Contents

Lesson 1

Talking cryptography is usually done in the "confidentiality" realm, where two characteristics in a communication channel are desirable: it must be *secret*, and *authentic*.

Modern confidentiality/authentication systems are forged under *Kerckhoffs's principle*, which states that a secure system shall only rely on the encryption keys, and not on the underlying algorithm's secrecy; in short, *"No security by obscurity"*. However sharing the key between two parties without the risk of eavesdropping is a costly operation.

1.1 Secret communication

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

The typical objects defined and used in secret communication topics are:

- $\mathcal{K}$: Key space
- $\mathcal{M}$: Message space
- $\mathcal{C}$: Ciphertext space
- $\mathsf{Enc} \in \mathcal{K} \times \mathcal{M} \to \mathcal{C}$: Encryption routine
- $\mathsf{Dec} \in \mathcal{K} \times \mathcal{C} \to \mathcal{M}$: Decryption routine

Enc and Dec form a *cryptographic secrecy scheme*, or just *secrecy scheme* for brevity, and as such, it must abide by the rule:

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

Definition 1. (*Shannon's "Perfect secrecy"*): Let M be any distribution over the message space $\mathcal{M}$, and K a uniform distribution over $\mathcal{K}$. Then, the cryptosystem $\Xi : (\mathsf{Enc}, \mathsf{Dec})$ is deemed *perfectly secret* iff the ciphertext obtained by applying the encryption routine to a message sampled by M using tke key in K is effectively useless in retrieving any info about the message itself, apart from the key-supplied decryption routine. Formally:

$$\forall M \sim \mathcal{R}and(\mathcal{M}), \forall C \sim \mathcal{R}and(\mathcal{C}) \implies \Pr[M = m] = \Pr[M = m \mid C \xmapsto{\$} c]$$

$\diamond$

Note how this definition doesn't involve the encryption key.

The perfect secrecy definition can be rephrased in different ways, bringing more details to light:

1. $\Pr[M = m] = \Pr[M = m \mid C \xmapsto{\$} c]$
2. $M \perp\!\!\!\perp C$
3. $\forall m_1, m_2 \in \mathcal{M}, c \in \mathcal{C} \implies \Pr[\mathsf{Enc}(K, m_1) = c] = \Pr[\mathsf{Enc}(K, m_2) = c]$

A remark has to be done here:

$$\begin{aligned} &\Pr[\mathsf{Enc}(K, m) = c] \\ =&\Pr[\mathsf{Enc}(K, M) = c \mid M \xmapsto{\$} m] \\ \neq&\Pr[\mathsf{Enc}(K, M) = c] \end{aligned}$$

which is exactly the difference between picking a specific message m (as in: *choosing* m), and picking it at random, just as M describes.

Proposition 1. *All the previous statements are equivalent.* ◇

Proof. The proof is structured as a cyclic implication between the threee definitions:

- (1) $\implies$ (2): Start from one side of the independency ddefinition, and work through the other:

$$\begin{aligned} &\Pr[C = c \wedge M = m] \\ =&\Pr[C = c] \Pr[M = m \mid C \xmapsto{\$} c] && \text{(Conditioned prob.)} \\ =&\Pr[C = c] \Pr[M = m] && \text{(Using 1.)} \end{aligned}$$

  This proves that M and C are independent distributions.

- (2) $\implies$ (3): For the proof's purposes, let M be an arbitrary distribution over $\mathcal{M}$; recall that, by definition: $C := \mathsf{Enc}(K, M)$:

$$\begin{aligned} &\Pr[\mathsf{Enc}(K, m_1) = c] \\ =&\Pr[\mathsf{Enc}(K, M) = c \mid M \xmapsto{\$} m_1] && \text{(Introducing } M\text{)} \\ =&\Pr[C = c \mid M \xmapsto{\$} m_1] && (C \text{ defintion}) \\ =&\Pr[C = c] && \text{(Using 2.)} \\ =&\ldots && \text{(Same steps reversed, where } m_1 \mapsto m_2\text{)} \\ =&\Pr[\mathsf{Enc}(K, m_2) = c] \end{aligned}$$

- (3) $\implies$ (1):

$$
\begin{aligned}
& \Pr[C = c] \\
=& \sum_m \Pr[C = c \wedge M = m] && \text{(Total prob.)} \\
=& \sum_m \Pr[\mathsf{Enc}(K, M) = c \wedge M = m] && (C \text{ definition}) \\
=& \sum_m \Pr[\mathsf{Enc}(K, M) = c \mid M \xmapsto{\$} m] \Pr[M = m] && \text{(Cond. prob. def.)} \\
=& \sum_m \Pr[\mathsf{Enc}(K, m) = c] \Pr[M = m] && \text{(Cond. collapse)} \\
=& \Pr[\mathsf{Enc}(K, \overline{m}) = c] \sum_m \Pr[M = m] && \text{(Using 3.)} \\
=& \Pr[\mathsf{Enc}(K, \overline{m}) = c] && \text{(Total prob.)} \\
=& \Pr[\mathsf{Enc}(K, M) = c \mid M \xmapsto{\$} \overline{m}] && \text{(Introducing } M) \\
=& \Pr[C = c \mid M \xmapsto{\$} \overline{m}] && (C \text{ definition})
\end{aligned}
$$

Knowing this, and applying Bayes' theorem, we get back to the first definition:

$$
\begin{aligned}
& \Pr[C = c] = \Pr[C = c \mid M \xmapsto{\$} m] \\
\implies & \Pr[C = c] = \Pr[M = m \mid C \xmapsto{\$} c] \frac{\Pr[C = c]}{\Pr[M = m]} && \text{(Bayes' theorem)} \\
\implies & \Pr[M = m] = \Pr[M = m \mid C \xmapsto{\$} c]
\end{aligned}
$$

Thus, we conclude that all three definitions for perfect secrecy are equivalent. □

1.1.1 One Time Pad (OTP)

Let $\mathcal{K} = \mathcal{M} = \mathcal{C} = \mathbb{2}^l$, and define the following cryptographic scheme:

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

Proof of correctness goes like: $\mathsf{Dec}(k, \mathsf{Enc}(k, m)) = \mathsf{Dec}(k, k \oplus m) = k \oplus k \oplus m = m$.

Theorem 1. *The* One-time pad *scheme is perfectly secret.* ◇

Proof. Let $K \sim \mathcal{U}nif(\mathcal{K})$. Then, $\forall m_1, m_2, c \in \mathbb{2}^l$:

$$\begin{aligned}
&\Pr[\mathsf{Enc}(K, m_1) = c] \\
=&\Pr[K \oplus m_1 = c] \\
=&\Pr[K = c \oplus m_1] \\
=&|\mathcal{K}|^{-1} && \text{(K is uniform)} \\
=&\dots && \text{(Same steps reversed, where } m_1 \mapsto m_2\text{)} \\
=&\Pr[\mathsf{Enc}(K, m_2) = c]
\end{aligned}$$

This satisfies the third definition of perfect secrecy. □

By observing our recent proof, some insights (and problems) arise:

1. The key and the message's lengths must always match ($|k| = |m|$);
2. As the name suggests, keys are useful just for one encryption. Otherwise, given two encryptions witht he same key, an attacker may exploit the XOR's idempotency to extract valuable information from both ciphertexts[1]:

$$c_1 = k \oplus m_1 \wedge c_2 = k \oplus m_2 \implies c_1 \oplus c_2 = m_1 \oplus m_2$$

Combined with the fact that keys must be preemptively shared in a secure fashion, these problems make for a quite impractical cryptographic scheme. One can further this analysis and generalize it to all "perfect" schemes, giving a rather delusive conclusion:

Theorem 2. *For a secrecy scheme to be perfectly secret:* $|\mathcal{K}| \geq |\mathcal{M}|$ ◇

Proof. Perfection will be disproved by breaking the first definition. Let $M \sim \mathcal{U}nif(\mathcal{M})$, and take $c \in \mathcal{C} : \Pr[C = c] > 0$. Consider $D = \{\mathsf{Dec}(k, c) : k \in \mathcal{K}\}$ as the set of all images of the decryption routine with all keys in $\mathcal{K}$. The offending assumption is that $|\mathcal{K}| < |\mathcal{M}|$. Then, by how D is defined:

$$|D| \leq |\mathcal{K}| < |\mathcal{M}| \implies \exists m \in \mathcal{M} \setminus D$$

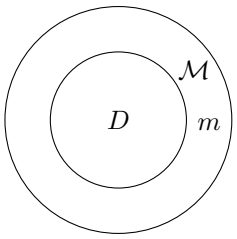


Figure 1.1: Where the messages stand

Fix this message m, and remember that by M's definition, $\Pr[M = m] = |\mathcal{M}|^{-1}$. Since $m \notin D$, there can be no key in $\mathcal{K}$ such that $\mathsf{Dec}(k, c) = m$. By

[1] This vulnerability of applying a function on a ciphertext, and expecting as a result the image of the original message by the same function, is called *malleability*, and is explored further in the notes.

observing that C strictly distributes over D, where m is not present, $\Pr[M = m \mid C \xmapsto{\$} c] = 0$. Summarizing up:

$$0 = \Pr[M = m \mid C \xmapsto{\$} c] \neq \Pr[M = om] = |\mathcal{M}|^{-1}$$

This clearly violates the first definition of perfect secrecy. □

Lesson 2

2.1 Authentic communication

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

2.1.1 Message Authentication Code scheme

Syntax:

- $\mathcal{K}$: Key space
- $\mathcal{M}$: Message space
- $\mathcal{C}$: Ciphertext space
- Φ: Tag (or signature) space
- $\mathsf{Tag} \in \mathcal{K} \times \mathcal{M} \to \Phi$: Tagging (or signing) function
- $\mathsf{Ver} \in \mathcal{K} \times \mathcal{M} \times \Phi \to \mathbb{2}$: Verifying (or validating) function

Tag and Ver form a *cryptographic authentication scheme*, or just *authentication scheme*, and it must abide by the rule:

$$\forall m \in \mathcal{M}, \forall k \in \mathcal{K} \implies \mathsf{Ver}(k, m, \mathsf{Tag}(k, m)) = 1$$

In the usual case (deterministic), the verifier consists only of an equality check, reusing the Tag routine: $\mathsf{Tag}(k, m) = \phi$

The security aspect to consider in these schemes is the signatures' *unforgeability*. Suppose that an attacker chooses a message m and obtains its tag ϕ without knowing the key k used in the process; then there shall be no better means of declaring a couple $(m', \phi') : \mathsf{Ver}(k, m', \phi') = 1$ other than random guessing, or knowing the key, of course.

Some more formal definitions of unforgeability follow:

Definition 2. *(ε-statistical one-time security)*: A given authentication scheme has ε-statistical one-time security iff, given a valid couple (m_1, ϕ_1), any adversary cannot *forge* a fresh valid couple (m_2, ϕ_2) without knowing the signature key k. Formally:

$$\forall m_1, m_2 \in \mathcal{M} : m_1 \neq m_2, \forall \phi_1, \phi_2 \in \Phi$$
$$\Downarrow$$
$$\Pr[\mathsf{Tag}(K, m_2) = \phi_2 \mid \mathsf{Tag}(K, m_1) \not\mapsto \phi_1] \leq \varepsilon$$

◇

Pairwise-independent hashing

Definition 3. Let $\mathcal{S}$ be a seeding space; define a family of *hash functions* to be the following object[2]:

$$H \in \mathcal{S} \to (\mathcal{M} \to \Phi) : s \mapsto h_s$$

Let S be a random variable in the seed space; the hash functions are deemed *pairwise-independent*[3] iff, for any two distinct messages m and m', the pair $(h_S(m), h_S(m'))$ distributes evenly in Φ^2. In other words:

$$\forall m, m' \in \mathcal{M} : m \neq m', \forall \phi, \phi' \in \Phi \implies \Pr[h_S(m) = \phi \wedge h_S(m') = \phi'] = \frac{1}{|\Phi|^2}$$

◇

As an example of such a family, consider the additive group of integers modulo p: $(\mathbb{Z}_p, +)$, where p is a prime integer. Define the family:

$$h_{(a,b)}(x) = ax + b \bmod p$$

where $\mathcal{S} = \mathbb{Z}_p^2$, and $\mathcal{M} = \Phi = \mathbb{Z}_p$.

Theorem 3. *The functions in the family H are pairwise-independent.* ◇

Proof. Let $S = (a, b)$ be a random seed for H; for any distinct messages m, m', and for any tags ϕ, ϕ':

$$\begin{aligned}
&\Pr[h_S(m) = \phi \wedge h_S(m') = \phi'] \\
=&\Pr[am + b = \phi \wedge am' + b = \phi'] \\
=&\Pr\left[\begin{pmatrix} m & 1 \\ m' & 1 \end{pmatrix} \cdot \begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} \phi \\ \phi' \end{pmatrix}\right] \\
=&\Pr\left[\begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} m & 1 \\ m' & 1 \end{pmatrix}^{-1} \cdot \begin{pmatrix} \phi \\ \phi' \end{pmatrix}\right] \\
=&\frac{1}{|\mathbb{Z}_p|^2}
\end{aligned}$$

which is exactly the definition of pairwise-independency. □

Theorem 4. *Define an authentication scheme to be such that its tagging routine is a hash function family: $\mathsf{Tag}(k, m) = h_k(m)$. Let this function family be pairwise-independent. Then the authentication scheme is $\frac{1}{|\Phi|}$-statistical one-time secure.* ◇

[2] This kind of notation consisting in putting an argument as a subscript to a generic, typically of higher-order function, is also called "currying"; it will be used extensively throughout the lessons.

[3] Care should be taken to not confuse *pairwise* independency with *mutual* independency: while the former acts only on pairs, the latter considers all possible subsets. The two notions are not necessarily equivalent.

Proof.

$$\forall m \in \mathcal{M}, \phi \in \Phi \implies \Pr[\mathsf{Tag}(K, m) = \phi] = \Pr[h_K(m) = \phi] = \frac{1}{|\Phi|}$$
$$\forall m \neq m' \in \mathcal{M}, \phi, \phi' \in \Phi \implies \Pr[\mathsf{Tag}(K, m) = \phi \wedge \mathsf{Tag}(K, m') = \phi'] = \frac{1}{|\Phi|^2}$$

Therefore:

$$\begin{aligned}
&\Pr[\mathsf{Tag}(K, m') = \phi' \mid \mathsf{Tag}(K, m) \$\!\!\mapsto \phi] \\
=&\frac{\Pr[\mathsf{Tag}(K, m') = \phi' \wedge \mathsf{Tag}(K, m) = \phi]}{\Pr[\mathsf{Tag}(K, m) = \phi]} \\
=&\frac{|\Phi|}{|\Phi|^2} = \frac{1}{|\Phi|}
\end{aligned}$$

□

Theorem 5. *Any $(2^{-\lambda})$-statistical t-time secure authentication scheme has a key of size $(t+1)\lambda$ for any $\lambda > 0$.* ◇

Proof. None given.

□

Lesson 3

3.1 Randomness Extraction

In most of our discourse, the subject of uniformly random variables is much recurrent; this chapter/lesson delves deeper into the topic. For starters, we devise some attempts to extract uniform randomness from "non-uniform" randomness sources.

Suppose to have a biased coin $B \sim \mathcal{B}er(p) : p \neq \frac{1}{2}$. How to craft a fair coin out of it? In his time, Von Neumann devised a simple algorithm, which is now known as the *Von Neumann extractor*:

1. Let $B \sim \mathcal{B}er(p)$ be a random variable
2. Sample $b_1 \xleftarrow{\$} B$
3. Sample $b_2 \xleftarrow{\$} B$
4. If $b_1 = b_2$ go to step 2
5. Else:
   - If $b_1 = 0 \wedge b_2 = 1$ output 1
   - If $b_1 = 1 \wedge b_2 = 0$ output 0

Some considerations can be made: The probability of both single cases in step 5 is $p(1-p)$, therefore the probability to reach it is $2p(1-p)$. Also, it is apparent that the number of possible failures in reaching step 5 follow a geometric distribution in p, thus the probability of increased number of failures decrease exponentially.

We now get back to our ultimate goal. Let X be any random variable over a space Ω, we wish to design an "extraction" algorithm Ext such that $U = \mathsf{Ext}(X)$ distributes uniformly over Ω. To help ourselves, we will deal with probability spaces of binary strings (2^n), and define a measure of "how much" a distribution is uniform over its space:

Definition 4. Let X be a random variable from a given probability distribution. Its *min-entropy* is defined as follows:

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

$\diamond$

Using this measure, we can already see an interesting case, which involves "constant" random variables:

$$\begin{aligned} X \sim \mathcal{C}onst(\overline{x}) &\implies \Pr[X = \overline{x}] = 1 \\ &\implies \Pr[X \neq \overline{x}] = 0 \\ &\implies H_\infty(X) = -\log_2(\Pr[X = \overline{x}]) = -\log_2(1) = 0 \end{aligned}$$

And in fact, a constant variable is useless in creating a uniform distribution: it always gives the same outcome, making everything deterministic. Therefore, such variables must be excluded in our search for a "universal extractor". On the other hand, looking at a uniform distribution:

$$\begin{aligned} X \sim \mathcal{U}nif(\Omega) &\implies \forall x \Pr[X = x] = \frac{1}{|\Omega|} \\ &\implies H_\infty(X) = -\log_2(\frac{1}{|\Omega|}) \end{aligned}$$

Knowing that Ω is be our usual domain choice of binary strings of a given length $\mathbb{2}^n$, the min-entropy becomes exactly n[4]. Using this measure, we can actually seek how much min-entropy we require in the original distribution X in order for the extractor to return a uniform distribution. Ideally, we would like a value as close to 0 as possible, because a min-entropy of zero leads to constant variables, which have been excluded beforehand. Alas, it turns out that:

Claim 1. *There is no such universal* Ext *algorithm that returns a uniform distribution from random variables X with min-entropy $H_\infty(X) \leq n - 1$* ◇

Proof. TODO 3: Help with the proof, things don't look good

□

So this approach is doomed, unless we factor in a preemptive small amount of true randomness in the algorithm. This is what a *seeded extractor* does:

$$\mathsf{Ext} : \underbrace{\mathbb{2}^d}_{seed(public)} \times \underbrace{\mathbb{2}^n}_{input} \to \underbrace{\mathbb{2}^l}_{output}$$

Before giving a formal definition of such an extractor, we require another notion of measure related to probability distributions:

Definition 5. *Statistical Distance*: Let X and Y be two random variables on the same probability space. Their *statistical distance* is defined as follows:

$$SD(X, Y) = \frac{1}{2} \sum_{x \in \Omega} |\Pr[X = x] - \Pr[Y = x]|$$

◇

[4]This also sheds some light in how string length is a frequent topic in the cryptography realm, as it usually expresses a cryptosystem's strength: the greater its min-entropy, the harder it is to find the right key from scratch for a ciphertext.

In an intuitive stance, this distance amounts to half the area delimited by the two distributions.

TODO 4: Image of the statistical distance

Definition 6. Let $\mathsf{Ext} \in \mathbb{2}^d \times \mathbb{2}^n \to \mathbb{2}^l$ be a seeded extractor, and $S \sim \mathcal{U}nif(\mathbb{2}^d)$. Then it is a (k, ε)-extractor iff:

$$\forall X : H_\infty(X) \geq k \implies SD((S, \mathsf{Ext}(S, X)), (S, \mathcal{U}nif(\mathbb{2}^l))) \leq \varepsilon$$

◇

Do note that S takes part in both sides of the statistical distance: this is to be interpreted that the seed is known at the time of extraction.

3.1.1 Universal hash functions

Getting back to our hash function families, we see that they too use an argument as a random seed, and attempt to be as uniform as possible; thus they behave in most ways as seeded extractors. Let's further develop the idea:

Definition 7. Let S be a uniform seed. A hash function family H is deemed *universal* iff:

$$\forall a \neq b \in \Omega \implies \Pr[h_S(a) = h_S(b)] = \frac{1}{2^l}$$

◇

Definition 8. Let X and Y be two IID random variables; a *collision* is the event of both evaluating to the same outcome. The probability of such an event is:

$$Col(X) = Col(Y) = \Pr[X = Y] = \sum_{x \in \Omega} \Pr[X = x \wedge Y = x] = \sum_{x \in \Omega} \Pr[X = x]^2$$

◇

Leftover hash lemma

TODO 5: —

Lesson 4

4.1 Negligible function

What is exactly a negligible function? Below here there is a possible interpretation of this notion, taken from an answer to a question in the Cryptography Stack Exchange website:

> "[...] in modern cryptographic schemes, we generally do not try to achieve perfect secrecy [...]. Instead, we define security against a specific set of adversaries whose computational power is bounded. Generally, we assume an adversary that is bounded to run in time polynomial to n, where n is the security parameter given to the key generation algorithm [...].
>
> 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 $\frac{1}{\mathcal{P}oly(n)}$ 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 $\mathcal{P}oly(n)$ guesses, it will not be significant since it will only have tried:
>
> $$\frac{\mathcal{P}oly(n)}{superpoly(n)}$$
>
> As n grows, the denominator grows far faster than the numerator and the success probability will not be significant."[5]

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

Definition 9. Let $f : \mathbb{N} \to \mathbb{N}$ be a function. Then it is deemed *polynomial*, and denoted as $f \in \mathcal{P}oly(\lambda)$, iff:

$$\exists c \in \mathbb{N} : f(\lambda) \in O(\lambda^c)$$

$\diamond$

Definition 10. Let $\nu : \mathbb{N} \to \mathbb{R}$ be a function. Then it is deemed *negligible*, and denoted as $\nu \in \mathcal{N}egl(\lambda)$, iff:

$$\forall f \in \mathcal{P}oly(\lambda) \implies \nu(\lambda) \in O\left(\frac{1}{f(\lambda)}\right)$$

$\diamond$

Note that these actually represent upper bounds for functions: a negligible function adheres to the polyomial function definition, whereas the opposite isn't true. To sum it up: $\mathcal{N}egl \subset \mathcal{P}oly$.

Exercise 6. Let $p(\lambda), p'(\lambda) \in \mathcal{P}oly(\lambda)$ and $\nu(\lambda), \nu'(\lambda) \in \mathcal{N}egl(\lambda)$. Then prove the following:

1. $p(\lambda) \cdot p'(\lambda) \in \mathcal{P}oly(\lambda)$
2. $\nu(\lambda) + \nu'(\lambda) \in \mathcal{N}egl(\lambda)$

Solution 1 (2). TODO 6: 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 $\forall n > n_0 \implies \nu(n) + \nu'(n) < \frac{1}{n^c}$.

Consider an arbitrary $c \in \mathbb{N}$. Then, since $c + 1 \in \mathbb{N}$, and both ν and ν' are negligible, there exist n_ν and $n_{\nu'}$ such that:

$$\begin{aligned} \forall n \geq n_\nu &\implies \nu(n) \leq n^{-(c+1)} \\ \forall n \geq n_{\nu'} &\implies \nu'(n) \leq n^{-(c+1)} \end{aligned}$$

Fix $n_0 = \max(n_\nu, n_{\nu'})$. Then, since $n_0 \geq 2$, $\forall n \geq n_0$ we have:

$$\begin{aligned} &\nu(n) + \nu'(n) \\ \leq &n^{-(c+1)} + n^{-(c+1)} \\ = &2n^{-(c+1)} \\ \leq &n^{-c} \end{aligned}$$

Therefore, we conclude that $\nu(n) + \nu'(n) \in \mathcal{N}egl(\lambda)$.

