Contents

Lesson 4

4.1 Negligible function

What is exactly a negligible function ?
Below here there is a possible interpretation of this notion:
"In real life, we can just consider adversaries with limited computational power; even if every non-perfectly secure authentication scheme can resist to unbounded computational power, the true unbounded computational power doesn't exist at all. So, it's reasonable to consider just "bounded" adversaries.
So consider a scheme π where the only attack against it is brute force attack. We consider π to be secure if it cannot be broken by a brute force attack in polynomial time.
The idea of **negligible probability** encompasses this exact notion. In π , let's say that we have a polynomial-bounded adversary.
Brute force attack is not an option.
But instead of brute force, the adversary can guess (a polynomial number of) random values and hope to chance upon the right one. In this case, we define security using negligible functions: The probability of success has to be smaller than the reciprocal of any polynomial function.
And this makes a lot of sense: If the success probability for an individual guess is a reciprocal of a polynomial function, then the adversary can try a polynomial amount of guesses and succeed with high probability. In sum then, if the overall success rate is $\frac{1}{poly(\lambda)}$ then we consider this a feasible attack and the scheme is insecure.
So, we require that the success probability must be less than the reciprocal of every polynomial function. This way, even if the adversary tries $poly(\lambda)$ guesses, it will not be significant since it will only have tried: $\frac{poly(\lambda)}{superpoly(\lambda)}$[1] As λ grows, the denominator grows far faster than the numerator and the success probability will not be significant." [2]

A Negligible function $\nu : \mathbb{N} \to [0, 1]$ is s.t. $\forall p(\lambda) \in poly(\lambda)$ then $\nu(\lambda) \in \mathcal{O}(\frac{1}{p(\lambda})$

Exercise 1. Let $p(\lambda), p(\lambda') \in poly(y)$ and $\nu(\lambda), \nu'(\lambda) \in \text{negl}\left(() \lambda\right)$.
Then prove

$$p(\lambda) * p'(\lambda) \in \text{poly}(\lambda) \tag{4.1}$$

[1]If we design a function hard for $superpoly(\lambda)$ possible attempts and the attacker completed $poly(\lambda)$ attempts, he has just $\mathcal{P}[\frac{poly(\lambda)}{superpoly(\lambda)}]$ of founding the key to break the scheme

[2]https://crypto.stackexchange.com/questions/5832/what-exactly-is-a-negligible-and-non-negligible-function

$$\nu(\lambda) + \nu'(\lambda) \in \text{negl}\left(()\,\lambda\right) \tag{4.2}$$

Solution 1 (4.2)**.** We need to show that for any $c \in \mathbb{N}$, we can find n_0 such that $\forall n > n_0$, $h(n) \leq n^{-c}$. So, consider an arbitrary $c \in \mathbb{N}$. Then, since $c + 1 \in \mathbb{N}$, and since f and g are negligible, there exists n_f and n_g such that:$\forall n \geq n_f$, $f(n) \leq n^{-(c+1)}$ and $\forall n \geq n_g$, $g(n) \leq n^{-(c+1)}$.

Choose $n_0 = max(n_f, n_g)$.
Then, $\forall n \geq n_0$ we have $h(n) = f(n) + g(n) \leq n^{-(c+1)} + n^{-(c+1)} = 2n^{-(c+1)}$ (since $n \geq n_0 \geq 2$). Thus $h(n) \leq n^{-c}$ and $h(n)$ is negligible.

4.2 ONE WAY FUNCTION

A OWF is a function "hard to invert".

Definition 1. The function

$$f : \{0,1\}^{n(\lambda)} \to 0, 1^{n(\lambda)} \tag{4.3}$$

is a OWF, if

$$\forall PPT \mathcal{A} \exists \nu(\lambda) \in \text{negl}\,(x) \tag{4.4}$$

such that

$$\mathcal{P}[GAME^{owf}_{f,\mathcal{A}}(\lambda) = 1] \leq \nu(\lambda) \tag{4.5}$$

◇

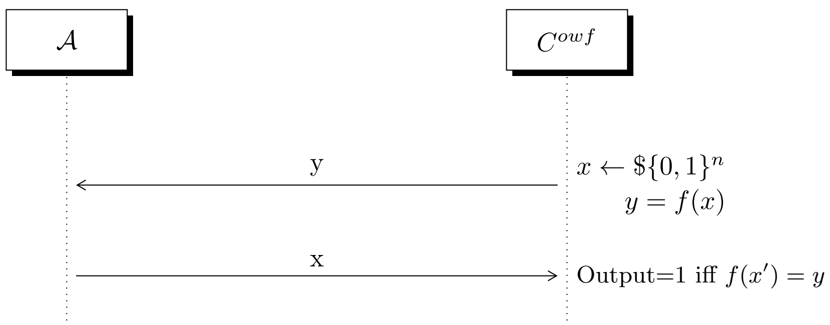


Figure 4.1: Game for OWF

Exercise 2. Show that $\exists \mathcal{A}$ inefficient and wins with probability 1 , and exists $\mathcal{A}$ efficient winning with probability 2^{-n} .

4.3 One way puzzle

A one-way function can be thought as a function which is very efficient in generating puzzles, and these puzzles are very hard to solve.
Furthermore, the person generating the puzzle knows a solution to it and can efficiently verify the validity of (possible other) solutions to the puzzle.

For a give couple (P_{GEN}, P_{VER}) of a puzzle generator and a puzzle verifier, we have :

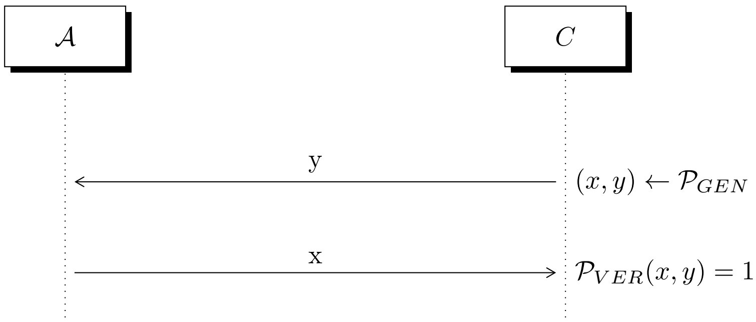


Figure 4.2: One way puzzle scheme

So, we can say that One-way Puzzle is a problem in NP(because solutions are easy to verify), while not in P(because a solution is hard to provide).

4.4 Impagliazzo Worlds

Suppose to have Gauss, a genius child, and his professor. The professor gives to Gauss some mathematical problems, and Gauss wants to solve them all.
Immagine now that, if using one-way functions, the problem is $f(x)$, and the solution to the problem is x.
According to Impagliazzo, we live in one of these possible worlds:

- Algorithmica , where P=NP , and all the problems easy to verify are also easy to solve, (so the Professor can try as hard as possible to break the scheme, but without success, since Gauss will solve them all using the verification procedure to compute the solution);
- Heuristica, NP problems are hard to solve in the worst case but easy on average , (so the professor , with some effort, can create a game difficult enough, but Gauss will solve it anyway; here there are some problems that the professor cannot find a solution to);
- Pessiland, NP problems are hard on average but no one-way functions exist
- Minicrypt , OWF exist but we don not have public-key cryptography,
- Cryptomania, public-key cryptography is possible, i.e. two parties can exchange secret messages over open channels.

4.5 Computational Indistinguishability

Distribution ensemble $X = X_{\lambda \in \mathbb{N}}$ and $Y = Y_{\lambda \in \mathbb{N}}$ are a sequence of distributions.

Definition 2. X and Y are computationally **indistinguishable** ($X \approx_c Y$) if $\forall . PPT.D, \exists \nu(\lambda) \in \text{negl}(\lambda)$ such that

$$|\mathcal{P}[D(X_\lambda) = 1] - \mathcal{P}[D(Y_\lambda) = 1]| \leq \nu(\lambda) \tag{4.6}$$

◇

Suppose we have this mental game: a Distinguisher D receives the value z. This value has been chosen by me , the Challenger, among X_λ and Y_λ , and the Distinguisher has to *distinguish* which was the source of z. What does this formula mean?
This formula means that, fixed 1 as one of the sources, the *probability* that D says "1!" when I pick z from X_λ is **not so far** from the *probability* that D says "1!" when I pick z from Y_λ .

So, this means that, when this property is verified by two random variables, there isn't too much *difference* between the two variables in terms of exposed information (reachable by D), otherwise the distance between the two probabilities should be much more than a *negligible* quantity .
What's the deep meaning of this formula? This is something to do.

Lemma 1. *If $X \approx_c Y$ then $\forall PPT.f$ we have $f(x) \approx_c f(y)$.* ◇

by contraddiction. We want to show that $f(x) \approx_c f(y)$. So, let's suppose this property is not true.
Assume $\exists PPT.f, D'$ and some $p'(\lambda) \in \text{poly}\lambda$ such that

$$|\mathcal{P}[D'(f(x)) = 1] - \mathcal{P}[\mathcal{D}'(f(y)) = 1]| > \frac{1}{p'(\lambda)} \tag{4.7}$$

.

So D' which can distinguish $f(x)$ and $f(y)$:

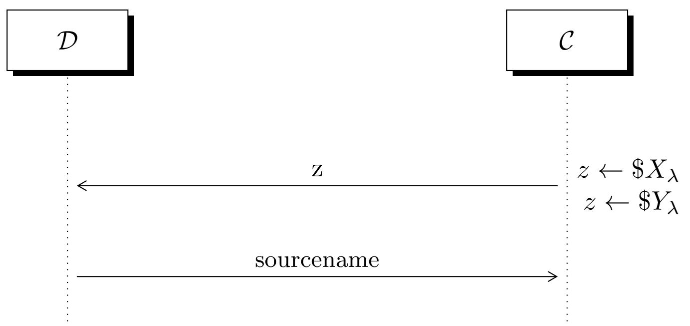


Figure 4.3: Distinguisher of $f(x) \approx_c f(y)$

. But , if this kind of distinguisher would exist, we could use this distinguisher to distinguish$X$ and Y .

We can build something like this:

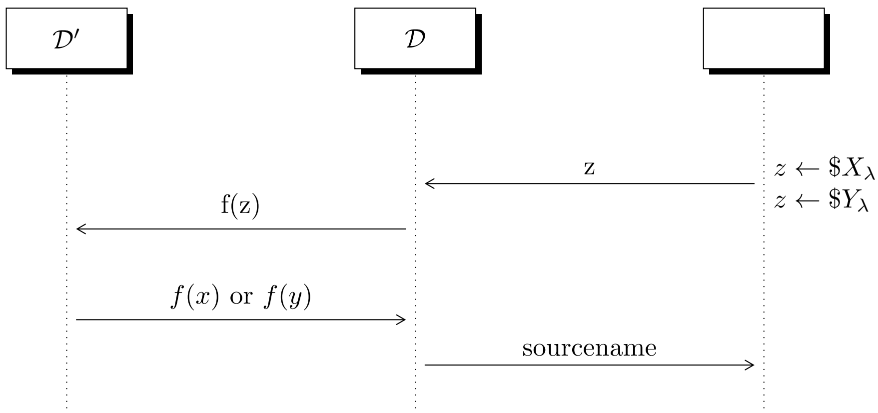


Figure 4.4: The reduction done

So, if D' could distinguish between f(x) and f(y), this means that its output can be used to distuinguish also the main problem in polinomial time (since D' is a PPT), the distinction about X_λ and Y_λ.

$\square$

4.6 Pseudorandom Generator (PRG)

A deterministic function $G : \{0,1\}^\lambda \rightarrow \{0,1\}^{\lambda+l(\lambda)}$ is a PRG if :

- G is polinomial time, so it runs in polinomial time
- $|G(s)| = \lambda + l(\lambda)$
- $G(U_\lambda) \approx_c U_{\lambda+l(\lambda)}$

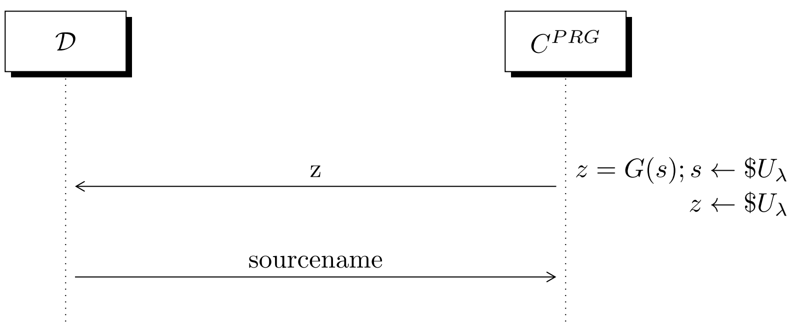


Figure 4.5: Pseudorandom generator game

So, if we take $s \leftarrow \$U_\lambda$, the output of G will be indistinguishable from a random draw from U_λ.

