Contents

Lesson 1

1.1 Problems in Cryptography

- Confidential communication

TODO 1: Image of Alice,Bob, Eve in ske

Kerckhoffs's principle: No security by obscurity.

Security should only rely on the secrecy of the key. However sharing the key between two parties is a costly operation.

Syntax:

- $\mathcal{K}$ = Key Space
- $\mathcal{M}$ = Message Space
- $\mathcal{C}$ = Ciphertext Space
- $Enc : \mathcal{K} \times \mathcal{M} \to \mathcal{C}$
- $Dec : \mathcal{K} \times \mathcal{C} \to \mathcal{M}$

Correctness:

$\forall m \in \mathcal{M}, \forall k \in \mathcal{K} : Dec(k, Enc(k, m)) = m$

Definition 1. Perfect Secrecy (SHANNON)

Let M be any distribution over $\mathcal{M}$.

Let K be uniform over $\mathcal{K}$.

We say that (Enc, Dec) is perfectly secret if $\forall M, \forall m \in \mathcal{M}, \forall c \in \mathcal{C}$.

$$Pr[M = m] = Pr[M = m|C = c]$$

Meaning that the ciphertext does not reveal anything about the message. ⋄

Theorem 1. *Let M, K, C as above. The following are equivalent:* ⋄

I $Pr[M = m] = Pr[M = m|C = c]$

II M and C are INDEPENDENT

III $\forall m, m' \in \mathcal{M}, \forall c \in C \; Pr[Enc(k, m) = c] = Pr[Enc(k, m') = c]$

Proof. $(I) \implies (II)$

$$Pr[M = m \wedge C = c] = Pr[C = c]Pr[M = m|C = c] = Pr[C = c]Pr[M = m]$$

They are independent.

$(II) \implies (III)$ Fix M, any $m \in M, c \in C$

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

$(III) \implies (I)$

$$Pr[C = c] = \sum_{m'} Pr[C = c \wedge M = m'] =$$
$$= \sum_{m'} Pr[Enc(K, M) = c \wedge M = m'] =$$
$$= \sum_{m'} Pr[Enc(K, M) = c|M = m']Pr[M = m'] =$$
$$= \sum_{m'} Pr[Enc(K, m') = c]Pr[M = m'] = \text{(Using III)}$$
$$= Pr[Enc(K, m) = c]\underbrace{\sum_{m'} Pr[M = m'] =}_{=1}$$
$$= Pr[Enc(K, m) = c] = Pr[Enc(K, M) = c|M = m] =$$
$$= \underbrace{Pr[C = c|M = m]}_{\substack{\text{The proof ends here} \\ \text{but we want (I)}}} \implies Pr[M = m]Pr[C = c|M = m] \implies$$
$$\implies P[M = m] = \frac{Pr[C = c]Pr[M = m|C = c]}{Pr[C = c|M = m]}$$
$$\implies Pr[M = m] = Pr[M = m|C = c]$$

□

1.2 One Time Pad (OTP)

$K = M = C = \{0, 1\}^l$
$K \leftarrow \$\{0, 1\}^l$

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

Theorem 2. ***OTP*** *is perfectly secret* ◇

Proof. Fix any $m, c \in \{0, 1\}^l$:

$$Pr[Enc(k, m) = c] = Pr[k \oplus m = c] =$$
$$= Pr[k = c \oplus m] = 2^{-l}\text{(because k is uniform)}$$
$$= Pr[Enc(k, m') = c]\forall m'$$

This satisfies (III) □

Problems:

1. $|k| = |m|$ (the key must have the same length of the message)

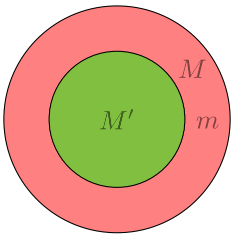
2. There must be one different key for each message (*One Time*).
   Otherwise:
   $c_1 = k \oplus m_1; c_2 = k \oplus m_2 \implies c_1 \oplus c_2 = m1 \oplus m_2$
   Not independent! (OTP is malleable[1])

Theorem 3. *In any perfectly secret SKE we must have* $|\mathcal{K}| \geq |\mathcal{M}|$ $\diamond$

Proof. Take M to be uniform over $\mathcal{M}$ and take c s.t. $Pr[C = c] > 0$.
Consider $M' = \{Dec(K, c) : k \in K\}$ (decryption of c with every key) and assume $|\mathcal{K}| < |\mathcal{M}|$

$$\implies |M'| \leq |K| \leq |M|$$
$$\implies \exists m \in M \setminus M'$$
$$\text{Now} Pr[M = m] = \frac{1}{|\mathcal{M}|}$$
$$Pr[\underbrace{M = m}_{m \text{ is not inside } M'} \quad | \quad \overbrace{C = c}^{\text{By assumption all the c "creates" M'}} \quad] = \emptyset$$

The above notion can be represented with the following picture:



Meaning that c is not an encryption of m! $\square$

[1]An encryption algorithm is "malleable" if it is possible to transform a ciphertext into another ciphertext which decrypts to a related plaintext. That is, given an encryption c of a plaintext m, it is possible to generate another valid ciphertext c', for a known Enc, Dec, without necessarily knowing or learning m.

Lesson 2

2.1 Message Authentication

T0D0 2: Msg auth image (bob, eve, tag)

2.1.1 Message Authentication Code (MAC or Tag)

Syntax:

- $\mathcal{K}$ = Key space, $\mathcal{M}$ = Message space, Φ = Tag space
- K is uniform over $\mathcal{K}$
- Tag: $\mathcal{K} \times \mathcal{M} \to \Phi$
- Verify: $\mathcal{K} \times \mathcal{M} \times \Phi \to \{0, 1\}$

Correctness:

$$\forall m \in \mathcal{M}, \forall k \in \mathcal{K}$$
$$\text{Verify}(k, m, \text{Tag}(k, m)) = 1$$

Assume Tag is <u>deterministic</u>, then $Vrf(k, m, \phi) = 1 \Leftrightarrow Tag(k, m) = \phi$
Security: Unforgeability $\to$ The attacker is not able to forge ϕ
Definition (ε-statistical one-time security): we say that Mac has ε-statistical one-time security if $\forall m, m' \in \mathcal{M} s.t. m \neq m', \forall \phi, phi' \in \Phi$

$$Pr[Mac(K, m') = \phi' | Mac(K, m) = \phi] \leq \varepsilon$$

T0D0 3: Not really sure of the Correctness of the following thins

Think of an "Unbounded" adversary choosing $m \in \mathcal{M}$ and obtaining $\phi = Mac(k, m)$. His goal is to forge on $m' \neq m'$ a tag ϕ' such that $Mac(k, m') = \phi'$ (with no knowledge of k).

Tool: **Pairwise Independent Hashing**

Definition 2. A family of (Hash) functions $H = \{h_s : \mathcal{M} \to \Phi\}_{s \in S}$ is Pairwise Independent if $\forall m, m' \in \mathcal{M} s.t. m \neq m'$ we have that $(h_s(m), h_s(m'))$ is uniform over Φ^2 for random choice of seed $s \in S$. Meaning that:

$$\forall m, m' s.t. m \neq m', \forall \phi, \phi' \in \Phi$$

$$Pr[h_s(m) = \phi \wedge h_s(m') = \phi'] = \frac{1}{|\Phi|^2}$$

◇

Construction: Let p be a prime number.
Define $h_{(a,b)}(x) = ax + b \mod p$ now $S = \mathcal{Z}_p^2, M = \Phi = Z_p = \{0, 1, ..., p-1\}$
Thus the following group will be defined $(\mathcal{Z}_p, +)$

Theorem 4. *Above H is Pairwise Independent* ◇

Proof.

$$\text{Fix} m, m' \in \mathcal{Z}_p s.t. m \neq m', \phi, \phi' \in \mathcal{Z}_p$$

$$Pr_{(a,b)}[h_{a,b}(m) = \phi \wedge h_{a,b}(m') = \phi'] =$$

$$= Pr_{(a,b)}[am + b = \phi \wedge am' + b = \phi'] =$$

$$= Pr_{(a,b)}[\begin{pmatrix} m & 1 \\ m' & 1 \end{pmatrix} \cdot \begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} \phi \\ \phi' \end{pmatrix}] =$$

$$= Pr_{(a,b)}[\begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} m & 1 \\ m' & 1 \end{pmatrix}^{-1} \cdot \begin{pmatrix} \phi \\ \phi' \end{pmatrix}] =$$

$$= \frac{1}{|\mathcal{Z}_p|^2} = \frac{1}{p^2}$$

Where $Pr_{(a,b)}$ denotes the probability over (a,b) random in $\mathcal{Z}_p$ □

Theorem 5. *Let $Mac(K, m) = h_k(m)$ for $\mathcal{H} = \{h_k : \mathcal{M} \to \Phi\}_k$ be Pairwise Independent. The Mac is $\frac{1}{|\Phi|}$-statistical one-time secure.* ◇

Proof.

$$\forall m \in \mathcal{M}, \phi \in \Phi$$

$$Pr[Mac(K, m) = \phi] = Pr_k[h_k(m) = \phi] = \frac{1}{|\Phi|}$$

$$\text{For any } m \neq m', \phi, \phi' \in \Phi$$

$$Pr[Mac(K, m) = \phi \wedge Mac(K, m') = \phi'] = \frac{1}{|\Phi|^2}$$

$$\implies Pr[Mac(K, m') = \phi' | Mac(K, m) = \phi] =$$

$$= \frac{\Pr[Mac(K, m) = \phi \wedge Mac(K, m') = \phi']}{Pr[Mac(K, m) = \phi]} =$$

$$= \frac{1}{|\Phi|^2} / \frac{1}{|\Phi|} = \frac{1}{|\Phi|}$$

□

Theorem 6. *Any t-time($\varepsilon = 2^{-\lambda}$)-statistical secure MAC has a key of size $(t+1)\lambda$ for any $\lambda > 0$.*
No proof ◇

Lesson 3

3.1 Randomness Extraction

Goal: extract "real" randomness from "imperfect" randomness.

Suppose to have a biased coin $Bs.t.Pr[B = 0] = p \neq \frac{1}{2}$ and
$Pr[B = 1] = 1 - p$.
Can you make it uniform?

3.1.1 Von Neumann Extractor

1. Sample $b_1 \leftarrow\!\!\$\ B; b_2 \leftarrow\!\!\$\ B$
2. If $b_1 = b_2$ go to 1.
3. Else:
    - If $b_1 = 0; b_2 = 1$ output 1
    - If $b_1 = 1; b_2 = 0$ output 0

Considerations:

- The probability of both cases in 3. is $p(p - 1)$
- The probability to reach 3. is $2p(p - 1)$
- $Pr[\text{Fail n-times}] = (1 - 2p(1 - p))^n$ which is very small

Goal: Let X be a Random Variable, I want to design an Extractor such that $Ext(X)$ is uniform. <u>PROBLEM:</u> Impossible as stated. What if X is constant? $\implies$ Add the requirement that X has uncertainty (ENTROPY)

Definition 3. MIN-ENTROPY of X (written as H_∞) is:

$$H_\infty(X) = -log_2 max_x Pr[X = x]$$

Where max_x is the x that maximize the probability. ◇

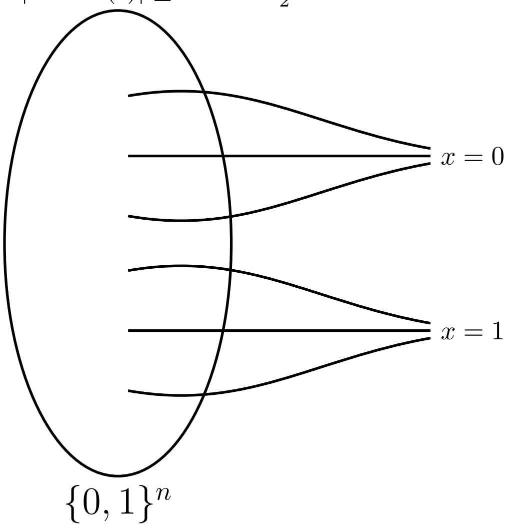
Example 1. Say $X = \{0, 1\}^n$

1. X is constant$(Pr[X = \bar{x}] = 1, Pr[X = x \neq \bar{x}] = 0)$
   $\implies H_\infty(X) = 0$

2. $X \equiv U_n \implies H_\infty(x) = n$ (since X is uniform $Pr[X = x] = 2^{-n}$)

Question: We want $Ext : \{0,1\}^n \rightarrow \{0,1\}$ working $\forall X s.t. H_\infty(X) \geq k$
Claim: Impossible even for $k = n - 1$

Proof. Let Ext be any candidate extractor, Let b be any binary value s.t. $|Ext^{-}1(b)|$ is maximal ($Ext^{-}1 : \{0,1\} \rightarrow \{0,1\}^n$)
$\implies |Ext^{-}1(b)| \geq 2^{n-1} = \frac{2^2}{2}$



Define X to be uniform over $Ext^{-}1(b)$ so $H_\infty(x) = n - 1$ but $Ext(X) = b$

CONSTANT.

T0D0 4: Help with the proof

□

Theorem 7. ***Way-Out-Seeded-Extractor*** *allow X to take SMALL uniformly random SEED(catalyst)*

$$Ext : \underbrace{\{0,1\}^d}_{seed(public)} \times \underbrace{\{0,1\}^n}_{input} \rightarrow \underbrace{\{0,1\}^m}_{output}$$

◇

Definition 4. We say that $Ext : \{0,1\}^d \times \{0,1\}^n \rightarrow \{0,1\}^l$ is a (k, ε)-extractor ("good") if $\forall x s.t. H_\infty(X) \geq k$ we have:

$$SD((S, Ext(S, x)), (S, U_l)) \leq \varepsilon$$

where $S \equiv U_d$, SD denotes the statistical distance and we are basically comparing two uniform values (x and U_l) ◇

Idea: Even if the seed is public the SD must not be influenced. Therefore the two compared variables should be uniform so that ε is small. In this way the output of the extractor can't be distinguished from a uniform string.

Definition 5. Statistical Distance: a distance between two random variables.
Say that X, X' are Random Variables over $\mathcal{X}$.
$SD(x, x') = \frac{1}{2} \sum_{x \in \mathcal{X}} |Pr[X = x] - Pr[X' = x]|$

T0D0 5: Image of the statistical distance

$\diamond$

T0D0 6: Leftover hash lemma

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 try (a polynomial number of) random values and hope to guess 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. If the overall success rate is $\frac{1}{poly(\lambda)}$ then we consider this attempt a feasible attack to the scheme, which makes the latter 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: $\frac{poly(\lambda)}{superpoly(\lambda)}$ [2]
>
> As λ grows, the denominator grows far faster than the numerator and the success probability will not be significant[3]."

[2] 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 finding the key to break the scheme

[3] "What exactly is a negligible (and non-negligible) function?" — Cryptography Stack Exchange

Definition

Let $\nu : \mathbb{N} \to [0,1]$ be a function. Then it is deemed **negligible** iff:

$$\forall p(\lambda) \in poly(\lambda) \implies \nu(\lambda) \in o(\frac{1}{p(\lambda)})$$

Exercise 8. Let $p(\lambda), p'(\lambda) \in poly(\lambda)$ and $\nu(\lambda), \nu'(\lambda) \in \text{negl}(\lambda)$. Then prove the following:

1. $p(\lambda) \cdot p'(\lambda) \in \text{poly}(\lambda)$
2. $\nu(\lambda) + \nu'(\lambda) \in \text{negl}(\lambda)$

Solution 1 (1.1)**.** T0D0 7: Questa soluzione usa disuguaglianze deboli; per essere negligibile una funzione dev'essere strettamente minore di un polinomiale inverso. Da approfondire

We need to show that for any $c \in \mathbb{N}$, then there is n_0 such that $\forall n > n_0 \implies h(n) \leq n^{-c}$.

So, consider an arbitrary $c \in \mathbb{N}$. Then, since $c + 1 \in \mathbb{N}$, and both f and g are negligible, there exists n_f and n_g such that:

$$\begin{aligned}
\forall n \geq n_f &\implies f(n) \leq n^{-(c+1)} \\
\forall n \geq n_g &\implies g(n) \leq n^{-(c+1)}
\end{aligned}$$

Fix $n_0 = \max(n_f, n_g)$. Then, since $n \geq n_0 \geq 2$, $\forall n \geq n_0$ we have:

$$\begin{aligned}
h(n) &= f(n) + g(n) \\
&\leq n^{-(c+1)} + n^{-(c+1)} \\
&= 2n^{-(c+1)} \\
&\leq n^{-c}
\end{aligned}$$

Thus we conclude $h(n)$ is negligible.

4.2 One-Way Functions

A One-Way Function (OWF) is a function that is "easy to compute", but "hard to invert".

Definition 6. Let $f : \{0,1\}^{n(\lambda)} \to 0, 1^{n(\lambda)}$ be a function. Then it is a OWF iff:

$$\forall \text{ PPT } \mathcal{A}\ \exists \nu(\lambda) \in \text{negl}(x) : \Pr\left[\text{GAME}^{\text{OWF}}_{f,A}(\lambda) = 1\right] \leq \nu(\lambda) \tag{4.1}$$

$\diamond$

Do note that the game does not look for the equality $x' = x$, but rather for the equality of their images computed by f.

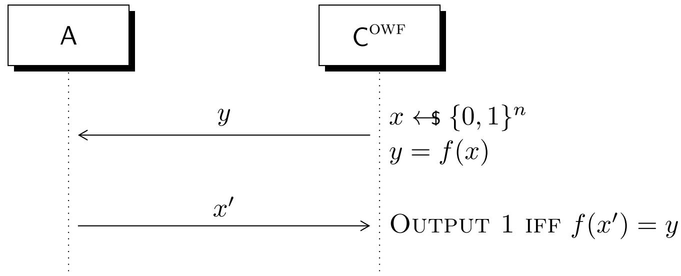


Figure 4.1: One-Way Function hardness

Exercise 9.

1. Show that there exists an inefficient adversary that wins $\textsc{Game}^{\text{OWF}}$ with probability 1
2. Show that there exists an efficient adversary that wins $\textsc{Game}^{\text{OWF}}$ with probability 2^{-n}

One-Way puzzles

A one-way function can be thought as a function which is very efficient in generating puzzles that are very hard to solve. Furthermore, the person generating the puzzle can efficiently verify the validity of a solution when one is given.

For a given couple $(\mathcal{P}_{\text{GEN}}, \mathcal{P}_{\text{VER}})$ of a puzzle generator and a puzzle verifier, we have:

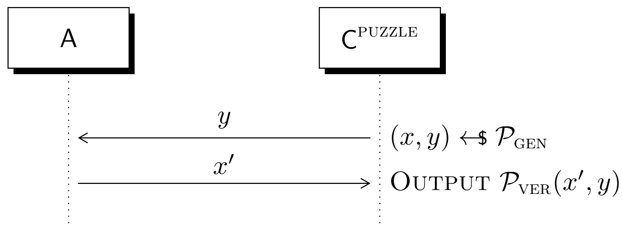


Figure 4.2: The puzzle game

So, we can say that one-way puzzle is a problem in NP, because the verifier enables efficient solution verification, but not in P because finding a solution from scratch is impractical by definition.