4.2 One-way functions

From here, we start defining an object that is fundamental to everyday cryptography: the *one-way* function, or OWF in short. Colloquially, a one-way function is a function that is "easy to compute", while being "hard to invert" a the same time, the concept of hardness being borrowed by complexity theory.

Definition 11. Let $f : 2^{n(\lambda)} \to 2^{n(\lambda)}$ be a function. Then it is a OWF iff:

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

◇

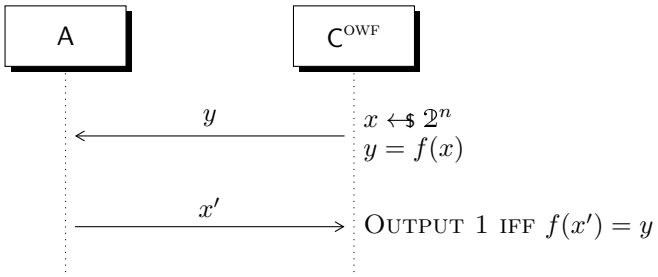


Figure 4.2: One-Way Function hardness

The structure of the "game" appearing in the definition is depicted in figure 4.2. Do note that the game does not check for $x = x'$, but rather for $f(x) = f(x')$; in a sense, the adversary is not trying to guess what the original x was: its goal is to find any value such that its image is y according to f, and such value may very well not be unique.

Exercise 7. Prove the following claims:

1. There exists an inefficient adversary that wins $\text{GAME}^{\text{OWF}}_{f,\mathsf{A}}$ with probability 1
2. There exists an efficient adversary that wins $\text{GAME}^{\text{OWF}}_{f,\mathsf{A}}$ with probability 2^{-n}

Solution 2 (7)**.**

1. Adversary uses a brute-force attack.
2. Adversary makes a random guess.

A one-way function can be thought as a function which is very efficient in generating "puzzles" that are very hard to solve from scratch. Furthermore, given a candidate solution, one can efficiently verify its validity. In a twist of perspective, for a given couple $(\mathcal{P}_{\text{GEN}}, \mathcal{P}_{\text{VER}})$ of a puzzle generator and a puzzle verifier, another "game" can be drawn as in figure 4.3.

It can also be said that the one-way puzzle problem is in NP, because witness checking is easy, but not in P because finding a solution to begin with is hard.

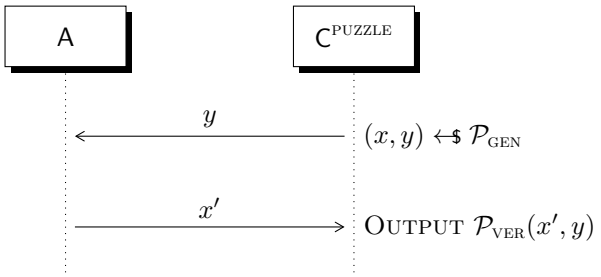


Figure 4.3: The puzzle game

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 12 (*Comp. indist.*)**.** Let X and Y be two distribution sequences; they are deemed *computationally indistinguishable*, written as "$X \approx_c Y$" iff:

$$\forall \mathsf{A} \in \text{PPT} \exists \nu(\lambda) \in \mathcal{N}egl(\lambda) : |\Pr[\mathsf{A}(X_\lambda) = 1] - \Pr[\mathsf{A}(Y_\lambda) = 1]| \leq \nu(\lambda)$$

$\diamond$

In words: any *efficient* adversary attempting to distinguish outputs between the two ensembles will succeed with a probability that is negligibly different than randomly guessing. Note the emphasis on "efficient", which makes this relationship between ensembles weaker than what would be a purely statistical one.

With the purpose of making these new concepts clearer, it is presented this mental game.

TODO 7: 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.

A *challenger* C chooses a value z among X_λ and Y_λ, and gives it to a *distinguisher* D. In turn, D has to correctly guess which was the source of z: either X_λ or Y_λ.

If we let X_λ and Y_λ to be *computationally indistinguishable*, then, fixed 1 as one of the sources, the probability that D says "1!" when C picks z from X_λ is *not so far* from the probability that D says "1!" when C picks 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 information avaliable to 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. *Let f be a function that has polynomial time-complexity. Then, for any two ensembles X and Y:*

$$X \approx_c Y \implies f(x) \approx_c f(y)$$

◇

Proof. This proof is by contradiction and uses a reduction. Let $X \approx_c Y$ be two indistinguishable ensembles, and $f \in \text{PPT}$ an arbitrary poly-time complex function. Assume there exists an adversary A to the challenge of distinguishing the ensembles' images $f(X)$ from $f(Y)$ that does efficiently succeed, as shown in figure 4.4.

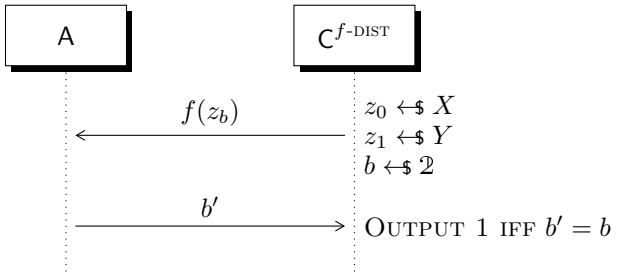


Figure 4.4: A distinguisher for f

Fix this adversary to be the distinguisher D_f. From here, another adversary $\mathsf{A} \in \text{PPT}$ can use D_f to effectively distinguish the original ensembles, as depicted in figure 4.5:

1. A asks for the original sample from the challenger
2. A applies f on the sample
3. A relays the resulting image to D_f
4. D_f replies with his outcome
5. A relays the outcome to the challenger

All of this is done in polynomial time, since all functions and machines involved in the process opeate in PPT. This contradicts the computational indistinguishablilty of X and Y.

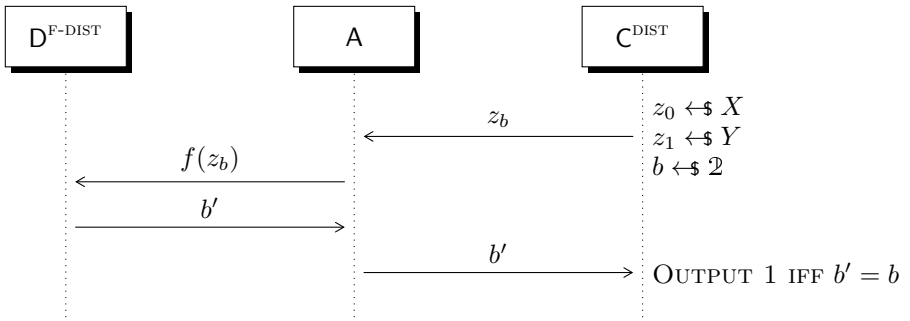


Figure 4.5: Distinguisher reduction

□

4.4 Pseudo-random generators

A deterministic function $G \in 2^\lambda \to 2^{\lambda+l(\lambda)}$ is called a *pseudo-random generator*, or PRG in short, iff:

- $G \in \text{PPT}(\lambda)$
- $|G(s)| = \lambda + l(\lambda)$
- Given U_n to be a distribution ensemble of n uniform random variables:

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

So, if we take $s \leftarrow\!\!\$\ U_\lambda$, the output of G will be indistinguishable from a random pick from $U_{\lambda+l(\lambda)}$.

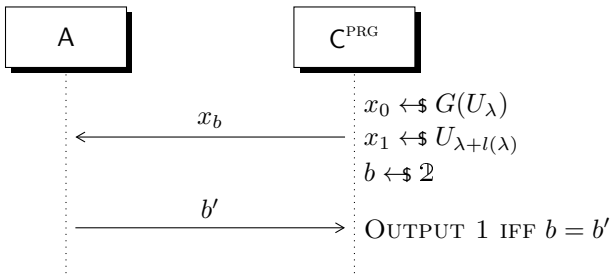


Figure 4.6: The pseudorandom game

Lesson 5

This chapter/lesson is devoted in constructing PRGs. We begin by assuming to have already a PRG $G \in \mathbb{2}^\lambda \to \mathbb{2}^{\lambda+1}$, that extends the string length by one bit, and prove that it is possible to extend such string by an indefinite amount while preserving pseudo-randomness.

5.1 Stretching a PRG

Consider this algorithm that uses G to construct G^l, as depicted in figure 5.7:

1. Let $s_0 \xleftarrow{\$} \mathbb{2}^\lambda$
2. $\forall i \in [l(\lambda)]$
   (a) let $G(s_{i-1}) = (s_i, b_i)$, where b_i is the extra bit generated by a single use of G
3. Compose $(b_1, b_2, ..., b_{l(\lambda)}, s_{l(\lambda)})$. This will be the returned string, which is $\lambda + l(\lambda)$ bits long

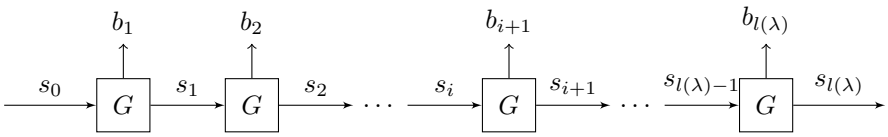


Figure 5.7: Constructing $G^{l(\lambda)}$ from $G(\lambda)$

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 like the one employed back in the OWF topic, and is commonly called the *"hybrid argument"*.

Lemma 2 (*Hybrid argument*)**.** *Let X, Y and Z be three any distribution ensembles of the same length. The following is true:*

$$X \approx_c Y \wedge Y \approx_c Z \implies X \approx_c Z$$

$\diamond$

Proof. $\forall \mathsf{D} \in \text{PPT}$, by using the triangle inequality:

$$\begin{aligned}
&|\Pr[\mathsf{D}(X)=1] - \Pr[\mathsf{D}(Z)=1]| \\
=&|\Pr[\mathsf{D}(X)=1] - \Pr[\mathsf{D}(Y)=1] + \Pr[\mathsf{D}(Y)=1] - \Pr[\mathsf{D}(Z)=1]| \\
\leq&|\Pr[\mathsf{D}(X)=1] - \Pr[\mathsf{D}(Y)=1]| + |\Pr[\mathsf{D}(Y)=1] - \Pr[\mathsf{D}(Z)=1]| \\
\leq&\nu(n) + \nu'(n)
\end{aligned}$$

where $\nu, \nu' \in \mathcal{N}egl(n)$. By the sum property of negligible functions, the result is still negligible, proving the lemma. □

In essence, the hybrid argument proves that computational indistinguishability is a transitive relationship, which enables us to design "hybrid" games in order to bridge differences two arbitrary ones. This property will be very useful in all future proofs, as it will be shown for the coming theorem:

Theorem 8. *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:*

$$G : 2^\lambda \to 2^{\lambda+1} \implies \forall l(\lambda) \in \mathcal{P}oly(\lambda)\ \exists G^l \in 2^\lambda \to 2^{\lambda+l(\lambda)} \qquad \diamond$$

Proof. First off, do observe that, since both G and l are polynomial in λ, then so is $G^{l(\lambda)}$, because it combines G $l(\lambda)$-many times. 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
- $H^i_\lambda := \begin{cases} b_1, ..., b_i \leftarrow\$\ 2 \\ s_i \leftarrow 2^{\lambda+i} \\ (b_{i+1}, ..., b_{l(\lambda)}, s_{l(\lambda)}) := G^{l(\lambda)-i}(s_i) \end{cases}$
- $H^{l(\lambda)}_\lambda := U_{\lambda+l}$

Focusing on two subsequent generic hybrids, as shown in figures 5.8 and 5.9, 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.

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

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

where the first i bits are chosen uniformly at random, and the remaining ones are obtained by subsequent applications of G. It can be observed that:

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

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

$$G^{l(\lambda)}(U_\lambda) = H^0_\lambda \approx_c H^1_\lambda \approx_c \ldots \approx_c H^{l(\lambda)}_\lambda = U_{\lambda+l(\lambda)}$$

which proves that G^l is indeed a PRG. □

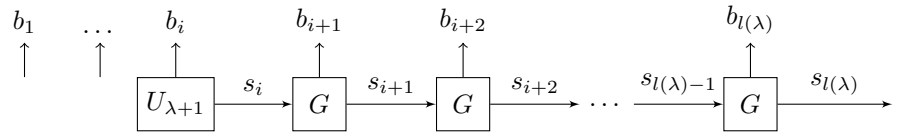


Figure 5.8: H^i_λ

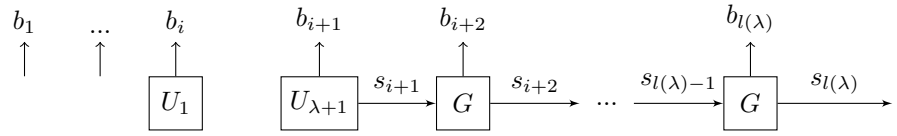


Figure 5.9: H^{i+1}_λ

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 $H^0_\lambda \approx_c \dots \approx_c H^l_\lambda$ where $H^i_\lambda \not\approx_c H^{i+1}_\lambda$. Thus there exists a distinguisher $\mathsf{D}^{\text{I-TH}}$ able to tell apart H^i_λ from H^{i+1}_λ, as shown in figure 5.10:

$$\exists i \in [0,l], \exists \mathsf{D}^{\text{I-TH}} \in \text{PPT} : |\Pr[\mathsf{D}^{\text{I-TH}}(H^i_\lambda) = 1] - \Pr[\mathsf{D}^{\text{I-TH}}(H^{i+1}_\lambda) = 1]| \notin \mathit{Negl}(\lambda)$$

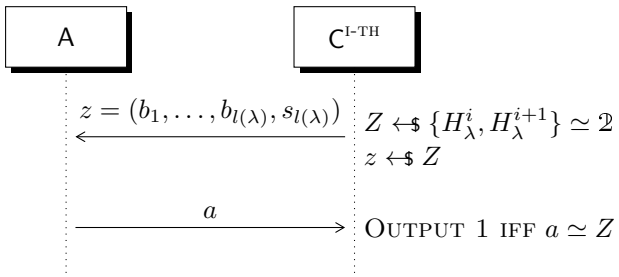


Figure 5.10: Distinguisher for H^i_λ and H^{i+1}_λ

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 5.11. 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.

□

5.2 Hardcore predicates

Now that we've seen how to reuse a one-bit stretch PRG in order to obtain an arbitrary length of pseudorandom bits, we turn to the problem of constructing a 1-bit stretch PRG itself. Let f be a OWF, and consider the following questions:

- Given an image $f(x)$, which bits of the input x are hard to extract?

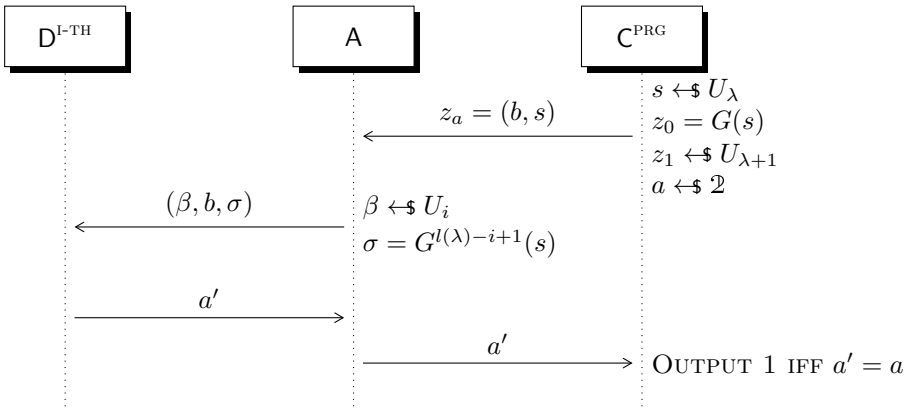


Figure 5.11: Reducing to a distinguisher for G, where $\beta = (b_1, \ldots, b_{i-1})$ and $\sigma = (b_{i+1}, \ldots, b_{l(\lambda)}, s_{l(\lambda))}$

- Is it always true that, given f, the first bit of $f(x)$ is hard to compute for any choice of x?

Example 1. Given an OWF f, then $f'(x) = x_0||f(x)$ is a OWF.

Two definitions for hardcore predicates are given:

Definition 13. Let $f : \mathbb{2}^n \to \mathbb{2}^n$ be a poly-time complex function. A poly-time complex predicate $\mathfrak{hc} : \mathbb{2}^n \to \mathbb{2}$ is said to be *hard-core* for f iff:

$$\forall \mathsf{A} \in \text{PPT} \implies \Pr(\mathsf{A}(f(x)) = \mathfrak{hc}(x) \mid x \leftarrow\!\$\ \mathbb{2}^n) \in \mathcal{N}egl(\lambda)$$

$\diamond$

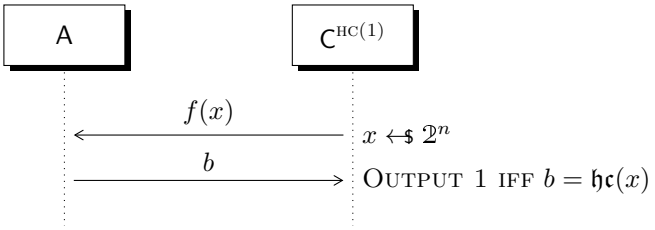


Figure 5.12: The hardcore game, f and $\mathfrak{hc}$ are known

Definition 14. A polynomial time function $\mathfrak{hc} : \mathbb{2}^n \to \mathbb{2}$ is hard-core for a function f iff:

$$(f(X), h(X)) \approx_{\text{C}} (f(X), U_2)$$

where X is a uniform distribution ensemble over $\mathbb{2}^n$, and $U_2 \sim \mathcal{U}nif(\mathbb{2})$. $\diamond$

Having made this definition, some observations are in order: we're going to rule out a cheesy solution

Claim 2. *There is no universal hardcore predicate $\mathfrak{HC}$ for all functions.* $\diamond$

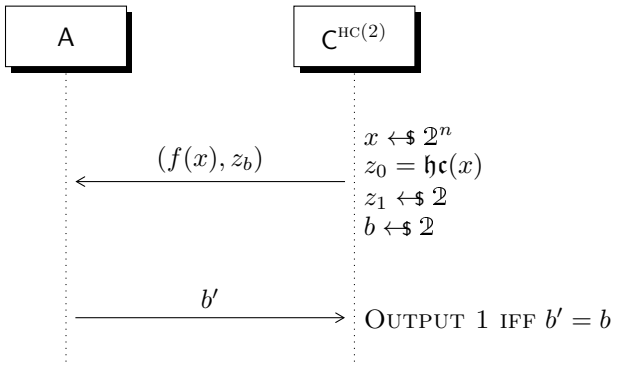


Figure 5.13: Another hardcore game, f and $\mathfrak{hc}$ are known

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's image itself. This contradicts the universality of $\mathfrak{HC}$. □

However, it is always possible to construct a hardcore predicate for a OWF, from another OWF:

Theorem 9 (Goldreich-Levin, '99)**.** *Let f be a* OWF *and consider $g(x, r) = (f(x), r)$ for $r \in \mathbb{2}^n$. Then g is a* OWF*, and:*

$$h(x, r) = \langle x, r \rangle = \bigoplus_{i=1}^{n} x_i \oplus r_i = \sum_{i=1}^{n} x_i \oplus r_i \bmod 2$$

is hardcore for g. ◇

Proof. TODO 8: TO BE COMPLETED (...did we actually do this? è una bella menata dimostrare questo)

□

Exercise 10. Prove that $f \in \mathrm{OWF} \implies g \in \mathrm{OWF}$ (Hint: do a reduction).

Solution 3 (10)**.** Let $\mathsf{D}^{\text{G-OWF}}$ be a machine that is efficient in inverting g, and consider the reduction shown in figure 5.14. By how g is defined, r' must be equal to r; therefore x' must be a valid pre-image of y in f. This contradicts the property of f being a OWF.

5.2.1 One-way permutations

A *one-way permutation*, or OWP in short, is defined exactly as the name itself suggests: a bijective OWF.

$$f \in \mathbb{2}^n \leftrightarrow \mathbb{2}^n \wedge f \in \mathrm{OWF} \implies f \in \mathrm{OWP}$$

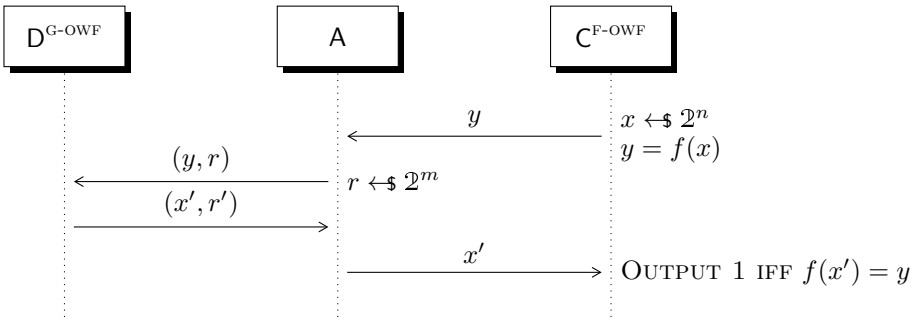


Figure 5.14: Efficiently inverting f

Corollary 1. *If $f \in \mathbb{2}^n \to \mathbb{2}^n$ is a* OWP *then, by the theorem of Goldreich-Levin defining $g(), h()$:*

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

is a PRG. ◇

Proof. The theorem states that if f is an OWF, then so is g. It's trivial to prove the analogue for OWPs. Moreover h is hardcore for g, tus:

$$\begin{aligned} G(U_{2n}) &\equiv (g(U_{2n}), h(U_{2n})) \\ &\equiv (f(U_n), U_n, h(U_{2n})) \\ &\approx_{\mathrm{C}} (f(U_n), U_n, U_1) \qquad \text{(definition 1 of hardcore predicate)} \\ &\equiv U_{2n+1} \end{aligned}$$

□

We've been successful in constructing a 1-bit stretch PRG; from here, by using the results in the previous section, we can construct a PRG that returns binary strings of an arbitrary length that are also pseudo-random.

Lesson 6

6.1 Computationally secure encryption

Having a better idea of what can and can't be accomplished in the cryptographic world, by means of theorems and proofs, we can focus now on the goal of defining a cryptographic system that meets our requirements. In this lesson, we focus specifically on the secrecy-oriented schemes, thus dealing with encryption and decryption of messages.

The requirements of a "good" encryption scheme are collectivey called those of *computationally secure encryption*: the characerizing requirement is to design a task, or routine, that is *computationally hard* for an attacker to revert. In detail: this task usually involves a secret key[6], and is accomplished in polynomial time, and any attacker who wishes to revert it has no efficient means of doing it wothout knowing such key. Other properties include:

1. *one-way*ness with respect to the encryption key: given $c = \mathsf{Enc}(k, m)$, it should be hard to recover k
2. *one-way*ness with respect to the original message: given $c = \mathsf{Enc}(k, m)$, it should be hard to recover m
3. In a stricter sense: no information whatsoever must "leak" from the message

To start visualizing these concepts, let $\Pi = (\mathsf{Enc}, \mathsf{Dec})$ be a secrecy scheme, and consider the game depicted in figure 6.15 where the adversary "wins" the game when the challenger outputs 1.