Lesson 5

4.7 Stretching a PRG

Theorem 3. *If there exists a PRG* $G : \{0,1\}^\lambda \to \{0,1\}^{\lambda+1}$ *, then* $\forall l(\lambda) \in \text{poly}\lambda$ *there exists a PRG with stretch* $G : \{0,1\}^\lambda \to \{0,1\}^{\lambda+l(\lambda)}$ $\diamond$

Proof. So, consider this algorithm/construction:

1. Let $s_0 \leftarrow \$\{0,1\}^\lambda$
2. $\forall i \in [l]$, let $(s_i, b_i) = G(s_{i-1})$
3. Output $b_1, b_2, ..., b_l, s_l$, so the output is a string of bit $\lambda + l(\lambda)$ long

.

Following this algorithm, we should obtain the following representation of G^l: .

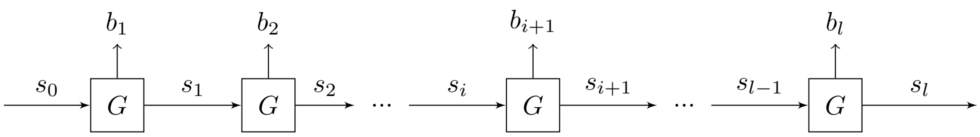


Figure 4.6: graphical representation of G^l

To show that the theorem is valid, we will use a proof by contraddiction: trying to show that G^λ is not a PRG, we will see that G should be not a PRG, contraddicting the theorem.

Different sources give different interpretations of this proof. As of now , the following one is an outline of the proof that seems to make sense, more or less. This proof should , however, be marked as TO BE REVIEWED.

The step points of this proof are:

1. Prove that $H_\lambda^i \approx_c H_\lambda^{i+1}$, $\forall i \in [0, l]$;
2. Prove the **hybrid argument**: if $X \approx_c Y$ and $Y \approx_c Z$, then $X \approx_c Z$;
3. with the hybrid argument, prove that

$$G^l(U_\lambda) = H_\lambda^0 \approx_c H_\lambda^1 \approx_c ... \approx_c H_\lambda^l = U_{l+\lambda}$$

4. now, since $H^i_\lambda \approx_c U_{\lambda+l}$, it's possible to use the contraddiction (masked as a proof by reduction)

.

To prove point **??**, define the following names:

- $H^0_\lambda := G^l(U_\lambda)$
-
$$H^i_\lambda := \begin{cases} b_1, ..., b_i \leftarrow \$\{0,1\} \\ s_i \leftarrow \{0,1\}^\lambda \\ (b_{i+1}, ..., b_l, s_l) := G^{l-i}(s_i) \end{cases}$$
- $H^l_\lambda := U_{\lambda+l}$

.

So H^i is just taking in input the number x_i , executing $l - i$ times G and obtaining a sequence of bytes.

Now, just have a look at those figures:

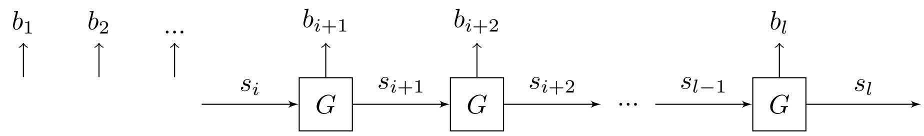


Figure 4.7: H^i_λ

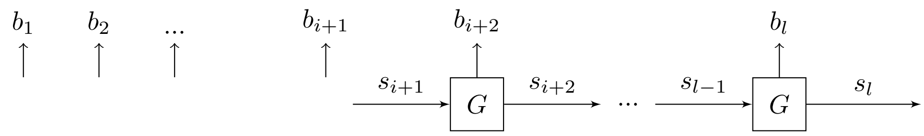


Figure 4.8: H^{i+1}_λ

.

H^i and H^{i+1} differ just for the input given to the $(i + 1)$-th step of the algorithm:

- in H^i, this input is pseudorandom;
- in H^{i+1}, this input comes from U_λ

.

Now consider a function [3]

$$f_i(s_{i+1}, b_{i+1}) = \left\{ \begin{array}{c} b_1, ..., b_i \leftarrow \$\{0,1\} \\ \forall j \in \{i+2, l\}, G(s_{j-1}) = \{s_j, b_j\} \\ \text{output} := \{b_1, ..., b_l, s_l\} \end{array} \right.$$

[3] Why do we define f_i? Such that we know that the first input given to G in the function will be considered s_{i+1}.

Given this function , it's possible to notice that:

- $f_i(U_{\lambda+1})$ has the same distribution of H^{i+l}
- $f_i(G(U_\lambda))$ has the same distribution of H^i

.
Given **??**, since by definition

$$G(U_\lambda) \approx_c U_{\lambda+1}$$

then also

$$f_i(G(U_\lambda)) \approx_c f_i(U_{\lambda+1})$$

and so $H^i \approx_c H^{i+1}$.

Now ,to prove **??** :

$$X \approx_c Z \Rightarrow \tag{4.8}$$
$$|\mathcal{P}[D(X) = 1] - \mathcal{P}[D(Z) = 1]| \leq \nu(\lambda) \tag{4.9}$$
$$|\mathcal{P}[D(X) = 1] - \mathcal{P}[D(Y) = 1] + \mathcal{P}[D(Y) = 1] - \mathcal{P}[D(Z) = 1]| \leq \tag{4.10}$$
$$\leq |\mathcal{P}[D(X) = 1] - \mathcal{P}[D(Y) = 1]| + |\mathcal{P}[D(Y) = 1] - \mathcal{P}[D(Z) = 1]| \tag{4.11}$$
$$\leq \nu(\lambda) + \nu(\lambda) = \nu(\lambda) \tag{4.12}$$

Now, to prove **??**, it's just needed to notice that

$$H^i_\lambda \approx_c ... \approx_c H^{l-1}_\lambda \approx_c H^l_\lambda \approx_c U_{l+\lambda} \tag{4.13}$$

for what's valid in point **??** and point **??**.

Now, use a contraddiction.
Suppose G^l is not a a PRG $\Rightarrow$

$$G^l(U_\lambda) \not\approx_c U_{\lambda+l} = H^l \not\approx_c H^0 \Rightarrow$$
$$\exists i \in [0, l], \exists PPT.D', p'(\lambda) \in \text{poly}\lambda$$
$$|\mathcal{P}[D'(H^i) = 1] - \mathcal{P}[D'(H^{i+l}) = 1]| \geq \frac{1}{p'(\lambda)}$$

.
This formula comes from observing that, since $H^l \not\approx_c H^0$, there must be a point in the chain $H^0 \approx_c H^1 \approx_c ... \approx_c H^l$ where $H^i \not\approx_c H^{i+1}$; so there exist D' capable of distinguish them.

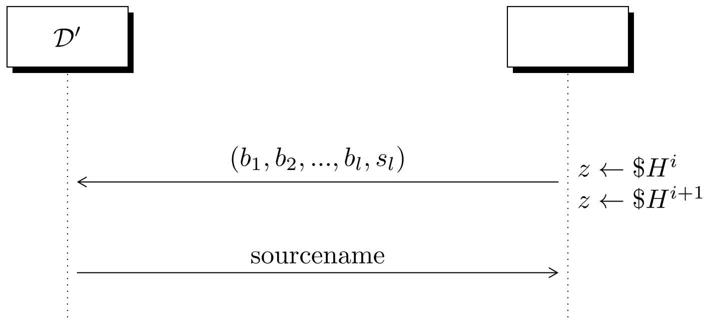


Figure 4.9: Distinguisher for H^i and H^{i+1}

If such a distinguisher exists, it can be also used to distinguish the output of funciton G from $U_{\lambda+1}$:

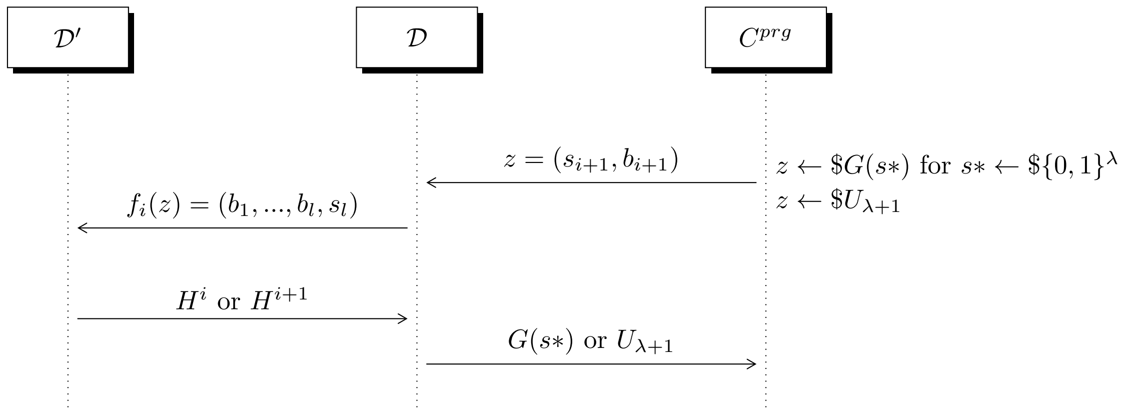


Figure 4.10: If (s_{i+1}, b_{i+1}) comes from $G(s*)$,D' finds H^i, otherwise it finds H^{i+1}

So we have a contraddiction, because we cannot distinguish a PRG , by definition. □

4.8 Hard-core predicate

Now , consider a tipical one-way function f, s.t. $f(x) = y$.

Question 1. Which bits of the input x are hard to compute given $y = f(x)$? Is always true that , given f and $f(x)$, the first bit is hard to compute for every x?

Example 1. Given an OWF f, then $f'(x) = x[0]||f(x)$ is a OWF.

Definition 1. A polinomial time function $h : \{0,1\}^n \to \{0,1\}$ is **hard core** for $f : \{0,1\}^n \to \{0,1\}^n$ if

$$\forall.PPT.A, \exists \nu(\lambda) \in \text{negl}(\lambda) \text{ such that}$$
$$\mathcal{P}[A(f(x)) = h(x) | x \leftarrow \$\{0,1\}^n] \leq \nu(\lambda)$$

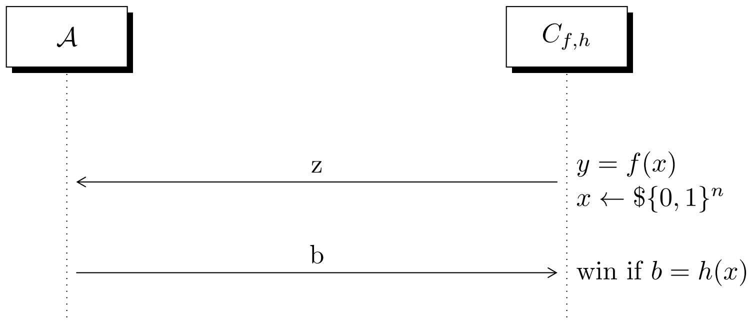


Figure 4.11: Hard-core function, game of definition 1

There is also an alternative definition:

Definition 2. A Polinomial Time function $h : \{0,1\}^n \to \{0,1\}$ is hard-core for f if

$$(f(x), h(x)) \approx_c (f(x), b)$$

where $x \leftarrow \$\{0,1\}^n$ and $b \leftarrow \$\{0,1\}$.

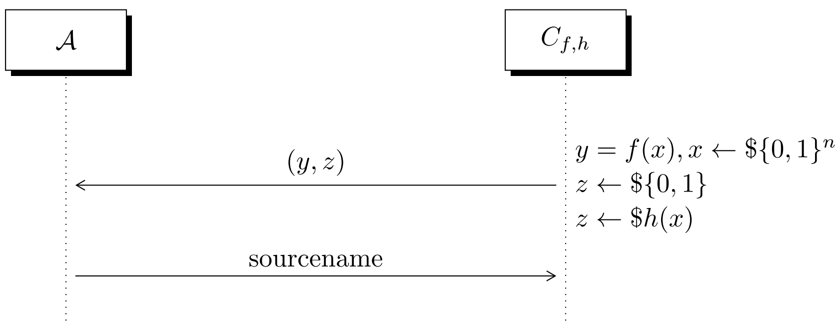


Figure 4.12: Hard-core function, game of definition 2

Claim 1. There is no *universal* hard-core function h.

A good h should be chosen for each different one-way function f.
Immagine h that works for all of the OWFs.
What about $f'(x) = h(x)||f(x)$? If h is hardcore for f and f', by the definition 1 of **hardcore function** h is applied on the same x and will return the same bit in $\{0,1\}$ at every interrogation.
TO BE REVIEWED.

Theorem 4 (Goldreich-Levin, '99)**.** *Let f be an OWF and consider $g(x,r) = (f(x), r)$ for $r \in \{0,1\}^n$. Then g is a OWF and*

$$h(x,r) =< x, r >= \sum_{i\text{-}th\ bit} x_i r_i mod 2 = ...$$

==
TO BE COMPLETED
==
is hard core for g. ◇

Exercise 5. Prove that g is OWF if f is OWF. (by reduction)

4.9 One Way Permutation

$f : \{0,1\}^n \rightarrow \{0,1\}^n$ is an OWF and

$$\forall x, |x| = |f(x)| \wedge x \neq x' \Rightarrow f(x) \neq f(x')$$

Corollary 1. If $f : \{0,1\}^n \rightarrow \{0,1\}^n$ is a OWP, then for $g(), h()$ as in the GL theorem,

$$G(s) = (g(s), h(s))$$

is a PRG.