4.2.1 Impagliazzo's 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.

Imagine now that, if using one-way functions, the problem is $f(x)$, and its solution is x. According to Impagliazzo, we live in one of these possible worlds:

- *Algorithmica*: P = NP, meaning all efficiently verifiable problems are also efficiently solvable.

The professor can try as hard as possible to create a hard scheme, but he won't succeed because Gauss will always be able to efficiently break it using the verification procedure to compute the solution

- *Heuristica*: NP problems are hard to solve in the worst case but easy on average.

  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*: One-way functions exist but public-key cryptography is impractical

- *Cryptomania*: Public-key cryptography is possible: two parties can exchange secret messages over open channels

4.3 Computational Indistinguishability

Distribution ensembles $X = \{X_{\lambda\in\mathbb{N}}\}$ and $Y = \{Y_{\lambda\in\mathbb{N}}\}$ are distribution sequences.

Definition 7. X and Y are **computationally indistinguishable** ($X \approx_c Y$) if $\forall$ PPT $D, \exists\nu(\lambda) \in \text{negl}(\lambda)$ such that:

$$|\Pr[D(X_\lambda) = 1] - \Pr[D(Y_\lambda) = 1]| \leq \nu(\lambda)$$

$\diamond$

T0D0 8: AP181129-2344: There may be room for improvement, but I like how it's worded: it puts some unusual perspective into the cryptographic game, and it could be a good thing since it closely precedes our first reduction, and the whole hybrid argument mish-mash.

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. $\forall$ PPT $f,\ X \approx_c Y \implies f(x) \approx_c f(y)$ $\diamond$

Proof. We want to show that $f(x) \approx_{\mathrm{C}} f(y)$. So, let's suppose this property is not true.

Assume $\exists$ PPT f, A and some $p'(\lambda) \in \text{poly}(\lambda)$ such that

$$|\Pr[D'(f(x)) = 1] - \Pr[D'(f(y)) = 1]| \geq \frac{1}{p'(\lambda)} \tag{4.2}$$

.

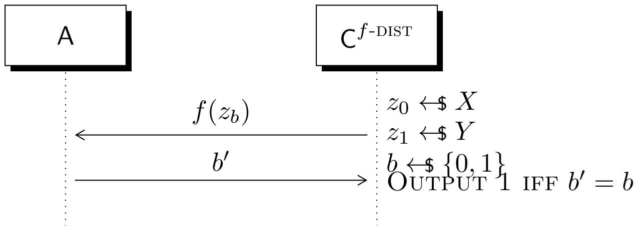


Figure 4.3: A distinguisher for f

In the game depicted by figure 4.3, the adversary can make use of f, because it is PPT. Thus, if such a distinguisher exists, then it can be used to distinguish X from Y, as shown in figure 4.4. So, if A could efficiently distinguish between $f(x)$ and $f(y)$, then D can efficiently distinguish between x and y. This contradicts the hypothesis $X \approx_{\mathrm{C}} Y$, which in turn means $f(X) \approx_{\mathrm{C}} f(Y)$. $\square$

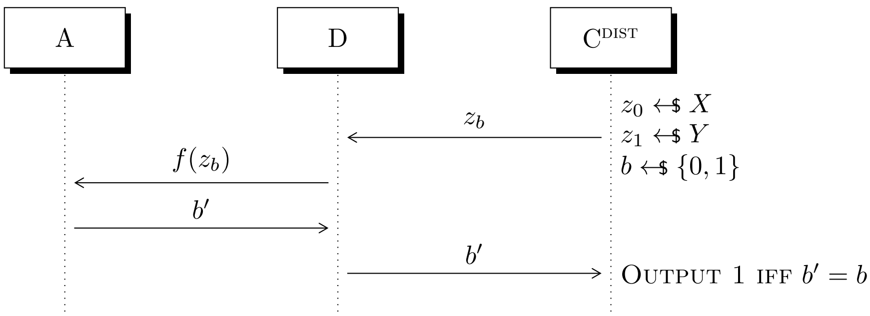


Figure 4.4: Distinguisher reduction

4.4 Pseudorandom Generators

A deterministic function $G : \{0,1\}^\lambda \to \{0,1\}^{\lambda+l(\lambda)}$ is called a PseudoRandom Generator, or PRG iff:

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

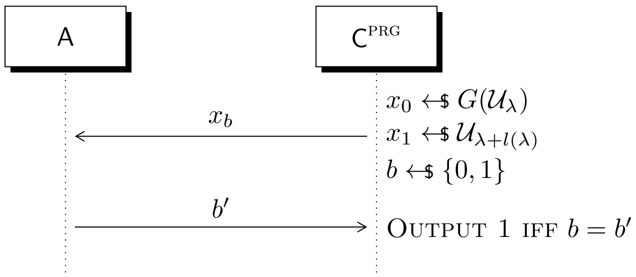


Figure 4.5: The pseudorandom game

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

Lesson 5

5.1 Stretching a PRG

Consider this algorithm that uses G to construct G^l, as depicted in figure 5.6:

1. Let $s_0 \leftarrow\!\$\ \{0,1\}^\lambda$
2. $\forall i \in [l(\lambda)]$, let $G(s_{i-1}) = (s_i, b_i)$, where b_i is the extra bit generated by G
3. Compose $(b_1, b_2, ..., b_{l(\lambda)}, s_{l(\lambda)})$ and output the result; the outputted string will be $\lambda + l(\lambda)$ bits long

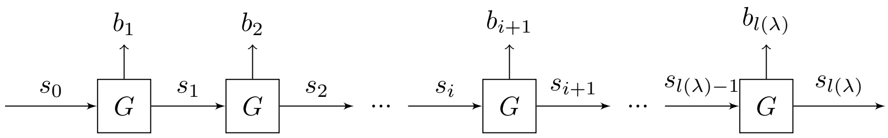


Figure 5.6: Constructing $G^{l(\lambda)}$ form $G(\lambda)$

To prove that this construct is a valid PRG, we will make use of a known technique for proving many other results, which relies heavily on reductions, and is called **the hybrid argument**.

Lemma 2. ***(Hybrid Argument)*** *Let $X = \{X_n\}, Y = \{Y_n\}, Z = \{Z_n\} : X \approx_{\text{C}} Y \wedge Y \approx_{\text{C}} Z$. Then $X \approx_{\text{C}} Z$.* $\diamond$

Proof. $\forall$ PPT D, by using the triangle inequality:

$$\begin{aligned}
&|\Pr[D(X_n) = 1] - \Pr[D(Z_n) = 1]| \\
&= |(\Pr[D(X_n) = 1] - \Pr[D(Y_n) = 1]) + (\Pr[D(Y_n) = 1] - \Pr[D(Z_n) = 1])| \\
&\leq |\Pr[D(X_n) = 1] - Pr[D(Y_n) = 1]| + |\Pr[D(Y_n) = 1] - \Pr[D(Z_n) = 1]| \\
&\leq negl(n) + negl(n) \\
&= negl(n)
\end{aligned}$$ $\square$

In essence, the hybrid argument proves that computational indistinguishability is transitive across a "hybrid" game, or by extension more than one. This property will be very useful in all future proofs.

Theorem 10. *If there exists a PRG $G(\lambda)$ with one bit stretch, then there exists a PRG $G^{l(\lambda)}$ with polynomial stretch relative to its input length:*

$$G : \{0,1\}^\lambda \to \{0,1\}^{\lambda+1} \implies \forall l(\lambda) \in \text{poly}\lambda\, \exists G^l : \{0,1\}^\lambda \to \{0,1\}^{\lambda+l(\lambda)}$$ $\diamond$

Proof. First off, do observe that, since both G and l are polynomial in λ, then $G^{l(\lambda)}$, which combines G l-many times is polynomial in λ too. To prove that $G^{l(\lambda)}$ is indeed a PRG, we will apply the hybrid argument. The hybrids are defined as:

- $H^0_\lambda := G^{l(\lambda)}(U_\lambda)$, which is the original construct
- $H^i_\lambda := \begin{cases} b_1, ..., b_i \leftarrow\!\!\$ \ \{0,1\} \\ s_i \leftarrow \{0,1\}^{\lambda+i} \\ (b_{i+1}, ..., b_{l(\lambda)}, s_{l(\lambda)}) := G^{l(\lambda)-i}(s_i) \end{cases}$
- $H^{l(\lambda)}_\lambda := U_{\lambda+l}$

Focusing on two subsequent generic hybrids, as shown in figures 5.7 and 5.8, it can be observed that the only difference between the two resides in how b_{i+1} is generated: in H^i it comes from an instance of G, whereas in H^{i+1} is chosen at random. H^0_λ is the starting point where all bits are pseudorandom, which coincides with the $G^{l(\lambda)}$, and $H^{l(\lambda)}_\lambda$ will generate a totally random string.

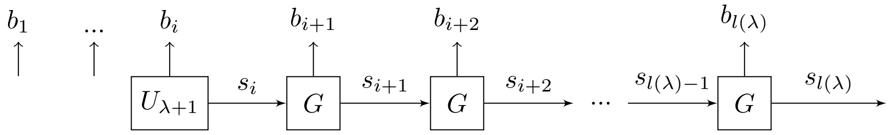


Figure 5.7: H^i_λ

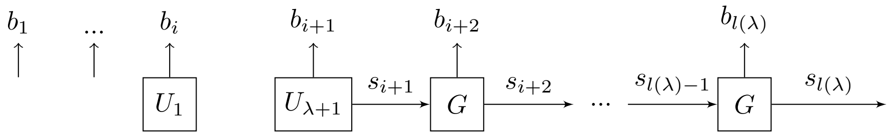


Figure 5.8: H^{i+1}_λ

So let's fix a step i in the gradual substitution, and consider the following function f_i:

$$f_i(s_{i+1}, b_{i+1}) = (b_1, \dots, b_i, b_{i+1}, b_{i+2}, \dots, b_{l(\lambda)}, s_{l(\lambda)})$$

where the first i bits are chosen uniformly at random, and the remaining ones are obtained by subsequent applications of G ($\forall j \in \{i+2, \dots, l(\lambda)\} \implies G(s_{j-1}) = (s_j, b_j)$). It can be observed that:

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

Since by PRG definition $G(U_\lambda) \approx_{\mathrm{C}} U_{\lambda+1}$, by lemma 1, we can deduce that $f_i(G(U_\lambda)) \approx_{\mathrm{C}} f_i(U_{\lambda+1})$, which in turn, by how f is defined, implies $H^i \approx_{\mathrm{C}} H^{i+1}$. This holds for an arbitrary choice of i, so by extension:

$$G^{l(\lambda)}(U_\lambda) = H^0_\lambda \approx_{\mathrm{C}} H^1_\lambda \approx_{\mathrm{C}} \dots \approx_{\mathrm{C}} H^{l(\lambda)}_\lambda = U_{\lambda+l(\lambda)}$$

which proves that G^l is indeed a PRG. $\square$

Proof. *(Contradiction)*: This is an alternate proof that, instead of looking for a function f to model hybrid transitioning, aims for a contradiction.

Suppose G^l is not a PRG; then there must be a point in the hybrid chain $H_\lambda^0 \approx_{\mathrm{C}} \ldots \approx_{\mathrm{C}} H_\lambda^l$ where $H_\lambda^i \not\approx_{\mathrm{C}} H_\lambda^{i+1}$. Thus there exists a distinguisher D' able to tell apart H_λ^i from H_λ^{i+1}, as shown in figure 5.9:

$$\exists i \in [0, l], \exists \text{ PPT } D' : |\Pr[D'(H_\lambda^i) = 1] - \Pr[D'(H_\lambda^{i+l}) = 1]| \geq \frac{1}{\mathrm{poly}(\lambda)}$$

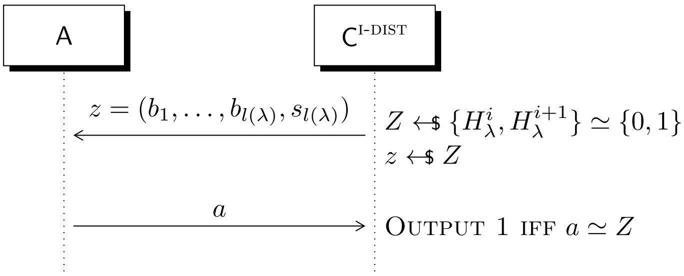


Figure 5.9: Distinguisher for H_λ^i and H_λ^{i+1}

If such a distinguisher exists, it can be also used to distinguish an output of G from a $\lambda + 1$ uniform string by "crafting" a suitable bit sequence, which will distribute exactly as the hybrids in question, as shown in the reduction in figure 5.10. This contradicts the hypothesis of f being a PRG, which by definition is to be indistinguishable from a truly random distribution. Therefore, G^l is indeed a PRG.

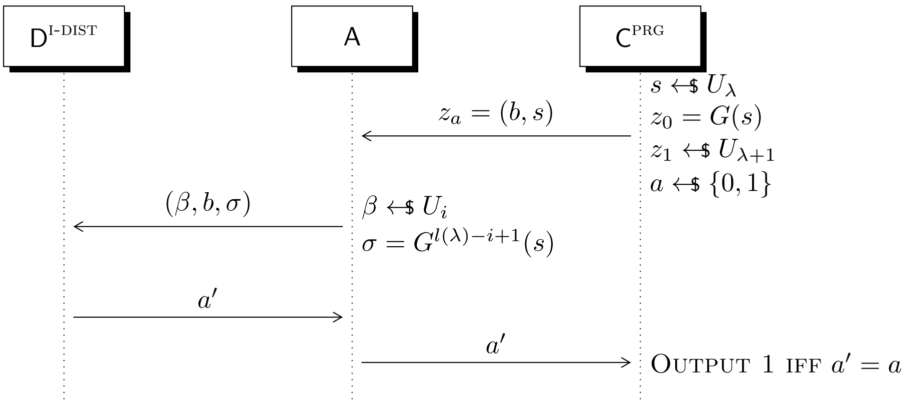


Figure 5.10: Reducing to a distinguisher for G, where $\beta = (b_1, \ldots, b_{i-1})$ and $\sigma = (b_{i+1}, \ldots, b_{l(\lambda)}, s_{l(\lambda)})$

□

5.2 Hardcore predicate

We've seen how to reuse a PRG in order to obtain an arbitrary length of pseudorandom bits starting from a PRG with one bit stretch. How to construct a 1-bit PRG?

Consider a typical 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 it always true that, given f, the first bit of $f(x)$ is hard to compute $\forall x$?

Example 2. Given an OWF f, then $f'(x) = x_0||f(x)$ is a OWF.

Definition 1. A polynomial time function $\mathfrak{hc} : \{0,1\}^n \to \{0,1\}$ is **hard core** for a given function $f : \{0,1\}^n \to \{0,1\}^n$ if, $\forall$ PPT A:

$$\Pr(A(f(x)) = \mathfrak{hc}(x) | x \leftarrow\!\$ \{0,1\}^n) \in \text{negl}(\lambda))$$

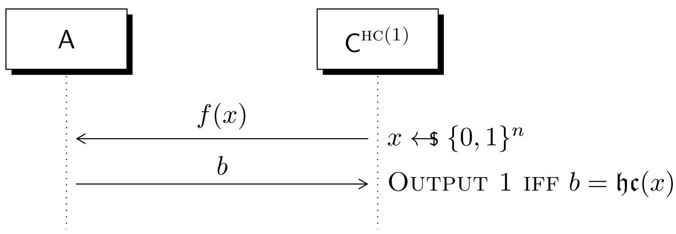


Figure 5.11: The hardcore game, f and $\mathfrak{hc}$ are known

There is also an alternative definition:

Definition 2. A polynomial time function $\mathfrak{hc} : \{0,1\}^n \to \{0,1\}$ is hard-core for a function f iff:

$$(f(x), h(x)) \approx_{\mathrm{C}} (f(x), b)$$

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

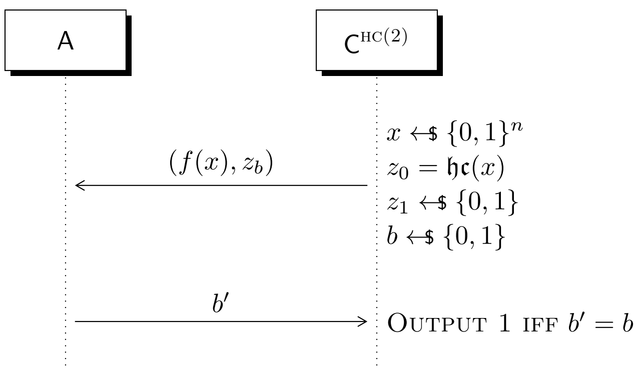


Figure 5.12: Another hardcore game, f and $\mathfrak{hc}$ are known

Having made this definition, some observations are in order:

Claim 1. There is no *universal* hardcore predicate $\mathfrak{HC}$.

Proof. Suppose there exists such a predicate $\mathfrak{HC}$. Let $f'(x) = \mathfrak{HC}(x)||f(x)$ for a given function f. Then $\mathfrak{HC}$ cannot be a hardcore predicate of f', because any image obtained by f reveals the predicate image itself. This contradicts the universailty assumption. □

Nevertheless, it has been proven always possible to construct a hardcore predicate for a OWF, given another OWF:

Theorem 11 (Goldreich-Levin, '99)**.** *Let f be a* OWF *and consider $g(x, r) = (f(x), r)$ for $r \in \{0,1\}^n$. Then g is a* OWF *and:*

$$h(x,r) =< x, r >= \sum_{i=1}^{n} x_i r_i \mod 2 = \bigoplus_{i=1}^{n} x_i r_i$$

is hard core for g. ◇

Proof. T0D0 9: TO BE COMPLETED (...did we actually do this? è una bella menata dimostrare questo)

□

Exercise 12. Prove that g is a OWF if f is a OWF (Hint: do a reduction).

5.3 One Way Permutation

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

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

Corollary 1. If $f : \{0,1\}^n \to \{0,1\}^n$ 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

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

□

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

Lesson 6

6.1 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 experiment depicted in figure 6.13 for the encryption scheme $\Pi = (Enc, Dec)$, where the adversary wins the game when the challenger outputs 1.