Definition 15. The scheme Π is said to be *computationally one-time secure* iff:

$$\forall \mathsf{A} \in \text{PPT} \implies \text{GAME}^{\text{IND}}_{\Pi,\mathsf{A}}(\lambda, 0) \approx_{\text{C}} \text{GAME}^{\text{IND}}_{\Pi,\mathsf{A}}(\lambda, 1)^{7}$$

or, rephrased in probability terms:

$$\forall \mathsf{A} \in \text{PPT} \implies |\Pr[\text{GAME}^{\text{IND}}_{\Pi,\mathsf{A}}(\lambda, 0) = 1] - \Pr[\text{GAME}^{\text{IND}}_{\Pi,\mathsf{A}}(\lambda, 1) = 1]| \in \mathcal{N}egl(\lambda)$$

$\diamond$

This last definition shows how such a scheme is compliant with the three properties exposed beforehand. In particular:

[6] This is the case for symmetric-key schemes, though many other kinds exist: some involving "public" keys, some others not having any key at all

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

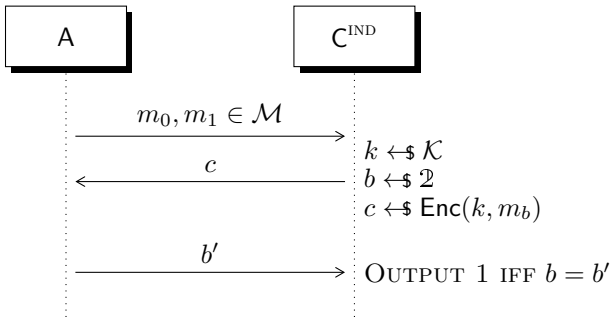


Figure 6.15: $\textsc{Game}^{\text{IND}}_{\Pi,\mathsf{A}}(\lambda, b)$

Todo 9: TO BE REVIEWED

1. *It is hard to recover the key.* If not, then an adversary Acan efficiently recover the key and use it to decrypt the ciphertext, which in turn enables him to perfectly distinguish m_0 from m_1 on any instance;

2. *It is hard to recover the message.* This is analogous, and even more obvious than the preceding point. Nevertheless, this is a necessary condition for a secrecy scheme to be "good", and it mustn't be forgotten;

3. *No information about the message whatsoever may leak from the ciphertext.* This may seem subtler than the previous point, but it warrants caution. Observe how an adversary A, if it has the ability to extract even a tiny bit of information of the original message from the ciphertext, then it is actually able to make an educated guess on which message was encrypted in the first place, putting him at an advantage. This leads the probabilities described in the definition to be sensibly more unbalanced than negligible, forfeiting the desired secrecy.

By extension, we may ask ourselves what scheme may or may not be *computationally two-time secure*. For instance, let $\Pi_\oplus = (\mathsf{Enc}, \mathsf{Dec})$ be a secrecy scheme using a PRG $G : \mathbb{2}^\lambda \to \mathbb{2}^n$, structured as follows:

- $\mathcal{K} = \mathbb{2}^\lambda$, $\mathcal{M} = \mathcal{C} = \mathbb{2}^n$
- $\mathsf{Enc}(k, m) = G(k) \oplus m$
- $\mathsf{Dec}(k, c) = c \oplus G(k) = m$

To be two-time secure means that, even if an adversary A gets hold of a valid plaintext-ciphertext couple $(\overline{m}, \overline{c})$, he is unable to decrypt any future ciphertexts[8], apart from the obvious $\overline{c}$. However, observe that A is now able to extract valuable information for decrypting future ciphertexts:

$$\overline{c} = \mathsf{Enc}(k, \overline{m}) = G(k) \oplus \overline{m} \implies \overline{c} \oplus \overline{m} = G(k)$$

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

so now, for any second ciphertext A receives, he can "mimick" the decryption routine, and efficiently uncover the underlying plaintext. This proves that $\Pi_\oplus$ is not two time-secure; nevertheless, it is still one-time secure:

Theorem 11. *If G is a* PRG*, then $\Pi_\oplus$ is computationally one-time secure* ◇

Proof. This proof is another example that showcases the use of hybrid games. Recalling the one-time security definition, we need to show that:

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

Consider the hybrid game in figure 6.16, where the original encryption routine is changed to use a completely random value, instead of using $G(k)$[9]. As an exercise, compare it with the original one-time secure definition in figure 6.15, to check that it perfectly matches.

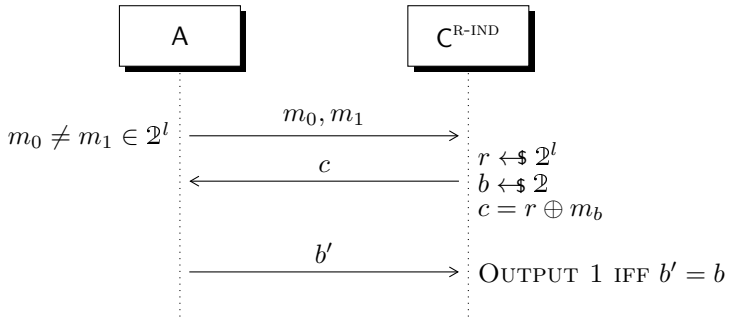


Figure 6.16: $\text{HYB}_{\Pi_\oplus,\mathsf{A}}(\lambda, b)$

The proof begins by affirming that:

Claim 3.

$$\forall \mathsf{A} \in \text{PPT} \implies \text{HYB}_{\Pi_\oplus,\mathsf{A}}(\lambda, 0) \equiv \text{HYB}_{\Pi_\oplus,\mathsf{A}}(\lambda, 1)$$

◇

To prove it, notice that r is chosen uniformly at random, and independently of b. Thus, no matter how the messages m_0 and m_1 are structured, r will effectively make the chosen message completely unrecognizable. In formal terms, let B and R be the random variables for C's picks of b and r respectively; then:

$$\begin{aligned}
&|\Pr(B = 0 \mid C = c) - \Pr(B = 1 \mid C \xmapsto{\$} c)| \\
=&|\Pr(B = 0 \mid R \oplus m_B \xmapsto{\$} c) - \Pr(B = 1 \mid R \oplus m_B \xmapsto{\$} c)| && \text{(C definition)} \\
=&\frac{|\Pr(R \oplus m_B = c \mid B \xmapsto{\$} 0)\Pr(B = 0) - \Pr(R \oplus m_B = c \mid B \xmapsto{\$} 1)\Pr(B = 1)|}{\Pr(R \oplus m_B = c)} \\
& && \text{(Bayes' theorem)} \\
=&\frac{|\Pr(R \oplus m_0 = c)\Pr(B = 0) - \Pr(R \oplus m_1 = c)\Pr(B = 1)|}{\Pr(R \oplus m_B = c)} && \text{(Cond. collapse)} \\
=&\frac{\left|\frac{1}{2^l}\frac{1}{2} - \frac{1}{2^l}\frac{1}{2}\right|}{\Pr(R \oplus m_B = c)} = 0
\end{aligned}$$

[9] The observant student may recognize that this modification yields exactly the "one-time pad" secrecy scheme discussed in lesson 1

Having proven that A's success is equivalent to straight guessing in $\text{HYB}_{\Pi_\oplus,\mathsf{A}}$, we now relate the hybrid game to the original one, affirming that:

Claim 4.

$$\forall \mathsf{A} \in \text{PPT}, \forall b \in \mathbb{2} \implies \text{HYB}_{\Pi_\oplus,\mathsf{A}}(\lambda, b) \approx_{\text{C}} \text{GAME}^{\text{IND}}_{\Pi_\oplus,\mathsf{A}}(\lambda, b)$$

$\diamond$

The proof proceeds by reduction as depicted in figure 6.17, by assuming the existence of a distinguisher D^{IND} for $c = G(k) \oplus m_b$ and $c = r \oplus m_b$, and using it to break G's pseudo-random generation property.

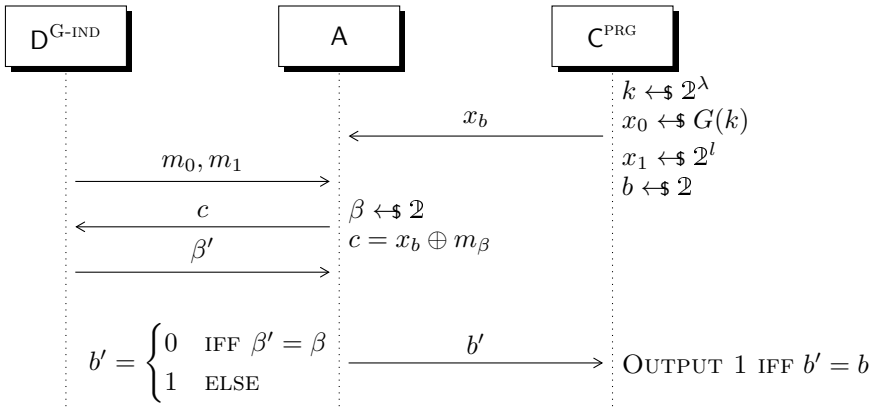


Figure 6.17: Reducing to breaking a PRG

Again, notice how the games of indistinguishability and pseudo-random generation are reliably reproduced on their respective sides. This construct's value centers on how D^{IND} will perform in its own challenge:

- if x_b is a random value, then D^{IND} will perform as depicted in the hybrid game, thus giving right or wrong answers at random;
- if b is the result of G, then D^{IND} has a better chance in finding the right answer by its own design;

From these observations, especially the second point, the adversary A has a better chance of winning the PRG game by asserting that x_b comes from G whenever D^{IND} makes a correct guess; conversely, A will preferrably declare that x_b is truly random whenever D^{IND} fails a guess, as the probability of the latter getting fooled by a random value is sensibly greater than by a pseudo-random one.

Either way, by the existence of D^{IND}, A gains an edge in efficiently recognizing G, which cannot happen by G's definition; the claim is proven. The theorem's proof can be completed by putting the pieces together, as is usual in the hybrid argument:

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

which finally states that $\text{GAME}^{\text{IND}}_{\Pi_\oplus,\mathsf{A}}(\lambda, 0) \approx_{\text{C}} \text{GAME}^{\text{IND}}_{\Pi_\oplus,\mathsf{A}}(\lambda, 1)$. $\square$

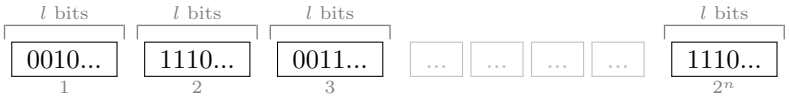
6.2 Pseudorandom functions

PRGs are used in practice as a stepping stone for building *pseudo-random functions*, PRF henceforth, which are the principal construct in several cryptographic schemes. Before formally introducing what a PRF is, we begin instead by defining what a *truly random function* is:

Definition 16. A random function $R : 2^n \to 2^l$ 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[10];
- if x is not fresh, then $R(x)$ is directly returned instead. ◇

It should be noted that, if such functions are to be implemented in computers, they would occupy too much space in memory. Suppose all the possible outputs of R have been generated and stored as an array in memory; then its total size in bits will be $l \cdot 2^n$:



Such a function becomes cumbersome and difficult to maintain in practice; therefore, it is desirable to find a kind of function which looks as a random function possible, but does not require to be wholly memorized, while not forgetting to maintain poly-time complexity. Pseudo-randomness comes to the rescue here:

Definition 17. Let f be a function, then it is deemed pseudo-random (therefore, f is a PRF) iff it is computationally indistinguishable from a true random function. ◇

In detail, PRFs are actually designed as function families f_k[11], where k is a parameter that indexes the funcitions inside the family. To model the PRFs' indistinguishability from random functions, let $F \in 2^\lambda \to (2^{n(\lambda)} \to 2^{l(\lambda)})$, usually denoted simply by f_k, be a PRF, and define $\mathfrak{R}(n, l)$ to be the domain that collects the random functions from $2^{n(\lambda)}$ to $2^{l(\lambda)}$.

Consider the indistinguishability game drawn in figure 6.18; although one may thing that PRGs and PRFs aren't much different, the game tells a different story, which is best put by an introductory paragraph about PRFs in their Wikipedia page:

> "Pseudorandom functions are not to be confused with pseudo-random generators (PRGs). The guarantee of a PRG is that a single output appears random if the input was chosen at random. On the other hand, the guarantee of a PRF is that all its outputs appear random, regardless of how the corresponding inputs were chosen, as long as the function was drawn at random from the PRF family."[12]

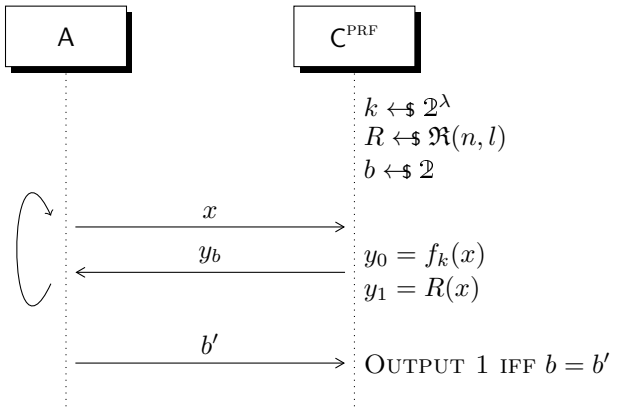


Figure 6.18: The PRF indistinguishability game

In this game, A is allowed to make multiple queries to C^{PRF}, as opposed to the PRG indistinguishability game where it can perform just one query before making its guess. To reiterate the PRF definition in terms of this game:

Definition 18. A function family $F = f_k$ is a PRF iff:

$$\text{Game}^{\text{PRF}}_{f_k,\mathsf{A}}(\lambda, 0) \approx_{\text{c}} \text{Game}^{\text{PRF}}_{f_k,\mathsf{A}}(\lambda, 1)$$

◇

Exercise 12. Prove the following statements:

- No PRG is secure against unbounded attackers;
- No PRF is secure against unbounded attackers.

6.2.1 GGM-tree

This section is dedicated to a concrete example of a PRF which is built from the ground up using a PRG. This construct has been designed from Oded Goldreich, Shafi Goldwasser and Silvio Micali, and its tstructure is akin to a binary tree, hence its name: GGM-*tree*.

Construction 1. Let $G \in 2^\lambda \to 2^{2\lambda}$ be a PRG such that it doubles the length of its argument, and denote the images' first and second halves as $G_0(k)$ and $G_1(k)$ respectively, so that:

$$k \mapsto (G_0(k), G_1(k))$$

Since the principal mechanism makes use of the halves being the same length of the argment, in the same spirit, we will denote the action of using one half of an image of G as argument of G itself in a shorter fashion, as demonstrated in the following example:

$$G_a(G_b(G_c(k))) =: G_{abc}(k)$$

[10] This property is also called *lazy sampling*.
[11] Does this remind you of something else? If not, look back in lesson 2 and 3.
[12] ➢ Pseudorandom function family — Wikipedia

This leads to the final step: let f_k be a function family, where $k \in 2^\lambda$, such that:

$$f_k(r) = G_r(k)$$

This is our candidate PRF. To visualize it, consider the tree structure depicted in figure 6.19: at each level of the tree, a single bit of r is used to decide which half of G's image will be used in the next level. For example, $f_k(01\ldots10)$ would evaluate as $G_0(G_1(\ldots G_1(G_0(k))))$.

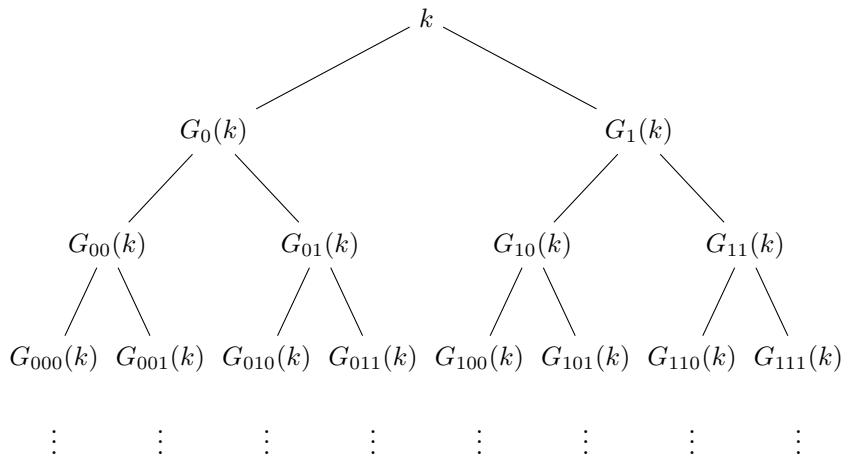


Figure 6.19: The GGM-tree for G

$\diamond$

The proof of PRF-ness will be discussed in the next lesson.

Lesson 7

GGM-tree (cont'd)

As stated in the previous lesson, given a PRG $G \in \mathbb{2}^\lambda \to \mathbb{2}^{2\lambda}$, we can build a function family f_k by repeatedly taking halves of G's images, and plugging them back into G. Our goal is to prove the following theorem:

Theorem 13. *If G is a* PRG*, then f_k is a* PRF*.* ◇

Proof. Before starting to prove the PRF-ness of f_k, we make a brief consideration about its time complexity: computing $f_k(x)$ consists in computing G and taking half of the resulting image as many times as is the length of x. Since the length of x is polynomial in λ, so is the number of G's iterations; combine this with the fact that G is itself polynomial by definition, and we conclude that f_k is polynomial too.

Having cleared any doubts about f_k's time complexity, we now turn to the essential point of interest: its pseudo-randomness. The proof will proceed by induction over the lentgth of x, which is also the height of the tree-like structure modeling the algorithm.

Base case ($n = 1$): f_k's domain is restricted to $\mathbb{2}$, meaning that its images will be respectively the two halves on a single iteration of $G(k)$; they are, of course, pseudorandom by G's definition:

$$(f_k(0), f_k(1)) = (G_0(k), G_1(k)) \approx_{\text{C}} U_{2\lambda}$$

therefore, in this case, F_k is pseudorandom.

Inductive step: Let $f'_k : \mathbb{2}^{n-1} \to \mathbb{2}^\lambda$ be a PRF. Define f_k as follows:

$$f_k : \mathbb{2}^n \to \mathbb{2}^\lambda : (b, x) \in \mathbb{2} \times \mathbb{2}^{n-1} \mapsto G_b(f'_k(x))$$

It must be proven that if f'_k is a PRF, then so is f_k. To help ourselves, we'll define some hybrid games as usual. We will proceed by hybrid games, depicted in figures 7.20 and 7.21:

Lemma 3. $\text{HYB}^1_{f_k,\mathsf{A}}(\lambda, 0) \approx_{\text{C}} \text{HYB}^1_{f_k,\mathsf{A}}(\lambda, 1)$ ◇

Proof. Assume $\exists \mathsf{D}^{\text{N-TH}} \in \text{PPT}$ that can distinguish f_k from $\overline{R} \circ G$ at the n-th step; then an adversary A can use it to break in turn f'_k's PRF-ness, and distingush it from $\overline{R} \circ G$ as shown in figure 7.22:

□

Before tackling $\text{HYB}^2_{\overline{R},\mathsf{A}}(\lambda, b)$, it is best to introduce another lemma:

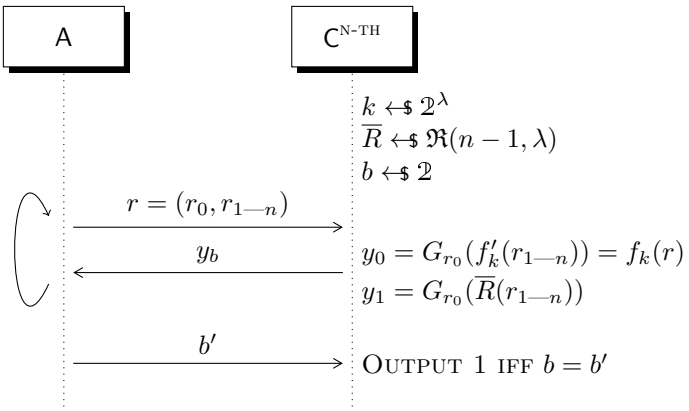


Figure 7.20: $\text{HYB}^1_{f_k,\mathsf{A}}(\lambda, b)$: The GGM construct is put against randomly driven PRG

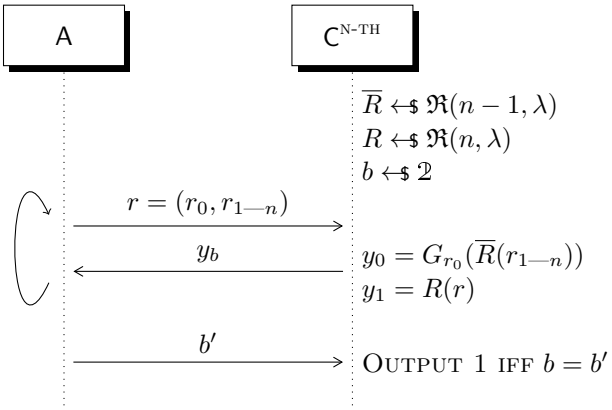


Figure 7.21: $\text{HYB}^2_{\overline{R}\circ G,\mathsf{A}}(\lambda, b)$: The randomly driven PRG against a true random function

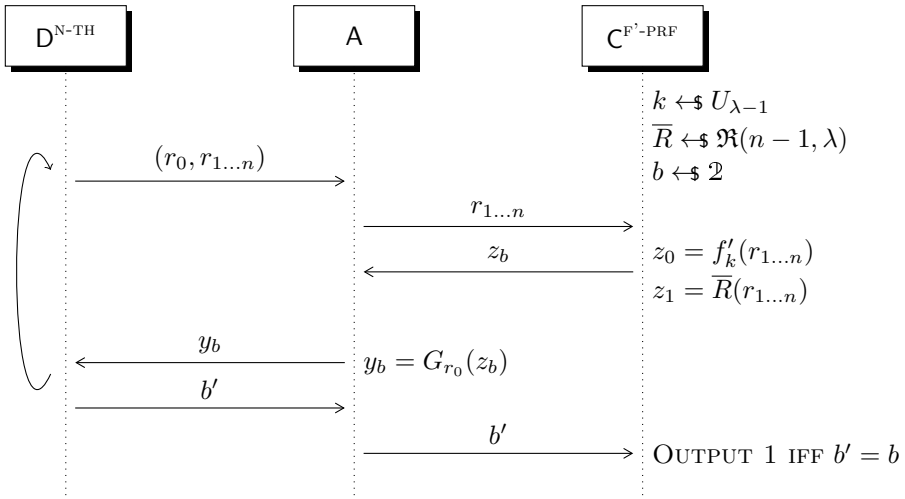


Figure 7.22: Using $\mathsf{D}^{\text{N-TH}}$ to break f'_k

Lemma 4. *If* $G : 2^\lambda \to 2^{2\lambda}$ *is a* PRG*, then:*

$$\forall K_i \sim \mathcal{U}nif(\lambda)\text{IID} \implies (G(K_1), \ldots, G(K_t)) \approx_c (U_{2\lambda}, \ldots, U_{2\lambda})$$

◇

Proof. TODO 10: Idea: all values are independent and pseudorandom on their own, hybridize progressively...