Proof. By GL , if f is an OWP, so is g. This means that if we want to invert g, since g depends on f we have to invert a OWP.
Moreover h is hardcore for g. Hence

$$G(U_{2n}) \equiv (g(U_{2n}), h(U_{2n})) \equiv \underbrace{(f(U_n), U_n, h(U_{2n})) \approx_c (f(U_n), U_n, U_1)}_{\text{definition 1 of hard core pred.}} \equiv U_{2n+1}$$

$\square$

We are stretching just 1 bit, but we know we can stretch more than one.

Lesson 6

4.10 Computationally secure encryption

Question 2. How to define the concept of **computationally secure encryption** ?

Find a task/scheme that is computationally hard for an attacker to break (supposing the attacker is polyλ, we want a scheme which requires an amount of time near ,as much as possible, to *superpoly*(λ) to be broken).

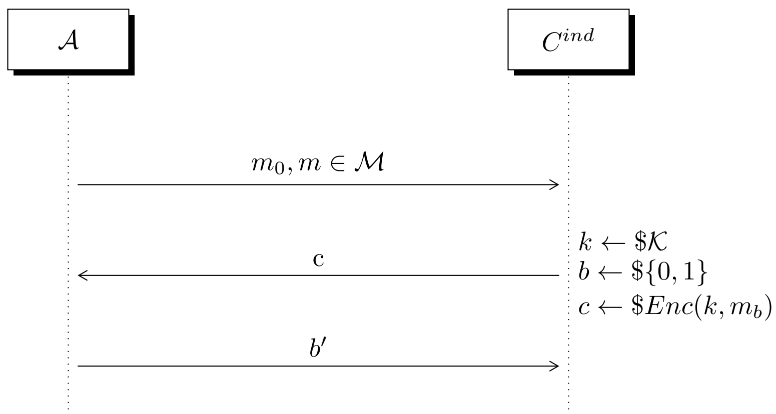
This scheme should have these properties:

- **one wayness** w.r.t. key (given $c = Enc(k, m)$, it should be hard to recover k)
- **one wayness** w.r.t. message (given $c = Enc(k, m)$, hard to obtain the message)
- no **information leakage** about the message

.

Consider the following experiment for $\Pi = (Enc, Dec)$, named

$$GAME^{ind}_{\Pi,\mathcal{A}}(\lambda, b)$$



In the image b' means that if $m_{b'} = m_b$ the adversary wins.

Definition 3. We say that Π is computationally **one time secure** if

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

or, alternatively $\forall.PPT.\mathcal{A}\exists\nu(\lambda) \in \text{negl}(\lambda)$

$$|\mathcal{P}[Game^{ind}_{\Pi,\mathcal{A}}(\lambda, 0) = 1] - \mathcal{P}[Game^{ind}_{\Pi,\mathcal{A}}(\lambda, 1) = 1]| \leq \nu(\lambda)$$

4

$\diamond$

This last definition is compliant with the three properties before exposed, in particular:

if a scheme is **one time secure** $\Rightarrow$ the scheme has each one of these 3 properties

- **compliance with point 1** : suppose point 1 is not valid, and k is not hard to discover for $\mathcal{A}$. But then $\mathcal{A}$ is able to perfectly distinguish m and m_0 with $\mathcal{P}[1]$ every time, and so the scheme couldn't be one time secure;
- **compliance with point 2** : suppose point 2 is not valid, and then the encrypted message can be easily discovered by $\mathcal{A}$. But then , as before, $\mathcal{A}$ can win every game with $\mathcal{P}[1]$, so the scheme couldn't be one time secure;
- **compliance with point 3** : suppose point 3 is not valid, and some information about m is leaked in c, for example the first bit of c is the same bit of m . $\mathcal{A}$ could forge $m_0 == m$ such that they have the same bits but just the first is different. When $\mathcal{A}$ obtains c, he can look at the first bit and distinguish which was the message encrypted. Thus, the scheme wouldn't be one time secure. TO BE REVIEWED.

What is not **two time secure** ?

Construction 1. Let $G : \{0,1\}^\lambda \to \{0,1\}^l$. Consider the following schema $\Pi_\oplus$:

- $\mathcal{K} = \{0,1\}^\lambda \Rightarrow k \leftarrow \$\{0,1\}^\lambda$
- $Enc(k, m) = G(k) \oplus m$, $m \in \{0,1\}^l$
- $Dec(k, c) = c \oplus G(k) = m$

$\diamond$

This construction isn't 2-time secure.Assume the pair

$$(\bar{m}, \bar{c} = G(k) \oplus \bar{m})$$

is known. Now , given $c = G(k) \oplus m$, where c and m are unknown, we can force the schema and do the following

$$\bar{c} = G(k) \oplus m = c \oplus m \oplus \bar{m} \Rightarrow c \oplus \bar{c} = m \oplus \bar{m}$$

and obtain m.

Theorem 6. *If G is a PRG, then $\Pi_\oplus$ is computationally one-time secure* $\diamond$

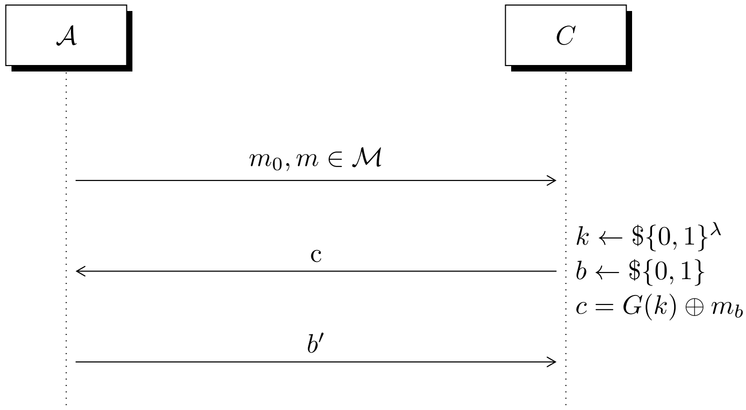


Figure 4.13: Game for $\Pi_\oplus$ schema ($Game_{\Pi_\oplus,\mathcal{A}}$)

Proof. We need to show that

$$Game^{ind}_{\Pi_\oplus,\mathcal{A}}(\lambda, 0) \approx_c Game^{ind}_{\Pi_\oplus,\mathcal{A}}(\lambda, 1)$$

So first consider the following **hybrid game** :

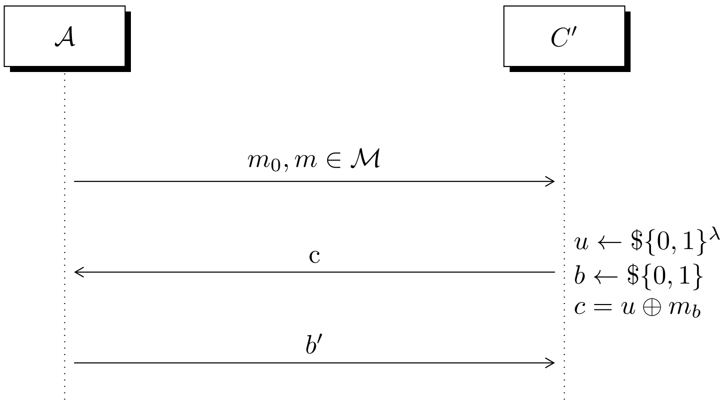


Figure 4.14: Hybrid game ($\mathcal{HYB}_{\Pi_\oplus,\mathcal{A}}(\lambda, b)$)

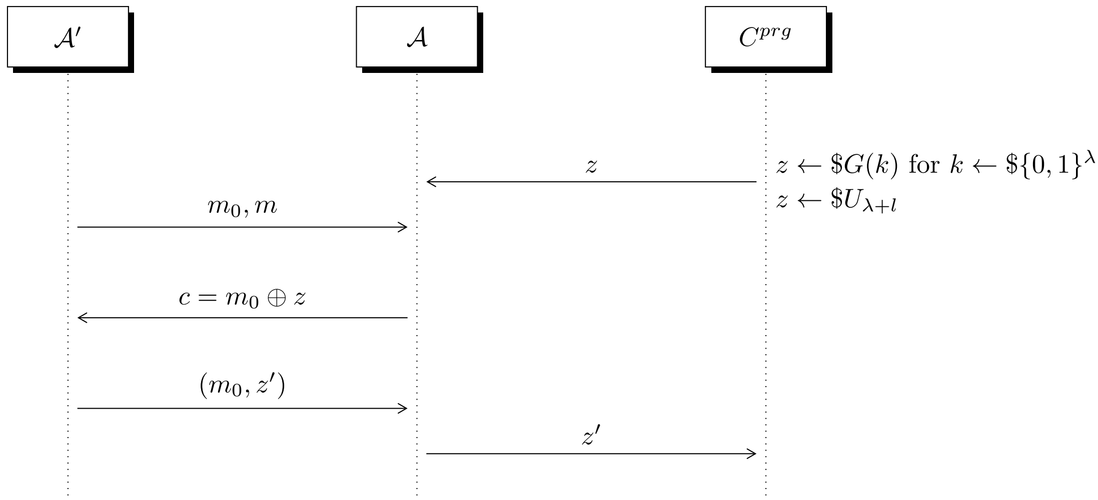
Lemma 2. $\mathcal{HYB}_{\Pi_\oplus,\mathcal{A}}(\lambda, 0) \equiv \mathcal{HYB}_{\Pi_\oplus,\mathcal{A}}(\lambda, 1)$ ◇

This is true because distribution of c doesn't depend on $b \in \{0, 1\}$.

Lemma 3. $\forall b \in \{0, 1\}, \mathcal{HYB}_{\Pi_\oplus,\mathcal{A}}(\lambda, b) \approx_c Game^{ind}_{\Pi,\mathcal{A}}(\lambda, b)$ ◇

Proof. Simple reduction to PRG, supposing that the statement isn't true.
This means that there exists $\mathcal{A}'$ capable of distinguish $c = m_b \oplus G(k)$ and $c = m_b \oplus u$.
We will prove this first with $b = 0$ (the other case is the same).

[4]$Game^{ind}$ refers to the indistinguishability of the messages sent by the attacker during the game



If $\mathcal{A}'$ could distinguish these two sources, then C^{prg} could be distinguished, but this is impossible. □

Now , for the two lemmas just seen, we have

$$Game^{ind}_{\Pi,\mathcal{A}}(\lambda, 0) \approx_c \mathcal{HYB}_{\Pi_\oplus,\mathcal{A}}(\lambda, 0) \equiv \mathcal{HYB}_{\Pi_\oplus,\mathcal{A}}(\lambda, 1) \approx_c Game^{ind}_{\Pi,\mathcal{A}}(\lambda, 1)$$

□

4.11 Pseudorandom functions

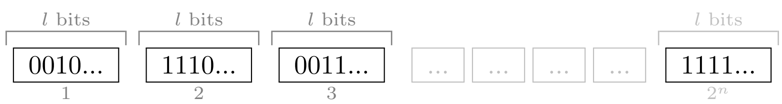
A random function

$$R : \{0,1\}^n \to \{0,1\}^l$$

. is a function that takes in input x and :

- returns a new $R(x) = y \leftarrow \$\{0,1\}^l$ if x has never been saw before and records that value (so ***two distinct inputs can collide***)
- returns the recorded $R(x)$ otherwise

. We could generate these functions, but they occupy too much space: supposing all the possible outputs of R have been generated and stored in an array in memory, the occupied bits in memory are $2^n l$.



In particular, the family $\mathcal{R} = \{R : \{0,1\}^n \to \{0,1\}^l\}$, also indicated as $\mathcal{R}(\lambda, n, l)$, containing all the possible random functions has cardinality $2^{2^n l}$.

Intuition: a pseudo-random function is indistinguishable (computationally speaking) from a truly random one.

Call $\mathcal{F} = \{F_k : \{0,1\}^{n(\lambda)} \rightarrow \{0,1\}^{l(\lambda)}\}_{k \in \{0,1\}^\lambda}$ the family of pseudorandom functions with key k. To give a definition of pseudo-random function, consider the following games:

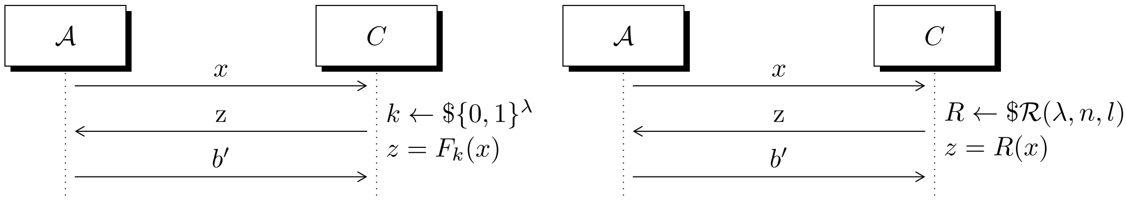


Figure 4.15: $Real_{\mathcal{F},\mathcal{A}}(\lambda)$ vs $Rand_{\mathcal{R},\mathcal{A}}(\lambda)$

where $b' \in \{0,1\}$ is a convention and 1 is assigned to *Real* or *Rand*; so in this game the adversary **recognizes** which machine he is talking with.