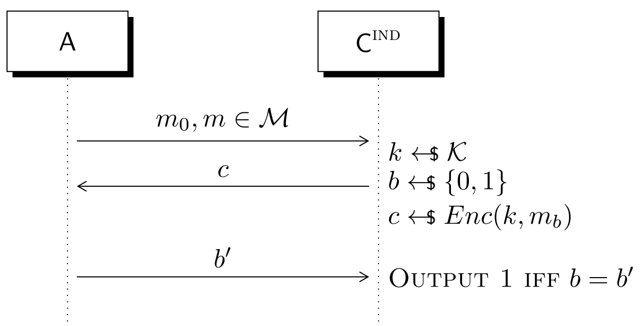


Figure 6.13: $\textsc{Game}^{\text{IND}}_{\Pi,\mathsf{A}}(\lambda, b)$

Definition 8. Π is said to be computationally **one time secure** iff:

$$\textsc{Game}^{\text{IND}}_{\Pi,\mathsf{A}}(\lambda, 0) \approx_c \textsc{Game}^{\text{IND}}_{\Pi,\mathsf{A}}(\lambda, 1)^4$$

[4] $\textsc{Game}^{\text{IND}}_{\Pi,\mathsf{A}}$ refers to the indistinguishability of the messages sent by A during the game

or, alternatively $\forall$ PPT A:

$$|\Pr[\text{GAME}^{\text{IND}}_{\Pi,\mathsf{A}}(\lambda, 0) = 1] - \Pr[\text{GAME}^{\text{IND}}_{\Pi,\mathsf{A}}(\lambda, 1) = 1]| \in \text{negl}(\lambda)$$

$\diamond$

This last definition is compliant with the three properties exposed beforehand. In particular, if a scheme is **one time secure**, then it has each one of these 3 properties:

T0D0 10: TO BE REVIEWED

- **compliance with point 1**: suppose point 1 is not valid, and k is not hard to discover for A. Then A is able to perfectly distinguish m and m_0 every time, therefore the scheme cannot be one time secure;
- **compliance with point 2**: suppose point 2 is not valid, and then the encrypted message can be easily discovered by A. Then, as before, A can win every game with $\Pr[1]$, 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. A could forge $m_0 = m$ such that they have the same bits but just the first is different. When 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.

What is not **two time secure**? Here is an apparently safe scheme, $\Pi_\oplus$, where $G : \{0,1\}^\lambda \to \{0,1\}^l$ is a PRG:

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

To prove that $\Pi_\oplus$ is not **two time secure**, assume an adversary A knows the pair $(\bar{m}, \bar{c} = G(k) \oplus \bar{m})$. Then, given any ciphertext $c = G(k) \oplus m$, where m is unknown, A can efficiently compute the following:

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

and thus, by XORing the result with $\bar{m}$, obtain the entire unknown message[5]. Nevertheless, this scheme is still **one time secure**:

Theorem 13. *If G is a* PRG*, then $\Pi_\oplus$ is computationally* ***one time secure*** $\diamond$

Proof. We need to show that, $\forall$ PPT A:

$$\text{GAME}^{\text{IND}}_{\Pi_\oplus,\mathsf{A}}(\lambda, 0) \approx_{\text{C}} \text{GAME}^{\text{IND}}_{\Pi_\oplus,\mathsf{A}}(\lambda, 1)$$

Consider the hybrid game in figure 6.14, where the original encryption routine is changed to use a random value instead of $G(k)$[6]. Compare with the

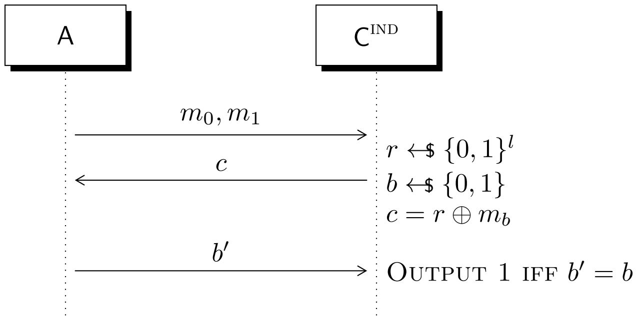


Figure 6.14: $\text{HYB}_{\Pi_{\oplus},\mathsf{A}}(\lambda, b)$

original **one time secure** definition in figure 6.13 to observe that it perfectly matches.

Lemma 3. $\text{HYB}_{\Pi_{\oplus},\mathsf{A}}(\lambda, 0) \equiv \text{HYB}_{\Pi_{\oplus},\mathsf{A}}(\lambda, 1)$ $\diamond$

Proof. This is true because distribution of c does not depend on $b \in \{0, 1\}$.

TODO 11: Eh?

$\square$

Lemma 4. $\forall b \in \{0,1\}, \text{HYB}_{\Pi_{\oplus},\mathsf{A}}(\lambda, b) \approx_{\text{C}} \text{GAME}^{\text{IND}}_{\Pi_{\oplus},\mathsf{A}}(\lambda, b)$ $\diamond$

Proof. The proof proceeds by reduction as depicted in figure 6.15, by assuming there exists a distinguisher D^{IND} for $c = G(k) \oplus m_b$ and $c = r \oplus m_b$, and using it to break the PRG itself.

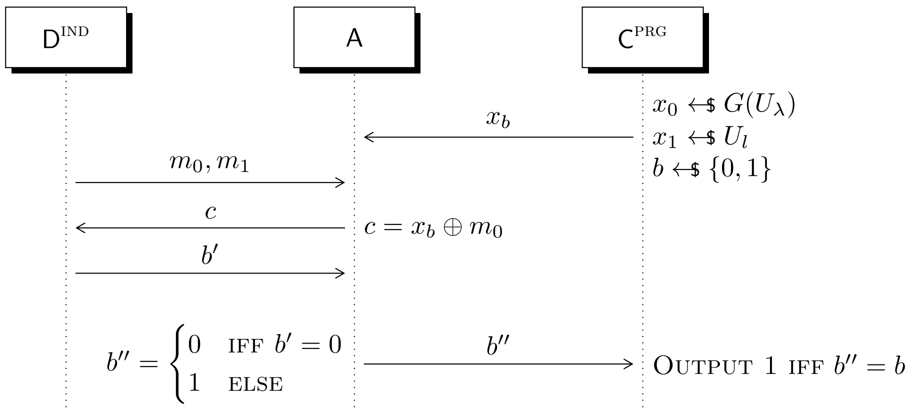


Figure 6.15: Reducing to breaking a PRG

[5]This example models a technique called "Chosen Plaintext Attack", which will be discussed in depth later

[6]The observant student may recognize that this modification yields exactly the One-Time Pad encryption scheme discussed in lesson 1

Do observe that the adversary always encrypts m_0, but this game can be modified for m_1 by switching b'''s assignments without loss of generality; the crucial point is to see if the distinguisher guesses which message the adversary has encrypted:

- if D^{IND} is wrong, then A can deduce with high probability that the value given by C^{PRG} is in fact truly random;
- if D^{IND} is right, then A has a sensibly greater probability of having received a pseudorandom value from C^{PRG}.

Either way, by the existence of D^{IND}, A gains an edge in efficiently breaking a PRG, which is absurd. $\square$

Finally, by the two above lemmas, the proof can be concluded by transitivity:

$$\text{GAME}^{\text{IND}}_{\Pi_\oplus,\mathsf{A}}(\lambda,0) \approx_{\text{C}} \text{HYB}_{\Pi_\oplus,\mathsf{A}}(\lambda,0) \equiv \text{HYB}_{\Pi_\oplus,\mathsf{A}}(\lambda,1) \approx_{\text{C}} \text{GAME}^{\text{IND}}_{\Pi_\oplus,\mathsf{A}}(\lambda,1)$$

$\square$

6.2 Pseudorandom functions

PRGs are practically used as a stepping stone for building *PseudoRandom Functions* (PRF), which in turn form the basis for most, if not all cryptographic schemes. Before introducing what a PRF formally is, a more intuitive definition of a *truly random function* is given.

Definition 9. A random function $R : \{0,1\}^n \to \{0,1\}^l$ is a function that, depending on what is known about its previous applications:

- if x is "fresh" (in formal terms, R has never been applied to x beforehand), then a value y is chosen UAR from R's codomain, and it is permanently associated as the image of x in R[7];
- if x is not fresh, then $R(x)$ is directly returned instead.

$\diamond$

It sholud be noted that such functions occupy too much space, if we were to memorize it anywhere: supposing all the possible outputs of R have been generated and stored as an array in memory, its size in bits will be $2^n l$.

So we look for a kind of function which is "random", but does not require to be memorized as a whole map, while still being efficiently computable. This is where pseudorandomness comes in: a function is deemed pseudorandom (a PRF) when it is (computationally) indistinguishable from a truly random one.

[7]this property is also called *lazy sampling*

Usually, PRFs come as function families F_k, where k is a key that identifies a single function inside the family itself[8].

With this in mind, let $\mathcal{F} = \{F_k : \{0,1\}^{n(\lambda)} \to \{0,1\}^{l(\lambda)}\}_{k \in \{0,1\}^\lambda}$ be the *keyed* function family that defines a PRF. Consider the two games in figure 6.16, each one involving respectively $\mathcal{F}$ and a random function R, where $\mathcal{R} = \{R : \{0,1\}^n \to \{0,1\}^l\}$, also written as $\mathcal{R}(\lambda, n, l)$ is the set of all random functions from n-bit strings to l-bit strings.

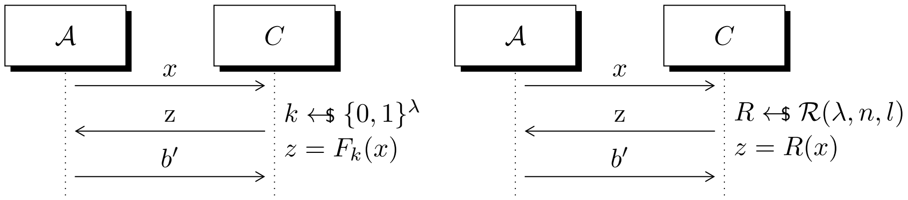


Figure 6.16: $Real_{\mathcal{F},\mathcal{A}}(\lambda)$ vs $Rand_{\mathcal{R},\mathcal{A}}(\lambda)$

$b' \in \{0,1\}$ 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 10. F_k is a PRF iff:

$$Real_{\mathcal{F},\mathcal{A}}(\lambda) \approx_{\text{C}} Rand_{\mathcal{R},\mathcal{A}}(\lambda)$$

◇

Exercise 14. Show that no PRG is secure against **unbounded attackers**.

Exercise 15. Show the same as above, but for any PRF instead.

6.2.1 GGM-tree

In this section, it is described how to construct a PRF starting from a given PRG, using a technique devised form Goldreich, Goldwasser and Micali, called the GGM-*tree*.

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

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

Consider this GGM-tree, which describes how G will be used in the PRF construction:

Figure 6.17: The GGM-tree for G

Build $\mathcal{F} = \{F_k : \{0,1\}^n \to \{0,1\}^\lambda\}$ such that:

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

◇

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

[8]Those unfamiliar with the "currying" technique may see the key as an additional argument to pass to the "main" function F

Lesson 7

WARNING : I was absent

Theorem 16. *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. T0D0 12: 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 5. *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:

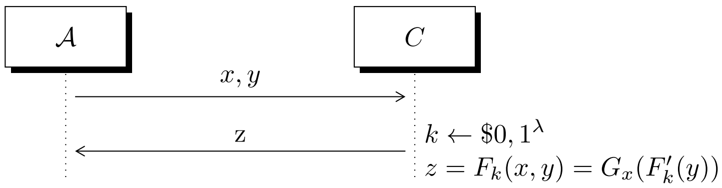


Figure 7.18: $\mathcal{HYB}^0_{\mathcal{F},\mathcal{A}}(\lambda)$

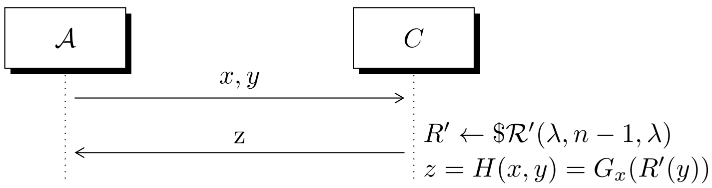


Figure 7.19: $\mathcal{HYB}^1_{\mathcal{R}',G,\mathcal{A}}(\lambda)$

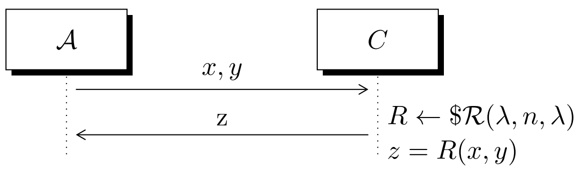
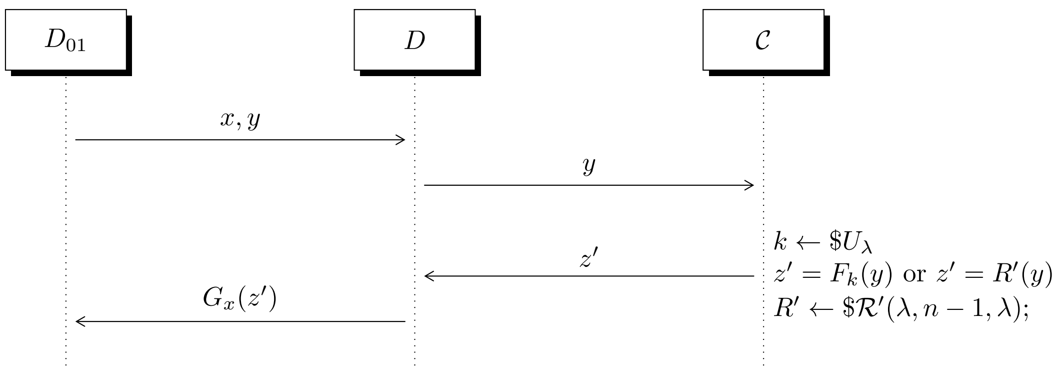


Figure 7.20: $\mathcal{HYB}^2_{\mathcal{R},\mathcal{A}}(\lambda)$

Lemma 6. $\mathcal{HYB}^0 \approx_c \mathcal{HYB}^1 \approx_c \mathcal{HYB}^2$. $\diamond$

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 ... □

7.1 Cpa-security

Now it's time to define a stronger notion of security, which is widely used in cryptology for first assessments on cryptographic schemes. Let $\Pi := (Enc, Dec)$ be a SKE scheme, and consider the game depicted in figure 7.21. Observe that this time, the adversary can "query" the challenger for the ciphertexts of any messages of his choice, with the only reasonable restriction that the query amount must be polynomially bound by λ. This kind of game/attack is called the *Chosen Plaintext Attack*, because of the adversary's capability of obtaining ciphertexts from messages. The usual victory conditions found in n-time security games, which are based on ciphertext distinguishability, apply.

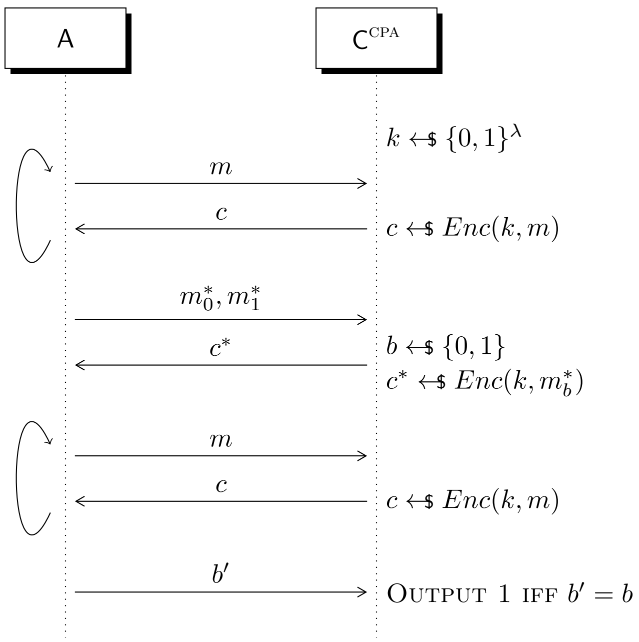


Figure 7.21: The CPA-security game: $\text{Game}^{\text{CPA}}_{\Pi,\mathsf{A}}(\lambda, b)$

Definition 11. A scheme is CPA-secure if $\text{Game}^{\text{CPA}}_{\Pi,\mathsf{A}}(\lambda, 0) \approx_C \text{Game}^{\text{CPA}}_{\Pi,\mathsf{A}}(\lambda, 1)$ ⋄

Having given this definition of security, recall the $\Pi_\oplus$ scheme defined in the previous lesson. It is easy to see that $\Pi_\oplus$ is not CPA-secure for the same reasons that it is not computationally 2-time secure; however this example sheds some new light about a deeper problem:

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

This is true, because nothing prevents the adversary from asking the challenger to encrypt either m_0 or m_1, or even both, before starting the actual challenge; just as in the 2-time case for $\Pi_\oplus$, he will know the messages' ciphertexts in advance, so he will be able to tell which message the challenger has encrypted every time. The solution for obtaining a CPA-secure encryption scheme consists of returning different ciphertexts for the same message, even better if they look random. This can be achieved by using PRFs.

Consider the following SKE scheme $\Pi_\mathcal{F}$, with $\mathcal{F} = \{F_k : \{0,1\}^n \to \{0,1\}^l\}$ being a PRF:

- $Enc(k, m) = (c_1, c_2) = (r, F_k(r) \oplus m)$, where $k \leftarrow\$ \{0,1\}^\lambda$ and $r \leftarrow\$ \{0,1\}^n$
- $Dec(k, (c_1, c_2)) = F_k(c_1) \oplus c_2$

Observe that the random value r is part of the ciphertext, making it long $n + l$ bits; also more importantly, the adversary can and will always see r. The key k though, which gives a *flavour* to the PRF, is still secret.

Theorem 17. *If $\mathcal{F}$ is a* PRF*, then $\Pi_\mathcal{F}$ is* CPA*-secure.* ◇

Proof. We have to prove that $\text{GAME}^{\text{CPA}}_{\Pi_\mathcal{F},\mathsf{A}}(\lambda, 0) \approx_C \text{GAME}^{\text{CPA}}_{\Pi_\mathcal{F},\mathsf{A}}(\lambda, 1)$; to this end, the hybrid argument will be used. Let the first hybrid $\text{HYB}^0_{\Pi,\mathsf{A}}$ be the original game, the second hybrid $\text{HYB}^1_{\Pi,\mathsf{A}}$ will have a different encryption routine:

- $r \leftarrow\$ \{0,1\}^n$
- $R \leftarrow\$ \mathcal{R}(\lambda, n, l)$
- $c = (r, R(r) \oplus m)$, where m is the plaintext to be encrypted

and then the last hybrid $\text{HYB}^2_{\Pi,\mathsf{A}}$ will simply output $(r_1, r_2) pickUARU_{n+l}$.

Lemma 7. $\forall b \in \{0,1\} \implies \text{HYB}^0_{\Pi,A}(\lambda, b) \approx_C \text{HYB}^1_{\Pi,A}(\lambda, b)$. ◇

Proof. As usual, the proof is by reduction: suppose there exists a distinguisher D capable of telling the two hybrids apart; then D can be used to break $\mathcal{F}$'s property of being a PRF. The way to use D is to make it play a CPA-like game, as shown in figure 7.22[9], where the adversary attempting to break $\mathcal{F}$ decides which message to encrypt between m_0 and m_1 beforehand, and checks whether D guesses which message has been encrypted. Either way, the adversary can get a sensible probability gain in guessing if the received values from the challenger were random, or generated by $\mathcal{F}$. Thus, assuming such D exists, A can efficiently break $\mathcal{F}$, which is absurd.