□

Now for the final lemma:

Lemma 5. $\text{HYB}^2_{\overline{R}\circ G,\mathsf{A}}(\lambda, 0) \approx_c \text{HYB}^2_{\overline{R}\circ G,\mathsf{A}}(\lambda, 1)$ ◇

Proof. This proof is trickier: the problem lies in the inherent difference between the task of detecting a PRG from that of detecting a PRF. The aforemenioned lemma helps us with the aspect of polynomial queries on a PRG, bringing it in line with the PRF game; yet, we're far from done.

Let's delve into the details: from A's perspective, the game of distinguishing $\overline{R} \circ G$ from R , where D performs a number of queries q polynomial in λ, is perfectly modeled by the act of distinguishing a sequence of evaluations $(G(K_1), \ldots, G(K_q))$ from $(U_{2\lambda}, \ldots, U_{2\lambda})$. Therefore, the game can be twisted in a way that A receives either one of the two whole sequences beforehand, and respects the rules by sending to D the right piece of information on each of its queries, in order to simulate the $\overline{R} \circ G/R$ game correctly.

At this point, the simple way for A to send the right info on each query i would be to check whether the query's first bit r_0 would be 0 or 1, and give back to D either the first or the second half of the i-th value of the sequence he received from C; this is exactly how GGM operates. However, there's a catch: D might make two queries where the r1—n parts are the same, the only difference is in the first bit: essentially, D is asking for both parts of G(r) from two distinct queries. In this case, if A just sends the i-th value's corresponding half, the simulation would break, because the halves come from different samples.

Thus, the adversary must keep track of which suffixes he has been already queried about, and make sure to send the value's other half (there are only two) whenever he is queried on a suffix for a second time. This final touch solves the problem of simulating the game for D, and now the task of A is changed to an equivalent one of breaking the lemma stated at the beginning.

TODO 11: Prata's notes:
Let T1, T2 be empty tables.
Given query x input x1—n let xbar=x1—i
if xbar notin T1 then x ¡-$ binarylambda and add k_xbar to T2
if xbar in T2 let k_xbar = T2[T1[xbar]]
output y = G_x_n(g_x_n-1(...G_x_i+1(k_xbar)))
H_0: GGM treee
H_n: random function
exploit lemma prg =¿ prf

□

In the end the hybrids are proven to mutually indistinguishable, therefore the inductive step is correct, proving the theorem.

□

7.1 CPA-security

Now it's time to define a stronger notion of security, which is widely used in cryptology for first assessments of cryptographic strength. Let $\Pi := (\mathsf{Enc}, \mathsf{Dec})$ be a SKE scheme, and consider the game depicted in figure 7.23. 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.

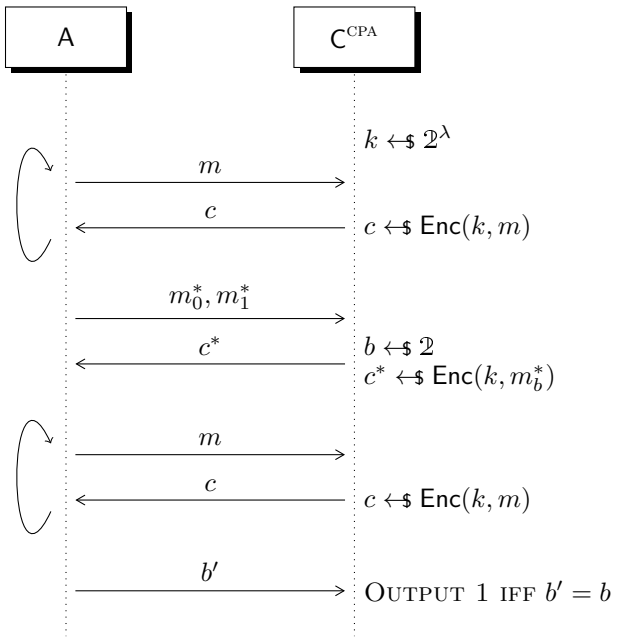


Figure 7.23: The CPA-security game: $\mathrm{GAME}^{\mathrm{CPA}}_{\Pi,\mathsf{A}}(\lambda, b)$

Definition 19. A scheme is CPA-secure if $\mathrm{GAME}^{\mathrm{CPA}}_{\Pi,\mathsf{A}}(\lambda, 0) \approx_c \mathrm{GAME}^{\mathrm{CPA}}_{\Pi,\mathsf{A}}(\lambda, 1)$ ◇

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 Π_{f_k}, where f_k is a PRF structured as follows:

- $\mathsf{Enc}(k, m) = (c_1, c_2) = (r, f_k(r) \oplus m)$, where $k \xleftarrow{\$} \mathbb{2}^{\lambda}$ and $r \xleftarrow{\$} \mathbb{2}^{n}$
- $\mathsf{Dec}(k, (c_1, c_2)) = f_k(c_1) \oplus c_2$

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 14. *If f_k is a* PRF*, then Π_{f_k} is* CPA*-secure.* ◇

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

- $r \xleftarrow{\$} \mathbb{2}^{n}$
- $R \xleftarrow{\$} \mathfrak{R}(n, l)$
- $c = (r, R(r) \oplus m)$, where m is the plaintext to be encrypted

and then the last hybrid $\textsc{Hyb}^{2}_{\Pi,\mathsf{A}}$ will simply output $(r_1, r_2) \xleftarrow{\$} U_{n+l}$.

Lemma 6. $\forall b \in \mathbb{2} \implies \textsc{Hyb}^{0}_{\Pi,\mathsf{A}}(\lambda, b) \approx_{\text{C}} \textsc{Hyb}^{1}_{\Pi,\mathsf{A}}(\lambda, b)$. ◇

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 f_k's property of being a PRF. The way to use D is to make it play a CPA-like game, as shown in figure 7.24[13], where the adversary attempting to break f_k decides which message to encrypt between m_0 and m_1 beforehand, and checks whether it 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 f_k. Thus, assuming such D exists, A can efficiently break f_k, contradictiong its PRF-ness.

□

Lemma 7. $\forall b \in \mathbb{2} \implies \textsc{Hyb}^{1}_{\Pi,\mathsf{A}}(\lambda, b) \approx_{\text{C}} \textsc{Hyb}^{2}_{\Pi,\mathsf{A}}(\lambda, b)$. ◇

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.

[13] An observant student may notice a striking similarity with a previously exposed reduction in figure 6.17

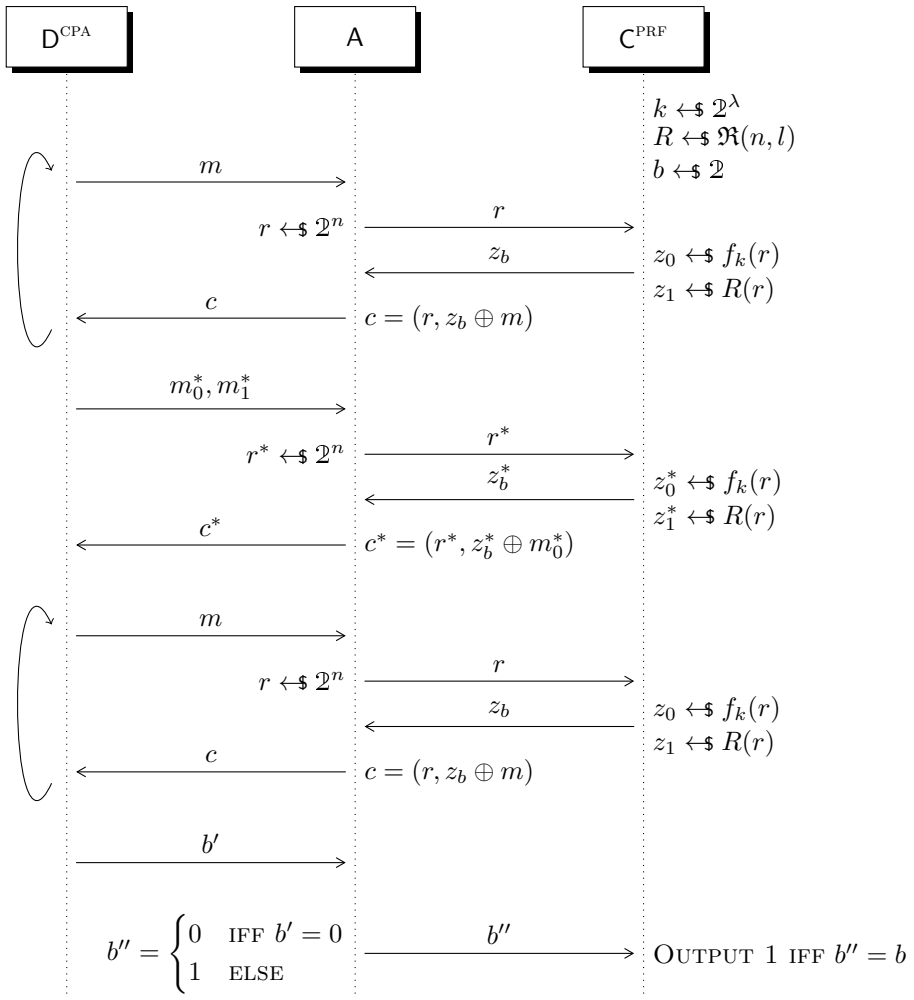


Figure 7.24: Breaking a PRF, for fixed message choice of m_0

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 $\overline{r}$ is queried twice in both runs, then on the second query the adversary will receive the same image in $\text{HYB}^1_{\Pi,\mathsf{A}}$, but almost certainly a different one in $\text{HYB}^2_{\Pi,\mathsf{A}}$. 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 REPEAT this collision event on $\overline{r}$ between 2 consecutive games. Then:

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

which proves that the REPEAT influences negligibly on the two hybrids' equivalence. Thus $\text{HYB}^1_{\Pi,\mathsf{A}}(\lambda, b) \approx_{\text{C}} \text{HYB}^2_{\Pi,\mathsf{A}}(\lambda, b)$[14].

□

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

□

[14] Do note that the hybrids lose their originally supposed perfect equivalence ($\text{HYB}^1_{\Pi,\mathsf{A}}(\lambda, b) \equiv \text{HYB}^2_{\Pi,\mathsf{A}}(\lambda, b)$) because of the REPEAT event. Nevertheless, the lemma is still proven because it includes computational bounds into A

Lesson 8

8.1 Domain extension

Up until now, encryption has been dealt with messages of fixed size around a polynomial function to λ. How to deal with messages with aritrary size? Setting a maximum bound to message length seems impractical, both for waste reasons when messages are too short, and for practicality when messages eventually get too long. The solution takes the form of a "block-cipher", where a message of a given size is split into equally-sized blocks, and then encrypted using a fixed-size encryption scheme. Various instances of this technique, called *modes*, have been devised.

8.1.1 Electronic Codebook mode

The operation of ECB-mode is straightforward: Given a message split into blocks $(m_1, \ldots, m_t)$, apply the scheme's encryption routine to each block, as shown in figure 8.25:

$$c_i = F_k(r) \oplus m_i \quad \forall i \in \{0, \ldots, t\}$$

Decryption is trivially implemented by XOR-ing the ciphered blocks with $F_k(r)$.

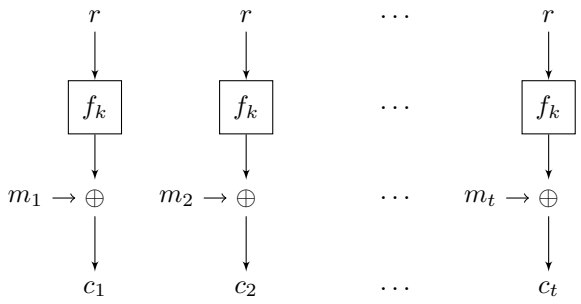


Figure 8.25: ECB-mode block-cipher in action, using a PRF as the encryption routine

This approach has the advantage of being completely parallelizable, as each block can clearly be encrypted separately; however there is a dangerous flaw in being not CPA-secure, even when using a PRF-based encryption scheme. To understand why, observe that random nonces for ciphertext randomization are chosen per-message; this means the encryption of message blocks become deterministic in the message scope, enabling an adversary to attack the scheme

within a single plaintext. It is sufficient to choose an all-0 or all-1 message to realize that all its blocks would encrypt to the same ciphered block.

8.1.2 Cipher block chaining mode (CBC)

This mode serializes block encryption by using the preceding ciphered block in the formula:

$$c_i = P_k(r) \oplus m_i \quad \forall i \in \{0, \ldots, t\}$$

This time, a *pseudorandom permutation*(PRP) is used instead of a PRF; they will be discussed later on. The diagram in figure 8.26 shows a general view of CBC-mode's operation. The decryption process is analogous but in a reversed fashion, by computing the preimage of a ciphered block and XOR-ing it with the preceding ciphered block:

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

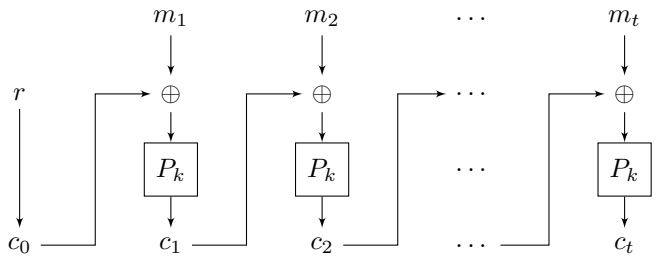


Figure 8.26: CBC-mode block-cipher in action, using a PRP as the encryption routine

8.1.3 Counter mode

Also denoted as CTR in short, this mode closely resembles ECB-mode but uses a "rolling" nonce instead of a static one, as shown in figure 8.27. At each successive block, the nonce is incremented by 1 and then used in a single block encryption. Since the nonce is in $\mathbb{2}^n$, the increment is done modulo 2^n so that the value will wrap around to 0 it it ever overflows. Decryption is analogous.

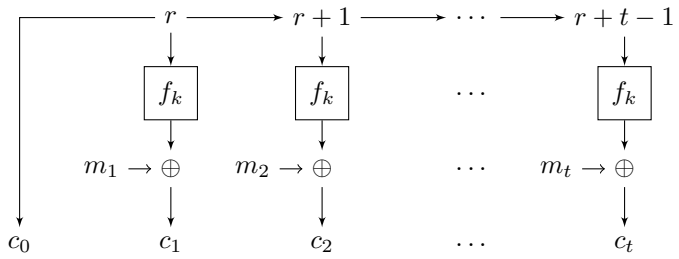


Figure 8.27: Counter-mode block-cipher in action, using a PRF as the encryption routine

This apparently innocuous change to EBC is enough to ensure CPA-security, at the cost of perfect parallelization.

Theorem 15. *Assume f_k is a* PRF*, then the counter-mode block cipher is* CPA*-secure for variable length messages*[15]*.* ◇

Proof. Figure 8.28 models a CPA attack to a counter-mode block-cipher. The proof will proceed by hybrid argument starting from this game, therefore the statement to verify will be $\text{GAME}^{\text{CPA}}_{\text{CTR},\mathsf{A}}(\lambda, 0) \approx_{\text{C}} \text{GAME}^{\text{CPA}}_{\text{CTR},\mathsf{A}}(\lambda, 1)$.

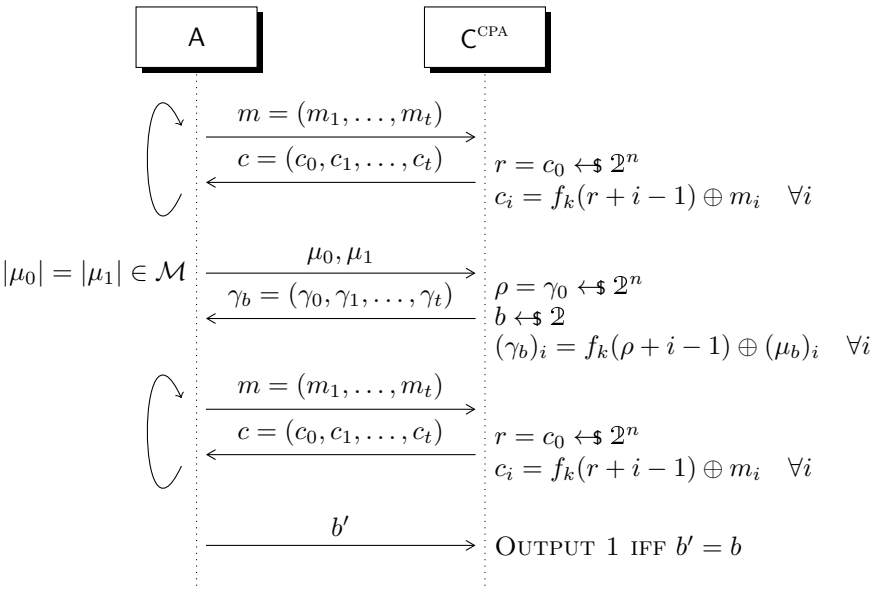


Figure 8.28: A chosen plaintext attack to counter-mode block-cipher

Define the two hybrid games from the original CPA game as follows:

- $\text{HYB}^1_{\text{CTR},\mathsf{A}}(\lambda, b)$: A random function R is chosen UAR from $\mathfrak{R}(n, n)$ at the beginning of the game, and is used in place of F_k in all block encryptions;
- $\text{HYB}^2_{\text{CTR},\mathsf{A}}(\lambda, b)$: The challenger will pick random values from $\mathbb{2}^n$ as ciphered blocks, disregarding any encryption routine.

Lemma 8. $\text{GAME}^{\text{CPA}}_{\text{CTR},\mathsf{A}}(\lambda, b) \approx_{\text{C}} \text{HYB}^1_{\text{CTR},\mathsf{A}}(\lambda, b) \quad \forall b \in \mathbb{2}$ ◇

Proof. The proof is left as exercise.

Hint: Since the original game and the first hybrid are very similar, we can use a distinguisher which plays the CPA-game; since this is a lemma, our goal in the reduction is to break the precondition contained in the theorem statement. □

Lemma 9. $\text{HYB}^1_{\text{CTR},\mathsf{A}}(\lambda, b) \approx_{\text{C}} \text{HYB}^2_{\text{CTR},\mathsf{A}}(\lambda, b) \quad \forall b \in \mathbb{2}$ ◇

[15] *Variable length messages* exactly means every message $m = (m_1, \ldots, m_t)$ is made of t blocks, and t can change from any message to a different one.

Proof. Since m_i doesn't affect the distribution of the result at all, for any i, if $R(r^*)$ behaves like a true random extractor, then the two hybrids are indistinguishable in the general case ($R(r+i) \oplus m_i \approx R(r+i)$). However, there is a sneaky issue: if in both games it happens that a given nonce r_i is used in both one query encryption and the challenge mesage encryption at any step, the subsequent encrypted blocks will be completely random in the second hybrid, whereas in the first hybrid the function's images, albeit random, become predictable, enabling a CPA.

Nevertheless, it can be proved that these "collisions" happen with negligible probability within $\text{HYB}^1_{\text{CTR},\mathsf{A}}$. Let:

- q = number of encryption queries in a game run
- t_i = number of blocks for the i-th query
- τ = number of blocks for the challenge ciphertext
- OVERLAP event: $\exists i, j, \iota : r_i + j = \rho + \iota$

The OVERLAP event exaclty models our problematic scenario. Now it suffices to show that it occurs negligibly. For simplicity, assume the involved messages are of same length, that is $t_i = \tau =: t$. Denote with OVERLAP_i to be the event that the i-th query overlaps the challenge sequence as specified above.

Fix some ρ. One can see that OVERLAP_i happens if:

$$\rho - t + 1 \leq r_i \leq \rho + t - 1$$

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

- the sequence $\rho, \ldots, \rho + t - 1$ comes before the sequence $r_i, \ldots, r_i + t - 1$, and they overlap just for the last element $\rho + t - 1 = r_i$ or
- the sequence $r_i, \ldots, r_i + t - 1$ comes before the sequence the sequence $\rho, \ldots, \rho + t - 1$, and they overlap just for the last element $r_i + t - 1 = \rho$.

Then:

$$\begin{aligned}
\Pr[\text{OVERLAP}_i] &= \frac{(\rho + t - 1) - (\rho - t + 1) + 1}{2^n} \\
&= \frac{2t - 1}{2^n} \\
\Pr[\text{OVERLAP}] &\leq \sum_{i=1}^{t} \Pr[\text{OVERLAP}_i] \\
&\leq 2\frac{t^2}{2^n} \in \mathcal{N}egl\lambda
\end{aligned}$$

which proves that our collision scenario happens with negligible pobability, thus the two hybrids are indistinguishable.

□

Having proven the indistinguishability between the hybrids, the conclusion is reached:

$$\text{GAME}^{\text{CPA}}_{\text{CTR},\mathsf{A}}(\lambda, 0) \approx_{\text{C}} \text{GAME}^{\text{CPA}}_{\text{CTR},\mathsf{A}}(\lambda, 1)$$

□

Lesson 9

9.1 Message Authentication Codes and unforgeability

After having explored the security conerns and challenges of the SKE realm, it is time to turn the attention to symmetric authentication schemes, or MAC schemes. Recall that a MAC scheme is a couple $(Tag, Verify)$, with the purpose of auhenticating the message's source. In this chapter, the tagging function will be denoted as Tag_k, akin to a PRF.

The desirable property that a MAC scheme should hold is to prevent any attacker from generating a valid couple (m^*, ϕ^*), even after querying a tagging oracle polynomially many times[16]. The act of generating a valid couple from scratch is called *forging*, and the aforementioned propery is defined as *unforgeability against chosen-message attacks* (or UFCMA, in short); its game diagram is shown in figure 9.29. Do note that m^* is stated to be outside the query set M, expressing the "freshness" of the forged couple[17]. In formal terms:

Definition 20. A MAC scheme Π is UFCMA-secure iff:

$$\forall \mathsf{A} \in \text{PPT} \implies \Pr[\text{GAME}^{\text{UFCMA}}_{\Pi,\mathsf{A}}(\lambda) = 1] \in \mathcal{N}egl(\lambda) \qquad \diamond$$

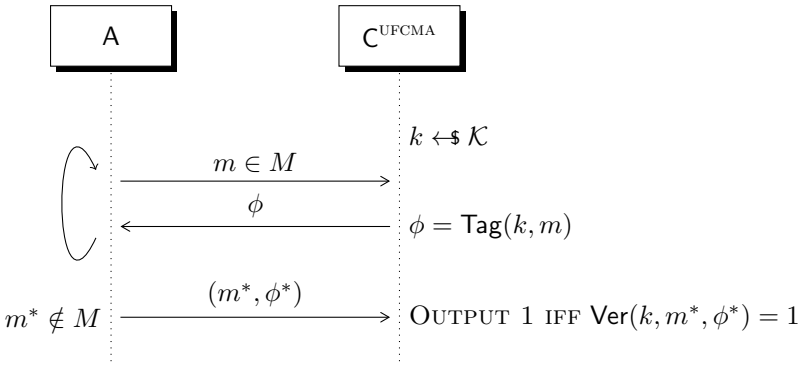


Figure 9.29: $\text{GAME}^{\text{UFCMA}}_{\Pi,\mathsf{A}}(\lambda)$

[16] Do note the similarities (and differences) between unforgeability in this setting and CPA-security in the secrecy setting

[17] Observe how this setup bears a striking resemblance to the PRF game. We're onto something...

