the Tag is deterministic ("UNIQUE TAGS"), then it's impossible to have $\tilde{c} = c_i$ but $\tilde{\tau} \neq \tau_i$ and $Vrfy = \text{accept}$.
- Therefore, any valid forgery from $\mathscr{A}_{AUTH}$ *must* have $\tilde{c} \notin \{c_i\}$.
- This is exactly the winning condition for $\mathscr{A}_{UF-CMA}$!
- Thus, $Pr[\mathscr{A}_{AUTH} \text{ wins}] \leq Pr[\mathscr{A}_{UF-CMA} \text{ wins}] \leq negl(\lambda)$.

Since EtM is both CPA-secure and $AUTH$-secure, it is CCA-secure. □