□

Lemma 8. $\forall b \in \{0,1\} \implies \text{HYB}^1_{\Pi,A}(\lambda, b) \approx_C \text{HYB}^2_{\Pi,A}(\lambda, b)$. ◇

[9] An observant student may notice a striking similarity with a previously exposed reduction in figure 6.15

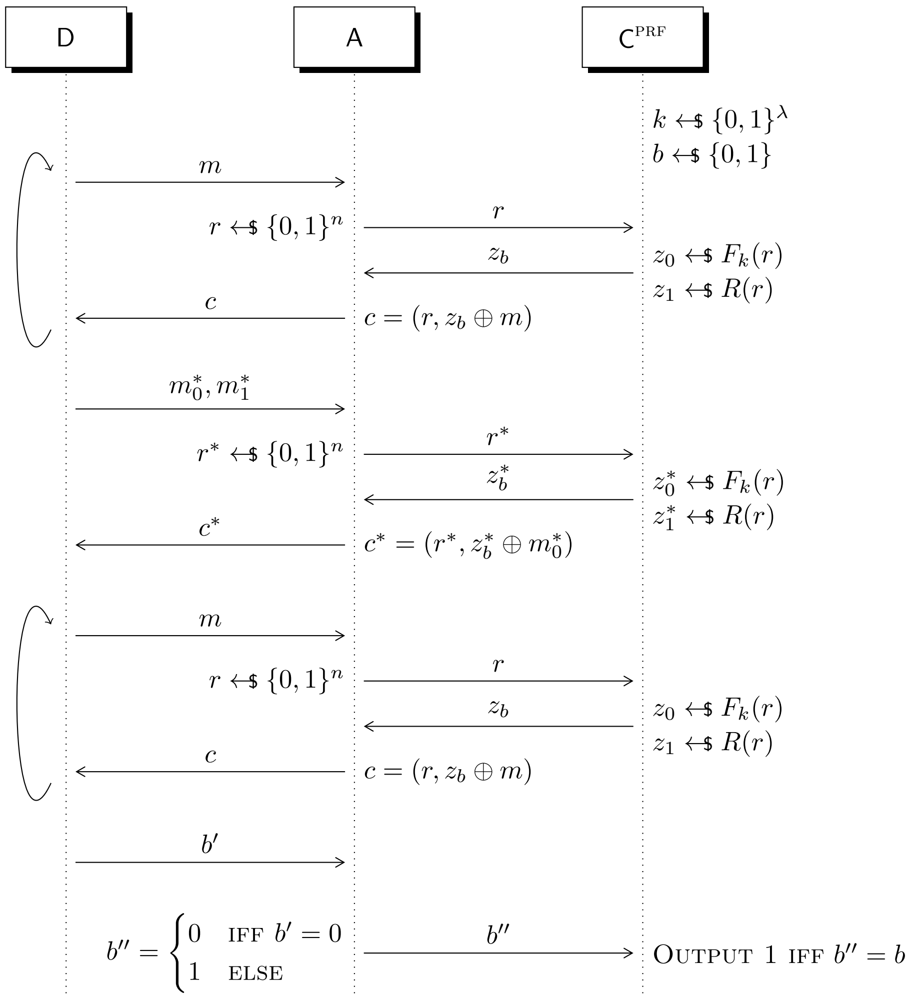


Figure 7.22: Breaking a PRF, for fixed message choice of m_0

Proof. Firstly, it can be safely assumed that any ciphertext $(r_i, R(r_i) \oplus m_b)$ distributes equivalently with its own sub-value $R(r_i)$, because of R's true randomness, and independency from m_b.

Having said that, the two hybrids apparently distribute uniformly, making them perfectly equivalent; however there is a caveat: if both games are run and one value $\bar{r}$ is queried twice in both runs, then om the second query, the adversary will receive the same image in $\textsc{Hyb}^1_{\Pi,\mathsf{A}}$, but almost certainly a different one in $\textsc{Hyb}^2_{\Pi,\mathsf{A}}$. This is because the first hybrid uses a function, which is deterministic by its nature, whereas the image in the second hybrid is picked completely randomly from the codomain. Nevertheless, this sneaky issue about "collisions" can be proven to happen with negligible probability.

Call $\textsc{Repeat}$ this collision event on $\bar{r}$ between 2 consecutive games. Then:

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

which proves that the $\textsc{Repeat}$ influences negligibly on the two hybrids' equivalence. Thus $\textsc{Hyb}^1_{\Pi,\mathsf{A}}(\lambda, b) \approx_{\mathrm{C}} \textsc{Hyb}^2_{\Pi,\mathsf{A}}(\lambda, b)$[10].

□

With the above lemmas, and observing that $\textsc{Hyb}^2_{\Pi,\mathsf{A}}(\lambda, 0) \equiv \textsc{Hyb}^2_{\Pi,\mathsf{A}}(\lambda, 1)$, we can reach the conclusion that $\textsc{Hyb}^0_{\Pi,\mathsf{A}}(\lambda, 0) \approx_{\mathrm{C}} \textsc{Hyb}^0_{\Pi,\mathsf{A}}(\lambda, 1)$, which is what we wanted to demonstrate.

□

[10]Do note that the hybrids lose their originally supposed perfect equivalence ($\textsc{Hyb}^1_{\Pi,\mathsf{A}}(\lambda, b) \equiv \textsc{Hyb}^2_{\Pi,\mathsf{A}}(\lambda, b)$) because of the $\textsc{Repeat}$ event. The lemma, though, is still proven.

Lesson 8

8.1 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 blockcipher (we can think at this like a PRP).

8.1.1 Electronic CodeBook (ECB)

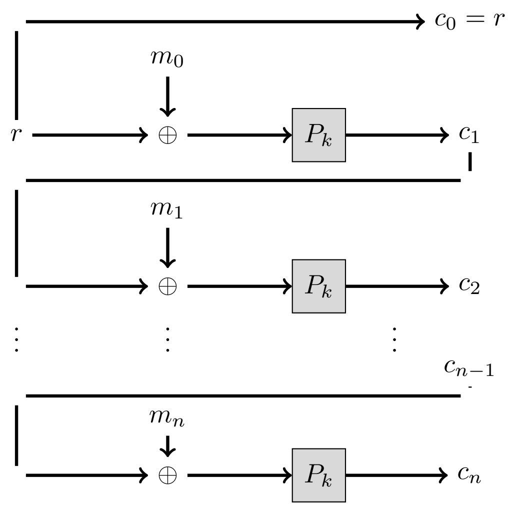
The principle is that I split the plaintext into many blocks (of length n) and I encrypt all of them individually (maybe in parallel):

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

So the final ciphertext will be: $c = c_1, c_2, ..., c_t$.

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

8.1.2 Cipher block chaining (CBC)



Here P_k needs to be a permutation since we must be able to do P_k^{-1}.

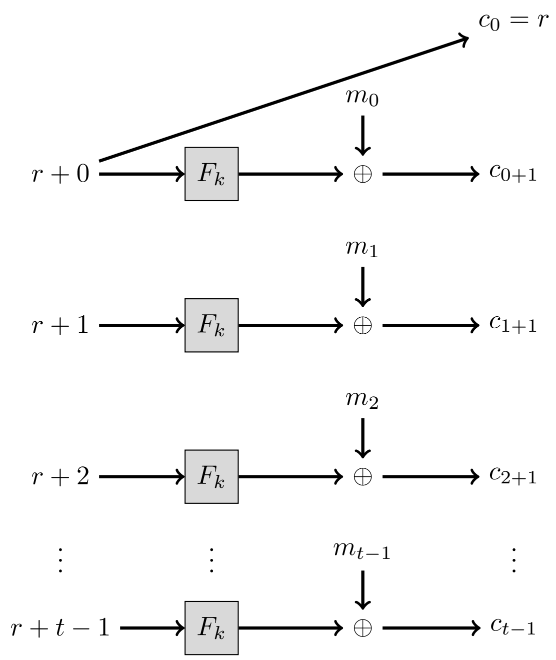
To encrypt we simply do $\forall i \in [1, t]$

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

The decryption is given by:

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

8.1.3 Counter Mode (CTR)



Here we "don't care" about the invertibility of F_k since we can simply re-compute it on r given the key.

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 stream cipher, because 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.

$$F_k(c_0) \oplus c_1 = m_0 \implies F_k(c_0 + i) \oplus c_i = m_1$$

Theorem 18. *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.* $\diamond$

.

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:

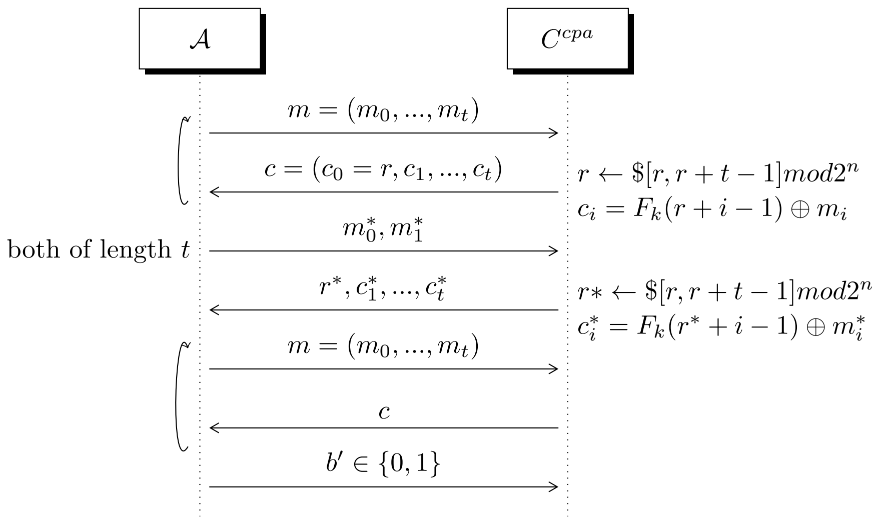


Figure 8.23: $Game^{cpa}_{CTR,\mathcal{A}}(\lambda, b)$

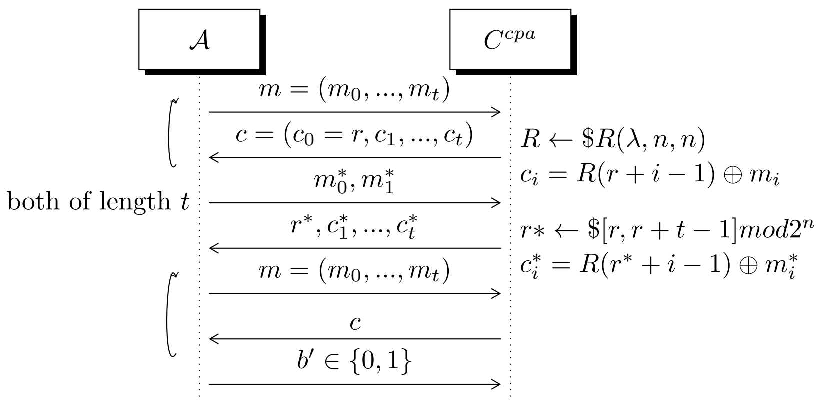


Figure 8.24: $\mathcal{HYB}_1(\lambda, b)$

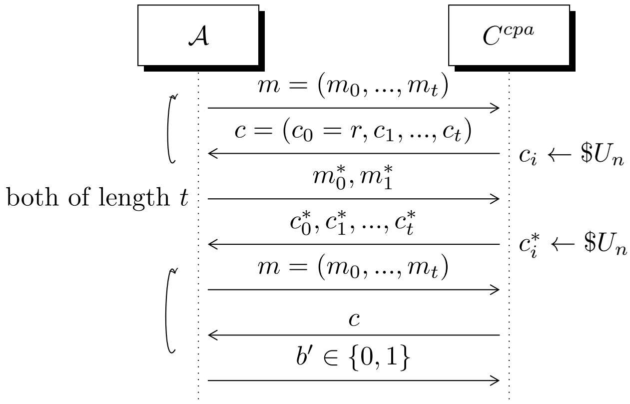


Figure 8.25: $\mathcal{HYB}_2(\lambda, b)$

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

Proof.

Exercise 19.

Lemma 9. *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 10. $\mathcal{HYB}_1(\lambda, b) \approx_c \mathcal{HYB}_2(\lambda), \forall b \in \{0, 1\}$ ◇

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 12. 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** ,

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

$\square$

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

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

.

$\square$

Lesson 9

9.1 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:

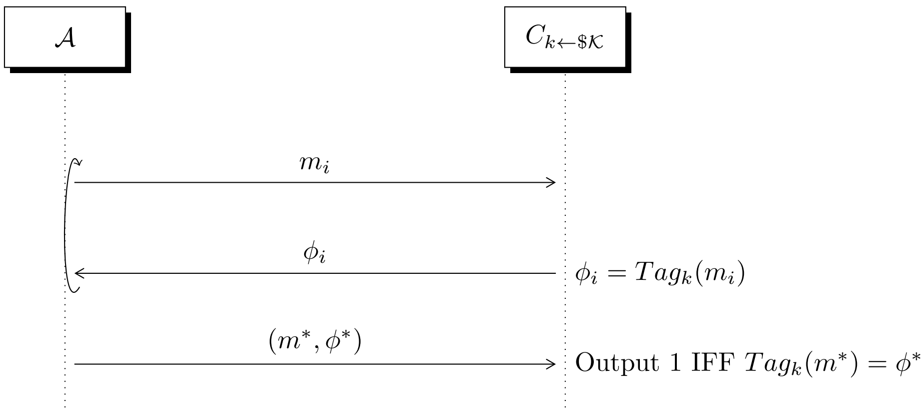


Figure 9.26: $Game^{ufcma}_{\Pi,\mathcal{A}}(\lambda)$

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 13. 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 20. *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:

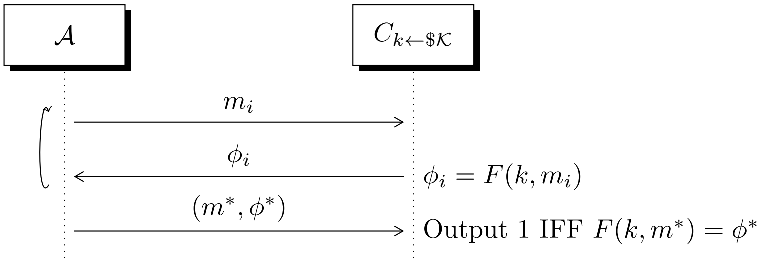


Figure 9.27: $Game^{ufcma}_{\mathcal{F},\mathcal{A}}(\lambda)$

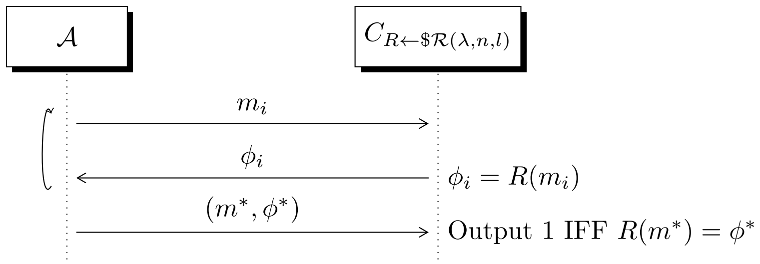
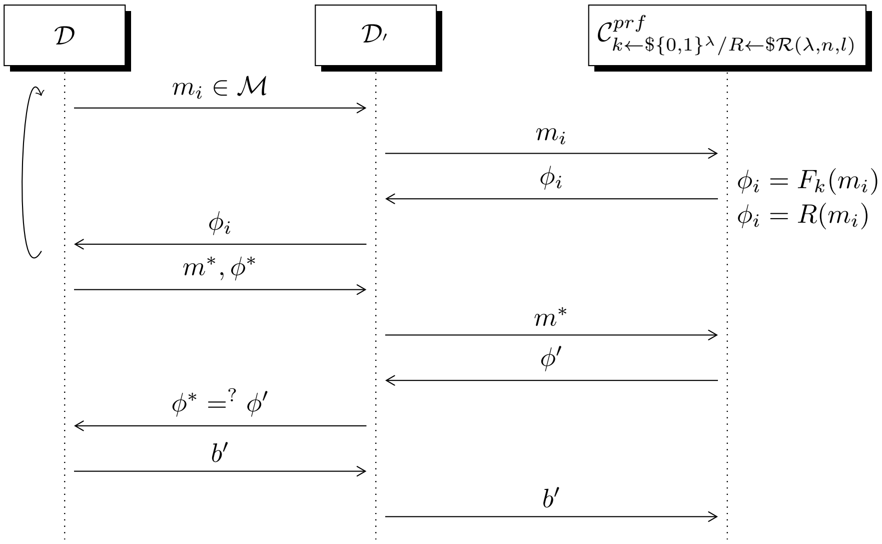


Figure 9.28: $\mathcal{HYB}^{ufcma}(\lambda)$

Lemma 11. $Game^{ufcma}_{\mathcal{F},\mathcal{A}}(\lambda) \approx_c \mathcal{HYB}^{ufcma}_1(\lambda)$ $\diamond$

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

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 $\frac{1}{2^l}$. □

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.

9.2 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 14. $\mathcal{H}$ 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 21. 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 22. *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 $t \geq 1$).* ◇

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:

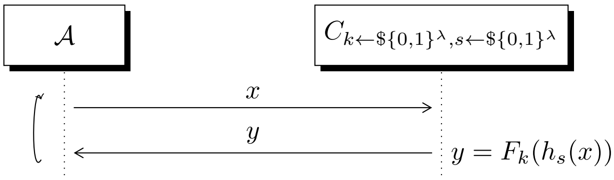


Figure 9.29: $Real_{\mathcal{F},\mathcal{A}}(\lambda)$

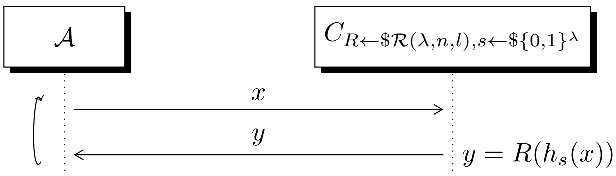


Figure 9.30: $\mathcal{HYB}_{\mathcal{R},\mathcal{A}}(\lambda)$

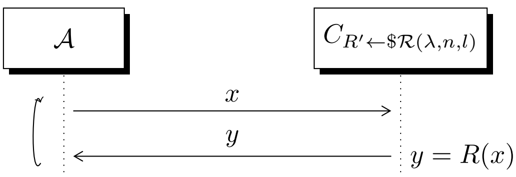


Figure 9.31: $Rand_{\mathcal{R}',\mathcal{A}}(\lambda)$

Lemma 13. *Real* $\approx_c \mathcal{HYB}$

◇

Exercise 23. Prove it!

Lemma 14.

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

◇

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,

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

□

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

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.

9.2.1 $\mathcal{H}$ family using Galois Fields

Construction 2. 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 $\frac{1}{\mathbb{F}} = \frac{1}{2^n} \in \text{negl}\,(\lambda)$.

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

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 $s \leftarrow \$\mathbb{F}^t$. We state that

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

$\diamond$

Exercise 24. 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

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

)

Lesson 10

10.1 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 $F_s j$

T0D0 13: something related to $f_s(1, .)$ and $f_s(0, .)$, but I didn't get it.

-

.

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

Exercise 25. 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.

10.1.2 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 16. *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 26. *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) ◇

10.1.3 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 17. *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 27. *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 28. 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.

==

10.2 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.