Having defined a good notion of security in the MAC scheme domain, we turn our attention to a somewhat trivial scheme, and find out that it is indeed UFCMA-secure:

Theorem 16. *Let f_k be a* PRF*, and define the following* MAC *scheme:*

$$\Pi = (\mathsf{Tag}, \mathsf{Ver}) := (f_k(m), [f_k(m) = \phi])$$

Then Π *is* UFCMA*-secure.* ◇

Proof. The usual proof by hybridization to random functions entails. The original game is identical to the UFCMA game, where the tagging function is the PRF, whereas the hybrid game will have it replaced with a truly random function, as shown in figure 9.30.

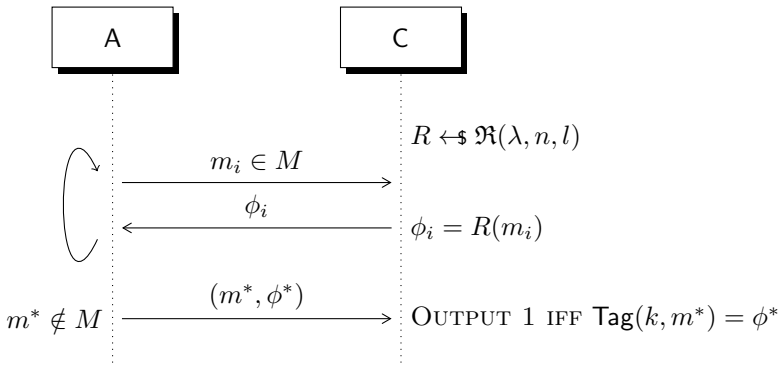


Figure 9.30: $\textsc{Hyb}^1_{\Pi,\mathsf{A}}(\lambda)$

Lemma 10. $\textsc{Game}^{\text{UFCMA}}_{\Pi,\mathsf{A}}(\lambda) \approx_{\text{C}} \textsc{Hyb}^1_{\Pi,\mathsf{A}}(\lambda)$ ◇

Proof. By assuming there is a distinguisher $\mathsf{D}^{\text{UFCMA}}$ capable of disproving the lemma, it can be used to distinguish the PRF itself, as depicted by the reduction in figure 9.31 □

Lemma 11. $\forall \mathsf{A} \in \text{PPT} \implies \Pr[\textsc{Hyb}^1_{\Pi,\mathsf{A}}(\lambda) = 1] \leq 2^{-l}$ ◇

Proof. This is true because attacker has to predict the output $R(m^*)$ on a fresh input m^* to win the game, which can happen at most with probability 2^{-l}. □

Thus, the conclusion is that Π is UFCMA-secure. □

9.2 Domain extension for MAC schemes

The previous scheme works on fixed length messages; as in the encryption domain, there are techinques for tagging variable length messages which are UFCMA-secure. However, before showing them, some other apparently secure modes are described here to give some possible insights on how to tackle the problem.

Assume the message $m = (m_1, \ldots, m_t) \in \mathcal{2}^{n \cdot t}$ for some $t \geq 1$. Given the tagging function $\mathsf{Tag}_k \in \mathcal{2}^n \to \mathcal{2}^l$, an attempt to tag the whole message may be to:

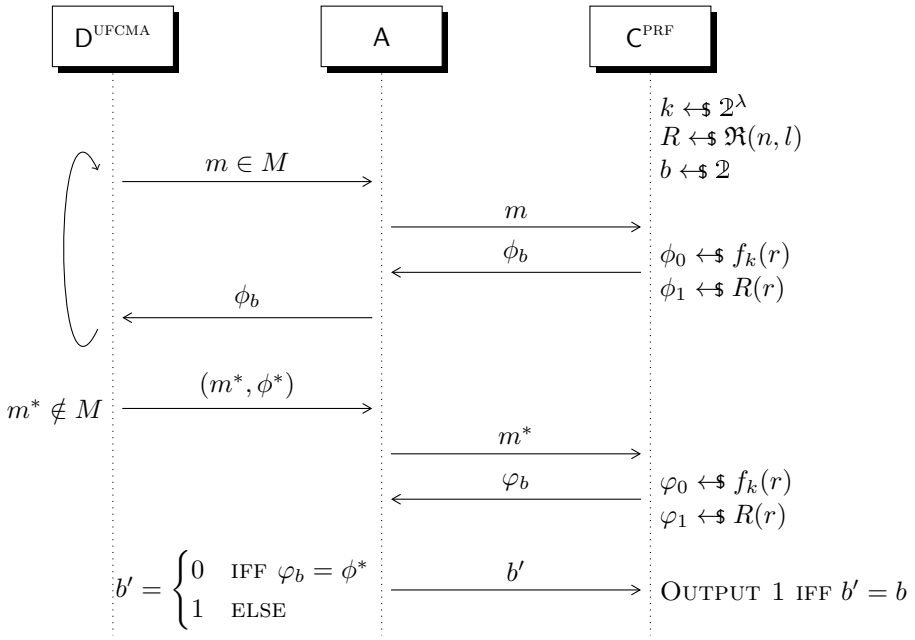


Figure 9.31: Distinguishing a PRF by using $\mathsf{D}^{\text{UFCMA}}$

- XOR all the message blocks, and then tag: $\phi = \mathsf{Tag}(k, \bigoplus_{i=1}^{t} m_i)$. This approach opens to some easy forgeries: given an authenticated message (m, ϕ), an adversary can create a valid couple (m', ϕ), where m' is the original message with two flipped bits in two distinct blocks at the same offset; the resulting XOR would be the same.
- define the tag to be a t-sequence of tags, one for each message block. Again, there is an easy way to forge authenticated messages: an adversary can just flip the position of two arbitrary distinct message blocks along with their relative tags, resulting in a different, yet authentic message.
- attempt a variant of the above approach, by adding the block number to the block itself to avoid the previous forging. Yet again, this is not UFCMA-secure: the adversary may just make two queries on two distinct messages, obtain the two tag sequences, and then forge an authenticated message by choosing at each position i whether to pick the message-tag blocks from the first or second query. Possibilities are endless...

9.2.1 Universal hash functions

A devised solution which has been proven to be secure relies on the following definition: a function family H which can be used to "shrink" variable length messages and then composed with a PRF f_k:

$$H \in 2^\lambda \to (2^{n \cdot t} \to 2^n) : s \mapsto h_s$$

$$\mathsf{Tag}((k, s), m) = f_k(h_s(m))$$

So what are the properties of the such a function family $H \circ F$? The main problem are *collisions*, since for each $m \in 2^{n \cdot t}$ it should be hard to find $m' \neq m$ such that $h_s(m) = h_s(m')$. But collisions do exist for functions in $H \circ F$, because they map elements from $2^{n \cdot t}$ to 2^t, and since the codomain is smaller than the domain, the functions cannot be injective in any way.

To overcome this problem, we can consider two options:

- assume collisions are hard to find, even when $s \in 2^\lambda$ is known; in this case we have a family of *collision-resistant hash functions*;
- let s be secret, and assume collisions are hard to find because it is hard to know how h_s works.

Definition 21. A family of hash functions h_s is deemed *ε-universal* iff:

$$\forall x \neq x' \in 2^{n \cdot t}, S \sim \mathcal{U}nif(2^\lambda) \implies \Pr[h_S(x) = h_S(x')] \leq \varepsilon$$

◇

If $\varepsilon = 2^{-n}$, meaning the collision probability is minimized, then the family is also called *perfectly universal*; in the case where $\varepsilon \in \mathcal{N}egl(\lambda)$ isntead, it is defined as *almost universal* (AU). Care should be taken in telling the difference between universality and pairwise independence, which states:

$$(h_S(x), h_S(x')) \equiv U_{2n}$$

Lemma 12. *Show that any pairwise independent hash function is perfectly universal.* ◇

Proof. The proof is left as exercise.

TODO 12: (should I use *Col* for solving this? What is the difference and when should I 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 17. *Assuming f_k is a* PRF *with n-bit domain and h_s is an* AU *hash function family, then $H \circ F$ is a* PRF *on $(n \cdot t)$-bit domain, for $t \geq 1$.* ◇

Proof. This proof too will proceed by hybridizing the original game up to the ideal random one. Consider the three sequences depicted in figures 9.32, 9.33 and 9.34:

Lemma 13.

$$Real_{F,\mathsf{A}}(\lambda) \approx_{\mathrm{C}} \mathrm{HYB}_{R,\mathsf{A}}(\lambda)$$

◇

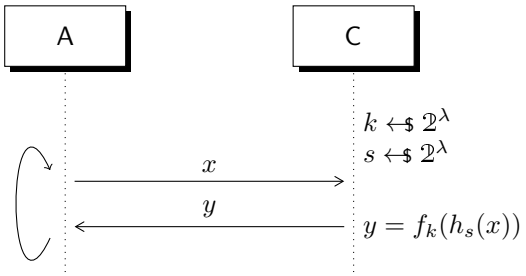


Figure 9.32: $Real_{F,\mathsf{A}}(\lambda)$

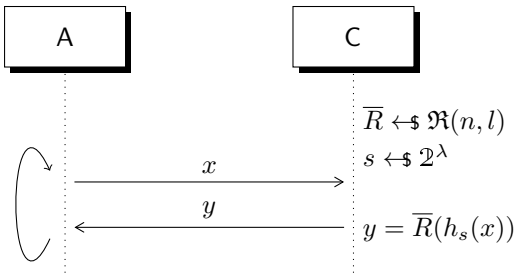


Figure 9.33: $\textsc{Hyb}_{R,\mathsf{A}}(\lambda)$

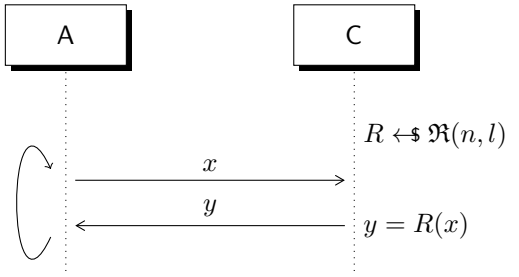


Figure 9.34: $Rand_{R',\mathsf{A}}(\lambda)$

Proof. The proof is left as exercise. □

Lemma 14.

$$\textsc{Hyb}_{R,\mathsf{A}}(\lambda) \approx_{\text{C}} \mathit{Rand}_{R',\mathsf{A}}(\lambda)$$

◇

Proof. Again, collisions come into play there, but there are multiple cases that complicate things. Given two queries with arguments x_1, x_2 returning the same image y, the random game can model two scenarios:

- the arguments are equal, but with negligible probability
- the arguments are distinct

while the hybrid can model three of them:

- the arguments are equal, again with negligible probability
- the arguments are distinct, *and so are their hashes*
- the arguments are distinct, *but not their hashes*

We want to show that the collision at hash level is negligible: as long as they don't happen, the random function $\overline{R}$ is run over a sequence of distinct points, and behaves just as the random game's function R does. So let BAD be the event:

$$\exists i \neq j \in [q] : h_s(x_i) = h_s(x_j)$$

where q denotes the adversary's query count. It suffices to show that $\Pr[\textsc{Bad}] \in \mathcal{N}\mathit{egl}(\lambda)$.

Since we don't care what happens after a collision, we can alternatively consider a mental experiment where we answer all queries at random, and only at the end sample $s \xleftarrow{\$} 2^\lambda$ and check ofr collisions: this does not change the value of $\Pr(\textsc{Bad})$. Now queries are independent of s, and this eases our proof:

$$\begin{aligned}
\Pr[\textsc{Bad}] &= \Pr[\exists x_i \neq x_j, h_S(x_i) = h_S(x_j)] \\
&\leq \sum_{i \neq j} \Pr[h_S(x_i) = h_S(x_j)] && h_s \text{ is AU}) \\
&\leq \binom{q}{2} \mathcal{N}\mathit{egl}(\lambda) \in \mathcal{N}\mathit{egl}(\lambda)
\end{aligned}$$

By ruling out this event, the lemma is proven. □

So now we have $\mathit{Real} \approx_{\text{C}} \textsc{Hyb}_{\Pi,\mathsf{A}} \approx_{\text{C}} \mathit{Rand}$ □

Corollary 2. *Let* $\Pi = (\mathsf{Tag}, \mathsf{Ver})$ *be a variable length message* MAC *scheme where, given a* PRF f_k *and an* AU *hash funcion family* h_s*, the tagging function is defined as* $f_k(h_s)$*. Then* Π *is* UFCMA*-secure.* ◇

9.2.2 Hash function families from finite fields

A generic 2^n-order finite field has very useful properties: adding two of its elements is equal to XOR-ing their binary representations, while multiplying them is done modulo 2^n. It is possible to define a hash function family that makes good use of these properties, and is suitable for a UFCMA-secure MAC scheme.

Construction 2. Let $\mathbb{F} = GF(2^n)$ be a *finite field* (or "Galois field") of 2^n elements, and let $m = (m_1, \ldots, m_t) \in \mathbb{F}^t$ and $s = (s_1, \ldots, s_t) \in \mathbb{F}^t$. The desired hash function family will have this form:

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

◇

Lemma 15. *The above function family h_s is almost universal.* ◇

Proof. In order for h_s to be almost universal, collisions must happen negligibly. Suppose we have a collision with two distinct messages m and m':

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

Let $\delta_i = m_i - m'_i$ and assume, without loss of generality, that $\delta \neq 0$. Then, by using the previous equation, when a collision happens:

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

Since the messages are different from each other, there is at least some i-th block that contains some of the differences. Assume, without loss of generality, that some of the differences are contained in the first block ($i = 1$); the sum can then be split between the first block itself $\delta_1 s_1$ and the rest:

$$\begin{aligned}
\sum_{i=1}^{t} \delta_i s_i &= 0 \\
\delta_1 s_1 + \sum_{i=2}^{t} \delta_i s_i &= 0 \\
\delta_1 s_1 &= -\sum_{i=2}^{t} \delta_i s_i \\
s_1 &= \frac{-\sum_{i=2}^{t} \delta_i s_i}{\delta_1}
\end{aligned}$$

which means when a collision happens, s_1 must be exactly equal to the sum of the other blocks, which is another element of $\mathbb{F}$. But since every seed is

chosen at random among $\mathbb{F}$, the probability of picking the element s_1 satisfying the above equation is just $|\mathbb{F}|^{-1} = 2^{-n} \in \mathcal{N}egl(\lambda)$. By repeating this reasoning for every difference-block, a sum of negligible probabilities is obtained, which is in turn negligible; therefore the hash function family h_s is almost universal.

□

H with Galois fields elements and polynomials

Exercise 18. Let $\mathbb{F} = GF(2^n)$ be a finite field of 2^n elements, and let $m = (m_1, \ldots, m_t) \in \mathbb{F}^t$ and $s \xleftarrow{\$} \mathbb{F}^t$. Construct the following hash function family:

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

Prove that this construction is AU.

Solution 4. (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

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

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

$$q_{m-m'}(s) = 0$$

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 polynomial's rank. The rank is $t - 1$, and the probability of picking a root from $\mathbb{F}$ as seed of h_s is:

$$\Pr[s = root] = \frac{t-1}{2^n} \in \mathcal{N}egl(\lambda)$$

Lesson 10

10.1 Domain extension for PRF-based authentication schemes

10.1.1 Hash function families from PRFs

Another way to obtain domain extension for a MAC scheme, using the $H \circ F$ approach, is to construct the hash function family from another PRF. We expect to have:

- $\forall \mathsf{A} \in \text{PPT}, S \sim \mathcal{U}nif(2^\lambda) \implies \Pr[h_S(m) = h_S(m')] \in \mathcal{N}egl(\lambda)$;
- We need two PRFs: one is f_k, and the other is f_s

10.1.2 XOR-mode

Let f_s be a PRF, and 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 f_s is $n + log_2(t)$ bytes long.

Lemma 16. *Above H is computational* AU. ◇

Proof. The proof is left as exercise.

(Hint: The pseudorandom functions can be defined as $f_s = f'_k(0, \dots)$ and $f_k = f'_k(1 \dots)$).

Possible proof: we have to show that

$$\forall m \neq m' \implies \Pr[h_s(m) = h_s(m')] \in \mathcal{N}egl(\lambda)$$

This means that:

$$\Pr\left[\left(\bigoplus_{i=1}^{t} f_s(m_i, i)\right) = \left(\bigoplus_{i=1}^{t} f_s(m'_i, i)\right)\right]$$

$$= \Pr[\forall i \quad f_s(m_i, i) \oplus f_s(m'_i, i) = \bigoplus_{j=1, j\neq i}^{t} f_s(m_j, j) \oplus f_s(m'_j, j) = \alpha]$$

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.

□

10.1.3 CBC-mode MAC scheme

This mode is used in practice by the TLS standard. It's used with a PRF f_s, setting the starting vector as $\text{IV} = 0^n = c_0$ and running this PRF as part of CBC. The last block otained by the whole process is the message's signature:

$$h_s(m) = f_s(m_t \oplus f_s(m_{t-1} \oplus f_s(\dots f_s(m_2 \oplus f_s(m_1 \oplus \text{IV}))\dots)))$$

Lemma 17. CBC MAC *defines completely an* AU *family.* ◇

Proof. (not proven) □

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

$$\text{E-CBC}_{K,S}(m) = f_k(h_s^{\text{CBC}}(m))$$

Theorem 19. *Actually if f_k is a* PRF*,* CBC MAC *is already a* MAC *scheme with domain $n \cdot t$ for arbitrarily fixed $t \in \mathbb{N}$.* ◇

Proof. (not proven) □

10.1.4 XOR MAC

Instead of $H \circ F$, now the Tag routine outputs $\phi = (\eta, f_k(\eta) \oplus h_s(m))$ where $\eta \xleftarrow{\$} 2^n$ is random and it's called *nonce*. Authentication is done as:

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

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.

TODO 13: AXU property definition is missing, rest of the section is left as-is

Lemma 18. *XOR mode gives computational AXU (Almost Xor Universal)* ◇

Proof. (not proven) □

Theorem 20. *If F is a* PRF *and H is computational AXU, then XOR-MAC is a MAC.* ◇

Proof. (not proven) □

Summary

TODO 14: not sure what to do with this bullet list...

With variable input lenght:

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

10.2 CCA-security

Going back to the encryption realm, a new definition of attack to a SKE scheme will be introduced. Now the adversary can query a decryption oracle, along with the CPA-related encryption oracle, for polynomially many queries. This attack is called the *Chosen Ciphertext Attack*[18], and schemes that are proven to be CCA-secure are also defined as *non-malleable*, on the reasoning that an attacker cannot craft fresh valid ciphertexts from other valid ones.

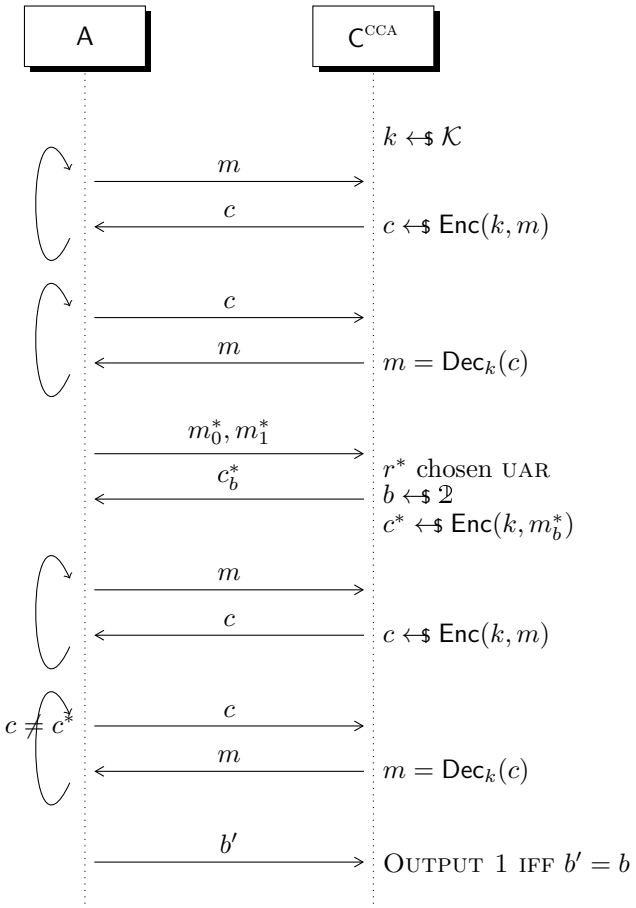


Figure 10.35: The chosen ciphertext attack, on top of CPA

Exercise 21. Show that the scheme Π_F defined in theorem 14, while CPA-secure, is not CCA-secure.

Proof. Let m_0 and m_1 be the messages the adversary sends to the challenger as the challenge plaintexts; on receiving the ciphertext $c_b = (r, f_k(r \oplus m_b) : b \leftarrow\$ \mathbb{2}$, the adversary crafts another ciphertext with an arbitrary value α:

$$\widehat{c_b} = (r, f_k(r) \oplus m_b \oplus \alpha)$$

[18] Different versions of the CCA notion exist. The one defined here is also called CCA2, or *adaptive Chosen Ciphertext Attack*

and queries the decryption oracle on it. The latter will decrypt the new ciphertext and return a plaintext, which can be easily manipulated by the adversary to reveal exactly which message was encrypted during the challenge:

$$\begin{aligned}\mathsf{Dec}(k, (\widehat{c_b}) &= \mathsf{Dec}(k, (r, f_k(r) \oplus m_b \oplus \alpha)) \\ &= f_k(r) \oplus f_k(r) \oplus m_b \oplus \alpha \\ &= m_b \oplus \alpha\end{aligned}$$

Therefore, the adversary certainly wins after just one decryption query, proving the scheme's vulnerability to CCA attacks.

□

10.3 Authenticated encryption

Instead of tackling the CCA-security problem upfront, it might be useful to consider a construction that achieves both secrecy and authentication: that is, a scheme that encrypts messages and authenticates their respective senders at the same time. In the encryption setting, such a scheme is defined as being CPA-secure with an additional AUTH property denoting the scheme's resistance to forgeries, much like its MAC cousins. The game shown in figure 10.36 models this AUTH property of a scheme $\Pi = (\mathsf{Enc}, \mathsf{Dec})$, with an additional quirk to the decryption routine:

$$\mathsf{Dec} \in \mathcal{K} \times \mathcal{C} \to M \cup \{\perp\}$$

where the $\perp$ value is returned whenever the decryption algorithm is supplied an invalid or malformed ciphertext.