Definition 4. $\mathcal{F}$ is a PRF family if

$$Real_{\mathcal{F},\mathcal{A}}(\lambda) \approx_c Rand_{\mathcal{R},\mathcal{A}}(\lambda)$$

$\diamond$

Exercise 7. Show that no PRG is secure against **unbounded attackers** .

Exercise 8. Show the same (as above) for PRF.

4.11.1 GGM Tree

Construction 2. Let $G : \{0,1\}^\lambda \rightarrow \{0,1\}^{2\lambda}$ be a PRG and let us write

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

Now consider this tree, called **GGM tree**, which describes the use of $G(k)$:
GGM TREE IMAGE
Build $\mathcal{F} = \{F_k : \{0,1\}^n \rightarrow \{0,1\}^\lambda\}$ such that

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

$\diamond$

For example, in the tree with height $n = 3$, for $x = 001$ we have $F_k(001)$, which is $G_0(G_0(G_1(k)))$.

Lesson 7

WARNING : I was absent

Theorem 9. *If G is a PRG, then F_{GGM} is a PRF.* $\diamond$

(this definition should mean implicitly that the k key has been chosen by the challenger before the proof starts)

Proof. TO BE REVIEWED. Now use the induction on the height n of the GGM tree for $\nu(\lambda) \in \text{poly}\lambda$.

Base Case $\Rightarrow n = 1$ follows by security of PRG G function, because

$$(F_k(0), F_k(1)) = (G_0(k), G_1(k)) \approx_c U_{2\lambda}$$

This means that, chosen k, the two values returned by F are indistinguishable from 2 values taken at random and inserted in the truth table of a possible random function. Since these values are indistinguishables, the source functions are indistinguishables and F_k is pseudorandom.

Now, for the **Inductive step** :

Lemma 4. *Let $F' : \{0,1\}^{n-1} \to \{0,1\}^\lambda$ be a PRF. Now define $F_k(x, y) = G_x(F'_k(y))$ where $F_k : \{0,1\}^n \to \{0,1\}^\lambda$.*
If $\{F'_k\}$ is a PRF so is $\{F_k\}$. $\diamond$

Consider the following images:

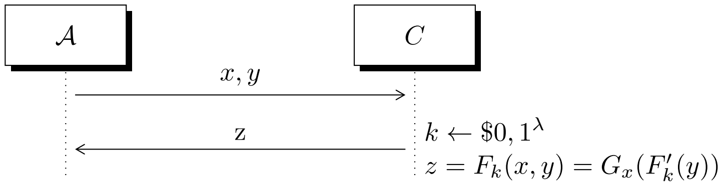


Figure 4.16: $\mathcal{HYB}^0_{\mathcal{F},\mathcal{A}}(\lambda)$

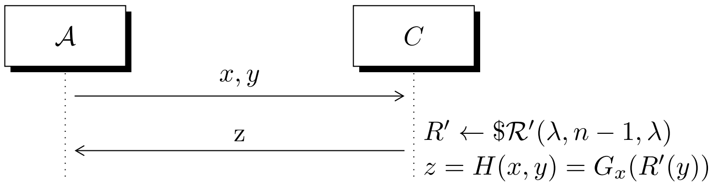


Figure 4.17: $\mathcal{HYB}^1_{\mathcal{R}',G,\mathcal{A}}(\lambda)$

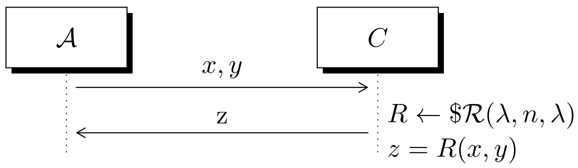
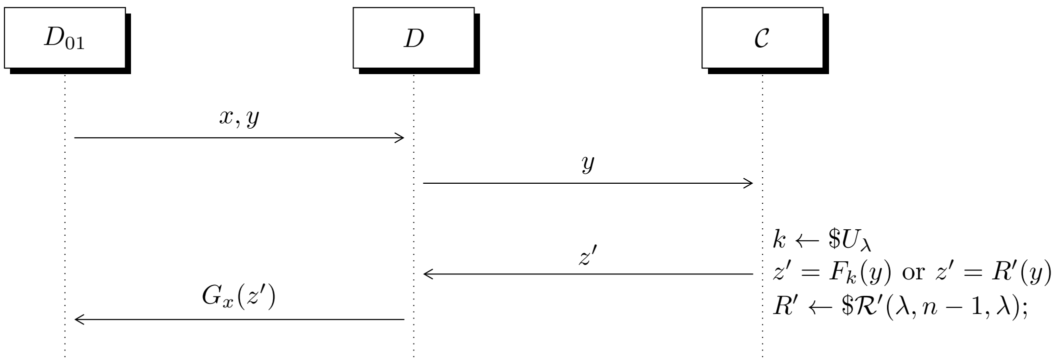


Figure 4.18: $\mathcal{HYB}^2_{\mathcal{R},\mathcal{A}}(\lambda)$

Lemma 5. $\mathcal{HYB}^0 \approx_c \mathcal{HYB}^1 \approx_c \mathcal{HYB}^2$. $\diamond$

Claim 2. $\mathcal{HYB}^0 \approx_c \mathcal{HYB}^1$

Assume $\exists . PPT . D_{01}$ that can distinguish F_k and H ; then there may exist a distinguisher D as in the image which breaks the assumption made by inductive step.



Claim 3. $\mathcal{HYB}^1 \approx_c \mathcal{HYB}^2$

For this proof we use the following simple lemma.

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

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

for $k_1, ..., k_t \leftarrow \$U_\lambda$

Assume that it exists a distinguisher $D_{1,2}$ which is capable of distinguish $H(x, y)$ and $R(x, y)$.

...TO REVIEW, NOT UNDERSTOOD AT ALL ... □

4.12 CPA SECURITY

Suppose G is a PRG.
Given $Enc(k, m) = G(k) \oplus m$, and a known $(\bar{m}, \bar{c})$ where $c = G(k) \oplus m$, then this function is not 2 time secure, since $c \oplus \bar{c} = m \oplus \bar{m}$, easy to invert.
(??? some glue arguments missing ???)

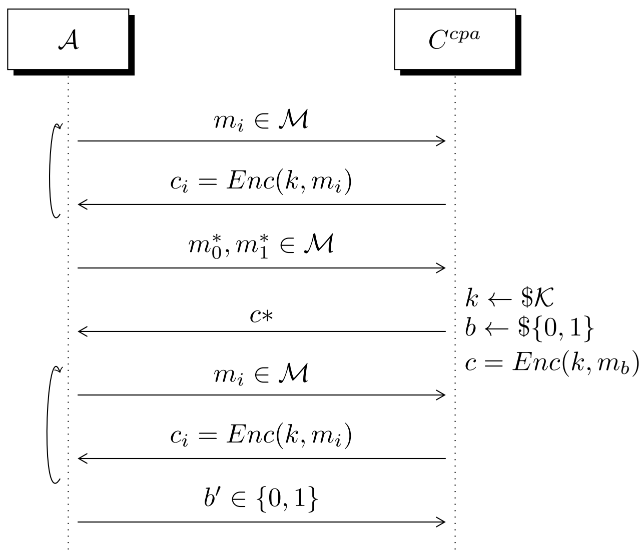


Figure 4.19: $Game^{cpa}_{\Pi,\mathcal{A}}(\lambda, b)$

The adversary wins if finds which m_b^* message was previously encrypted. For sure m_i can be equal to m_0 or m_1.
The two cyclic requests, before and after the starred messages, can be arbitrarily long (also 0, there is no constraint).

Definition 5. A scheme is CPA-secure if $Game^{cpa}_{\Pi,\mathcal{A}}(\lambda, 0) \approx_c Game^{cpa}_{\Pi,\mathcal{A}}(\lambda, 1)$ ◇

Observation 1. *No deterministic scheme can achieve CPA security.* ◇

This is because the Adversary is not limited in what he can ask to the Challenger: if he asks m_0 and m_1 before sending the starred messages, he will know in advance the encrypted form of the messages, and he will be able to distinguish with $\mathcal{P}[Game^{cpa}] = 1$ the two games.

So, a way to obtain a CPA-secure encryption scheme consists of returning different cyphertexts for the same message, and we can build a scheme which tries to generate this kind of output using PRFs.

Consider the following SKE scheme Π.
Let $\mathcal{F} = \{F_k : \{0,1\}^n \to \{0,1\}^l\}$ be a PRF:

- $k \leftarrow \$U_\lambda$
- $Enc(k, m)$, picking a random $r \leftarrow \$\{0,1\}^n$ with an output of $c = (c_1, c_2) = (r, F_k(r) \oplus m) \leftarrow \$\{0,1\}^{n+l}$
- $Dec(k, (c_1, c_2)) = F_k(c_1) \oplus c_2$

. In this scheme, the only secret thing is k, which gives a *flavour* to the random function; Adversary can see $c = (r, F_k(r) \oplus m)$, so he can see r.

Theorem 10. *If $\mathcal{F}$ is a family of PRF functions, then Π is CPA-secure* $\diamond$

To prove this, we have to prove that $Game^{cpa}_{\Pi,\mathcal{A}}(\lambda, 0) \approx_c Game^{cpa}_{\Pi,\mathcal{A}}(\lambda, 1)$.

Proof. Consider hybrid arguments

$$\mathcal{HYB}_0 \equiv Game^{cpa}_{\Pi,\mathcal{A}}(\lambda, 0)$$

and

$$\mathcal{HYB}_1$$

, like $\mathcal{HYB}_0$ but with another distribution of $Enc(k, m_b)$:

- picking $r \leftarrow \$\{0,1\}^n$
- $R \leftarrow \$\mathcal{R}(\lambda, n, l)$
- output obtained is $(r, R(r) \oplus m_b)$

and

$$\mathcal{HYB}_2(\lambda, b)$$

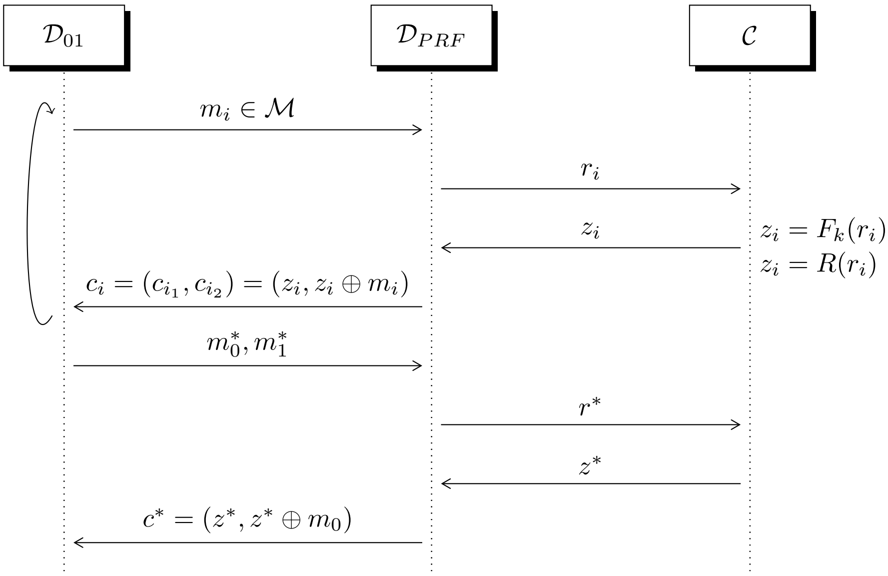
which simply outputs $(r_1, r_2) \leftarrow \$U_{n+l}$.

Lemma 6. *$\mathcal{HYB}_0 \approx_c \mathcal{HYB}_1$ for each $b \in \{0,1\}$* $\diamond$

Proof. Suppose these two hybrids are distinguishable; but then we can use the related distinguisher to break the starting assumption saying $\mathcal{F}$ is a PRF.

Since they are two similar games (but just for the encryption function), we can use the CPA game for this reduction.

Fix $b = 0$ and build the following



After the last message, D_{01} will reply with a $b' \in \{0, 1\}$ which expresses which one of the encryption functions has been used; since this result can be used for solving the PRF game, the starting assumption fails and the lemma is proven.

□

Lemma 7. $\mathcal{HYB}_1 \approx_c \mathcal{HYB}_2$

◇

Proof. Even if the output of the first hybrid is $(r_i, R(r_i) \oplus m_b)$, the distribution of the second member of this couple doesn't depend on m_b, hence we can assume that $R(r_i) \oplus m_b \equiv R(r_i)$.