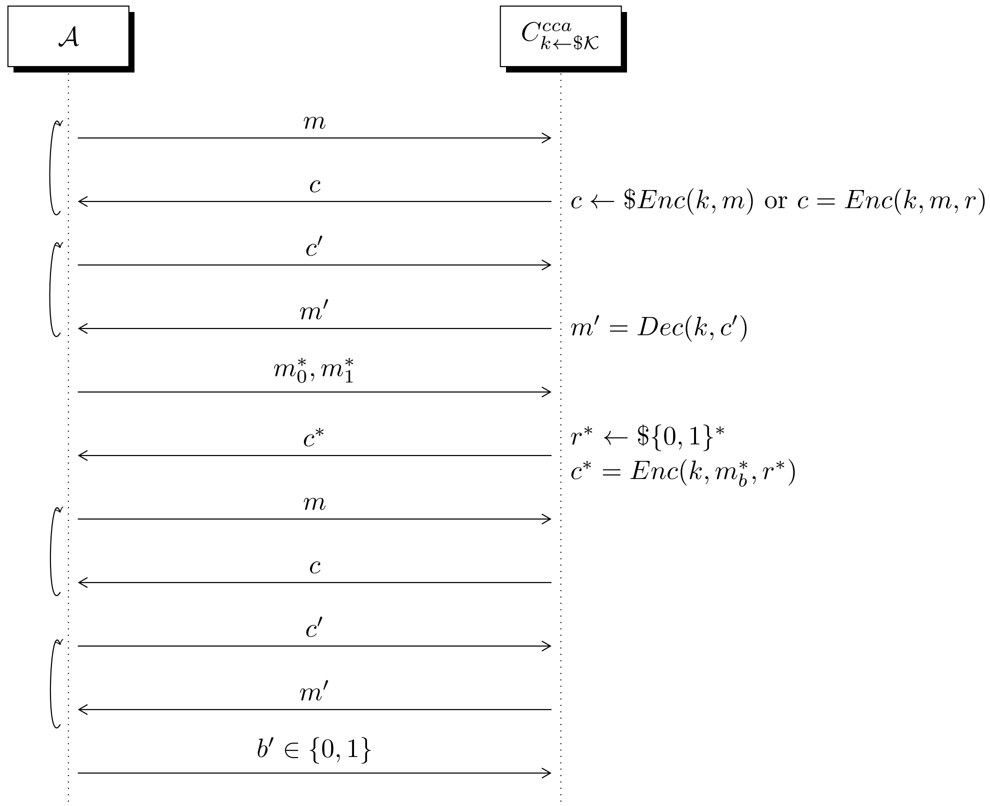
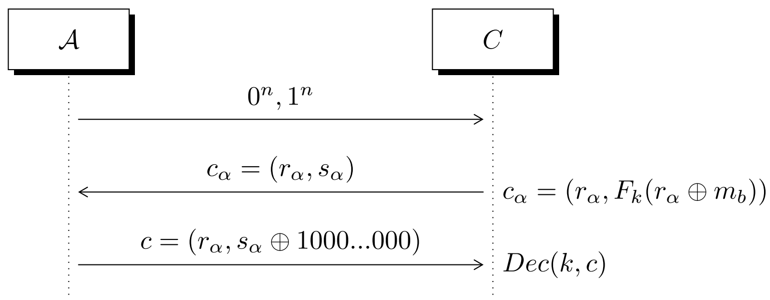


Figure 10.32: $Game^{cca}_{\Pi,\mathcal{A}}(\lambda, b)$

Exercise 29. 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

□

10.3 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:

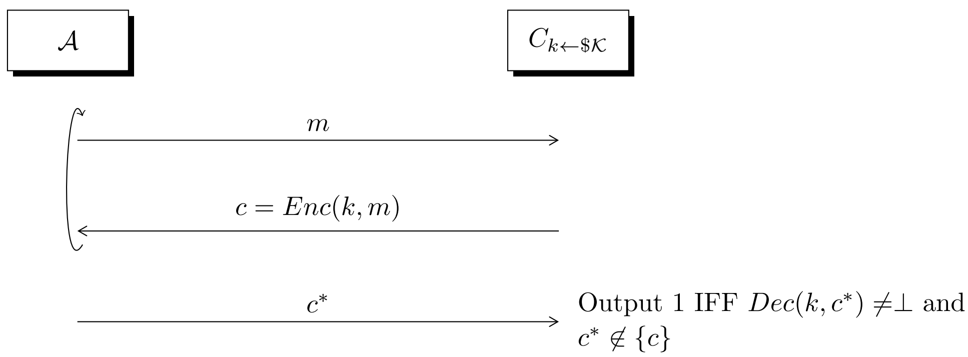


Figure 10.33: $Game^{auth}_{\Pi,\mathcal{A}}(\lambda)$

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 30. *Let Π be an SKE (shared key encryption). If Π has* ***CPA + auth*** *security, then it also has* ***CCA*** *security.* ◇

Exercise 31. 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 $\mathcal{C}$.
==

10.3.1 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*** :

3.

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

==
DA RIVEDERE: come i ϕ sono randomici o no? A guardare la prima parte della prossima lezione , sembrerebbe di si.

4. ***Encrypt-then-MAC*** :

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

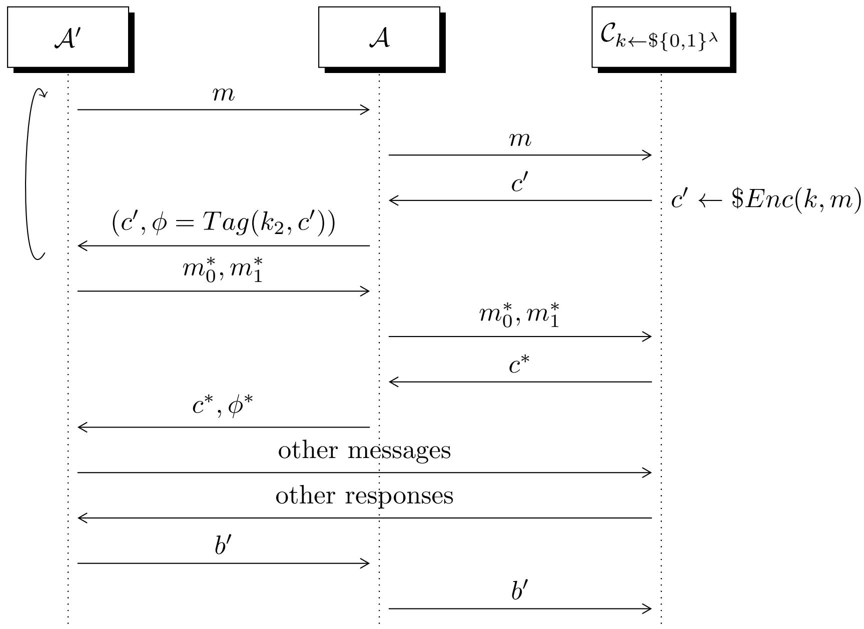
==

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 32. *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.

T0D0 14: 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 33. Prove it!
Similar to the cpa-security proof.

$\square$

Lesson 11

11.1 Authenticated encryption (Age of Ultron)

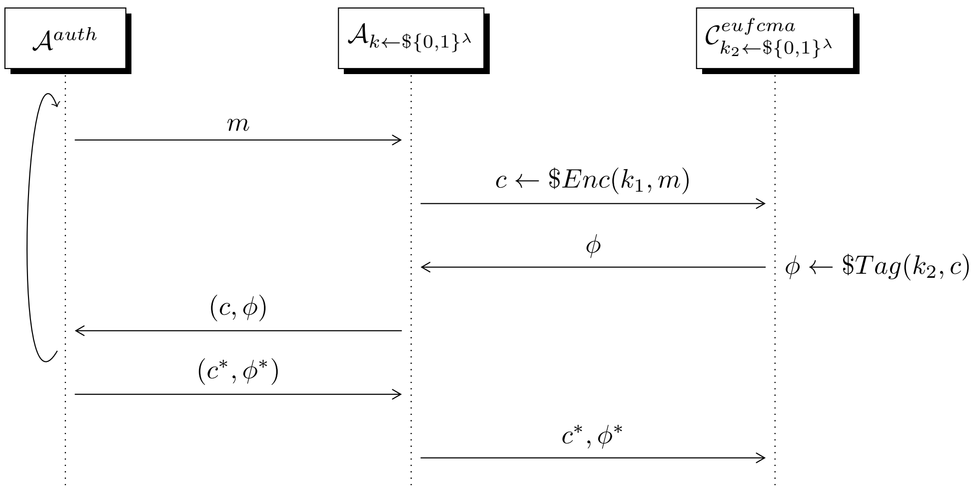
Last time we proved CPA-security of Π. Today we will explore the *auth* property. Consider Π as

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

Lemma 18. *If $Tag(.,.)$ is **EUF-CMA** , then Π has auth-property.* ⋄

What is **EUF-CMA** ?
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, we show that



The game returns 1 IFF

$$c^* \leftarrow \$Enc(k_1, .) \wedge \phi^* \leftarrow \$Tag(k_2, c^*) \wedge (c^*, \phi) \notin \{(c, \phi)\}$$

From A^{auth} perspective, all the couples (c_i, ϕ_i) received are made with the following schema:

$$c_i \in Enc(k_1, m \in \mathcal{M}) \wedge \phi_i \leftarrow \$Enc(k_2, c_i)$$

Since $\mathcal{A}^{auth}$ wins $Game^{auth}$, the challenge couple $(c*, \phi^*)$ which breaks $Game^{auth}$ will be produced to be decrypted as

$$Dec(k, (c^*, \phi^*)) \rightarrow Dec(k_1, c^*) \in \mathcal{M} \wedge Dec(k_2, \phi^*) = c^*$$

But if this happens , then $\mathcal{A}$ can use the same challenge couple of $\mathcal{A}^{auth}$ to win $Game^{ufcma}$, which is impossible.

It could happen that, for $c^* = c$ previously seen, ϕ^* is a new fresh tag, never seen before. Just in this case the *auth* game would be valid because (c^*, ϕ^*) would have never been seen before, but **not** the eufcma game, because c^* was previously sent to the challenger. $\square$

Now we want an ufcma secure scheme able to resist against message-tag challenge couples where the tag is fresh but the message has been already requested to the challenger.

11.2 Pseudorandom permutations

Pseudorandom permutations are like PRFs, but efficiently invertible. Consider the following family of functions:

$$\mathcal{F} = \{F_k : \{0,1\}^n \rightarrow \{0,1\}^n\}_{k \in \{0,1\}^\lambda}$$

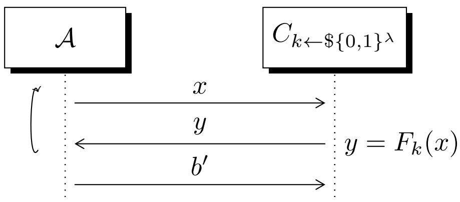


Figure 11.34: $Real_{\mathcal{F},\mathcal{A}}(\lambda)$

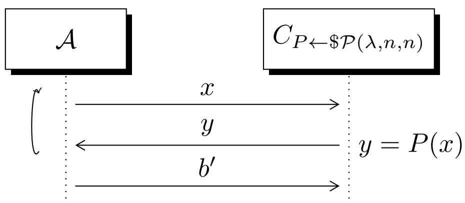


Figure 11.35: $Ideal_{\mathcal{P},\mathcal{A}}(\lambda)$

and the two games are indistinguishable

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

11.2.1 Feistel Network

Let $F : \{0,1\}^n \to \{0,1\}^n$ and ψ_F the invertible function

$$\psi_F(\overbrace{x}^{\text{n bits}}, \overbrace{y}^{\text{n bits}}) = (y, x \oplus F(y)) = (x', y')$$
$$\psi_F^{-1}(\overbrace{x'}^{\text{n bits}}, \overbrace{y'}^{\text{n bits}}) = (F(x') \oplus y', x') = (x, y)$$

Is this function pseudorandom?

This is not pseudorandom, because the first n bits of the output of ψ_F are always equal to y, while in a PRF the probability that, given two different (x, y) and (x', y) in input, the first bits are equal is very low.

T0D0 15: FEISTEL IMAGE

The $l - th$ level outputs something like

$$\psi_F[l](x, y) = \psi_{F_{k_l}}(\psi_{F_{k_{l-1}}}(...(\psi_{F_{k_1}}(x, y))...))$$

Two XORed rounds of this function don't create a PRP. In particular, imagine 2 queries (x, y) and (x', y) such that

$$\psi_{F,F'}(x, y) \oplus \psi_{F,F'}(x', y) = (x \oplus F(y) \oplus x' \oplus F(y), \dots)$$

Since for 2 random queries with the same y, the first member of the output is always equal to $x \oplus x'$ with probability 1, this XORed rounds cannot constitute a PRP (which, instead, for 2 queries with the same second member outputs a first member equal to $x \oplus x'$ with negligible probability).

Lemma 19. *For every* ***unbounded*** *distinguisher making $q \in poly(\lambda)$ queries, the following are statistically close as long as $y_1, \dots, y_q$ are* ***y-nique*** *, i.e. $\forall i \neq j, y_i \neq y_j$* ◇

figures

Lesson 15

15.1 Public key encryption recap

$\text{Game}_{\Pi,\mathcal{A}}^{pke-cca}$:

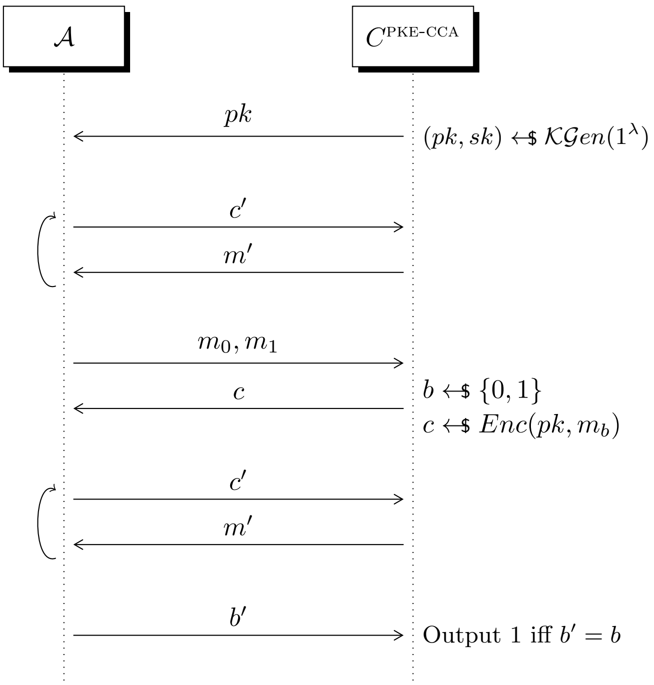


Figure 15.36: CCA on a PKE scheme

(Reminder): Encryptions are made like this: $Enc(k, m) = (r, F_k(r) \oplus m), r \simeq \mathcal{U}nif(\{0,1\}^\lambda)$.

Every time an encryption is made, a fresh value r is picked UAR.

15.1.1 Trapdoor permutation

A TDP(TrapDoor Permutation) is a OWP family structured has these features:

- A key pair is chosen UAR by a key generator algorithm: $(pk, sk) \leftarrow\!\!\$\ \mathcal{KG}en(1^\lambda)$
- There is a function family $f_{pk} \subseteq (V_{pk} \to V_{pk})$ such that:
  - Computing f_{pk} is efficient
  - Sampling from the domain ($x \leftarrow\!\!\$\ V_{pk}$) is efficient
- There is an efficient function g_{sk} that "inverts" f_{pk} (sk is the "trapdoor"):

$$g(sk, f(pk, x)) = x$$

- No efficient adversary is able to invert f_{pk} wihout knowing sk

Note: Since pk is public, any adversary gets the capability of encrypting messages for free, without requiring n external challenger/oracle!

Therefore, if left deterministic, a TDP is not CPA-secure.

Here, in this scheme, we combine randomness and the notion of hardcore predicate $\mathfrak{hc}$:

- $(pk, sk) \leftarrow\!\!\$\ \mathcal{KG}en(1^\lambda)$
- $r \leftarrow\!\!\$\ \Xi_{pk}$
- $c := Enc(pk, m) = (f_{pk}(r), \mathfrak{hc}(r) \oplus m)$
- Correctness: $Dec(sk, c) = \mathfrak{hc}(g_{sk}(c_1)) \oplus c_2$

Theorem 34. *If $(\mathcal{KG}en, f, g)$ is a TDP, and $\mathfrak{hc}$ is hardcore for f, then the above scheme is CPA-secure.* ◇

Proof: (Execrcise)

T0D0 16: Apparently, the reduction here is not easy at all. Some hints are needed to solve this exercise.

15.1.2 TDP examples

One example stems form the factoring problem: let's look again at $\mathbb{Z}_n^\times$, where $n = pq$, $p, q \in \mathbb{P}$;

Theorem 35. *(Chinese remainder, or CRT): The following isomorphisms to $\mathbb{Z}_n^\times$ are true:*

- $\mathbb{Z}_n \simeq \mathbb{Z}_p \times \mathbb{Z}_q$
- $\mathbb{Z}_n^\times \simeq \mathbb{Z}_p^\times \times \mathbb{Z}_q^\times$

Note that the theorem is more general, and holds for any p, q that are co-prime. ◇

How to use this theorem for constructing a PKE scheme:

Reminder (Euler's theorem):

REVIEW: $\forall x \in \mathbb{Z}_n \implies x^{\varphi(n)} = x \mod n$
maybe the correct one is
$\forall x \in \mathbb{Z}_n \implies x^{\varphi(n)} = 1 \mod n$

Reminder: $\forall p, q \in \mathbb{P} \implies \varphi(pq) = (p-1)(q-1)$

So let a be the public key such that $\gcd(a, \varphi(n)) = 1$, then $\exists! b \in \mathbb{Z}_n : ab = 1 \mod \varphi(n)$, b will be our private key.

Define encryption as $f(a, m) = m^a \mod n$, and then decryption as $g(b, c) = c^b \mod n$.

Observe that

$$g(b, f(a, m)) = (m^a)^b = m^{ab} = m^{k\varphi(n)+1} = (m^{\varphi(n)})^k m = m \mod n$$

, because $ab = 1 \mod \varphi(n)$.

So we conjecture that the above is a valid TDP-based PKE scheme. This is actually called the "RSA assumption":

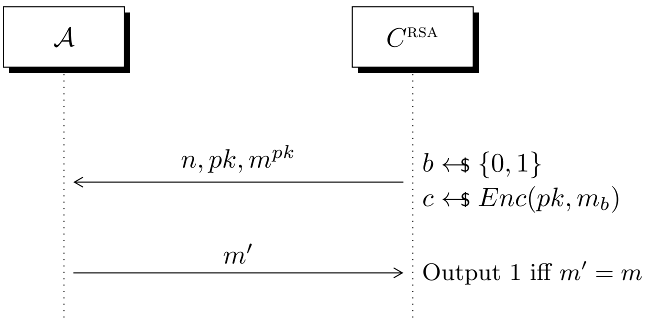


Figure 15.37: Depiction of the RSA assumption

Relation to the factoring problem: RSA $\implies$ FACT

Proof: Given p, q, an adversary can compute $\varphi(n) = (p-1)(q-1)$, and then find the inverse of the public key in $\mathbb{Z}_{pq}^{\times}$.