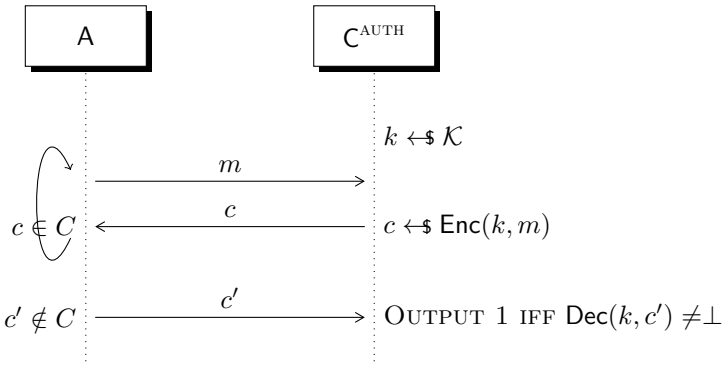


Figure 10.36: $\textsc{Game}^{\textsc{auth}}_{\Pi,\mathsf{A}}(\lambda)$

Theorem 22. *Let* Π *be a* SKE *scheme. If it is* CPA*-secure, and has the* AUTH *property, then it is also* CCA*-secure.* ◇

Proof. The proof is left as an exercise.

Hint: consider the experiment where $\mathsf{Dec}(k, c)$:

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

The approach would be to 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 C.

□

10.3.1 Combining SKE & MAC schemes

Let $\Pi_1 = (\mathsf{Enc}, \mathsf{Dec})$ be a SKE scheme, and $\Pi_2 = (\mathsf{Tag}, \mathsf{Ver})$ be a MAC scheme; there are 3 ways to combine them into an authenticated encryption scheme:

- *Encrypt-and-Tag*:
  1. $c \leftarrow\!\!\$\ \mathsf{Enc}(k_1, m)$
  2. $\phi \leftarrow\!\!\$\ \mathsf{Tag}(k_2, m)$
  3. $c^* = (c, \phi)$
- *Tag-then-encrypt*:
  1. $\phi \leftarrow\!\!\$\ \mathsf{Tag}(k_2, m)$
  2. $c \leftarrow\!\!\$\ \mathsf{Enc}(k_1, (\phi, m))$
  3. $c^* = c$
- *Encrypt-then-Tag*:
  1. $c \leftarrow\!\!\$\ \mathsf{Enc}(k_1, m)$
  2. $\phi \leftarrow\!\!\$\ \mathsf{Tag}(k_2, c)$
  3. $c^* = (c, \phi)$

Of the three options, only the last one is proven to be CCA-secure for arbitrary scheme choices; the other approaches are not secure "a-priori", with some couples proven to be secure by themselves. Notable examples are the *Transport Layer Security* (TLS) protocol, which employs the second strategy, and has been proven to be secure because of the chosen ecnryption scheme; *Secure shell* (SSH) instead uses the first strategy.

Theorem 23. *If an authenticated encryption scheme* Π *is made by combining a* CPA*-secure* SKE *scheme* Π_1 *with a* strongly unforgeable MAC *scheme* Π_2 *in the* Encrypt-then-tag *method. Then* Π *is* CPA*-secure and auth-secure.* ◇

Proof. Assume that Π is not CPA-secure; then an adversary can use the resulting distinguisher D^{CPA} to direct a succesful CPA against Π_1, as shown in figure 10.37: the point is to run the two components of Π separately.

TODO 15: Professor says that we have to show that $\text{GAME}^{\text{CPA}}_{\Pi,\mathsf{A}}(\lambda, 0) \approx_{\text{C}} \text{GAME}^{\text{CPA}}_{\Pi,\mathsf{A}}(\lambda, 1)$, but why? Isn't this proof enough?

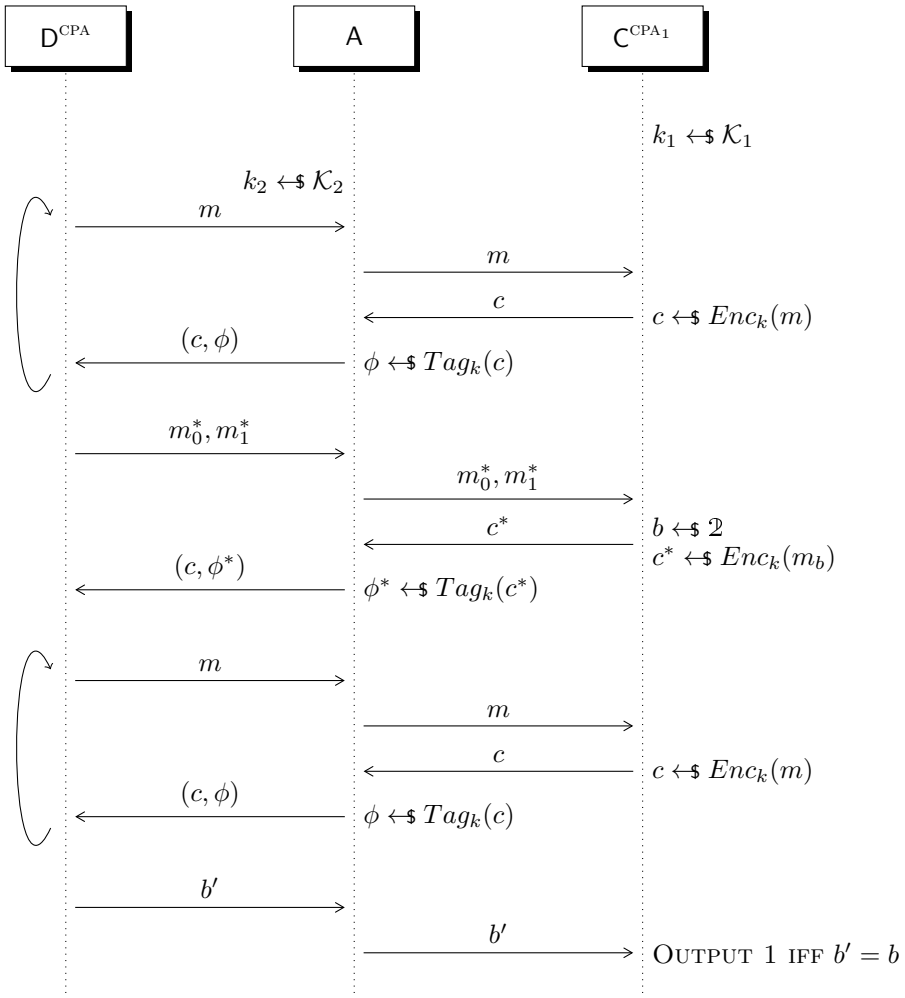

$\mathsf{D}^{\mathrm{CPA}}$
A
$\mathsf{C}^{\mathrm{CPA}_1}$
$k_1 \leftarrow\!\$\ \mathcal{K}_1$
$k_2 \leftarrow\!\$\ \mathcal{K}_2$
m
m
c
$c \leftarrow\!\$\ Enc_k(m)$
(c, ϕ)
$\phi \leftarrow\!\$\ Tag_k(c)$
m_0^*, m_1^*
m_0^*, m_1^*
c^*
$b \leftarrow\!\$\ \mathbb{2}$
$c^* \leftarrow\!\$\ Enc_k(m_b)$
(c, ϕ^*)
$\phi^* \leftarrow\!\$\ Tag_k(c^*)$
m
m
c
$c \leftarrow\!\$\ Enc_k(m)$
(c, ϕ)
$\phi \leftarrow\!\$\ Tag_k(c)$
b'
b'
Output 1 iff $b' = b$

Figure 10.37

□

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

Exercise 24. Prove it!

Similar to the cpa-security proof.

Lesson 11

11.1 Authenticated encryption (continued)

Having proven that an authenticated encryption scheme in an *Encrypt-then-Tag* mode is CPA-secure, it remains to prove that it has the AUTH property. Before this, a new unforgeability definition is needed:

Definition 22. Let $\Pi = (\mathsf{Tag}, \mathsf{Ver})$ be a MAC scheme. Then Π is *strongly* resistant to *existential forgery under chosen message attack*, or SEF-CMA-secure in short, iff:

$$\Pr[\text{GAME}^{\text{UFCMA}}_{\Pi,\mathsf{A}}(\lambda) = 1] \in \mathcal{N}egl(\lambda)$$

i.e. it is UFCMA, but with the additional restriction that the tag ϕ^* of the forged message must be "fresh" itself. ◇

Note the small difference in security between UFCMA and SEF-CMA.

Theorem 25. *Let* $\Pi = (\mathsf{Enc}, \mathsf{Dec}, \mathsf{Tag}, \mathsf{Ver})$ *be an authenticated encryption scheme, composed by a* SKE *scheme* Π_1 *and a* MAC *scheme* Π_2*. If* Π_2 *is* SEF-CMA*, then* Π *has the* AUTH *property.* ◇

Proof. The proof is analogous to the previous proof regarding the scheme's CPA security. Suppose that Π has not the AUTH property; then an adversary can use the distinguisher D^{AUTH} to successfully forge authenticated messages with fresh signatures against Π_2, as depicted in figure 11.38.

From D^{AUTH}'s perspective, all the couples (c_i, ϕ_i) received are made from m as follows:

$$c_i \xleftarrow{\$} \mathsf{Enc}(k_1, m) \wedge \phi_i \xleftarrow{\$} \mathsf{Tag}(k_2, c_i)$$

Since D^{AUTH} is able to break the AUTH property of Π, the couple it outputs (c^*, ϕ^*) will be such that:

$$\mathsf{Dec}(k_1, c^*) = \widehat{m^*} \wedge \mathsf{Ver}(k_2, c^*, \phi^*) = 1$$

Note that it is not important that the ciphertext decodes to the original plaintext; the mater at hand is that A can now use the same couple to break Π_2's UFCMA property, which is a contradiction.

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. □

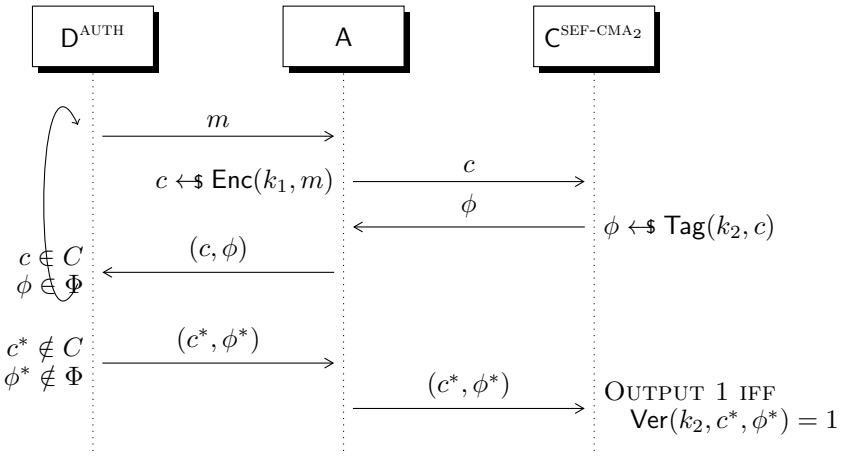


Figure 11.38: Breaking authenticity of Π_2

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

Todo 16: Luby-Rackoff is cited here, but it's related to both PRFs-PRPs and the Feistel network analysis

Nothing prevents a PRF g_k to be bijective; in this case, it is referred to as a *pseudorandom permutation*, or PRP in short. Their definition is analogous to a generic PRF, as shown in figure 11.39: PRPs are computationally indistinguishable from a random permutation.

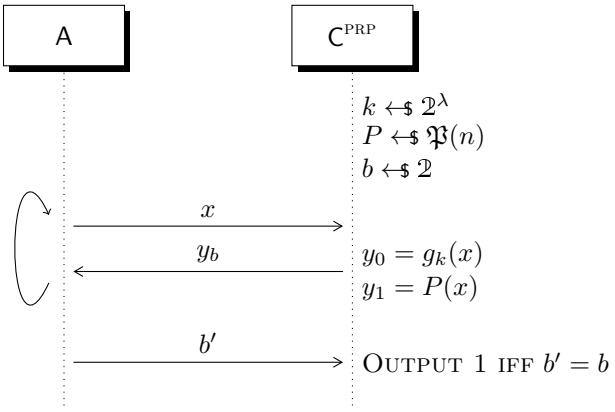


Figure 11.39: $\text{Game}^{\text{PRP}}_{g_k,\mathsf{A}}(\lambda, b)$

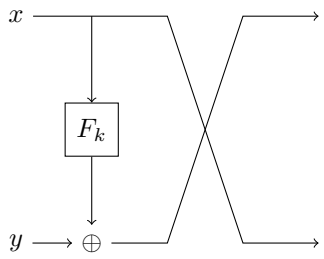


Figure 11.40: A single-round Feistel network

$$\textsc{Game}^{\text{PRP}}_{g_k,\mathsf{A}}(\lambda,0) \approx_{\text{C}} \textsc{Game}^{\text{PRP}}_{g_k,\mathsf{A}}(\lambda,1)$$

An important difference is that g_k is efficiently invertible, although knowledge of k is reqiuired in order to do so.

11.2.1 Feistel network

PRPs have been successfully constructed by using existing PRFs into what is called a *Feistel network*. As a starting point, let $f_k \in 2^n \to 2^n$ be a PRF, and define the function ψ_f as follows:

$$\begin{aligned}\psi_f(x,y) &= (y, x \oplus F(y)) = (x', y')\\ \psi_f^{-1}(x',y') &= (F(x') \oplus y', x') = (F(y) \oplus x \oplus F(y), y) = (x,y)\end{aligned}$$

While this construct is invertible and uses a PRF, it is not pseudorandom itself, because the first n bits of ψ_F's image are always equal to y, and thus visible to any adversary. A first attempt at fixing this vulnerability would be to apply the construct two times on two different PRFs $\psi^2_{F,F'}$, in an attempt to "hide" y. Yet, this approach still leaks valuable information:

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

However, this example with additional restrictions will be useful very soon, so it is reworded as the following lemma:

Lemma 19. *For any unbounded adversary making $q \in \mathcal{P}oly(\lambda)$ queries, he is unable to consistently win the game depicted in 11.41:*

$$\textsc{Game}^{\text{R-FEIS}}_{\Pi,\mathsf{A}}(\lambda,0) \approx_{\text{S}} \textsc{Game}^{\text{R-FEIS}}_{\Pi,\mathsf{A}}(\lambda,1)$$

as long as $y_1, \dots, y_q$ are mutually distinct. ◇

Proof. TODO 17: Idea: Hybridize over the queries before the challenge, from PR to random; prove that the stat distance between i and i+1 is negligible

□

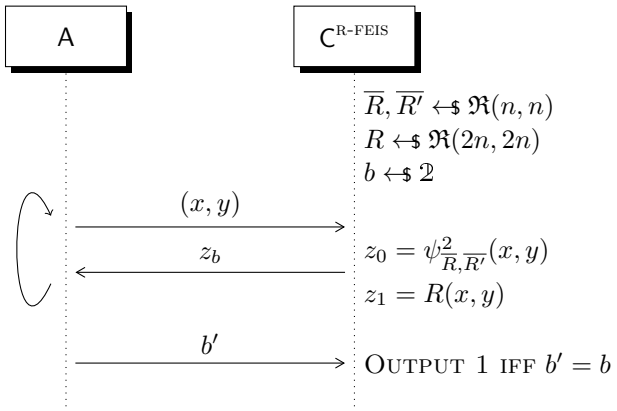


Figure 11.41: The randomic-Feistel distinguishing game: $\text{GAME}^{\text{R-FEIS}}_{\Pi,\mathsf{A}}s$

Going back to the Feistel networks in general, it should be easy to see that they can be made of an arbitrary number of rounds, by simply chaining output with input. The l-th iteration is denoted as:

$$\psi^l_\Phi(x,y) = \psi_F(\psi_{F''}(\ldots\psi_{F^{(l)}}(x,y)\ldots))$$

where Φ is the sequence of PRFs used at each single step. It can be shown that the rounds neeed to obtain a network that is indeed pseudorandom is just 3; also, the same PRF can be used, it is sufficient to change the seed on each iteration[19]:

Theorem 26. *Let F_i, F_j, F_k be a* PRF *over three seeds. Then $\psi^3_{i,j,k}$ is a* PRP. ◇

Proof. TODO 18: Idea: Four total games: original, swap prfs with random functions, swap the three functions with a single one (use previous lemma), swap random function with random permutation (avoid bad events generated by injection property)

□

[19] That is actually the purpose of using a PRF

Lesson 12

12.1 Hashing

Remember one solution to domain-exetension for PRFs, as a composition of a PRF F with an almost universal hash function H. Hash functions compress their arguments to some "fingerprint", which is asssumed to be unique. However, since this compression in this context inherently introduces information loss, it is not guaranteed that every message gets its own unique fingerprint; indeed, there will be some instances where two messages yield the same hash value, or in other words, the hashes *collide*. It is desirable for a hash function to be *resistant* to these events, meaning that it is hard to reproduce such collisions.

Definition 23. A hash function family H is deemed *collision-resistant*, denoted as $H \in \text{CRH}$ iff the probability of finding a collision is negligible, even when knowing the seed s. Formally:

$$\forall \mathsf{A} \in \text{PPT} \implies \Pr(\text{GAME}^{\text{CRH}}_{\Pi,\mathsf{A}}(\lambda) = 1) \in \mathcal{N}egl(\lambda)$$

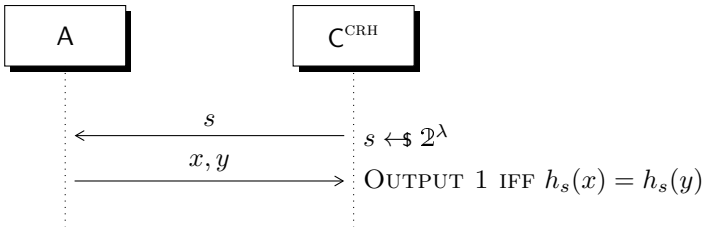


Figure 12.42: The *collision-resistance* game

◇

A note: before, we were dealing with unbounded adversaries, and the key was hidden. Now the key is public, but the adversary must be efficient.

Exercise 27. Let Π be a UFCMA authentication scheme over the message space $\mathbb{2}^n$. Show that $\Pi' = (\mathsf{Tag}', \mathsf{Ver}') : \mathsf{Tag}'_{k,s}(m) = \mathsf{Tag}_k(h_s(m))$ is UFCMA-secure over $\mathbb{2}^l$, where $l \in \mathcal{P}oly(n)$, as long as h_s itself is CRH.

12.1.1 Merkle-Damgård construction

A construct for starters has been defined by Ralph Merkle and Ivan Damgård, which revolves around the use of a CRH function $h_s \in \mathbb{2}^{n+c} \to \mathbb{2}^n$, where c is

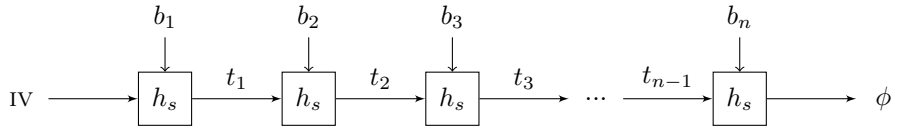


Figure 12.43: Basic outline of a Merkle-Damgård construction

the size of one "block" of a message. For now, consider the messages to be of an arbitrarily fixed length l, and let c be 1. The steps to follow are:

1. Let IV $\in 2^b$ be an *initialization vector*
2. Initialize $t_0 :=$ IV
3. For each bit b_i of the message to hash m:
   (a) Compute $t_i := h_s(b_i, t_{i-1})$
4. Return $\phi = t_n$

Figure 12.43 depicts a general view of the algorithm. Let it be denoted as another hash function $h'_s \in 2^l \to 2^n$.

Theorem 28. *The construction H' obtained by Merkle-Damgård is a* CRH *function.* ◇

Proof. Assume H' can be broken efficiently by a distinguisher D^{CRH}, meaning that finding two distinct block sequences that give the same hash is easy. Consider the reduction to H's CRH-ness in figure 12.44

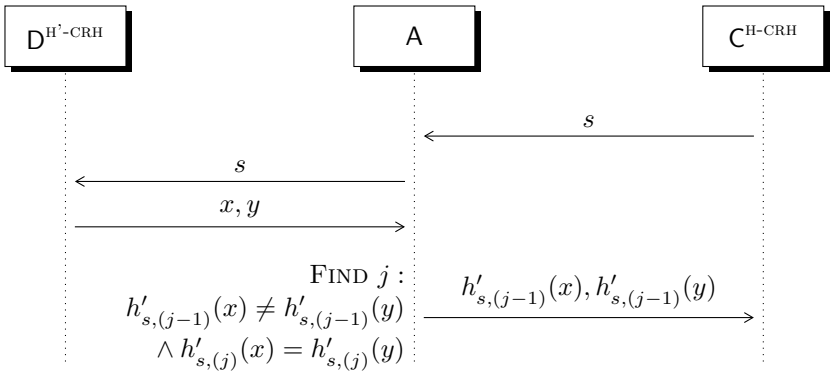


Figure 12.44: Breaking the underlying CRH-ness of H

Ignore same blocks: find largest j such that:

$$(b_j, x_{j-1}) \neq (b'_j, y_{j-1}) \wedge h_s(b_j, x_{j-1}) = h_s(b'_j, y_{j-1})$$

this implies the rest of th message is equal, then the resulting final hash will be equal, thus for $j > 0$ we have a collision.

□

Domain extension

In order to adapt the MD-construct to messages of variable length, some sort of "strengthening" is required:

Lemma 20 (Length padding)**.** *Let* $H_s \in \mathbb{2}^{n+l} \to \mathbb{2}^n$*, then:*

$$H'_s = H_s(\langle l' \rangle, H_s(x_{l'}, \ldots, H_s(x_1, 0^n)\ldots)), |l'|, |x_i| \in \mathbb{2}^c$$

◇

This is an example of padding which encodes the message length in itself.

Theorem 29. *The strengthened construct is collision resistant for variable-length messages.* ◇

Proof. Hint: similar as above, case by case □

Merkle trees

This is an alternative construction to the "linear" approach used beforehand: it starts from the message's blocks acting as leaves of a complete binary tree, and using a halving compression function on each node from the bottom up, and outputting the final tag as the image of the root invocation. It also has nice properties, such as easy verification of the presence of a single specific block in the message.

12.1.2 Compression functions

The compression functions are the central point of these kinds of "fingerprinting" tag schemes. They are, essentially, hash functions with a smaller codomain than its domain, thereby forcing the existence of collisions.