The difference between $\mathcal{HYB}_1$ and $\mathcal{HYB}_2$ comes if we play the game once more and we pick up the same c_1 of the first game: while in $\mathcal{HYB}_2$ r_2 is independent from r_1 and will be different from the previous r_2' with probability near to 1, in $\mathcal{HYB}_1$ (if r_i is the same of the previous game) $R(r_i)$ will be the same of the previous game (because $R()$ is a random function).

Anyway, we can show that this "collision" happens with very low probability.

Call **REPEAT** this event of collision of r_i between 2 consecutive games. To show the statement of the lemma, it suffices to show that $P[REPEAT] \in \text{negl}\,(\lambda)$, therefore the two distributions are indistinguishable (or distinguishable with a negligible probability) .

In fact, if I make q queries to $\mathcal{HYB}_1$,

$$\begin{aligned}
P[REPEAT] = P[\exists i, j \in q \text{ such that } r_i = r_j] \leq \\
\leq \sum_{i,j \wedge i \neq j} \mathcal{P}[r_i = r_j] = Col(U_n) = \\
= \sum_{i \wedge j, i \neq j} \sum_{e \in \{0,1\}^n} \mathcal{P}[r_1 = r_2 = e] = \\
= \sum_{i \wedge j, i \neq j} \sum_{e \in \{0,1\}^n} \mathcal{P}[r = e]^2 = \\
= \binom{q}{2} 2^n \frac{1}{2^{2n}} = \\
= \binom{q}{2} 2^{-n} \leq \\
\leq q^2 2^{-n} \in \operatorname{negl}(\lambda)
\end{aligned}$$

□

Since $\mathcal{HYB}_0 \equiv Game(\lambda, 0)$ but the same proofs can be made for $Game(\lambda, 1)$,
$Game(\lambda, 0) \equiv \mathcal{HYB}_0(Game(\lambda, 0)) \approx_c \mathcal{HYB}_2 \approx_c \mathcal{HYB}'_0(Game(\lambda, 1)) \equiv Game(\lambda, 1)$

□

Lesson 8

4.13 Domain extension

How can we encrypt long messages , say $m = (m_1, m_2, ..., m_t)$ where for $i \in [t], m_i \in \{0,1\}^n$?

Mode of operation: let's build P_k, a blockcypher.

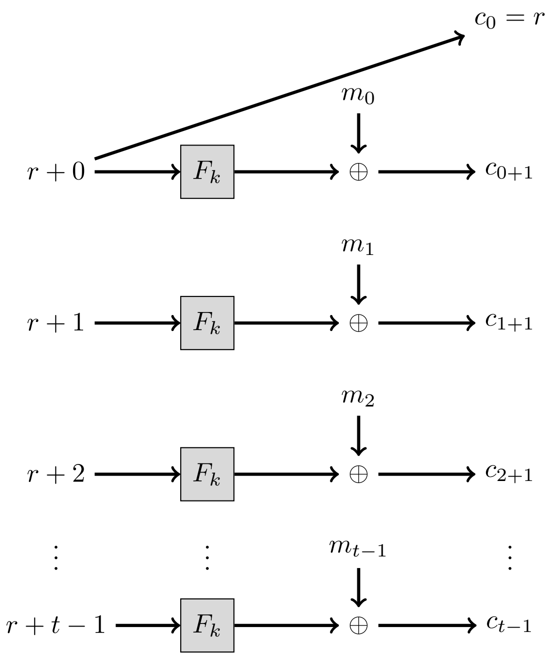
4.13.1 Electronic CodeBook (ECB)

(In notes, this name is associated with CTR-mode, or **counter mode**, but why?)
The principle is that I have many blocks of information and I encrypt all of them individually (maybe in parallel):

$$c_i = P_k(m_i) \forall i \in [t]$$

and outputs $c_1, c_2, ..., c_t$.

If I encrypt the same message, I obtain the same cyphertext. So it's not CPA-secure as the encryption function is **deterministic.**



In the image, each message is long exactly **n bits**, and

$$r \leftarrow \$\{0,1\}^n$$

so the sums made over r are done in **mod** 2^n, since r can be a number in the range $[0, 2^n - 1]$.

This is also called streamcipher, cause as input flows ECB produces the ciphertext.How to decrypt this schema?
Since I know r_i, if I compute $F_k(r_i)$ I can use the $\oplus$-operation to calculate the message.

4.13.2 Cipher block chaining (CBC)

==
==
CBC IMAGE ==
==
The schema to cipher here is

$$\forall i \in [1,t] c_i = P_k(C_{i-1} \oplus m_1)$$

How to decipher?

$$P_k^{-1}(i) = c_{i-1} \oplus m_i$$

so $m_i = P_k^{-1}(c_i) \oplus c_{i-1}$.

Theorem 11. *Assume $\mathcal{F}$ (F_k in the image above is part of this family) is a PRF, then CTR-mode yields a CPA-secure SKE for variable length messages.* ◇

.
Variable length messages means that every message

$$m = (m_1, ..., m_t)$$

has t subsets, and t can change from message m to another message $m' = (m'_1, ..., m'_{t'})$. Consider the following hybrids:

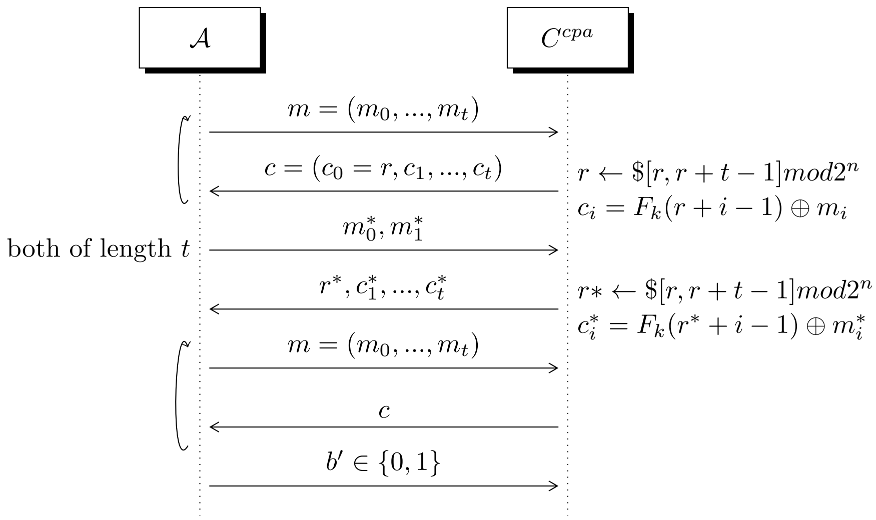


Figure 4.20: $Game^{cpa}_{CTR,\mathcal{A}}(\lambda, b)$

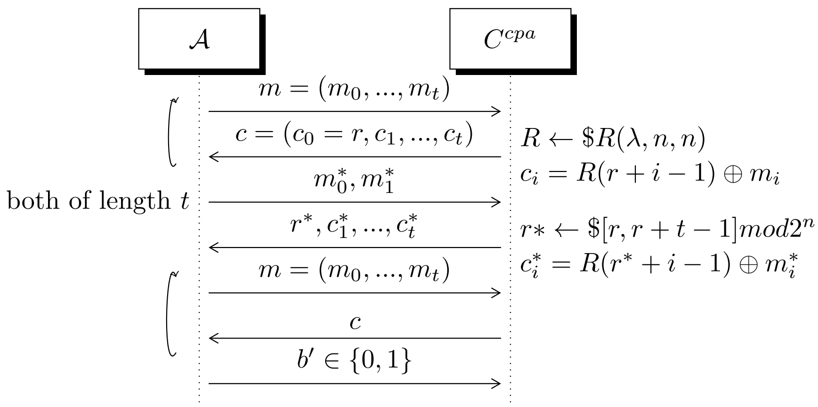


Figure 4.21: $\mathcal{HYB}_1(\lambda, b)$

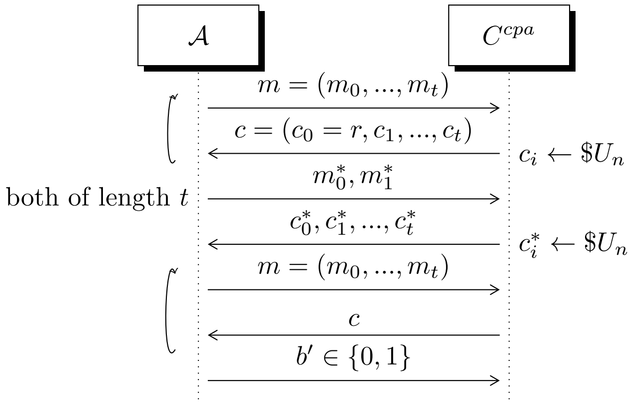


Figure 4.22: $\mathcal{HYB}_2(\lambda, b)$

Now we want to show that $Game^{cpa}_{\Pi,\lambda}(\lambda, 0) \approx_c Game^{cpa}_{\Pi,\lambda}(\lambda, 1)$

Proof.

Exercise 12.

Lemma 8. *Show that* $Game^{cpa}_{\Pi,\lambda}(\lambda, b) \approx_c \mathcal{HYB}_1(\lambda, b), \forall b \in \{0,1\}$ ◇

(Since this $Game(\lambda, b)$ is a CPA scheme and the second one is very similar, we can use a distinguisher which plays the CPA game; since this is a lemma, our precondition to "break" during the reduction is the precondition contained in the theorem statement)

Lemma 9. $\mathcal{HYB}_1(\lambda, b) \approx_c \mathcal{HYB}_2(\lambda), \forall b \in \{0,1\}$ ◇

The two hybrids are identical but for the encryption function, which is completely random in the second one while the first one uses a random function.

Since $R(r+i) \oplus m_i \approx R(r+i)$ (because m_i doesn't affect the distribution of the result at all), if $R(r^*)$ behaves like a true random extractor , the two hybrids are indistinguishable.

Now, if I examine a simple query and the consecutive challenge query (q, q^*) in a game in $\mathcal{HYB}_2$, the responses will be chosen at random with very low probability of being the same; but in a game in $\mathcal{HYB}_1$, the problem comes if the challenger creates an $r^* + j'$ for q^* which overlaps some chosen $r_i + j$ previously chosen for q.
This is bad, because $R(r_i + j) = R(r^* + j')$ since R is a random function, and the probability of outputting the same response ciphertext will be much higher in $\mathcal{HYB}_1$.

It's possible to show that these collisions happen very few times (with **negligible** probability) in $\mathcal{HYB}_1$ and that the two hybrids are distinct and distinguishable for a negligible factor.

Proof. Let :

- q= # of encryption queries
- t_i = # of blocks for the i-th query
- t^* = # of blocks for challenge

and let the function R run: we will have $R(r*), ...R(r^*+t^*-1)$ and $R(r_i), ..., R(r_i+t_i-1)$ sequences of random function outputs.

Definition 6. OVERLAP event: $\exists$ i, j, j', $r_i + j = r^* + j'$ (here $j \leq t_i$ and $j' \leq t^*$), for some query q_i. $\diamond$

If **OVERLAP** does not happen , the sequence $(R(r^*), ..., R(r^* + t^* - 1))$ is made of ***uniform*** and ***independent*** values. Thus $\mathcal{HYB}_1(b)$ is identical to $\mathcal{HYB}_2(b)$ for all $b \in \{0, 1\}$.

Now it suffices to show that $\mathcal{P}[OVERLAP] \in \text{negl}(\lambda)$.

For simplicity, assume q =(length of each query) and also $t_i = t^* = q$.
Let $OVERLAP_i$ be the event that the $i - th$ query $r_i, ..., r_i + q - 1$ partially or totally overlaps the challenge sequence $r^*, ..., r^* + q - 1$.

Fix some $r*$.
One can see that $OVERLAP_i$ happens if

$$r^* - q + 1 \leq r_i \leq r^* + q - 1$$

, which means that r_i should be chosen *at least* in a way that :

- the sequence $r^*, ..., r^* + q - 1$ comes before the sequence $r_i, ..., r_i + q - 1$, and they overlap just for the last element $r^* + q - 1 = r_i$ or
- the sequence $r_i, ..., r_i + q - 1$ comes before the sequence the sequence $r^*, ..., r^* + q - 1$, and they overlap just for the last element $r_i + q - 1 = r^*$

.
So now

$$\mathcal{P}[OVERLAP_i] = \frac{(r^* + q - 1) - (r^* - q + 1) + 1}{2^n} = \frac{2q - 1}{2^n}$$

It is obvious that, for definition of **OVERLAP** ,

$$\mathcal{P}[OVERLAP] \leq \sum_{i=1}^{q} \mathcal{P}[OVERLAP_i] \leq 2\frac{q^2}{2^n} \in \text{negl}(\lambda)$$

□

Since $\mathcal{HYB}_1 \approx_c \mathcal{HYB}_2$ and $\mathcal{HYB}_1 \equiv Game(b)$ with b equal to 0 and 1 respectively, we can state that

$$Game(0) \approx_c \mathcal{HYB}_2 \approx_c Game(1)$$

.