It hasn't been proven that FACT $\implies$ RSA

15.2 Textbook RSA

This is an insecure 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.

REVIEW:

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 standardized (PKCS#1,5)

REVIEW:

4. Still not CCA secure! counterexample?

15.2.1 Trapdoor Permutation from Factoring

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

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

Since Z_p^* is cyclic I can always write:

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

Moreover, since $(g^{\frac{p-1}{2}})^2 \equiv 1 \pmod{p}$ and $g^{\frac{p-1}{2}}$ cannot be 1 (since $g^0 \neq g^{\frac{p-1}{2}} \neq g^{p-1}$) but must be one of the $p-1$ elements of Z_p^*, then $g^{\frac{p-1}{2}} \equiv -1 \pmod{p}$.

Now it's possible to show that $f : \mathbb{QR}_p \to \mathbb{QR}_p$ is a permutation, and we are going to show a method to invert it, aka f^{-1}.

Assume $p \equiv 3 \pmod 4$ ([*]$p = 4t + 3 \Rightarrow t = \frac{p-3}{4}$), 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, in particular only the positive one.
Since we have that

$$p = k4 + 3 \Rightarrow \frac{p-1}{2} = \frac{4k+2}{2} = 2k + 1$$

so $\frac{p-1}{2}$ is odd.

Now, since we are considering just $\mathbb{QR}_p = \{y \in \mathbb{Z}_p^* : \exists x \in \mathbb{Z}_p^* x^2 = y\}$ and we can write each $x \in \mathbb{Z}_p^*$ as g^z for a $z \in \mathbb{Z}_p$,

$$y = x^2 \Leftrightarrow y = (g^z)^2 = g^{2z}$$

So, $y = g^{z'} \in \mathbb{QR}_p \Leftrightarrow z'$ is even. If z' is odd, then $y \notin \mathbb{QR}_p$.
Since $\frac{p-1}{2}$ is odd, then $g^{\frac{p-1}{2}} \notin \mathbb{QR}_p$, and since it is possible to generate all of the other numbers with odd exponents

$$g^{odd} = g^{\frac{p-1}{2} \pm even} = g^{\frac{p-1}{2}} g^{\pm even} \Rightarrow -1(g^{\pm even})$$

and g powered to odd exponents will have this form.

From that, it's possible to state the following

Lemma 20. $\forall z, z \in \mathbb{QR}_p \implies -z \notin \mathbb{QR}_p$ ◇

15.2.2 Rabin's Trapdoor permutation

Now we study a one way function built on previous deductions about number theory and modular arithmetic.
The *Rabin trapdoor permutation* is defined as

$$f(x) = x^2 \bmod n$$

where $n = p * q$ for primes $p, q = 3\text{mod}4$.

We can observe that the image of this function is $\mathbb{QR}_n$, a subset of $\mathbb{Z}_n^*$.
For **Chinese remainder theorem** it is possible to state that f maps as follows

$$x = (x_p, x_q) \mapsto (x_p^2, x_q^2)$$

since each element of $\mathbb{Z}_n$ has always two different forms , in $\mathbb{Z}_p$ and in $\mathbb{Z}_q$.

So
$$y \in \mathbb{QR}_n \Leftrightarrow y_p \in \mathbb{QR}_p \wedge y_q \in \mathbb{QR}_q$$
As before, the image of f is exactly

$$\mathbb{QR}_n = \{y : \exists x : y = x^2 \bmod n\}$$

If we try to invert the function f, even without applying the previous inversion algorithm, we easily note that among the 4 possible values

$$f^{-1}(y) = \{(x_p, x_q), (-x_p, x_q), (x_p, -x_q)(-x_p, -x_q)\}$$

only 1 is a quadratic residue since we said, in the last lemma, that only one out of $-x_k, x_k$ is a quadratic residue for $k = q, p$.

Therefore, we have that the Rabin's TDP is a permutation in $\mathbb{QR}_n$, and that the cardinality of $\mathbb{QR}_n$ is $\frac{|\mathbb{Z}_n^*|}{4}$.

Furthermore, with the following claim we can state that the Rabin cryptosystem is OWF thanks to the hardness of factoring.

Claim 1. *Given x, z such that x^2 $mod n \equiv z^2$ mod $n \equiv y$ mod n,*

$$x \neq \pm z \Rightarrow \textit{ we can factor } n$$

$\diamond$

Proof. Since $f^{-1}(y)$ has only one value out of four, $x \neq \pm z$ and $z \in \{(x_p, x_q), (-x_p, -x_q)\}$, then $x \in \{(x_p, -x_q), (-x_p, x_q)\}$ and

$$x + z \in \{(0, 2x_q), (2x_p, 0)\}$$

Now assume $x + z = (2x_p, 0)$ without loss of generality, since the proof for the other case is the same.
We have that $x + z \equiv 0 \bmod q$ and $x + z \not\equiv 0 \bmod p$.
But then $gcd(x + z, n) = q$, and we obtain q. $\square$

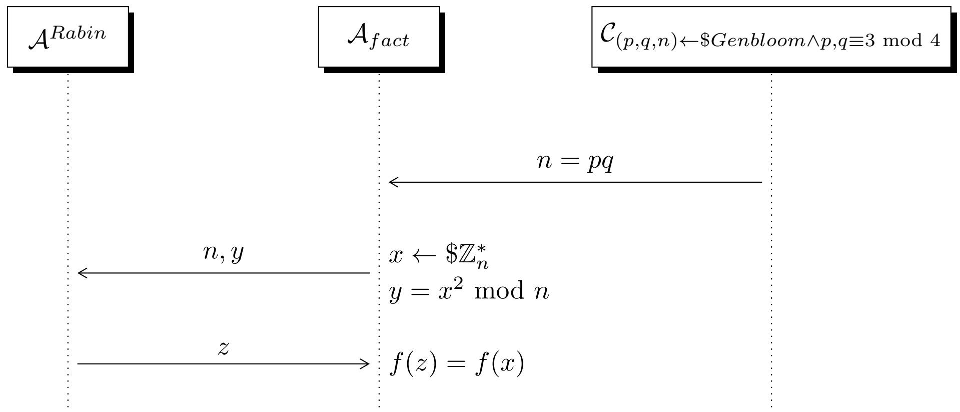
Theorem 36. *Squaring mod n (where n is a bloom integer[11]) is a **trapdoor permutation** under factoring.* $\diamond$

Since we have already shown tha Rabin's function is a permutation since it is invertible, we have to show that Rabin's function is also OWF.
In other words

$$\text{Factoring is hard } \Rightarrow f(x) \text{ is OWF , aka inverting it is hard}$$

The following proof is by contraddiction.

Proof. Assume that exists an adversary PPT who, given $y = x^2 \bmod n$, can find a $z \in \mathbb{Z}_n$ such that $z^2 \bmod n = y$ but $z \neq \pm x$.



We can build the following reduction to show that $\mathcal{A}$ choses x:
Once obtained $z \neq \pm x$ which $z^2 = y$ we can use **Claim 1**(just summing x and z and analyzing the result) to factorize n in polytime.

But factorizing n in polytime is not possible. $\square$

[11] a bloom integer n is $n = p, q$ for $p, q = 3mod4$, as the definition of Rabin's TDP

Lesson 16

16.1 Pke schemes over ddh assumption

16.1.1 El Gamal scheme

Let's define a new Π = (KGen, Enc, Dec). Generate the needed (**public**) parameters $(G, g, q) \leftarrow\$ GroupGen(1^\lambda)$[12].

The KeyGen algorithm is defined as follows:

- Pick $x \leftarrow\$ \mathbb{Z}_q$
- Output the key pair (pk, sk) as (g^x, x)

The encryption routine Enc(pk, m) will:

- Pick $r \leftarrow\$ \mathbb{Z}_q$
- Output $c = (c_1, c_2) = (g^r, pk^r \cdot m)$[13]

The decryption routine Dec(sk, c) will:

- Compute $\hat{m} = c_1^{-sk} \cdot c_2$

Correctness of this scheme follows from some algebric steps:

$$\begin{aligned}
\hat{m} &= Dec(sk, Enc(pk, m)) \\
&= Dec(x, Enc(g^x, m)) \\
&= Dec(x, (g^r, (g^x)^r \cdot m)) \\
&= (g^r)^{-x} \cdot (g^x)^r \cdot m \\
&= m
\end{aligned}$$

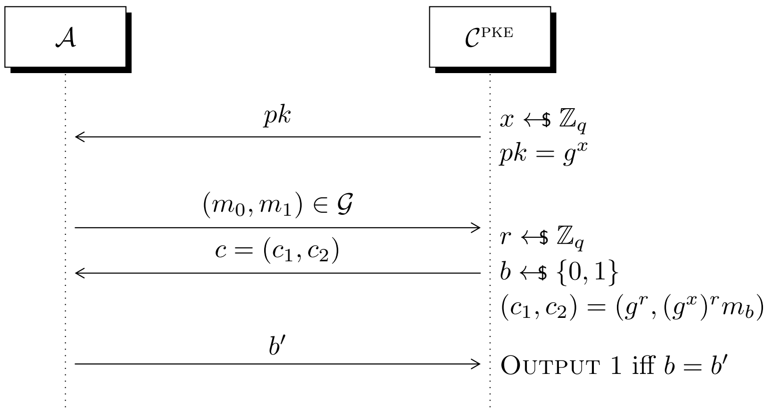
Theorem 37. *Assuming* DDH*, the El Gamal scheme is* CPA*-secure.* ◇

Proof. Consider the two following games $H_0(\lambda, b)$ and $H_1(\lambda, b)$ defined as follows. Observe that b can be fixed without loss of generality.

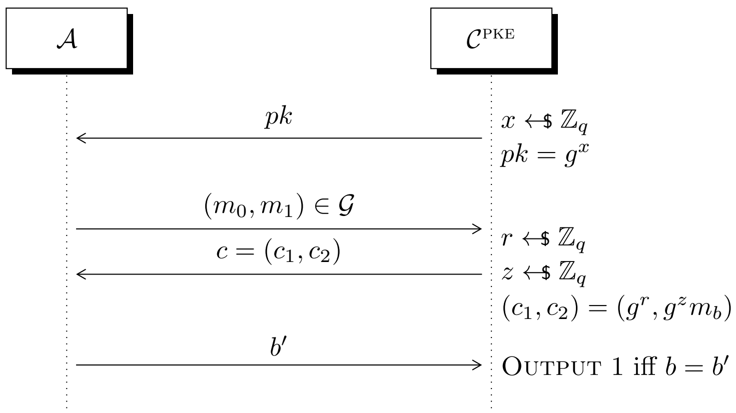
[12] G could be any "valid" group such as $\mathbb{QR}_p$ or an Elliptic Curve

[13] We need r because we want to re-randomize c

$H_0(\lambda, b)$:



$H_1(\lambda, b)$:



Note: it is important to note that we can measure the advantage of $\mathcal{A}$, so fixed its output $Adv_{\mathcal{A}}(\lambda) = |\underbrace{Pr[\overbrace{\mathcal{A} \to 1}^{b'=1} | b = 0]}_{\text{"}\mathcal{A}\ loses\text{"}} - \underbrace{Pr[\overbrace{\mathcal{A} \to 1}^{b'=1} | b = 1]}_{\text{"}\mathcal{A}\ wins\text{"}}|$. Since b is fixed the above formula will give a value λ *innegl*, generally the advantage of an adversary is: $\frac{1}{2} + \lambda$ (random guessing + a negligible factor).

Proof techinique: $H_0(\lambda, 0) \approx_c H_0(\lambda, 1) \equiv H_1(\lambda, 0) \approx_c H_1(\lambda, 1)$

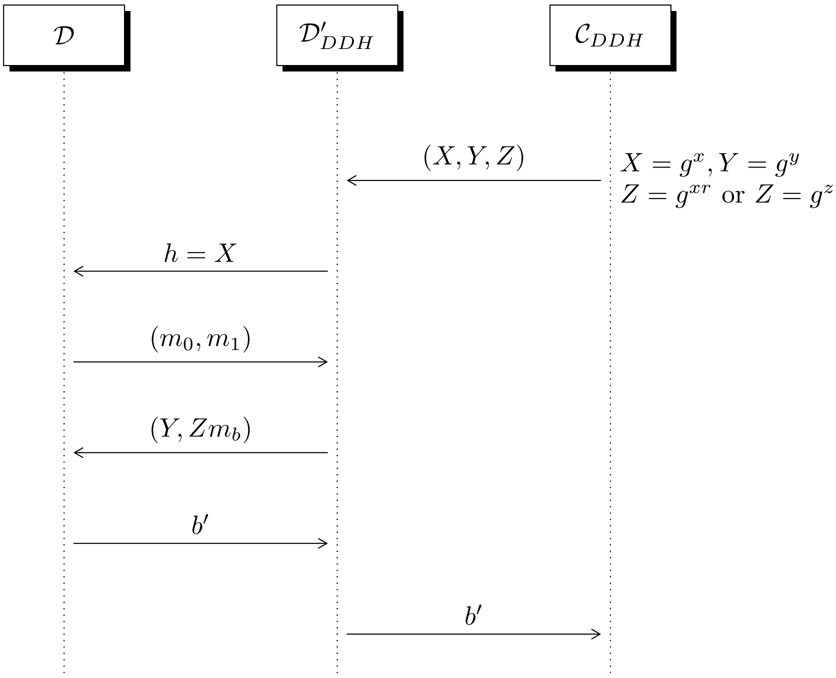
$$\implies H_0(\lambda, 0) \approx_c H_1(\lambda, 1)$$

Lemma 21. $\forall b \in \{0, 1\}, H_0(\lambda, 0) \approx_c H_0(\lambda, 1)$
Fix b. (Reduction to DDH)
Assume $\exists$ PPT D which is able to distinguish $H_0(\lambda, b)$ and $H_1(\lambda, b)$ with non negl. probability. ◇

Proof. Consider the following Game:

Contradiction: $\mathcal{D}$ should be able to compute log_g to distinguish the message.

□



Lemma 22. $H_1(\lambda, 0) \equiv H_1(\lambda, 1)$ $\diamond$

Proof. This follows from the fact that: $(g^x, (g^r, g^z m_0)) \equiv (g^x, (g^r, U_\lambda) \equiv$
$\equiv (g^x, (g^r, g^z m_1)))$ $\square$

Lemma 23. $H_1(\lambda, 1) \equiv H_0(\lambda, 1)$ $\diamond$

This is proved in the exact same way as **Lemma 20.** As a matter of fact it is the second part of the proof (where b is fixed to 1).

$\square$

Properties of of El Gamal PKE scheme

Some useful observations can be made about this scheme:

- It is **homomorphic**: Given two ciphertexts (c_1, c_2) and (c_1', c_2'), then doing the product between them yields another valid ciphertext:

$$
\begin{aligned}
&(c_1 \cdot c_1', c_2 \cdot c_2') \\
&= (g^{r+r'}, h^{r+r'}(m \cdot m'))
\end{aligned}
$$

  thus, decrypting $c \cdot c'$, gives $m \cdot m'$.

- It is **re-randomizable**: Given a ciphertext (c_1, c_2), and $r' \leftarrow\$ \mathbb{Z}_q$, then computing $(g^{r'} \cdot c_1, h^{r'} \cdot c_2)$ results in a "fresh" encryption for the same message: the random value used at the encryption step will change from the original r to $r + r'$

So this is not CCA-Secure!

These properties of the El Gamal scheme can be desirable in some use cases, where a message must be kept secret to the second party. In fact, there are some PKE schemes which are designed to be **fully homomorphic**, i.e. they are homomorphic for any kind of function.

Consider the following use case: a client C has an object x and wants to apply a function f over it, but it lacks the computational power to execute it. There is another subject S, which is able to efficiently compute f, so the goal is to let it compute $f(x)$ but the client wishes to keep x secret from him. This can be achieved using a FH-PKE scheme as follows:

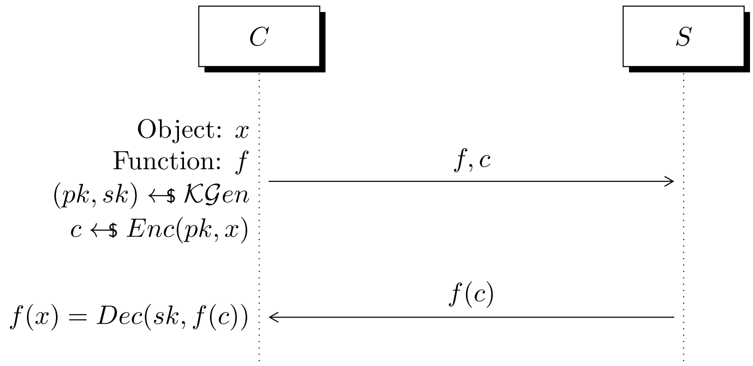


Figure 16.38: Delegated secret computation

However one important consideration must be made: All these useful characteristics expose an inherent malleability of any fully homomorphic scheme: any attacker can manipulate ciphertexts efficiently, and with some predictable results. This compromises even CPA security of such schemes.

16.1.2 Cramer-Shoup PKE scheme

This scheme is based on the standard DDH assumption, and has the advantage of being CCA secure. A powerful tool, called **Designated Verifier Non-Interactive Zero-Knowledge (DVNIZK)**, or alternatively **Hash-Proof System**, is used here.

Proof systems

Let L be a Turing-recognizable language in NP, and a predicate $\mathcal{R} \in X \times Y \to \{0, 1\}$ such that:

$$L := \{y \in Y : \exists x \in X\ \mathcal{R}(x, y) = 1\}$$

where x is called a "witness" of y.

In our instance, let $y = pq, x = (p, q)$

$\Pi = (\mathcal{S}etup, \mathcal{P}rove, \mathcal{V}erify)$

$(\omega, \tau) \leftarrow\$ \mathcal{S}etup(1^\lambda)$, where ω is the **Common Reference String**, and τ is the **trapdoor**.

Additional notes:

- ω is public $(= pk)$

- τ is part of the secret key
- $\tau = (x, y) : \mathcal{R}(x, y) = 1$
- There is presumably a common third-party, which samples from the setup and publishes ω, while giving τ to only B.