Let $(\mathsf{Gen} : 0 \mapsto (\mathfrak{pk}, \mathfrak{sk}), f, g)$ be a PKE scheme, where the functions f and g are keyed PRPs. A *claw* is a couple of values (x, x') such that:

$$f(\mathfrak{pk}, x) = g(\mathfrak{pk}, x')$$

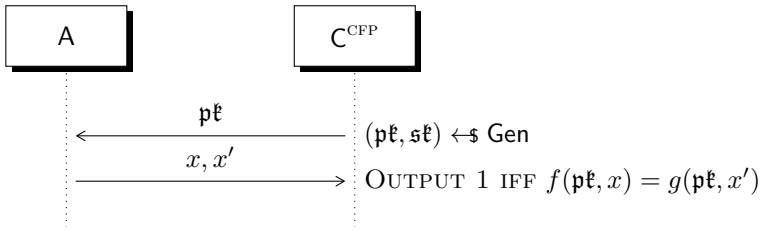


Figure 12.45: The game of claw-free permutations

Theorem 30. *Assuming* $\mathcal{F}$ *is claw-free, then* $h_{\mathfrak{pk}}$ *is* CRH *from* $n + l$ *bits to* n*.* ◇

Davies-Meyer construct

Definition 24. $x_{i+1} = E_k(x_i) \oplus x_i$, maps n+$\lambda$ to n. E is AES ◇

Lesson 13

13.1 Number theory

Theorem 31 (Fermat's last theorem)**.**

$$\forall x, y, z \in \mathbb{Z}, n > 2 \implies x^n + y^n \neq z^n$$

◇

Lemma 21.

$$\forall a \in \mathbb{Z}_n : \gcd(a, n) > 1 \implies a \notin \mathbb{Z}_n^\times$$

◇

Proof. Assume there exists $b \in \mathbb{Z}_n$ such that $ab \equiv_n 1$. Then, there exists a quotient for the division *by* nbetween a and b with remainder 1. Observe that $\gcd(a, n)$ divides $ab + qn$, which is equal to 1. It entails that $\gcd(a, n) = 1$, which is a contradiction. □

Lemma 22.

$$\forall a, b \in \mathbb{N} : a \geq b \neq 0 \implies \gcd(a, b) = \gcd(b, a \bmod b)$$

◇

Proof. Not given. □

Theorem 32. *Given two integers a and b, their greatest common divisor can be computed efficiently with respect to their lengths. Additionally, two other numbers u and v can be efficiently computed in order to satisfy Bézout's identity:* $\gcd(a, b) = au + bv$ ◇

Proof. Hint: Use previous lemma recursively; we stop at $r_{t+1} = 0$ for some t, thus:

$$\gcd(a, b) = \cdots = \gcd(r_t, r_{t+1}) = r_t$$

□

Claim 5. $r_{i+2} \leq r_i/2 \forall 0 \leq i \leq t - 2 \implies \#steps = \lambda - 1$ *if* $|b| \in 2^\lambda$ ◇

The hypohesis is a natural consequence of the repeated modulo operations.

Observation 2 (Exponentiation mod n: Square and multiply)**.** *Let $b \in \mathbb{2}^l$, where by writing b_i we denote b's i-th bit. Then:*

$$a^b \equiv_n a^{\sum_{i=0}^{l} 2^i b_i} \equiv_n \prod_{i=0}^{l} a^{2^i b_i}$$

◇

Theorem 33. *The number of primes lesser than or equal to x is a number greater than or equal to $\frac{x}{3 \log_2 x}$* ◇

There exist many algorithms that solve the problem of primality testing, with time complexity polynomial in the length of their numerical representation; of most relevance are those of Miller-Rabin, which is probabilistic, but consistently used in practice, and the completely deterministic Agrawal-Kayal-Saxena (AKS), which has a much greater polynomial rank, and has been deemed impractical for most uses.

What remained as a conjecture is the intractability of determining the factors of a λ-bit composite number, which is widely known as the *factorization hardness*; this in turn would imply that integer multiplication of two λ-bit primes is a OWF.

Definition 25. Given a group G, its order is the least i such that $a^i \equiv_n 1$ ◇

Corollary 3.

$$\forall a \in \mathbb{Z}_m^\times \implies a^{\phi(n)} \equiv_n 1 \land a^b \equiv_n a^{b \bmod \phi(n)} \land a^{p-1} \equiv_p 1$$

◇

Proof. $(\mathbb{Z}_n^\times, \cdot)$ is a group with $\phi(n)$ elements. Take $\langle a \rangle = \{a^0, \ldots, a^{d+1}\}$, where d is the order. We must have $\phi(n) = kd$ for some k. Therefore, $a^{\phi(n)} = a^{dk} \equiv_n 1$.

Also, $a^b \equiv_n a^{b\phi(n)+b \bmod \phi(n)} \equiv_n 1 \cdot 2^{b \bmod \phi(n)}$ □

Theorem 34. *Let G, H be two groups such that $H < G$, meaning the order of H divides the order of G* ◇

13.2 Standard model assumptions

We now turn our attention to some conjectures that form the basis for most of the cryptographic schemes that will follow. We will start from the weakest, and go up to the strongest. For all our purposes, let $\mathcal{GG}(1^\lambda)$ be a "group generator" with security parameter λ, and let a random sample $(G, g, q) \xleftarrow{\$} \mathcal{GG}(1^\lambda)$ be a triplet composed of the group itself G, one of its generators g, and its order q.

Discrete logarithm

Given g and g^x in a λ-bit group, there is no efficient algorithm for computing y such that $g^y = g^x$ withouut knowing x beforehand:

$$\forall \mathsf{A} \in \text{PPT} \implies \Pr[\textsc{Game}^{\text{DL}}_{\Pi,\mathsf{A}}(\lambda) = 1] \in \mathcal{N}egl(\lambda)$$

This means that the DL for a generic group G yields a OWF, whereas in a multiplicative group $\mathbb{Z}_p^\times$, we obtain a OWP.

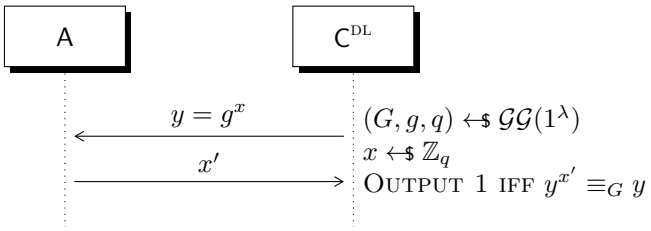


Figure 13.46: $\textsc{Game}^{\mathrm{DL}}_{\Pi,\mathsf{A}}(\lambda)$

Computational Diffie-Hellman

Given a group G and two elements in it g^x, g^y, it is impractical to compute g^{xy} without knowing x or y.

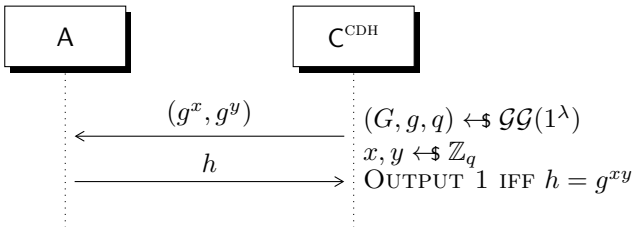


Figure 13.47: $\textsc{Game}^{\mathrm{CDH}}_{\Pi,\mathsf{A}}(\lambda)$

Decisional Diffie-Hellman

Given a group G and three elements in it g^x, g^y, g^z, it is impractical to distinguish g^{xy} from g^z by only knowing g^x and g^y, along with the originating triad (G, g, q)).

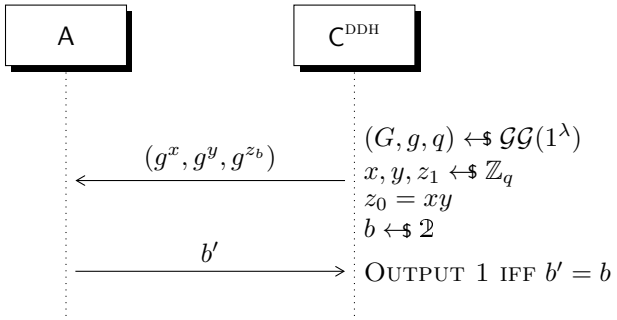


Figure 13.48: $\textsc{Game}^{\mathrm{DDH}}_{\Pi,\mathsf{A}}(\lambda)$

All these assumptions helped in constructind the *Diffie-Hellman key exchange* protocol, which is a way to establish a SKE channel from an unsafe channel, with any adversary unable to efficiently break the channel's secrecy. Do note that authentication is left out of the picture here.

Some relationships have been established between these assumptions: it is known that DDH $\implies$ CDH $\implies$ DL; and that CDH $\not\Longrightarrow$ DDH.

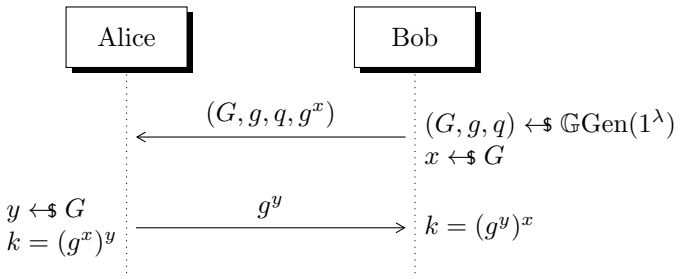


Figure 13.49: The Diffie-Hellman Key Exchange protocol

Lesson 14

Decisional Diffie-Hellman assumption (cont'd)

Claim: DDHis not hard for groups $\mathbb{Z}_p^\times$

Proof: Let $Quad_p$ be the group of quadratic residues modulo p, group operation is multiplication.

We can test if a give number y is in Quad by checking if $y^{(p-1)/2} \equiv_p 1$, because:

$$y = g^{2z} \implies y^{(p-1)/2} = g^{2z(p-1)/2} = g^{z(p-1)} \equiv_p 1$$

Otherwise:

$$y \neq g^{2z} \implies y^{(p-1)/2} \equiv_p g^{z'(p-1)} \cdot g^{(p-1)/2} \not\equiv_p 1$$

Claim: $g^{xy} \in Quad_p \implies 2|x \vee 2|y$

???

Can have a distinguisher; end of Proof

Nevertheless, some otehr groups are believed to harden quadratic residue membership; such groups are $Quad_p$ itself, or the elliptic curve groups.

Recall: DL is hard

Extend: $g^x, g^{y_1}, g^{xy_1}, g^{y_2}, g^{xy_2}, g^{y_3}, g^{xy_3}, \ldots$

prove this is hard by hybrid arg

Naor-Reingold encryption scheme

14.1 Public key encryption schemes

Lesson 15

15.1 Public key encryption recap

$\text{GAME}_{\Pi,\mathsf{A}}^{\text{PKE-CCA}}$:

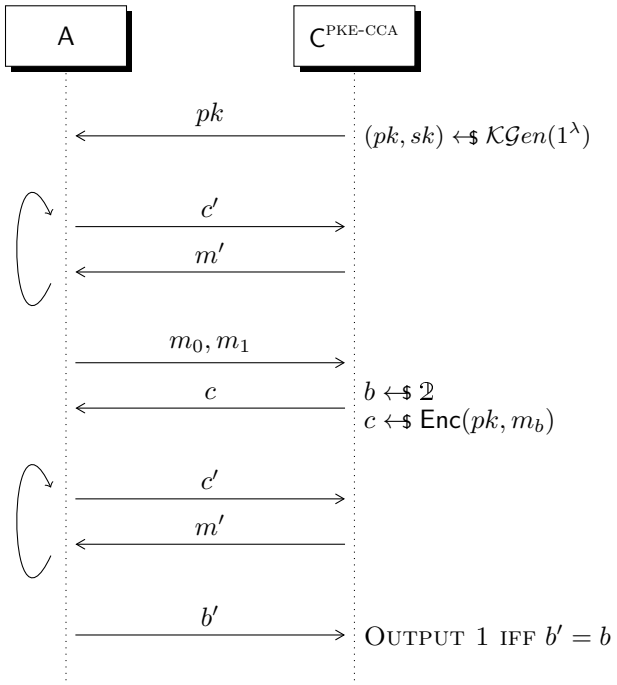


Figure 15.50: CCA on a PKE scheme

(Reminder): Encryptions are made like this: $Enc(k,m) = (r, F_k(r) \oplus m), r \simeq \mathcal{U}nif(2^\lambda)$.

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

15.1.1 Trapdoor permutation

A *trapdoor permutation* (or TDP) 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
  - Domain sampling ($x \leftarrow\!\!\$\ V_{pk}$) 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 an 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 35. *If $(\mathcal{KG}en, f, g)$ is a TDP, and $\mathfrak{hc}$ is hardcore for f, then the above scheme is* CPA*-secure.* ◇

Proof. The proof is left as exercise

TODO 19: Apparently, the reduction here is not easy at all, some hints are needed.

□

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 36. *(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):

TODO 20: $\forall x \in \mathbb{Z}_n \implies x^{\varphi(n)} = x \bmod n$
maybe the correct one is
$\forall x \in \mathbb{Z}_n \implies x^{\varphi(n)} = 1 \bmod n$

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 $\exists! b \in \mathbb{Z}_n : ab = 1 \bmod \varphi(n)$, b will be our private key.
Define encryption as $f(a, m) = m^a \bmod n$, and then decryption as $g(b, c) = c^b \bmod n$.
Observe that

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

because $ab = 1 \bmod \varphi(n)$.
So we conjecture that the above is a valid TDP-based PKE scheme. This is actually referred to as the RSA *assumption*, and is depicted in figure 15.51

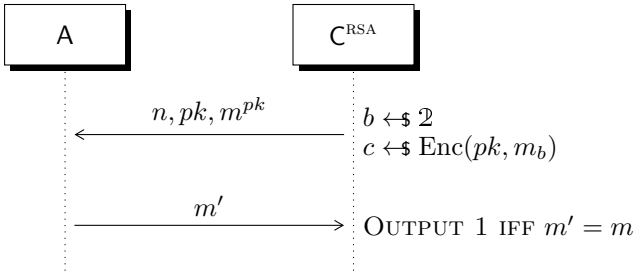


Figure 15.51: The RSA assumption

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) scheme.

- Setup: $pk \leftarrow\!\!\$\ \mathbb{Z}_n^{\times}, sk \equiv_{\varphi(n)} pk^{-1}$
- Encrypt: $Enc_{pk}(m) = m^{pk} \bmod n$
- Decrypt: $Dec_{sk}(c) = c^{sk} \bmod n$

- Correctness: $Enc_{pk}(Dec_{sk}(m)) = m^{pk \cdot sk} \equiv_n m$

Again, since the encryption routine is deterministic, the scheme is not CPA-secure. However, a hardcore predicate can be inserted to the routine: $\hat{m} = r||m$, where $r \leftarrow\!\!\$\ \mathbb{2}^l$. Now the encryption is pseudorandom.

Facts:

1. $l \in \omega(log(\lambda))$ otherwise it is possible to eficiently bruteforce it.

TODO 21:

2. If $m \in \mathbb{2}$ then I can prove it CPA secure under RSA (just use standard TDP)

3. If m is "in the middle" ($\mathbb{2} \leq m \leq \mathbb{2}^l$) RSA is believed to be secure and is standardized (PKCS#1,5)

TODO 22:

4. Still not CCA secure! counterexample?

15.2.1 Trapdoor Permutation from Factoring

Let's look at $f(x) = x^2 \mod n$ where $f : \mathbb{Z}_n^\times \to \mathbb{QR}_n (\subset \mathbb{Z}_n^\times)$, this is not a permutation in general.

Now let's consider the Chinese Reminder Theorem (CRT) representation:

$$x = (x_p, x_q) \to x_p \equiv x \mod p, x_q \equiv x \mod q$$
$$f(x) = x^2 \mod p; x \leftarrow\!\!\$\ \mathbb{Z}_p^\times$$

Since $\mathbb{Z}_p^\times$ is cyclic I can always write:

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

Moreover, since $(g^{\frac{p-1}{2}})^2 \equiv 1 \mod p$ 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 $\mathbb{Z}_p^\times$, then $g^{\frac{p-1}{2}} \equiv -1 \mod p$.

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 $p \equiv 3 \bmod 4$ ([*]$p = 4t + 3 \Rightarrow t = \frac{p-3}{4}$), then squaring modulo p is a permutation because, given $\underline{y = x^2 \bmod p}$ 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 23. $\forall z, z \in \mathbb{QR}_p \implies -z \notin \mathbb{QR}_p$ $\diamond$

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 $\mathbb{Z}_n^\times$.

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 $\frac{|\mathbb{Z}_n^\times|}{4}$.

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

Claim 6. *Given x, z such that x^2 $modn \equiv z^2$ mod $n \equiv y$ mod n,*

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

◇

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. □

Theorem 37. *Squaring mod n, where n is a* Blum integer[20] *is a trapdoor permutation under the factoring assumption.* ◇

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 A choses x, here $\mathcal{BG}en$ is a sampler for Blum integers:

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. □

[20] a Blum integer n is the product of two numbers p and q such that $p, q = 3 \bmod 4$, as the definition of Rabin's TDP

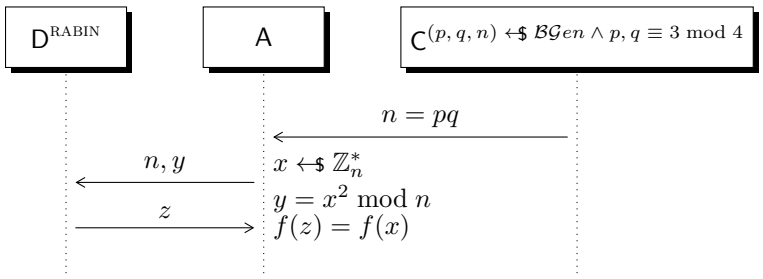

$\mathsf{D}^{\text{RABIN}}$
A
$\mathsf{C}^{(p,q,n) \leftarrow\$ \mathcal{BG}en \wedge p,q \equiv 3 \bmod 4}$
$n = pq$
n, y
$x \leftarrow\$ \mathbb{Z}_n^*$
$y = x^2 \bmod n$
z
$f(z) = f(x)$

Figure 15.52: —

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 $(G, g, q) \leftarrow\!\!\$\ GroupGen(1^\lambda)$[21].

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 $\text{Enc}(pk, m)$ will:

- Pick $r \leftarrow\!\!\$\ \mathbb{Z}_q$
- Output $c = (c_1, c_2) = (g^r, pk^r \cdot m)$[22]

The decryption routine $\text{Dec}(sk, c)$ 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 38. *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.

Note: it is important to note that we can measure the advantage of A, so fixed its output $Adv_{\mathsf{A}}(\lambda) = |\underbrace{Pr[\overbrace{\mathsf{A} \to 1}^{b'=1} | b = 0]}_{\text{"A } loses\text{"}} - \underbrace{Pr[\overbrace{\mathsf{A} \to 1}^{b'=1} | b = 1]}_{\text{"A } wins\text{"}}|$. Since b is

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

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

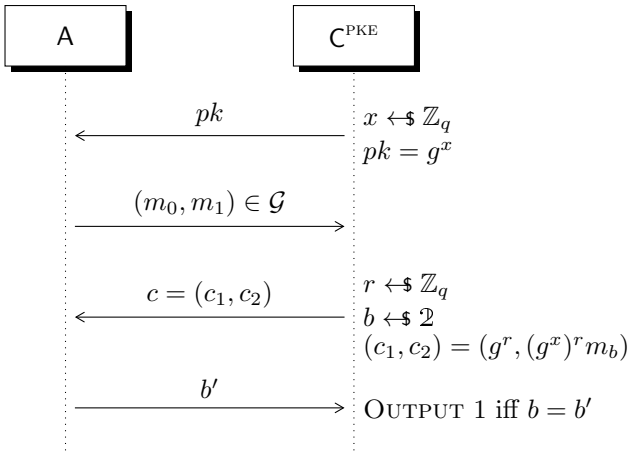


Figure 16.53: $H_0(\lambda, b)$

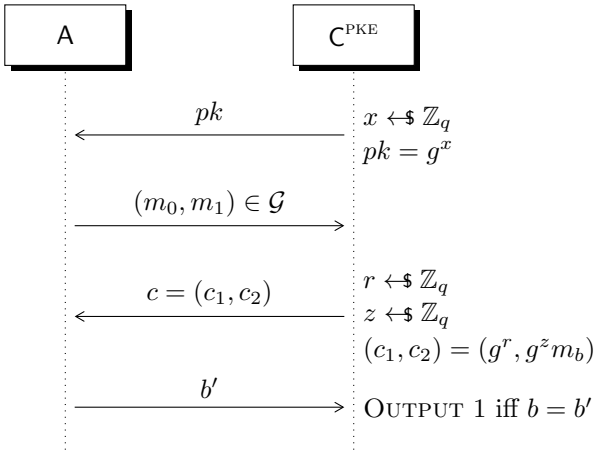


Figure 16.54: $H_1(\lambda, b)$

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 24. $\forall b \in \mathbb{2}, H_0(\lambda, 0) \approx_c H_0(\lambda, 1)$
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:

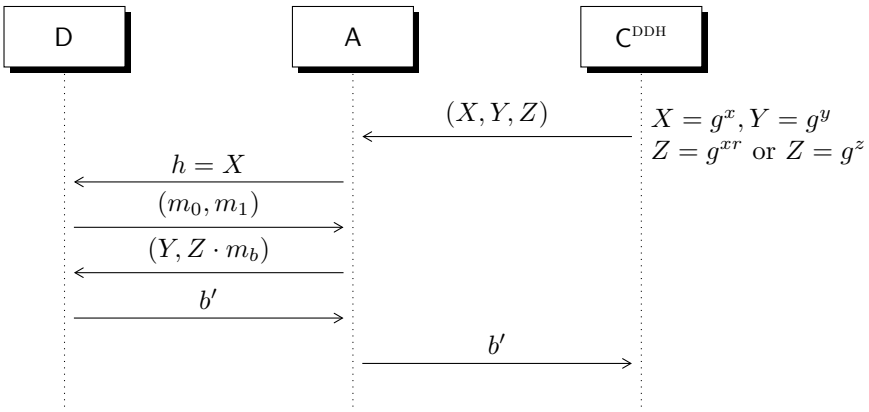


Figure 16.55: —

Contradiction: D should be able to compute log_g to distinguish the message.

□

Lemma 25. $H_1(\lambda, 0) \equiv H_1(\lambda, 1)$ ◇

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

Lemma 26. $H_1(\lambda, 1) \equiv H_0(\lambda, 1)$ ◇

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).

□

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:

  $$(c_1 \cdot c'_1, c_2 \cdot c'_2)$$
  $$= (g^{r+r'}, h^{r+r'}(m \cdot m'))$$

  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:

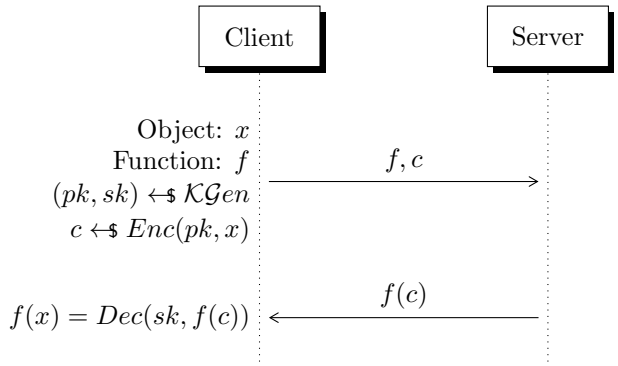


Figure 16.56: Delegated secret computation

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* (DV-NIZK), or alternatively **Hash-Proof System**, is used here.

Proof systems

Let L be a Turing-recognizable language in NP, and a predicate $\mathcal{R} \in X \times Y \to 2$ 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.