□

Lesson 9

4.14 Message Authentication and UFCMA-security

First, remember the $Tag()$ function and how a MAC works. Now $Tag()$ is defined using a key k, and we call it $Tag_k()$.
In particular we are looking for a cool property of a message authentication protocol, called ***universal unforgeability against chosen-message attacks***, which prevents the attacker from generating a valid couple (m^*, ϕ^*) after some queries containing messages and receiving the related tags.

This property is defined through a game called

$$Game^{ufcma}_{\Pi,\mathcal{A}}(\lambda)$$

and played in the following manner:

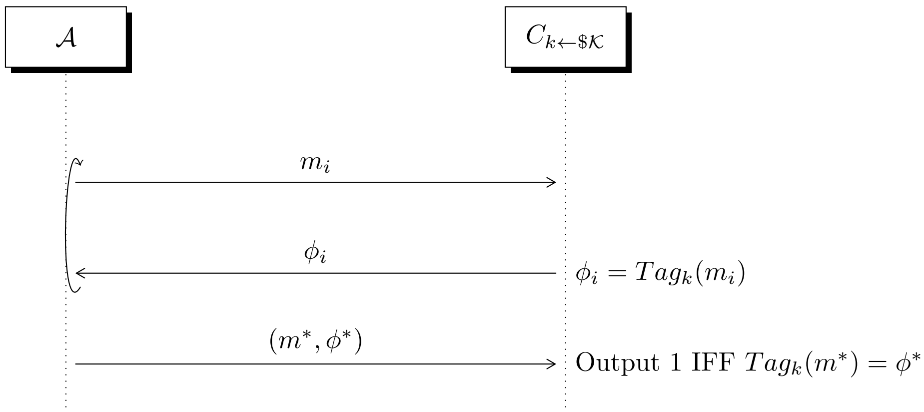


Figure 4.23: $Game^{ufcma}_{\Pi,\mathcal{A}}(\lambda)$

with m^* which must be a fresh new message never used before and with the Adversary $\mathcal{A}$ which doesn't know the key k used in the tag.

Definition 7. Π is ***ufcma-secure*** if $\forall.PPT.\mathcal{A}$

$$\mathcal{P}[Game^{ufcma}_{\Pi,\mathcal{A}}(\lambda) = 1] \in \text{negl}(\lambda)$$

$\diamond$

Now consider the following theorem:

Theorem 13. *Let* $\mathcal{F} = \{F_k : \{0,1\}^n \to \{0,1\}^l\}_{k \in \{0,1\}^\lambda}$ *be a PRF family. Then* Π *which uses* $Tag_k() = F_k()$ *is a UFCMA-secure MAC with n-bit domain.* $\diamond$

We show that the scheme which uses a PRF is indistinguishable from a scheme which uses a random function, and that a MAC scheme which uses a random function is breakable by an efficient attacker in negligible time.

Proof. Consider the two following hybrids:

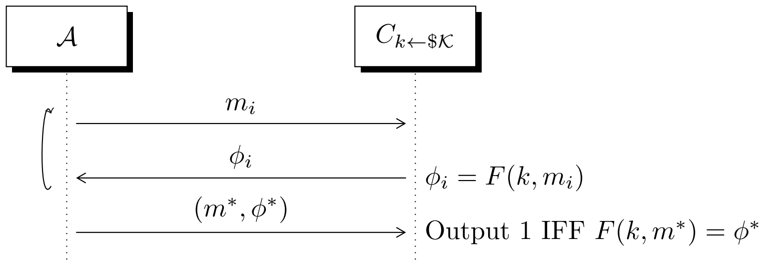


Figure 4.24: $Game^{ufcma}_{\mathcal{F},\mathcal{A}}(\lambda)$

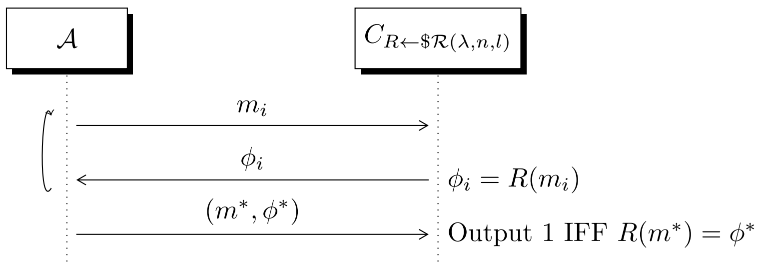
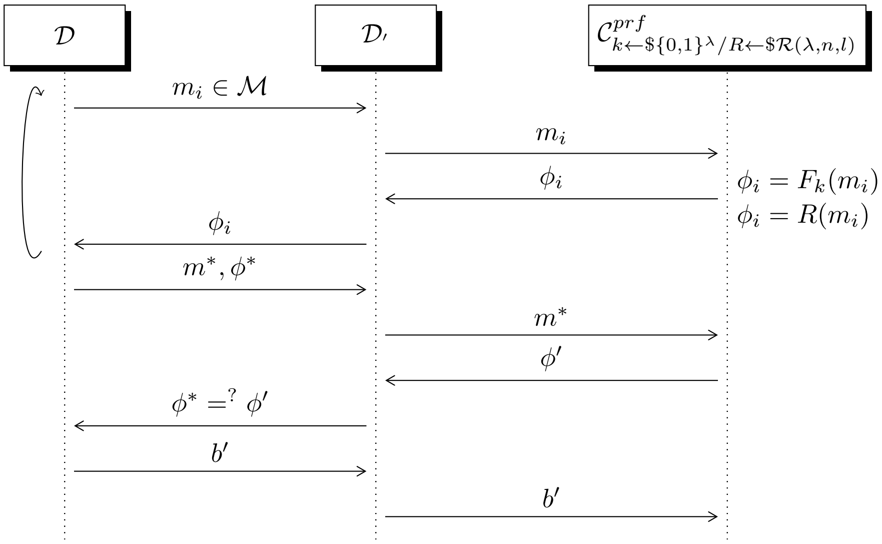


Figure 4.25: $\mathcal{HYB}^{ufcma}(\lambda)$

Lemma 10. $Game^{ufcma}_{\mathcal{F},\mathcal{A}}(\lambda) \approx_c \mathcal{HYB}^{ufcma}_1(\lambda)$ $\diamond$

Assume that these two are distinguishable by D.So we could have D' which is capable , with the following game, of distinguish a pseudo-random function from a true random function:



After D' receive ϕ' from C, he has to use the distinguisher D to distinguish the PRF function from the random one.
So he says to D if $\phi^* = \phi'$ or not, and D now is capable of understanding which game he is playing.

Lemma 11. *For all efficient adversaries* $\mathcal{P}[\mathcal{HYB}_1(\lambda) = 1] \leq 2^{-l}$ $\diamond$

This is true because attacker has to predict the output $R(m^*)$ on a fresh input m^* and to send to the Challenger this couple to win the game.
This can happen, at most, with probability $\frac{1}{2^l}$. $\square$

This was for fixed length messages, since we can encrypt messages long n. In the next subsection, we will see how to create a UFCMA-secure MAC for variable length messages.

4.15 Domain extension

Assume $m = (m_1, ..., m_t) \in (\{0,1\}^n)^t$for some $t \geq 1$. How can we tag m given just $Tag_k : \{0,1\}^n \to \{0,1\}^l$?

We can try various attempts:

- XOR all blocks: $\phi = Tag_k(\bigoplus_{i=1}^t m_i)$.
  This is not secure cause given (m, ϕ) it is possible to find $m^* \neq m$ s.t.
  $$\bigoplus m_i = \bigoplus m_i^*$$
  and output this the couple (m^*, ϕ) to win.
- Let $\phi_i = Tag_k(m_i)$ and the final message of the challenge has this form : $(m, \phi = (\phi_1, ..., \phi_t))$.
  This is not secure. Given $m = (m_1, ..., m_t)$, this message has an unique $\phi = (\phi_1, ..., \phi_t)$, and if I swap m_1 and m_t I obtain a fresh new message,

with a fresh new tag $\phi' = (\phi_t, \phi_2, ..., \phi_{t-1}, \phi_1)$. Using this new couple, the game is won.

- Try with $\phi_i = Tag_k(i||m_i)$, authenticating the position of the block . But this is not secure, and can be showed in **just 2 queries**. (I solved this in class during the break, with $t+1$ queries: t for retrieving the partial tag of the submessage+position , and the last query to merge all the obtained results in a fresh new message. But this solution can be improved.) (UPDATE: I send $m = m_{1_1}, ...m_{1_t}$ and I save the corresponding ϕ. Then I send $m' = m_{2_1}, ..., m_{2_t}$ and I save the ϕ'. Now I forge the new fresh prince of Bel Air $m^* = m_1, m_2, m_2, ..., m_2$ and I can forge also a valid ϕ^* because I have all the signed parts of this new tag.)

Now I feel cool in Los Angeles and I want to explore a new **IDEA**: the design of a shrinking functions family

$$\mathcal{H} = \{h_s : \{0,1\}^{nt} \to \{0,1\}^n\}_{s\in\{0,1\}^\lambda}$$

which can be used to shrink variable length messages and then apply a PRF on them.
This idea is cool, and I consider the induced family

$$\mathcal{F}(\mathcal{H}) = \{F_k(h_s(.))\}$$

Question 3. Which are the properties of this family?

The main problem are *collisions* , since for each $m \in \{0,1\}^{nt}$ it should be hard to find $m' \neq m$ such that $h_s(m) = h_s(m')$.
But we know that collisions exist, because we are trying to create a function

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

which maps elements in $\{0,1\}^{nt}$ to the elements in $\{0,1\}^t$, and since the second set is smaller, for the pidgeonhole principle, there will be elements of $\{0,1\}^t$ which will be reached by more than one element of $\{0,1\}^{nt}$.

To overocome this problem, we can consider 2 ways:

- assume collisions are hard to find given $s \in \{0,1\}^\lambda$ publicly, and we have a *collision resistant hashing*;
- let s be secret, and assume collisions are hard to find because it is hard to know how h_s works.

Definition 8. $\mathcal{H}$ is called ***universal*** family if

$$\forall x, x' \in \{0,1\}^{nt} \text{ such that } x \neq x'$$

$$\mathcal{P}_{s\leftarrow\$\{0,1\}^\lambda}[h_s(x) = h_s(x')] \leq \varepsilon$$

◇

For $\varepsilon = 2^{-n}$ we call it **perfectly universal**.
For $\varepsilon \in \text{negl}(\lambda)$ we call it **almost universal** (or **AU**).

Exercise 14. Show that any pairwise independent hash function is perfectly universal.(should I use Col for solving this? What is the difference and when I should use Col instead of one-shot-probability?) **ASK FOR SOLVING PROPERLY** (Thoughts: when I ask *what's the probability that , chosen 2 distinct x-es, their hashes are the same* ***on a certain value****?* , maybe I have to use one-shot, because one-shot refers to the prob. that the two inputs collide on a specific value, even if not specified.
Instead, if I consider *what's the prob. that , chosen 2 distinct x-es, their hashes are the same?*, maybe I have to calculate all the possible collisions, because I want to know if the 2 inputs can collide in general.)

Theorem 15. *Assuming $\mathcal{F}$ is a PRF with $n-$bit domain and $\mathcal{H}$ is AU, then $\mathcal{F}' = \mathcal{F}(\mathcal{H})$ is a PRF (and , if used in a MAC as tag function, makes it UFCMA) on $nt-$bit domain (for $t \geq 1$).* $\diamond$

We want to show that $\mathcal{F}' = \mathcal{F}(\mathcal{H})$ is a PRF , so we want to show that

$$Real_{\mathcal{F},\mathcal{A}}(\lambda) \approx_c Rand_{\mathcal{R}',\mathcal{A}}(\lambda)$$

Proof. Consider these 3 experiments/games:

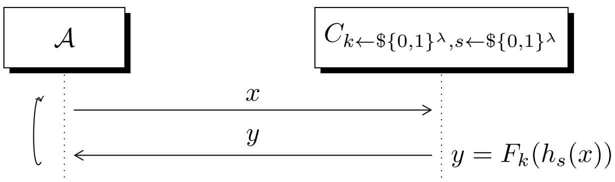


Figure 4.26: $Real_{\mathcal{F},\mathcal{A}}(\lambda)$

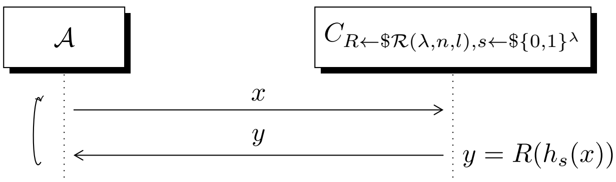


Figure 4.27: $\mathcal{HYB}_{\mathcal{R},\mathcal{A}}(\lambda)$

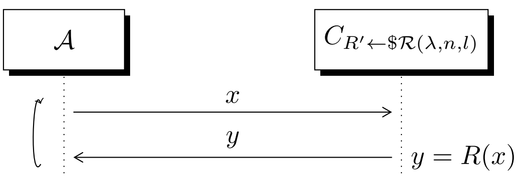


Figure 4.28: $Rand_{\mathcal{R}',\mathcal{A}}(\lambda)$

Lemma 12. *Real* $\approx_c \mathcal{HYB}$

$\diamond$

Exercise 16. Prove it!