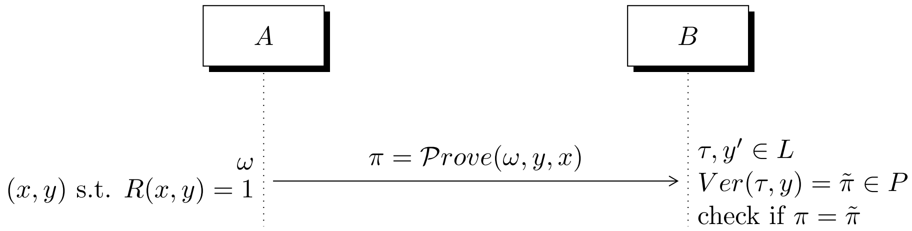


Figure 16.39: Overview of Cramer-Shoup operation

Proof system - purpose: a way to convince B that A knows something
Can compute the proof in two different ways, this is the core notion of ZK
No $\tau \implies ZK$

Properties

- *(implicit, against malicious Bob)* **Zero-knowledge**: Proof for x can be simulated without knowing x itself
- *(stronger, against malicious ALice)* **Soundness**: It is hard to produce a valid proof for any $y \notin L$
- *honest people* **Completeness**: $\forall y \in L, \forall(\omega, \tau) \leftarrow\!\!\$\ \mathcal{S}etup(1^\lambda)$:
  $\mathcal{P}rove(\omega, x, y) = \mathcal{V}erify(\tau, y)$

T0D0 17: to review and understand/better

t-universality

Definition 15. Let Π be DV-NIKZ[14].
We say it is *t-universal* if for any distinct

$$y_1, \ldots, y_t \text{ s.t. } y_i \notin L(\forall i \in [t])$$

we have

$$(\omega, Ver(\tau, y_1), \ldots, Ver(\tau, y_t)) = (\omega, v_1, \ldots, v_t)$$

where $(\omega, \tau) \leftarrow \$\mathcal{S}etup(1^\lambda)$ and $v_1, \ldots, v_t \leftarrow \$\mathcal{P}$ where $\mathcal{P}$ should be the proofs' space. $\diamond$

[14] Designated verifier non-interactive zero-knowledge

Enriching DV-NIKZ

Can we enrich DV-NIKZ with labels $l \in \{0,1\}^*$?

Suppose to have the following:

$$L' = L \| \{0,1\}^* = \{(y,l) : y \in L \wedge l \in \{0,1\}^*\}$$

Then our scheme changes : $Prove(\omega, (y,l), x) = \Pi; Ver(\tau, (y,l))$ and , for *t-universality* , now we can consider 2 distinct (y_i, l_i).

Membership Hard Language (MH)

Definition 16. Language L is ***MH*** if $\exists \bar{L}$ such that:

1. $L \cap \bar{L} = \emptyset$
2. $\exists$ PPT *Samp* outputting $y \leftarrow \$\mathcal{Y}$ together with $x \in \bar{\mathcal{X}}$ such that

   $$R(y,x) = 1$$

   (it's possible to say that $Samp(1^\lambda) \leftarrow \$(y,x)$)
3. $\exists$ PPT $\bar{Samp}$ outputting $y \leftarrow \$\bar{L}$
4. $\{y : (y,x) \leftarrow \$Samp(1^\lambda)\} \approx_c \{y : y \leftarrow \$\bar{Samp}(1^\lambda)\}$

$\diamond$

Lesson 17

17.1 Construction of a CCA-secure PKE

This section exposes a construction of a CCA-secure PKE scheme, using hash-proof systems, membership-hardness, and the n-universality property.

Let Π_1, Π_2 be two distinct hash-proof systems for some NP language L and the range of $Prove_2$ supports labels ($L' = L||\{0,1\}^\ell$).

Construct the CCA scheme as follows: $\Pi := (\mathcal{KG}en, Enc, Dec)$

- $(\overbrace{(\omega_1, \omega_2)}^{pk}, \overbrace{(\tau_1, \tau_2)}^{sk}) \leftarrow\!\!\$\ \mathcal{KG}en(1^\lambda)$, $\quad (\omega_1, \tau_1) \leftarrow\!\!\$\ Setup_1(1^\lambda), (\omega_2, \tau_2) \leftarrow\!\!\$\ Setup_2(1^\lambda)$
- Encryption routine: $Enc((\omega_1, \omega_2), m)$
  - $(y, x) \leftarrow\!\!\$\ Sample_1(1^\lambda)$
  - $\pi_1 \leftarrow\!\!\$\ Prove_1(\omega_1, y, x)$
  - $l := \pi_1 \cdot m$
  - $\pi_2 \leftarrow\!\!\$\ Prove_2(\omega_2, (y, l), x)$
  - $c := (c_1, c_2) = ((y, l), \pi_2)$
- Decryption routine: $Dec((\tau_1, \tau_2), (c_1, c_2))$
  - $\hat{\pi}_2 = Verify_2(\tau_2, c_1))$
  - IF $\hat{\pi}_2 \neq c_2$ THEN OUTPUT FALSE
  - Recall: $c_1 = (y, l)$
  - $\hat{\pi}_1 = Verify_1(\tau_1, y)$
  - OUTPUT $l \cdot \hat{\pi}_1^{-1}$

Correctness (assume $\hat{\pi}_i = \pi_i \forall i$):

$$\begin{aligned}
\hat{m} &= Dec(sk, Enc(pk, m)) \\
&= Dec((\tau_1, \tau_2), Enc((\omega_1, \omega_2), m)) \\
&= Dec((\tau_1, \tau_2), ((y, l), \pi_2)) \\
&= l \cdot \hat{\pi}_1^{-1} \\
&= \pi_1 \cdot m \cdot \hat{\pi}_1^{-1} \\
&= m
\end{aligned}$$

Some additional notes (may be incorrect):

- The message space of the second prover is the range of the first prover.
- The message space of the first prover is a multiplicative group
- The message space of the second prover is a polylogarithmic language in (λ)

Theorem 38. *Assuming π_1 is 1-universal, π_2 is 2-universal and L is Membership-hard, then the above scheme is* CCA*-secure.* ◇

Proof. Five different games will be defined, from $\text{GAME}^0_{\Pi,A}$ up to $\text{GAME}^4_{\Pi,A}$; the first game will be an analogous formalization of how the above PKE scheme works. It shall be proven that, for either fixed b in $\{0,1\}$:

$$\text{GAME}^0_{\Pi,A}(\lambda,b) = \text{GAME}^1_{\Pi,A}(\lambda,b) \approx_C \text{GAME}^2_{\Pi,A}(\lambda,b) \approx_S \text{GAME}^3_{\Pi,A}(\lambda,b) = \text{GAME}^4_{\Pi,A}(\lambda,b)$$

and finally that $\text{GAME}^4_{\Pi,A}(\lambda,0) = \text{GAME}^4_{\Pi,A}(\lambda,1)$, therefore concluding that $\text{GAME}^0_{\Pi,A}(\lambda,0) \approx_C \text{GAME}^0_{\Pi,A}(\lambda,1)$, and proving this scheme is CCA-secure.

T0D0 18: Non sono per niente sicuro riguardo all'origine di x ed y, né tantomeno dove sia definito il sampler per essi

The games are defined as follows:

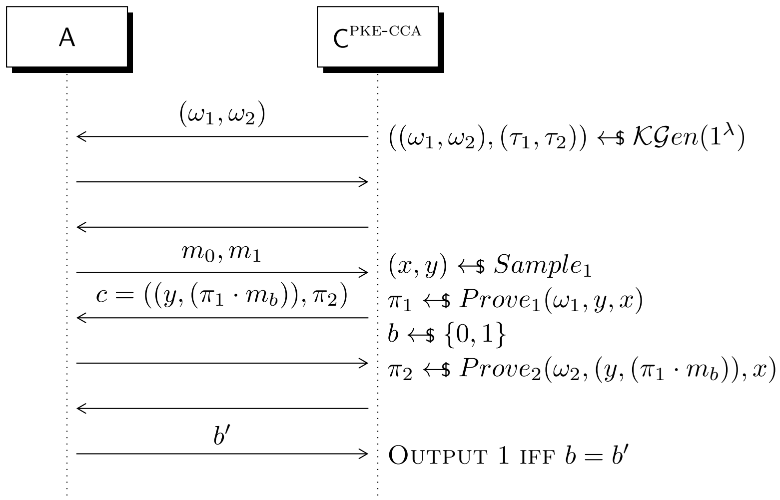


Figure 17.40: Original CCA game

T0D0 19: Non è stato chiaro sull'origine di x ed y, inoltre mi manca da scrivere le query di decifratura

□

Lemma 24.

$$\forall b, G_0(\lambda,b) \equiv G_1(\lambda,b)$$

◇

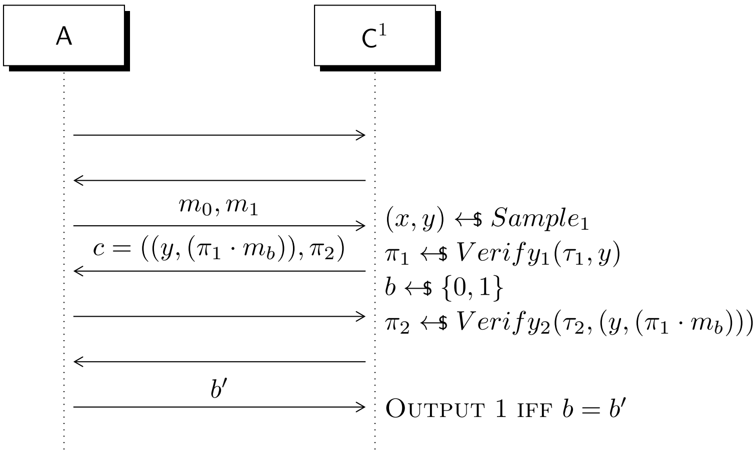


Figure 17.41: Use verifiers

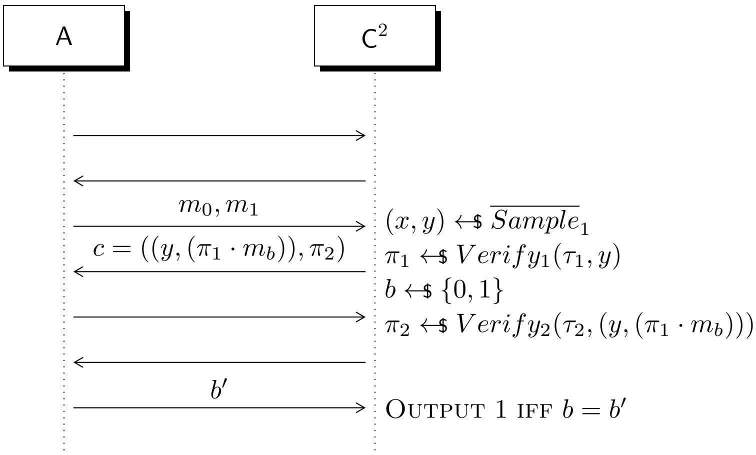


Figure 17.42: Sample statements outside the language

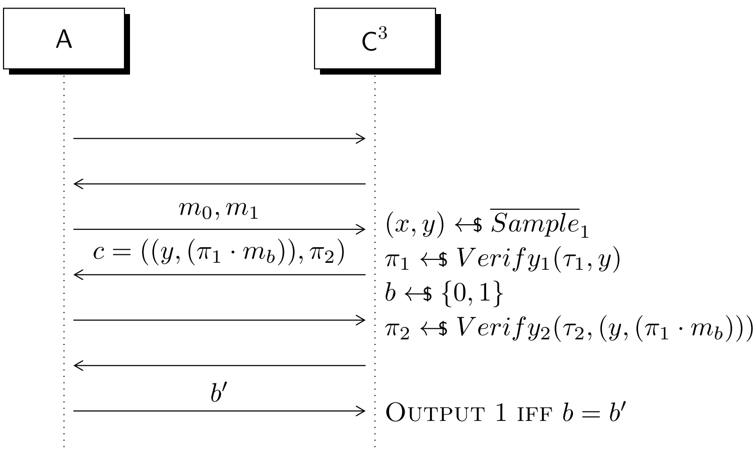


Figure 17.43: Modify decryption queries

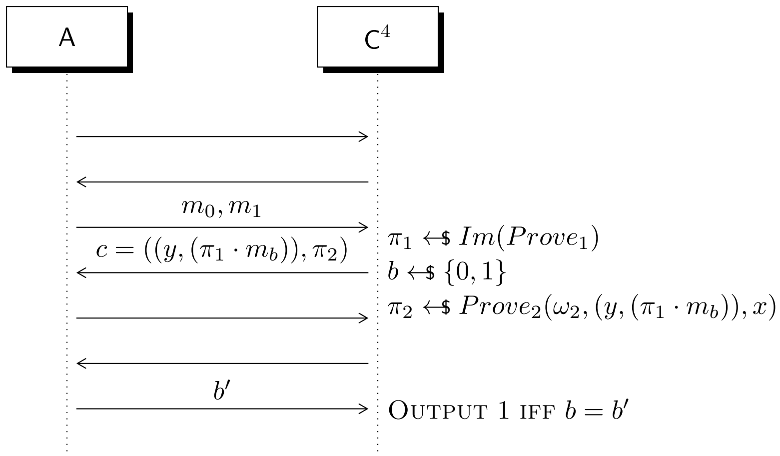


Figure 17.44: π_1 is chosen UAR

Proof. This follows by the correctness of Π_1 and Π_2

$$\Pi_1 = \tau_1 = Ver_1(\tau, y)$$
$$\Pi_2 = \tilde{\Pi}_2 = Ver_2(\tau, y)$$

with probability 1 over the choice of $(\omega_1, \tau_1) \leftarrow \$Setup_1(1^\lambda)$

$$\Pi_1 \leftarrow Prove_1(\omega, y, x), (\omega_2, \tau_1) \leftarrow \$Setup_2(1^\lambda)$$
$$\Pi_2 \leftarrow Prove(\omega_2, (y, l), x) \forall y \in L, l \in \mathcal{P}_1$$

□

Lemma 25.

$$\forall b, G_1(\lambda, b) \approx_c G_2(\lambda, b)$$

◇

Proof. Straight forward reduction from membership hardness. □

Lemma 26.

$$\forall b, G_2(\lambda, b) \approx_c G_3(\lambda, b)$$

◇

T0D0 20: I'm not completely sure the next proof is complete

Proof. Recall that the difference between G_2 and G_3 is that

$$(g^{(i)}, l^{(i)}, \Pi_2^{(i)}) \text{ such that } y^{(i)} \notin L$$

are answered $\perp$ in G_3, instead in G_2

$$\perp \text{ comes out as output} \Leftrightarrow \tilde{\Pi}_2^{(i)} = Ver(\tau, y^{(i)} \neq \Pi_2^{(i)})$$

It's possible to distinguish **two cases** , looking at $c = (y, l, \Pi_2)$:

1. if $(y^{(i)}, l^{(i)}) = (y, l)$ and $\tilde{\Pi}_2^{(i)} = \Pi_2^{(i)}$, it outputs $\perp$ if in the decryption scheme $(\Pi_2^{(i)} \neq \tilde{\Pi}_2^{(i)})$ it outputs $\perp$.
2. [15] otherwise $(y^{(i)}, l^{(i)}) \neq (y, l)$ if $y^{(i)} \notin L$ we want that $\Pi_2^{(i)}$ doesn't output exactly $Ver_2(\tau, (y^{(i)}, l^{(i)}))$, but it should output $\perp$.

EVENT BAD: If we look at the view of $\mathcal{A}$, the only information he knows is

$$(\omega_2, \tilde{\Pi}_2 = Ver_2(\tau_2, y))$$

for $y \notin L$.
The value

$$Ver_2(\tau_2, (y^{(i)}, x^{(i)}))$$

for $y^{(i)} \in L$ and $(y^{(i)}, l^{(i)}) \neq (y, l)$ is random.
So,

$$\mathcal{P}[BAD] = 2^{-|\mathcal{P}_2|}$$

□

Lemma 27.

$$\forall b, G_3(\lambda, b) \equiv G_4(\lambda, b)$$

◇

Proof. If we look at the view of $\mathcal{A}$, the only information known about τ_1 is ω_1, since the decryption oracle only computes for $y^{(i)} \in L$

$$Ver_1(\tau, y^{(i)}) = Prove_1(\omega_1, y^{(i)}, x^{(i)})$$

By 1-universality, $\Pi = Ver_1(\tau, y)$ for any $y \in L$ is random. □

Lemma 28.

$$G_4(\lambda, 0) \equiv G_4(\lambda, 1)$$

◇

Proof. The challenge ciphertext is independent of b. □

REVIEW: referencing something from another part of lesson 17

17.1.1 Instantiation of U-HPS (Universal Hash Proof System)

MHL(Membership Hard Language) from DDH

$\exists r$ using a DDH language such that $L_{DDH} = \{(c_1, c_2), \exists r | c_1 = g_1^r, c_2 = g_2^r\}$

Given a group $\mathcal{G}$ of order q with (g_1, g_2) as generators, we will have (g_1, g_2, c_1, c_2) but if we impose $g_1 = g$ and $g_2 = g^a$ then the previous construction becomes (g, g^a, c_1, c_2). But for definition $c_1 = g_1^{r_1}$ and $c_2 = g_2^{r_2}$ then I can write (g, g^a, g^r, g^{ar}).