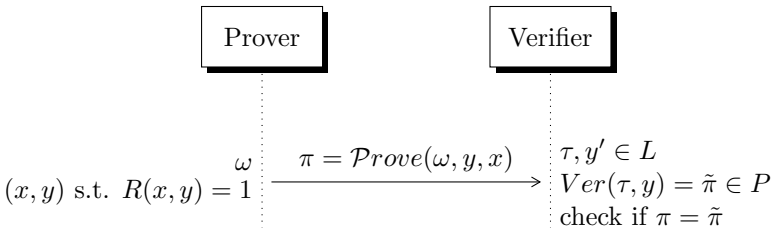


Figure 16.57: Overview of Cramer-Shoup operation

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**: $\forall y \in L, \forall(\omega, \tau) \leftarrow\!\!\$\ \mathcal{S}etup(1^\lambda)$: $\mathcal{P}rove(\omega, x, y) = \mathcal{V}erify(\tau, y)$

TODO 23: to review and understand/better

t-universality

Definition 26. Let Π be DV-NIKZ[23].
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 $(\omega, \tau) \leftarrow\!\!\$\ Setup(1^\lambda)$ and $v_1, \ldots, v_t \leftarrow\!\!\$\ \P$ where $\P$ should be the proofs' space. ◇

[23] Designated verifier non-interactive zero-knowledge

Enriching DV-NIKZ

Can we enrich DV-NIKZ with labels $l \in 2^*$?

Suppose to have the following:

$$L' = L \| 2^* = \{(y, l) : y \in L \wedge l \in 2^*\}$$

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 27. 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. $\{y : (y, x) \leftarrow\!\!\$\ Samp(1^\lambda)\} \approx_{\mathrm{C}} \{y : y \leftarrow\!\!\$\ \bar{Samp}(1^\lambda)\}$

$\diamond$

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 ($L' = L||\mathbb{2}^\ell$).

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

- $(\overbrace{(\omega_1, \omega_2)}^{pk}, \overbrace{(\tau_1, \tau_2)}^{sk}) \leftarrow\!\$\ \mathcal{KG}en(1^\lambda)$, $(\omega_1, \tau_1) \leftarrow\!\$\ Setup_1(1^\lambda), (\omega_2, \tau_2) \leftarrow\!\$\ Setup_2(1^\lambda)$
- 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 39. *Assuming π_1 is 1-universal, π_2 is 2-universal and L is a membership-hard language; then the above scheme is* CCA*-secure.* ◇

Proof. Five different games will be defined, from $\text{GAME}^0_{\Pi,\mathsf{A}}$ up to $\text{GAME}^4_{\Pi,\mathsf{A}}$; the first game will be an analogous formalization of how the above PKE scheme works. It shall be proven that, for arbitrarily fixed b in $\mathbb{2}$:

$$\text{GAME}^0_{\Pi,\mathsf{A}}(\lambda,b) \equiv \text{GAME}^1_{\Pi,\mathsf{A}}(\lambda,b) \approx_{\text{C}} \text{GAME}^2_{\Pi,\mathsf{A}}(\lambda,b) \approx_{\text{S}} \text{GAME}^3_{\Pi,\mathsf{A}}(\lambda,b) \equiv \text{GAME}^4_{\Pi,\mathsf{A}}(\lambda,b)$$

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

TODO 24: 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:

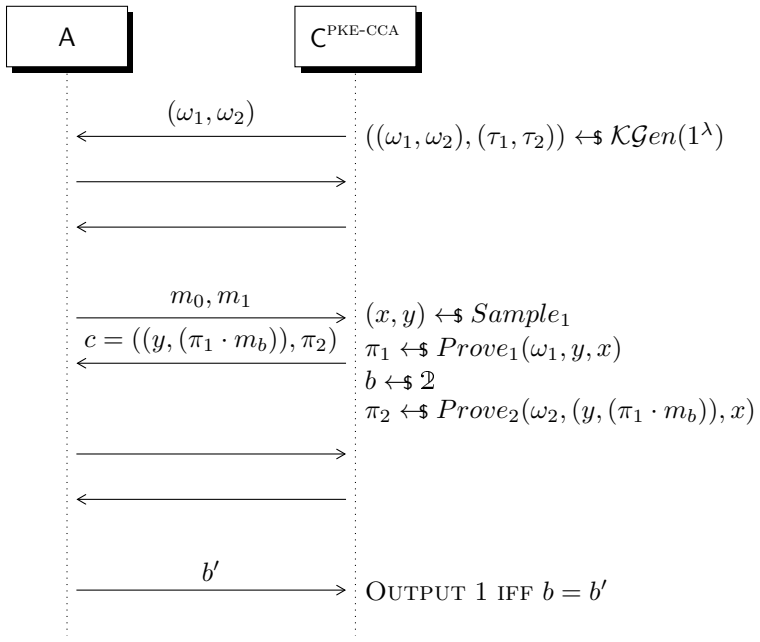


Figure 17.58: Original CCA game

TODO 25: Non è stato chiaro sull'origine di x ed y, inoltre mi manca da scrivere le query di decifratura

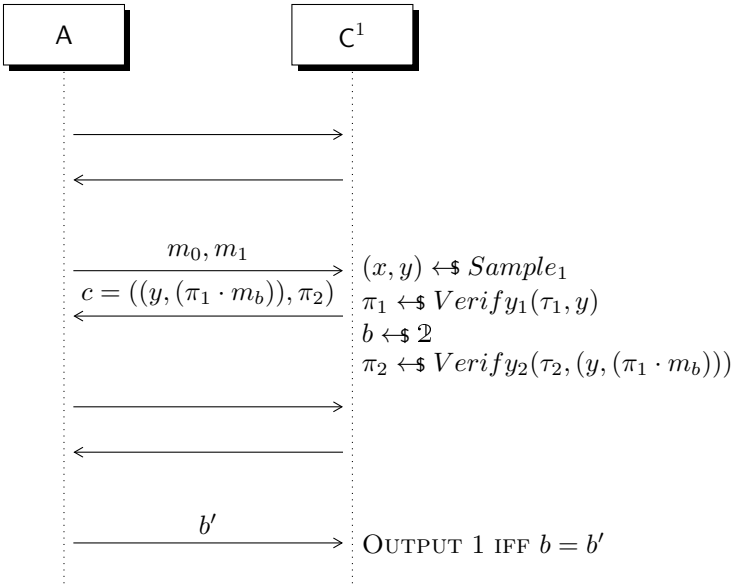


Figure 17.59: Use verifiers

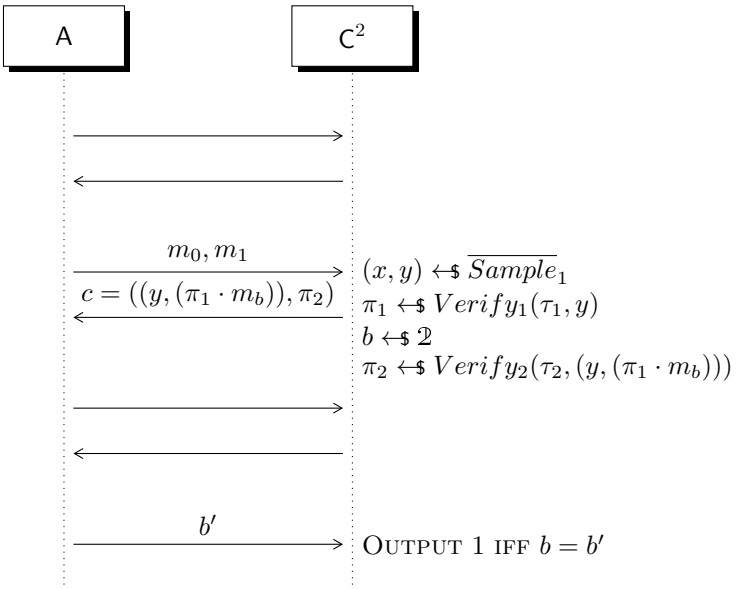


Figure 17.60: Sample statements outside the language

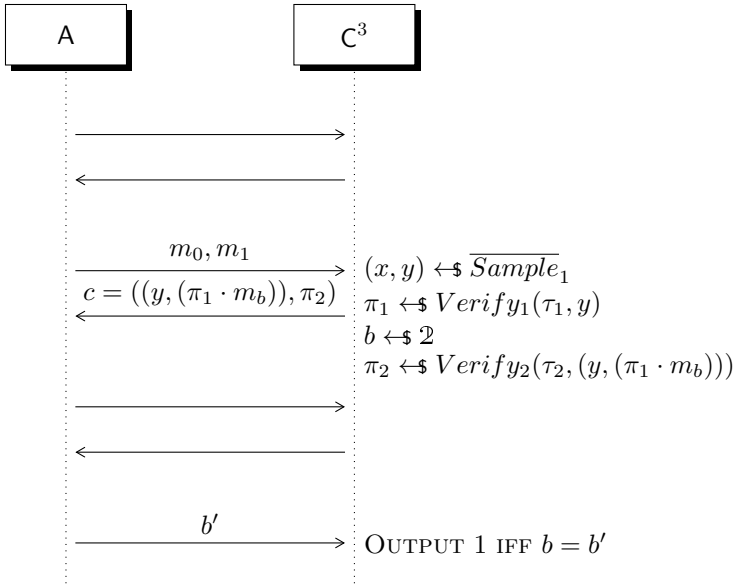


Figure 17.61: Modify decryption queries

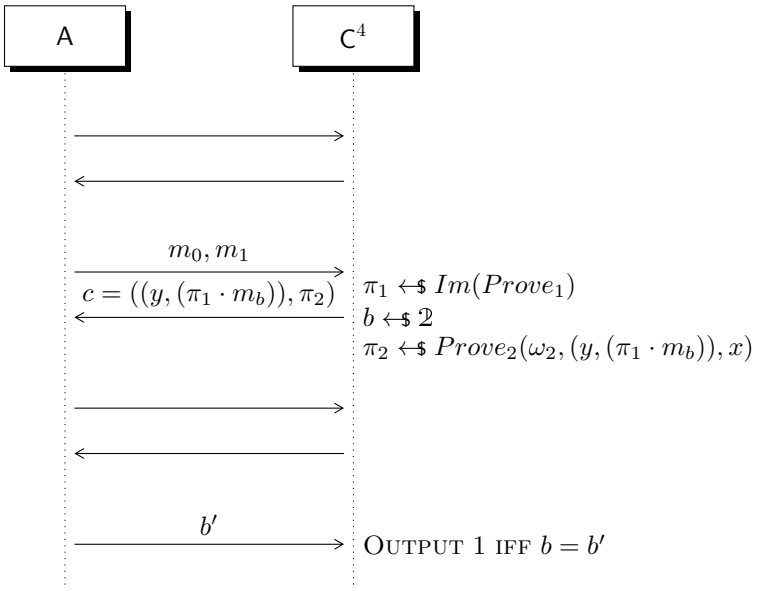


Figure 17.62: π_1 is chosen UAR

□

Lemma 27.

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

◇

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 $(\omega_1, \tau_1) \leftarrow\!\!\$\ Setup_1(1^\lambda)$

$$\Pi_1 \leftarrow Prove_1(\omega, y, x), (\omega_2, \tau_1) \leftarrow\!\!\$\ Setup_2(1^\lambda)$$
$$\Pi_2 \leftarrow Prove(\omega_2, (y, l), x) \forall y \in L, l \in \P_1$$

□

Lemma 28.

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

◇

Proof. Straight forward reduction from membership hardness. □

Lemma 29.

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

◇

TODO 26: 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. [24] 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 $\perp$.

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

EVENT BAD: If we look at the view of 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,

$$¶[BAD] = 2^{-|¶_2|}$$

□

Lemma 30.

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

◇

Proof. If we look at the view of 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 31.

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

◇

Proof. The challenge ciphertext is independent of b. □

TODO 27: 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)

- $Setup(1^\lambda)$: Pick $x_1, x_2 \leftarrow\!\!\$\ \mathbb{Z}_q$ 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 40. *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 $\Pi = c_1^{x_1} c_2^{x_2} = g_1^{r_1 x_1} g_2^{r_1 x_1 + \alpha r_2 x_2}$.

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

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

□

- $Setup(1^\lambda)$:
  - Pick $x_3, x_4, x_5, x_6 \leftarrow\!\!\$\ \mathbb{Z}_q$ 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$\rightarrow$ $H = \{H_s\}$
- $Prove(\omega, (c_1, c_2, l), r)$
  - Compute $\beta = H_s(c_1, c_2, l) \in \mathbb{Z}_p$
  - Output $\Pi = h_2^r h_3^{r\beta}$
- $Verify(\tau, (c_1, c_2, l))$
  - Compute $\beta = H_s(c_1, c_2, l) \in \mathbb{Z}_q$
  - 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 41. *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:

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

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

- $\beta = H_s(c_1, c_2, l)$
- $\beta' = H_s(c_1', c_2', l')$

- 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 28: split definition from correctness

- $(pk, sk) \xleftarrow{\$} \mathcal{KG}en$, 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:
  - $r \xleftarrow{\$} \mathbb{Z}_q$
  - $\beta = H_s(c_1, c_2, c_3) = (g_1^r, g_2^r, h_1^r m)$
  - $Enc(pk, m) = (c_1, c_2, c_3, (h_2 h_3^{\beta})^r)$
- 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

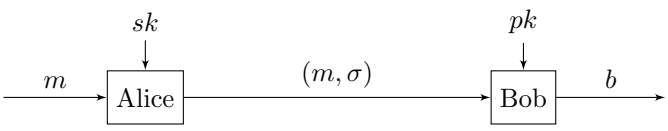


Figure 18.63: Asymmetric authentication

- 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:

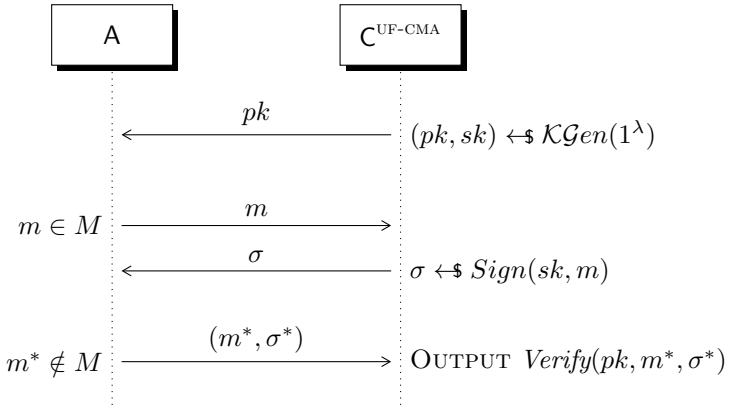


Figure 18.64: Unforgeable digital signatures

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:

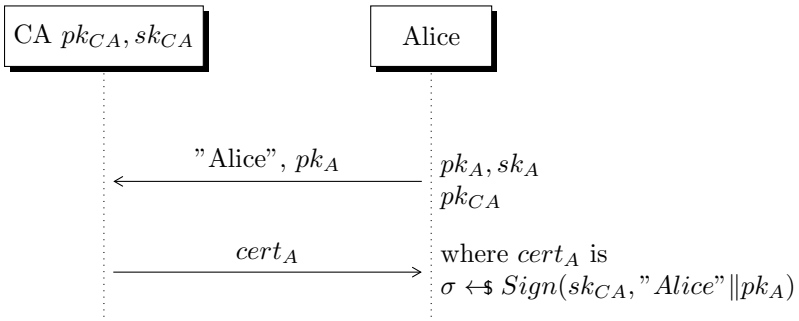


Figure 18.65

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.

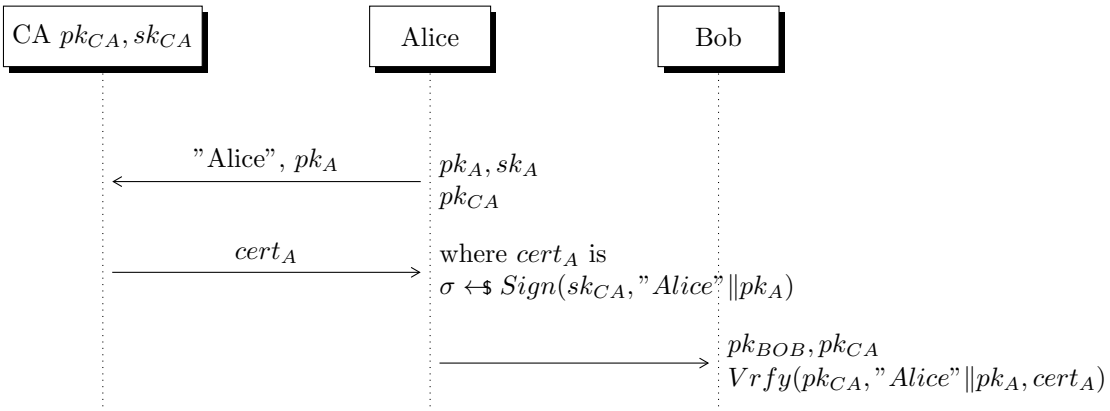


Figure 18.66

Theorem 42. *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 29: 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:

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

"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 $\hat{e}(g, g^c)$. □

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

- Generate some params $= (\mathcal{G}, \mathcal{G}_t, g, q, \hat{e}) \leftarrow BilGen(1^\lambda)$
- Pick $a \leftarrow\!\!\$\ \mathbb{Z}_q$ 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 $r \leftarrow \mathbb{Z}_q$ 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 43. *The Waters' signature scheme is* UFCMA ◇

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

TODO 30: Sequence is incomplete/incorrect, have to study it more...

TODO 31: 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 $x_0 \xleftarrow{\$} \{-kl, \ldots, 0\}, x_{1\text{—}k} \xleftarrow{\$} \{0, \ldots, l\}, y_{0\text{—}k} \xleftarrow{\$} \mathbb{Z}_q$

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:

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

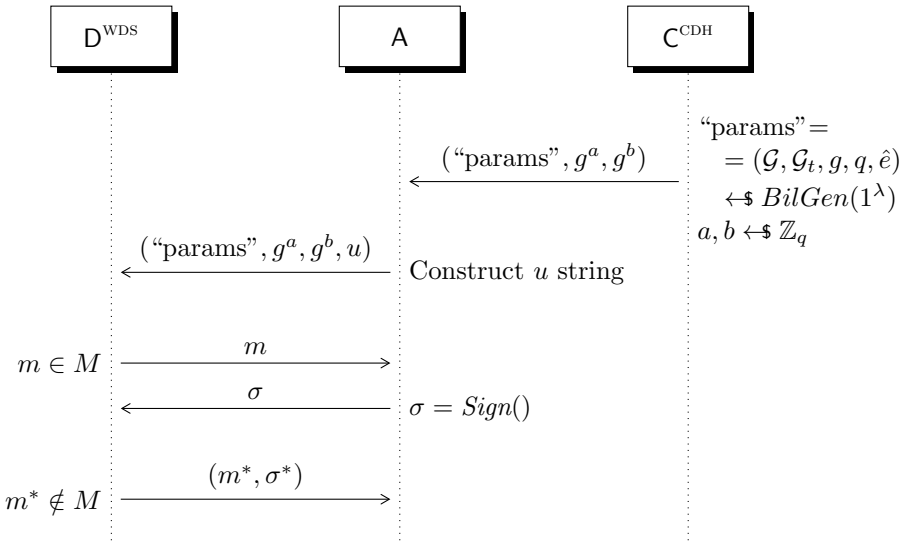


Figure 19.67: Reducing Waters' scheme to CDH

TODO 32: 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 $\bar{r} \leftarrow\!\!\$\ \mathbb{Z}_q \quad \bar{r} = r - a\beta^{-1}$

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

□

Lesson 20

20.1 Random Oracle Model (ROM)

The Random Oracle Model treats a given hash function H as a truly random function.

Recall: a truly random function R is defined to have a specific evaluation behaviour. Between subsequent evaluations:

- if the argument hasn't been submitted to the function beforehand, then a value is chosen UAR from the codomain, and assigned as the image of said argument in the function;
- otherwise, the function will return the image as assigned in the corresponding previous evaluation.

They do act, in fact, as truth tables[25].

20.2 Full domain hashing

Let (f, f^{-1}, Gen) be a TDP scheme over some domain $\mathcal{X}_{pk}$

Take RSA:

- $(m, pk, sk) \xleftarrow{\$} \text{Gen}_{\text{RSA}}(1^\lambda)$
- $f(pk, x) = x^{pk} \bmod n$
- $f^{-1}(sk, y) = y^{sk} \bmod n$

Build a similar asymmetric-authentication scheme as such:

- $(m, pk, sk) \xleftarrow{\$} \text{Gen}_{\text{RSA}}(1^\lambda)$
- $Sign_{sk,H}(x) : \sigma = f^{-1}(sk, H(m))$
- $Verify_{pk,H} : H(m) = f(pk, \sigma)$

Exercise: show RSA-sign is not secure without H. (Hint: The scheme becomes malleable)

Theorem: If the above scheme (full-domain hash) uses a TDP for f, f^{-1}, then it is (asymmetric)-UFCMAunder the random oracle model.

Proof: Reduce to TDP, program the random oracle.

Notes: this is a loose reduction

Some assumptions are made:

[25] Such tables are also aptly called *rainbow tables*.

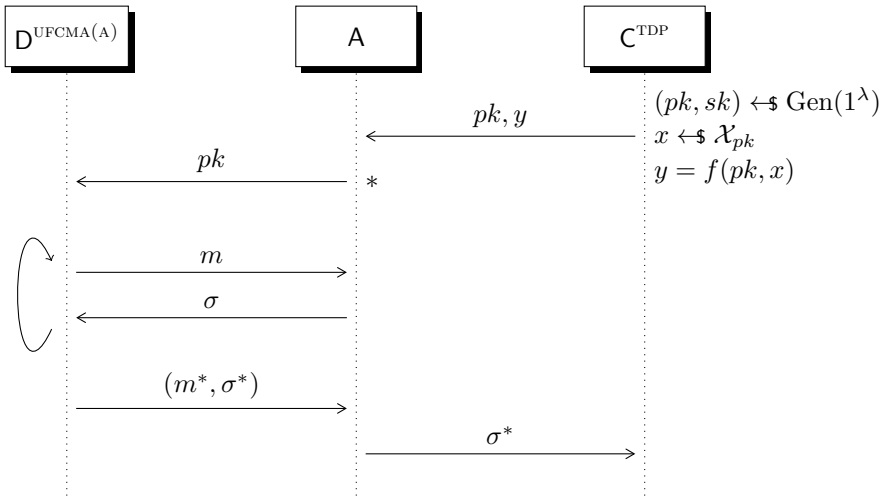


Figure 20.68

- The adversary makes the same number of RO queries as the number of signing queries done by the distinguisher (without loss of generality)
- The RO query must be done *before* the corresponding sign query, otherwise the advarsary cannot sign the messages, as specified by the scheme

The RO queries are actually an analogue of the definition of a random function, and it is the *programming* step of the oracle itself; then if the signing queries do not correspond to any RO query, abort the game.

20.3 ID Scheme

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

20.4.1 Fiat-Shamir scheme

Lesson 21

21.1 Full domain hashing

Lesson 22

22.1 Examples of ID schemes

Lesson 23

23.1 Bilinear DDH assumption

Lesson 24

24.1 CCA proof for ???