Lemma 13.

$$\mathcal{HYB} \approx_c Rand$$

$\diamond$

These 2 experiments are very similar but for the encryption step.
If I send 2 consecutives queries to both the experiments, while *Rand* will give me

- the same y with the same x or
- the same y with different $x_1 \neq x_2$ but with very low probability,

$\mathcal{HYB}$ could return the same y:

- with the same x input or
- with different $x_i \neq x_j \Rightarrow h_s(x_i) \neq h_s(x_j)$ but with very low probability (because R is a random function), or
- with different $x_i \neq x_j$ but with the same $h_s(x_i) = h_s(x_j)$,

.
We want to show that the last item doesn't happen too often, and that these 2 experiments are distinct for a negligible factor.

Proof. Let **BAD** be the event that

$$\exists i, j \in [q] \text{ with } i \neq j \text{ s.t. } h_s(x_i) = h_s(x_j)$$

As logn as **BAD** doesn't happen, the function R is run as a sequence of distinct points $R(h_s(x_1)), ..., R(h_s(x_q))$.
So in this case the distribution if the 2 games is identical, and it suffices to show that $\mathcal{P}[BAD] \in \text{negl}(\lambda)$.

The event **BAD**, which happens during a game made of q queries and q replies , is the same as collecting all the q queries, chosing the seed and then looking for collisions. If interpreted in this way,

$$\begin{aligned} \mathcal{P}[BAD] = \mathcal{P}_s[\exists x_i \neq x_j, h_s(x_i) = h_s(x_j)] \leq \\ \leq \sum_{i,j} \underbrace{\mathcal{P}[h_s(x_i) = h_s(x_j)]}_{h_s \text{ is AU by definition}} \leq \\ \leq \binom{q}{2} \text{negl}(\lambda) \in \text{negl}(\lambda) \end{aligned}$$

$\square$

So now we have *Real* $\approx_c \mathcal{HYB} \approx_c$ *Rand* $\square$

Now we want to show a possible shrinking/hash function which can be used in a UFCMA-secure MAC. In order to be used in a UFCMA-secure MAC as the input of a PRF, this function must be part of an ***almost universal*** family.

4.15.1 $\mathcal{H}$ family using Galois Fields

Construction 3. Take $\mathbb{F} = GF(2^n)$, a *Galois field* of 2^n elements.
Let $m = (m_1, ..., m_t) \in \mathbb{F}^t$ and $s = (s_1, ..., s_t) \in \mathbb{F}^t$. We state that

$$h_s(m) = \sum_{i=1}^{t} s_i m_i =< s, m >= q_m(s)$$

$\diamond$

A generic *galois field* has very interesting properties, like the following :

- addition of two elements is like applying the XOR operation on their binary forms;
- multiplication of two elements is like the product $mod 2^n$

.

Now, since this family must be almost universal to be used as part of UFCMA-secure MAC protocol, collisions must happen a negligible amount of time.
Suppose we have a collision with two different messages:

$$\sum_{i=1}^{t} m_i s_i = \sum_{i=1}^{t} m'_i s_i$$

Let $\delta_i = m_i - m'_i$, assuming without loss of generality that $\delta \neq 0$.
Now we have that, when a collision happens,

$$\sum_{i=1}^{t} m_i s_i = \sum_{i=1}^{t} m'_i s_i \Leftrightarrow \sum_{i=1}^{t} m_i s_i - \sum_{i=1}^{t} m'_i s_i = 0 \Leftrightarrow$$
$$\sum_{i=1}^{t} \delta_i s_i = 0$$

Taking m, m' such that $m \neq m'$ means that m is distinct from m' at least for one subsequence $m_i \neq m'_i$.
So we can assume, without loss of generality, that $i = 1$ is an index (or the only index) which m and m' differ on.

So we can split the last summation in 2 parts, choosing $\delta_1 s_1$ as the first element and $\sum_{i=2}^{t} \delta_i s_i$ as second element:

$$\delta_1 s_1 + \sum_{i=2}^{t} \delta_i s_i = 0 \Leftrightarrow$$
$$\delta_1 s_1 = -\sum_{i=2}^{t} \delta_i s_i \Leftrightarrow$$
$$s_1 = \frac{-\sum_{i=2}^{t} \delta_i s_i}{\delta_1}$$

and this means that when a collision happens s_1 must be exactly equal two the second member of the equation, which is an element of $\mathbb{F}$. But since every seed is chosen at random among $\mathbb{F}$, what's the probability of picking the element s_1 which zeroes the above equation ?
This probability is just $\frac{1}{\mathbb{F}} = \frac{1}{2^n} \in \operatorname{negl}(\lambda)$.

$\mathcal{H}$ with Galois fields elements and polynomials

Construction 4. Take $\mathbb{F} = GF(2^n)$, a *Galois field* of 2^n elements.
Let $m = (m_1, ..., m_t) \in \mathbb{F}^t$ and $s \leftarrow \$\mathbb{F}^t$. We state that

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

◇

Exercise 17. Prove that this construction is **almost universal**.
(possible proof: to be almost universal, looking at the definition, collisions with $m \neq m'$ must be negligible.
So consider a collision as above: it must be true that

$$\sum_{i=1}^{t} m_i s^{i-1} = \sum_{i=1}^{t} m'_i s^{i-1} \Leftrightarrow \sum_{i=1}^{t} m_i s^{i-1} - \sum_{i=1}^{t} m'_i s^{i-1} = 0 \Leftrightarrow q_{m-m'}(s) = 0$$

How can we make a polynomial equal to 0? We have to find the **roots** of the polynomial, which we know are at most the **grade** of the polynomial . So, the grade of this polynomial is $t-1$, and the probability of picking a root from $\mathbb{F}$ as seed of $h_s(.)$ is

$$\mathcal{P}[s = root] = \frac{t-1}{2^n} \in \operatorname{negl}(\lambda)$$

)

Lesson 10

4.16 Domain extension for PRFs/MACs

Almost universal approach : I have a family $\mathcal{F}(\mathcal{H})$ with $\mathcal{H}$ AU and with PRF $f \in \mathcal{F}$.

Computational AU : we want to build a family $\mathcal{H}$ using some other PRFs. We expect to have:

- $\mathcal{P}[h_s(m) = h_s(m'), s \leftarrow \$\{0,1\}^\lambda, (m, m') \leftarrow \$A(1^\lambda)] \in \text{negl}(\lambda)$;
- We need two PRFs. One is F_k, and the other is $F_s j$ == something related to $f_s(1,.)$ and $f_s(0,.)$, but I didn't get it. ==

.

4.16.1 XOR mode

Assume that we have this function

$$h_s(m) = F_s(m_1||1) \oplus ... \oplus F_s(m_t||t)$$

so that the input to the PRF $F_s(.)$ is $n + log_2 t$ bytes long.

Lemma 14. *Above $\mathcal{H}$ is computational AU if $\mathcal{F}$ is a PRF.* ◇

Exercise 18. Prove this !

Possible proof:

we have to show that

$$\mathcal{P}[h_s(m) = h_s(m')] \in \text{negl}(\lambda)$$

with $m \neq m'$.
This means that

$$\mathcal{P}[F_s(m_1||1) \oplus ... \oplus F_s(m_t||t) = F_s(m'_1||1) \oplus ... \oplus F_s(m'_t||t)] =$$

$$= \mathcal{P}[F_s(m_i\|i) \oplus F_s(m'_i\|i) = \alpha = \bigoplus_{j=1, j\neq i}^{t} F_s(m_j\|j) \oplus F_s(m'_j\|j)]$$

for each $i \in [1, t]$.
But α is one unique random number chosen over 2^n possible candidates, so the collision probability is negligible.

4.16.2 CBC MAC

This is part of the standard, used in TLS. It's used with a PRF F_s, setting the starting vector $IV = 0^n = c_0$ and running this PRF as part of CBC. The output of the CBC process is just the last block:

$$h_s(m_1, ..., m_t) = F_s(m_t \oplus F_s(m_{t-1} \oplus ... \oplus F_s(m_2 \oplus F_s(m_1 \oplus IV))))$$

Lemma 15. *CBC MAC defines completely an AU family.*
(not proven) ◇

We can use this function to create an **encrypted CBC**, or **E-CBC** :

$$E - CBC_{K,S}(m) = F_k(h_s^{CBC}(m))$$

.

Theorem 19. *Actually if $\mathcal{F}$ is a PRF, CBC-MAC is already a MAC with domain nt for arbitrary but fixed $t \in \mathbb{N}$.*
(not proven) ◇

4.16.3 XOR MAC

Instead of $\mathcal{F}(\mathcal{H})$ now the $Tag()$ function outputs $\phi = (\eta, F_k(\eta) \oplus h_s(m))$ where $\eta \leftarrow \$\{0,1\}^n$ is random and it's called *nonce* .
When I want to authenticate, I should send the

$$(m, (\eta, F_k(\eta) \oplus h_s(m)))$$

couple.
When I want to verify a message and I get the couple $(m, (\eta, v))$, I just check that $v = F_k(\eta) \oplus h_s(m)$. It should be hard to find a value called a such that, given $m \neq m'$,

$$h_s(m) \oplus a = h_s(m')$$

In fact, since an adversary who wants to break this scheme has to send a valid couple (m^*, ϕ^*) after some queries, he could:

- ask for message m and store the tag $(\eta, F_k(\eta) \oplus h_s(m))$
- try to find $a = h_s(m) \oplus h_s(m')$ and modify the previous stored tag adding $v \oplus a$,

so now he could send the authenticated message

$$(m', (\eta, F_k(\eta) \oplus h_s(m')))$$

which is a valid message.

==
WHAT IS IT ABOUT?

Lemma 16. *XOR mode gives computational AXU (Almost Xor Universal).*
(not proven) ◇

WAHT DOES IT MEAN AXU?? Has something to do with the first 2 definitions given in the start of this subsection ? ==

Theorem 20. *If $\mathcal{F}$ is a PRF and $\mathcal{H}$ is computational AXU, then XOR-MAC is a MAC.*
(not proven) ◇

==
NOT CLEAR WHAT TO DO

Exercise 21. Now with variable input lenght:

- AXU based XOR mode
- $\mathcal{F}(\mathcal{H})$ is insecure with polynomial construction $h_s(m) = q_m(s)$, but can be fixed.
- CBC-MAC not secure. (exercise)
- E-CBC is secure.

==

4.17 Chosen Ciphertext Attack security

In this kind of attack, the adversary has a decryption capability added to the previous encryption capability. A scheme which is CCA-secure is also **non-malleable**, since the attacker cannot modify the obtained ciphertexts to obtain new valid ciphertexts.

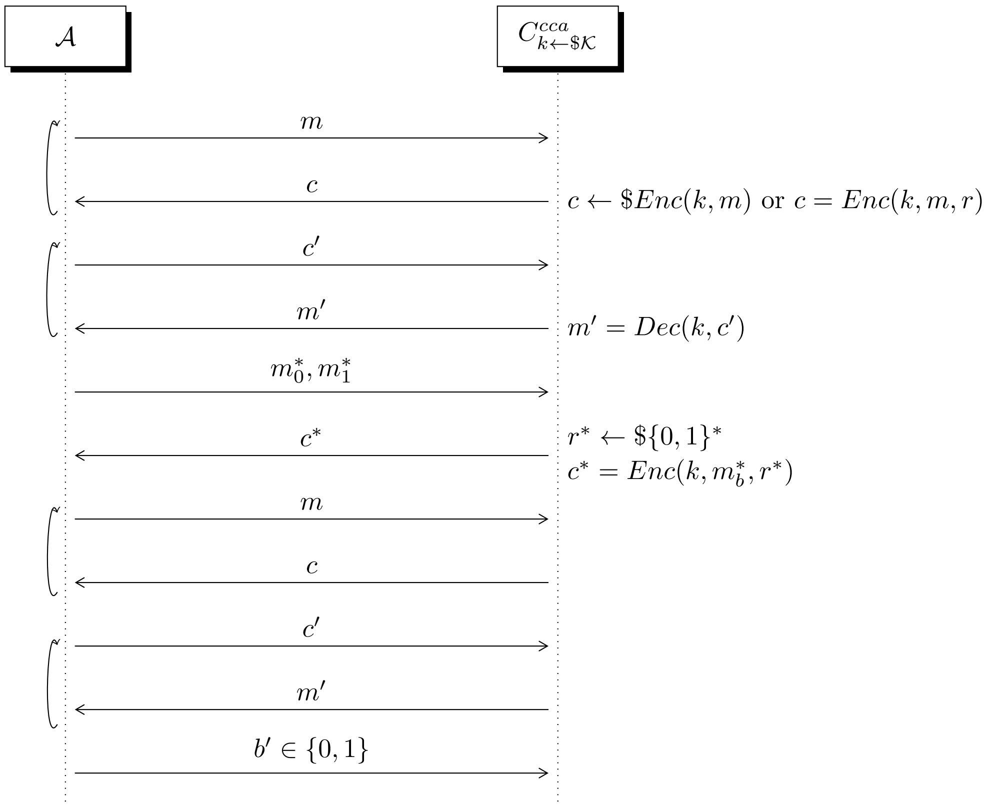
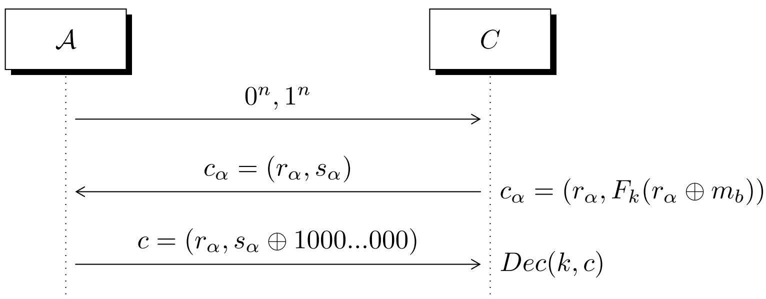


Figure 4.29: $Game^{cca}_{\Pi,\mathcal{A}}(\lambda, b)$

Exercise 22. Show that $(r, F_k(r) \oplus m)$ which is CPA-secure, is not CCA-secure.