Now we can define our U-HPS Π :=(Setup, Prove, Verify)

[15] when decryption oracle doesn't output the challenge

- *Setup*(1^λ): Pick $x_1, x_2 \leftarrow \$\mathcal{Z}_q$ and define:
  $\omega = h_1 = (g_1^{x_1}, g_2^{x_2})$, $\tau = (x_1, x_2)$
- *Prove*$(\omega, \underbrace{(c_1, c_2)}_{y}, r)$ output $\Pi = \omega^2$
- $Verify(\tau, \underbrace{(c_1, c_2)}_{y})$ output $\widetilde{\Pi} = c_1^{x_1} c_2^{x_2}$

Correctness: $\Pi = \omega^2 = (g_1^{x_1} g_2^{x_2})^r = g_1^{rx_1} g_2^{rx_2} = c_1^{x_1} c_2^{x_2} = \widetilde{\Pi}$

Theorem 39. *Above construction defines a 1-universal DVNIZK for* L_{DDH} $\diamond$

Proof. We want to prove that if we take any $(c_1, c_2) \notin L_D DH$ *the distribution* $(\omega = h_1, \widetilde{\Pi} = Verify(\tau, (c_1, c_2)))$ *uniformly distributed.*

Define a **MAP** $\mu(\overbrace{x_1, x_2}^{random}) = (\omega, \Pi) = (g_1^{x_1}, g_2^{x_2}, c_1^{x_1}, c_2^{x_2})$ it suffices to prove that μ is injective. This can easily be done with some constrains:

$\mu'(x_1, x_2) = log_{g_1}(\mu(x_1, x_2)) = (log_{g_1}(\omega), log_{g_1}(\Pi)$

For $r_1 \neq r_2$ then $c_1 = g_1^{r_1}, c_2 = g_2^{r_2} = g^{\alpha r_2}$. For $\alpha = log_{g_2} g_1$ then $\Pi = c_1^{x_1} c_2^{x_2} = g_1^{r_1 x_1} g_2^{r_1 x_1 + \alpha r_2 x_2}$.

$$\mu'(x_1, x_2) = \begin{pmatrix} z_1 \\ z_2 \end{pmatrix} = \begin{pmatrix} 1 & \alpha \\ r_1 & r_2\alpha \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$$

$$\text{Since Det} \begin{pmatrix} 1 & \alpha \\ r_1 & r_2\alpha \end{pmatrix} = \alpha(r_2 - r_1) \neq 0 \text{ the map is injective.}$$

$\square$

- *Setup*(1^λ):
  - Pick $x_3, x_4, x_5, x_6 \leftarrow \$\mathcal{Z}_q$ and define:
    * $\omega = (h_2, h_3, s) = (g_1^{x_3}, g_2^{x_4}, g_1^{x_5}, g_2^{x_6}, s)$ where s is a **seed** for a CRH$\rightarrow \mathcal{H} = \{H_s\}$
- *Prove*$(\omega, (c_1, c_2, l), r)$
  - Compute $\beta = H_s(c_1, c_2, l) \in \mathcal{Z}_p$
  - Output $\Pi = h_2^r h_3^{r\beta}$
- $Verify(\tau, (c_1, c_2, l))$
  - Compute $\beta = H_s(c_1, c_2, l) \in \mathcal{Z}_q$
  - Output $\widetilde{\Pi} = c_1^{x_3 + \beta x_5} c_2^{x_4 + \beta x_6}$

Correctness:
$\Pi = h_2^r h_3^{r\beta} = (g_1^{x_3} g_2^{x_4})^r (g_1^{x_5} g_2^{x_6})^r = c_1^{x_3} c_2^{x_4} c_1^{\beta x_5} c_2^{\beta x_6} = c_1^{x_3 + \beta x_5} c_2^{x_4 + \beta x_6} = \widetilde{\Pi}$

Theorem 40. *The above construction define a 2-universal DVNIZK for* L_{DDH} $\diamond$

Proof. Same goal and procedure as before
- Take any $(c_1, c_2) \notin L_{DDH}$
- Fix $(c_1, c_2, l) \neq (c_1', c_2', l')$ s.t. $(c_1, c_2), (c_1', c_2') \notin L_{DDH}$ which means:

- $(c_1, c_2) = (g_1^{r_1} g_2^{r_2})$
- $(c_1', c_2') = (g_1^{r_1'} g_2^{r_2'})$
- $\beta = H_s(c_1, c_2, l)$
- $\beta' = H_s(c_1', c_2', l')$

$\}$ $r_1 \neq r_2$ and $r_1' \neq r_2'$

- Let's define a MAP

$$\mu'(x_3, x_4, x_5, x_6) = (\omega, \widetilde{\Pi} = Ver(\tau, (c_1, c_2, l)), \widetilde{\Pi'} = Ver(\tau, (c_1', c_2', l'))) =$$
$$= (\underbrace{(h_2, h_3)}_{\omega}, c_1^{x_3 + \beta x_5} c_2^{x_4 + \beta x_6}, c_1'^{x_3 + \beta' x_5} c_2'^{x_4 + \beta' x_6} =$$
$$= ((g_1^{x_3} g_2^{x4}, g_1^{x_5} g_2^{x_6}), g_1^{r_1 x_3 + \beta r_1 x_5} g_2^{r_2 x_4 + \beta r_2 x_6}, g_1^{r_1' x_3 + \beta' r_1' x_5} g_2^{r_2' x_4 + \beta' r_2' x_6})$$

But I can rewrite g_2 as $g_2 = g_1^{\alpha}$ since they are generators. So:

$$((g_1^{x_3 + \alpha x_4}, g_1^{x_5 + \alpha x_6}), g_1^{r_1 x_3 + \beta r_1 x_5} g_1^{\alpha(r_2 x_4 + \beta r_2 x_6)}, g_1^{r_1' x_3 + r_1' \beta' x_5} g_1^{\alpha(r_2' x_4 + \beta' r_2' x_6)}) =$$
$$= ((g_1^{x_3 + \alpha x_4}, g_1^{x_5 + \alpha x_6}), g_1^{r_1 x_3 + \alpha r_2 x_4 + \beta r_1 x_5 + \alpha \beta r_2 x_6}, g_1^{r_1' x_3 + \alpha r_2' x_4 + r_1' \beta' x_5 + \alpha \beta' r_2' x_6}) =$$

$$\begin{pmatrix} z_1 \\ z_2 \\ z_3 \\ z_4 \end{pmatrix} = \begin{pmatrix} 1 & \alpha & 0 & 0 \\ 0 & 0 & 1 & \alpha \\ r_1 & \alpha r_2 & \beta r_1 & \alpha \beta r_2 \\ r_1' & \alpha r_2' & \beta' r_1' & \alpha \beta' r_2' \end{pmatrix} \cdot \begin{pmatrix} x_3 \\ x_4 \\ x_5 \\ x_6 \end{pmatrix}$$

$$\text{Since Det} \begin{pmatrix} . & . & . & . \\ . & . & . & . \\ . & . & . & . \\ . & . & . & . \end{pmatrix} = \alpha^2 (r_2 - r_1)(r_2' - r_1')(\beta - \beta') \neq 0$$

IFF:

$\overbrace{r_2 \neq r_1, r_2' \neq r_1'}^{\text{for construction}}, \beta \neq \beta' \rightarrow$ this last condition is true because we picked H_s collision resistant.

Otherwise H_s computed on different elements((c_1, c_2, l) and (c_1', c_2', l')) will have to output the same $\beta (= \beta')$. □

Lesson 18

Cramer-Shoup scheme construction

From the above two proof systems we can construct a PKE scheme, which is attributed to Cramer and Shoup:

TODO 21: split definition from correctness

- $(pk, sk) \leftarrow\$ \mathcal{KG}en$, where:
  - $pk := (h_1, h_2, h_3) = (g_1^{x_1} g_2^{x_2}, g_1^{x_3} g_2^{x_4}, g_1^{x_5} g_2^{x_6})$
  - $sk := (x_1, x_2, x_3, x_4, x5, x_6)$
- Encryption procedure:
  - $r \leftarrow\$ \mathbb{Z}_q$
  - $\beta = H_s(c_1, c_2, c_3) = (g_1^r, g_2^r, h_1^r m)$
  - $Enc(pk, m) = (c_1, c_2, c_3, (h_2 h_3^{\beta})^r)$
- Decryption procedure:
  - Check that $c_1^{x_3+\beta x_5} c_2^{x_4+\beta x_6} = c_4$. If not, output FALSE.
  - Else, output $\hat{m} = c_3 c_1^{-x_1} c_2^{-x_2}$

18.1 Digital signatures

In this section we explore the solutions to the problem of authentication with the use of an asymmetric key scheme. Some observations are in order:

- In a symmetric setting, a verifier routine could be banally implemented as recomputing the signature using the shared secret key and the message. Here, Bob cannot recompute σ as he's missing Alice's secret key (and for good reasons too...). Thus, the verifying routine must be defined otherwise

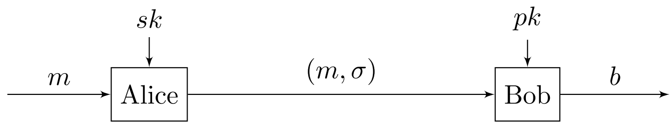


Figure 18.45: Asymmetric authentication

- In a vaguely similar manner to how an attacker could encrypt messages by itself in the asimmetric scenario, because the public key is known to everyone, any attacker can verify any signed messages, for free

Nevertheless, proving that a DS scheme is secure is largely defined in the same way as in the symmetric scenario, with the UF-CMA property:

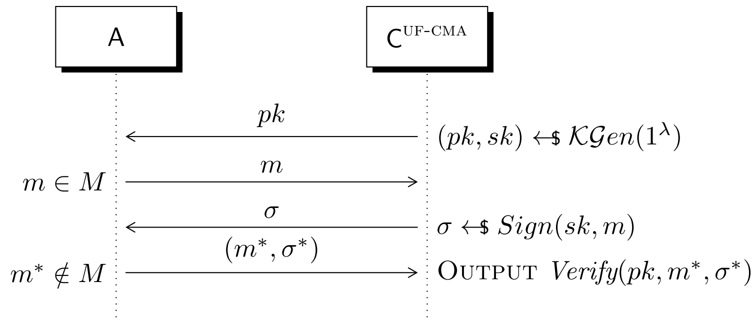


Figure 18.46: Unforgeable digital signatures

18.1.1 Public Key Infrastructure

The problem now is that Alice has a public key, but she wants "the blue check" over it, so Bob is sure that public key comes only from the true Alice.

To obtain the blue check, Alice needs an universally-trusted third party, called *Certification Authority*, which will provide a special *signature* to Alice for proving her identity to Bob.

The scheme works as follows:

The CA message pk' is also called as $cert_A$, the signature of the CA for Alice.

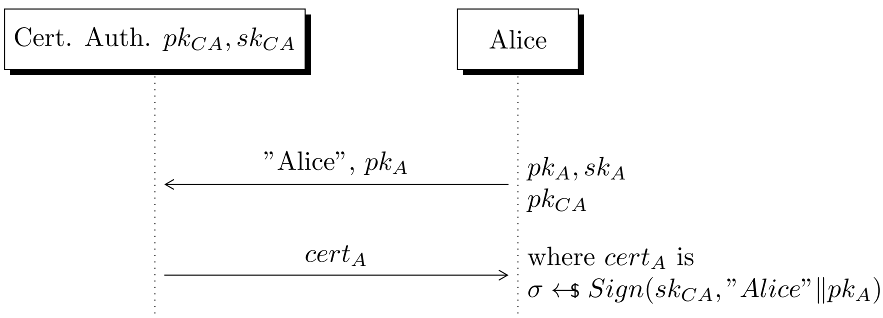
Now, when Bob wants to check the validity of the Alice's public key:

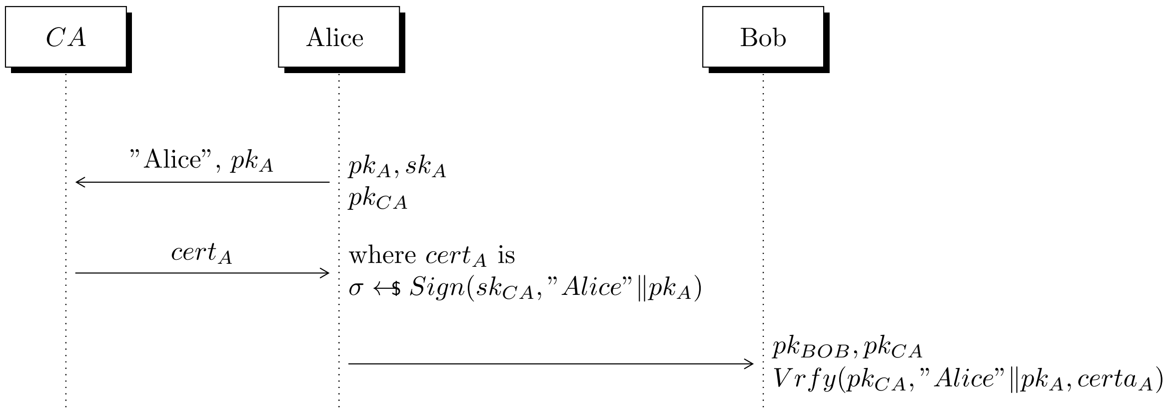
How can Bob recognize a valid certificate from an expired/invalid one?

The infrastructure provides some servers which contain the lists of the currently valid certificates.

Theorem 41. *Signatures are in **Minicrypt**.* ◇

This is a counterintuitive result, not proven during the lesson, but very interesting because it implies that we can create valid signatures only with hash functions, without considering at all public key encryption.





In the next episodes:

- Digital Signatures from TDP*
- Digital Signatures from ID Scheme*
- Digital Signatures from CDH

Where * appears, something called *Random Oracle Model* is used in the proof. Briefly, this model assumes the existance of an ideal hash function which behaves like a truly random function (outputs a random y as long as x was never queried, otherwise gives back the already taken y).

Lesson 19

TODO 22: Warining: e' venuto fuori un casino in questa lezione, ho cercato di riordinare le cose

19.1 Bilinear Map

Let's define a **Bilinear Group** as $(\mathcal{G}, \mathcal{G}_t, q, g, \hat{e}) \leftarrow BilGen(1^\lambda)$ where:

- $\mathcal{G}, \mathcal{G}_t$ are prime order groups (order q).
- g is a random generator of $\mathcal{G}$.
- $(\mathcal{G}, \cdot)$ is a multiplicative group and $\mathcal{G}_t$ is called target group.
- $\hat{e}$ is a (bilinear) MAP: $\mathcal{G} \times \mathcal{G} \to \mathcal{G}_t$ efficiently computable. Defined as follows:

$$\to \text{ Take a generator } g \text{ for } \mathcal{G} \text{ and an element } h \text{ in } \mathcal{G}.$$
$$\forall g, h \in \mathcal{G}, a, b \in \mathcal{Z}_q \hat{e}(g^a, h^b) = \hat{e}(g, h)^{ab} = \hat{e}(g^{ab}, h)$$
$$\text{and } \hat{e}(g, g) \neq 1 \text{ (Non degenerative)}$$

"I can move the exponents"

Venturi said something here related to Weil pairing over an elliptic curve. I found this. Interesting but not useful.
Assumption: CDH is HARD in $\mathcal{G}$.
Observation: DDH is EASY in $\mathcal{G}$!

Proof. $(g, g^a, g^b, g^c) \approx_c (g, g^a, g^b, g^{ab}) \implies \hat{e}(g^a, g^b) =_? \hat{e}(g, g^c)$ now just move the exponents and I can transform the first element in $\hat{e}(g, g^{ab})$ and check if it is equal to $\hat{e}(g, g^c)$. $\square$

Now $KGen(1^\lambda)$ will:

- Generate some params $= (\mathcal{G}, \mathcal{G}_t, g, q, \hat{e}) \leftarrow BilGen(1^\lambda)$
- Pick $a \leftarrow \$\mathcal{Z}_q$ then $g_1 = g^a$
- Pick $g_2 = g^b$ and $g_2, u_0, u_1, ..., u_k \leftarrow \$\mathcal{G}$.
- Then output:
  - $P_k = (params, g_1, g_2, u_0, ..., u_k)$

- $S_k = g_2^a = g^{ab}$

Sign(S_k, m):

- Divide the message m of length k in bits and not it as follows: $m = (m[1], ..., m[k])$
- Now define $\alpha(m) = u_0 \prod_{i=1}^{k} u_i^{m[i]}$
- Pick $r \leftarrow \mathcal{Z}_q$ and output the signature $\sigma = (\underbrace{g_2^a}_{S_k} \cdot \alpha(m)^r, g^r) = (\sigma_1, \sigma_2)$

Vrf(P_k, m, (σ_1, σ_2))

- Check $\hat{e}(g, \sigma_1) = \hat{e}(\sigma_2, \alpha(m)) = \hat{e}(g_1, g_2)$

Correctness: *"Just move the exponents"*

$\hat{e}(g, \sigma_1) = \overbrace{\hat{e}(g, g_2^a \cdot \alpha(m)^a)}^{\text{I can separate the second part for bilinearity}} = \hat{e}(g, g_2^a) \cdot \hat{e}(g, \alpha(m)^r) = \hat{e}(\underbrace{g^a}_{g_1}, g_2) \cdot \hat{e}(\underbrace{g^r}_{\sigma_2}, \alpha(m)) = \hat{e}(g_1, g_2) \cdot \hat{e}(\sigma_2, \alpha(m))$

We can say that we are moving the exponents from the "private domain" to the "public domain".

19.2 Waters signatures

Theorem 42. *The Waters' signature scheme is* UF-CMA ◇

Proof. The trick is to "program the 'u's " (Venturi)

T0D0 23: Sequence is incomplete/incorrect, have to study it more...

T0D0 24: The following explanation is roundabout, will rectify later

The following describes how the Waters' challenger constructs the u string. The main idea is, given k as the message length, to choose each single bit of u from 1 up to k such that:

$$\alpha(m) = g_2^{\beta(m)} g^{\gamma(m)}, \quad \beta(m) = x_0 + \sum_{i=1}^{k} m_i x_i, \quad \gamma(m) = y_0 + \sum_{i=1}^{k} m_i y_i$$

where $x_0 \leftarrow\!\$ \{-kl, \ldots, 0\}, x_{1—k} \leftarrow\!\$ \{0, \ldots, l\}, y_{0—k} \leftarrow\!\$ \mathbb{Z}_q$

In particular: $l = 2q_s$, where q_s is the number of sign queries made by the adversary.

Therefore, let $u_i = g_2^{x_i} g^{y_i} \quad \forall i \in [0, k]$. Then:

$$\alpha(m) = g_2^{x_0} g^{y_0} \prod_{i=1}^{k} (g_2^{x_i} g^{y_i})^{m_i} = g_2^{x_0 + \sum_{i=1}^{k} m_i x_i} g^{y_0 + \sum_{i=1}^{k} m_i y_i} = g_2^{\beta(m)} g^{\gamma(m)}$$

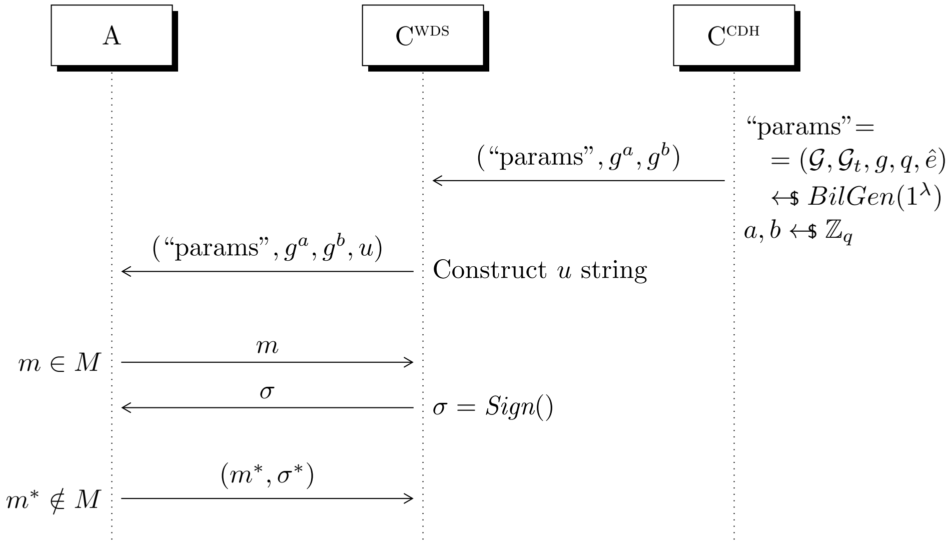


Figure 19.47: Reducing Waters' scheme to CDH

TODO 25: Partizioni, doppi if.... qui non ci ho capito 'na mazza

Step 1: $\sigma = (\sigma_1, \sigma_2) = (g_2^a \alpha(m)^{\bar{r}}, g^{\bar{r}})$, for $\bar{r} \leftarrow\$ \ \mathbb{Z}_q \quad \bar{r} = r - a\beta^{-1}$

$$\begin{aligned}
\sigma_1 &= g_2^a \alpha(m)^{\bar{r}} \\
&= g_2^a \alpha(m)^{r - a\beta^{-1}} \\
&= g_2^a (g_2^{\beta(m)} g^{\gamma(m)})^{r - a\beta^{-1}} \\
&= g_2^a g_2^{\beta(m)r - a} g^{\gamma(m)r - \gamma(m)a\beta^{-1}} \\
&= g_2^{\beta(m)r} g^{\gamma(m)r} g^{-\gamma(m)\beta^{-1}}
\end{aligned}$$

$\square$

Lesson 20

20.1 Random Oracle Model (ROM)

The Random Oracle Model treats an Hash function $\mathcal{H}$ as a truly random function.

20.2 ID Scheme

20.3 Honest Verifier Zero Knowledge (HVZK) and Special Soundness (SS)