Proof. with

$$Dec(k, c) = F_k(r_\alpha) \oplus s_\alpha \oplus 1000...000 =$$
$$F_k(r_\alpha) \oplus F_k(r_\alpha) \oplus m_b \oplus 1000...000 =$$
$$m_b \oplus 1000...000$$

At this point we have that the output is

- 1000...000 if m_b was 000...000
- 0111...111 if m_b was 0111...111

□

4.18 Authenticated encryption

Idea: what if we combine the target of authenticity with the target of encryption?
The first property is satisfied when the receiver is able to understand if the received message was sent exactly by the trusted sender; the last property is satisfied when no information of the sent message is contained in the ciphertext, thus only the chosen receiver can fully read the original sent message.
If we build up a schema with these 2 properties, we can obtain a new schema which should be *cpa-secure* (to enforce the encryption/privacy property) and essentially secure against forgeries of chosen message attacks (to enforce the authentication property).
In particular, a successful forgery can be obtained winning the following game:

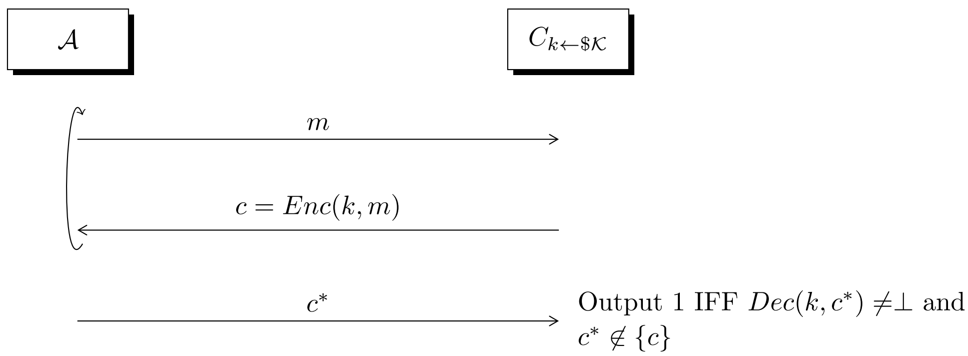


Figure 4.30: $Game^{auth}_{\Pi,\mathcal{A}}(\lambda)$

meaning that the c^* challenge ciphertext should be a **fresh** and a valid one ciphertext, considering that the $Dec(.)$ function is defined as

$$Dec : \mathcal{K} * \mathcal{C} \rightarrow M \cup \{\perp\}$$

where $\perp$ represents invalid/meaningless messages.

In the end, we want that our *authenticated encryption schema* is cpa-secure and has this last property, called **strong unforgeability** or **auth** .

Theorem 23. *Let Π be an SKE (shared key encryption). If Π has* ***CPA + auth*** *security, then it also has* ***CCA*** *security.* $\diamond$

Exercise 24. Prove it!
Hint: consider the experiment where $Dec(k, c)$:

- if c not fresh (i.e. output of previous encryption query m, output m)
- else output $\perp$

. ==
TO DO AND TO PROVE
Approach: reduce cca to cpa; given D^{cca}, we can build D^{cpa}. D^{cca} will ask decryption queries, but D^{cpa} can answer just with these two properties shown above, so it can reply just if he asked these (c, m) before to its challenger $\mathcal{C}$.
==

4.18.1 Three approaches to authenticated encryption

Let Π_1 be a **cpa-secure** SKE and Π_2 be a **auth** MAC schema.
We have 3 ways to combine these 2 schema to obtain a new one:

1. ***Encrypt-and-MAC*** :

$$c \leftarrow \$Enc(k_1, m)$$
$$\phi = Tag(k_2, m)$$
$$c^* = (c, \phi)$$

2. ***MAC-then-encrypt*** :

$$\phi = Tag(k_2, m)$$
$$c \leftarrow \$Enc(k_1, \phi \| m)$$
$$c^* = c$$

3. ***Encrypt-then-MAC*** :

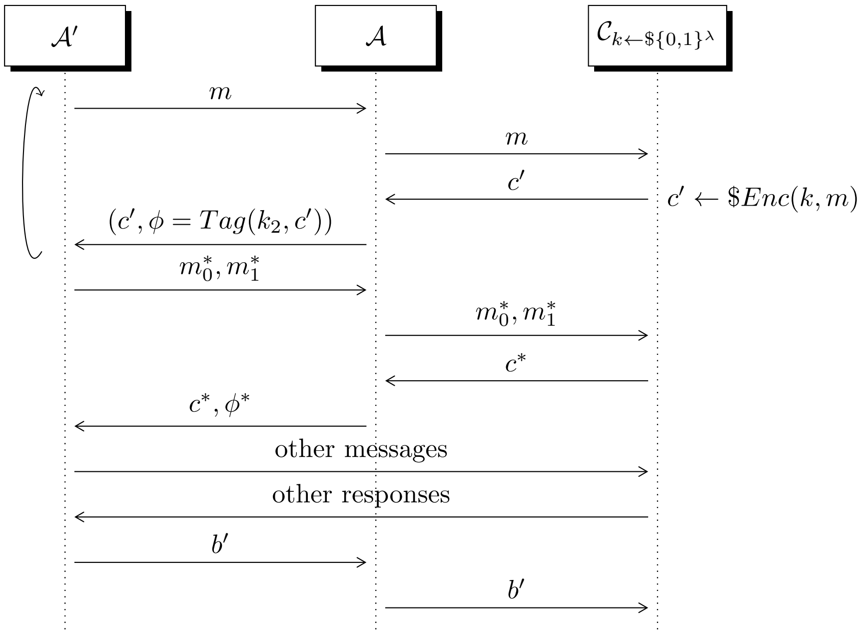
$$c \leftarrow \$Enc(k_1, m)$$
$$\phi = Tag(k_2, c)$$
$$c^* = (c, \phi)$$

In a paper is clearly stated that the first solution doesn't work taking random combination of CPA-secure and MAC schemes, because there are couple which, mixed, are not CCA-secure. Furthermore the second solution doesn't work for the same above reason, even if it's now part of the standard.

Instead, the third solution works always for a CPA-secure scheme Π_1 and strong unforgeability scheme Π_2 which, combined, they generate the final scheme Π.

Theorem 25. *If Π is made combining Π_1 cpa-secure and Π_2 auth-secure as stated in the third point, then Π is cpa-secure and auth-secure .* $\diamond$

Proof. By reduction, we negate that Π has both the properties and we find the contraddiction.
Suppose now that Π hasn't the cpa-security property.



Here we are supposing that $\mathcal{A}$ can break the cpa-security of a generic Π_1 scheme used by $\mathcal{C}$, while $\mathcal{A}'$ can break the cpa-security of a generic scheme Π. $\mathcal{C}$ can generate the $Tag(k_2, .)$ function, randomly chosing $k_2(((($ and simulating Π.

==
REVIEW
Professor says that we have to show that $Game^{cpa}(\lambda, 0) \approx_c Game^{cpa}(\lambda, 1)$, but why??? Isn't this proof enough?

==
Proved for the cpa-security property, now we have to prove, in a similar way, that the auth property must be holded by Π if Π_2 is an auth-secure scheme.

Exercise 26. Prove it!
Similar to the cpa-security proof.

□

Lesson 11

4.19 Authenticated encryption (Age of Ultron)

Last time we proved CPA-security of Π. Today we will explore the *auth* property. Consider Π as

$$Enc : \{0,1\}^\lambda * \mathcal{M} \to \mathcal{C}$$
$$Tag : \{0,1\}^\lambda * \mathcal{C} \to \Phi$$

Lemma 17. *If $Tag(.,.)$ is* ***EUF-CMA*** *, then Π has auth-property.* $\diamond$

What is **EUF-CMA** ?
i It's a property similar to **uf-cma** , but now I want that the challenge message (m^*, ϕ^*) is made by a fresh m^* and a valid **fresh** ϕ^*.
The difference is that in ufcma we didn't care about the freshness of ϕ^*.

Proof. Suppose Π has not the *auth* property.
So we have an $\mathcal{A}'$ which can win the **auth** challenge of Π.
On the other hand, we have a Π_2 schema which uses an **euf-cma** $Tag(.,.)$ function.
So, by reduction, ...

$\square$

4.20 Textbook RSA

This is an <u>insecure</u> toy example of the more complex *RSA* (Rivest Shamir Adleman) algorithm. The key generation algorithm: KGen = Gen$RSA(1^\lambda)$ outputs $P_k = (n, e)$ and $S_k = d$, then we have

$$Enc(P_k, m) = m^e \pmod n$$
$$Dec(S_k, c) = c^d \pmod n$$

Since the output of Enc is deterministic this is **not CPA secure**! However it can be used with HARD-CORE Predicate.
Preprocess the message to add randomness:

$$\hat{m} = r||m \text{ where } r \leftarrow \$\{0,1\}^l$$

now Enc is not deterministic.
Facts:

1. $l \in super(log(\lambda))$ otherwise it is possible to bruteforce in PPT.
2. If $m \in \{0,1\}$ then I can prove it CPA secure under RSA (just use standard TDP)
3. If m is "in the middle" ($\{0,1\} \leq m \leq \{0,1\}^l$) RSA is believed to be secure and is <u>standardized</u> (PKCS#1,5)
4. Still not CCA secure!

4.20.1 Trapdoor Permutation from Factoring

Let's look at $f(x) = x^2 \pmod{n}$ where $f : \mathcal{Z}_n^* \to \mathbb{QR}_n(\subset \mathcal{Z}_n^*)$, this is not a permutation in general.
Now let's consider the Chinese Reminder Theorem (CRT) representation:

$$x = (x_p, x_q) \to x_p \equiv x \pmod{p}, x_q \equiv x \pmod{q}$$
$$f(x) = x^2 \pmod{p}; x \leftarrow \$\mathcal{Z}_p^*$$

Since Z_p^* is cyclic I can always write:

$$Z_p^* = \{g^0, g^1, g^2, \ldots, g^{\frac{p-1}{2}-1}, g^{\frac{(p-1)}{2}}, \ldots, g^{p-2}\}$$
$$\mathbb{QR}_p = \{g^0, g^2, g^4, \ldots, \overbrace{g^{p-3}}^{g^{\frac{pi1}{2}-1} inZ_p^*}, \underbrace{g^0}_{g^{\frac{p-1}{2}} inZ_p^*}, \ldots\}$$
$$|\mathbb{QR}_p| = \frac{p-1}{2}$$

Moreover since $(g^{\frac{p-1}{2}})^2 \equiv 1 \pmod{p}$ then $g^{\frac{p-1}{2}} \equiv -1 \pmod{p}$.
Now assume $p \equiv 3 \pmod{4}$ ([*]$p = 4t + 3$ CRT), then squaring $Modp$ is a permutation because, given $\underline{y = x^2 \pmod{p}}$ if I compute:

$$(y^{t+1})^2 = \underbrace{y^{2t+2}}_{[*]\ 2t+2=\frac{p-3}{2}+2=\frac{p+1}{2}=\frac{p-1}{2}+1} = (x^2)^{\frac{p-1}{2}+1} = 1x^2 = x^2$$
$$\implies x = \pm y^{t+1}$$

But only 1 among the above $\pm y^{t+1}$ is a square, this is because $\frac{p-1}{2}$ is odd.

Lemma 18. *$\forall z, z \in \mathbb{QR}_p$ IFF $-z \notin \mathbb{QR}_p$* $\diamond$