

Cryptography

Michele Laurenti

January 6, 2017

Abstract

Notes taken during the Cryptography lectures held by Daniele Venturi (<http://danieleventuri.altervista.org/crypto.shtml>) in fall 2016 at Sapienza.

Contents

1	Introduction	1
1.1	Perfect secrecy	1
1.2	Authentication	4
1.3	Randomness Extraction	6
2	Computational Cryptography	7
2.1	One Way Functions	7
2.2	Computational Indistinguishability	8
2.3	Pseudo Random Generators	9
2.4	Hard Core Predicates	10
2.5	Symmetric Key Encryption Schemes	12
2.6	Chosen Plaintext Attacks and Pseudo Random Functions	14
2.7	Computationally Secure MACs	17
	Acronyms	19

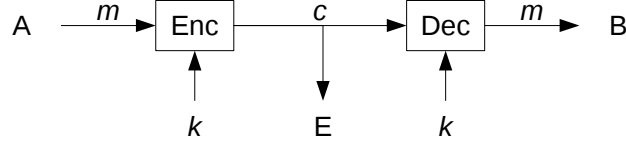


Figure 1: Message exchange between A and B using symmetric encryption. E is the eavesdropper.

1 Introduction

Solomon,
I'm concerned about security; I think, when we email each other,
we should use some sort of code.

Confidentiality is our goal. We want to encrypt and decrypt a (plaintext) message m , using a key, to obtain a cyphertext c . As per Kirkoff's principle, only the key is secret.

Our encryption schemes have the following syntax:

$$\Pi = (\text{Gen}, \text{Enc}, \text{Dec}).$$

A and B , the actors of our communication exchange (fig. 1), share k , the key, taken from some key space \mathcal{K} . The elements of our encryption scheme play the following roles:

1. Gen outputs a random key from the key space \mathcal{K} , and we write this as $k \leftarrow \$\text{Gen}$;
2. $\text{Enc} : \mathcal{K} \times \mathcal{M} \rightarrow \mathcal{C}$ is the encryption function, mapping a key and a message to a cyphertext;
3. $\text{Dec} : \mathcal{K} \times \mathcal{C} \rightarrow \mathcal{M}$ is the decryption function, mapping a key and a cyphertext to a message.

We expect an encryption scheme to be at least correct:

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

1.1 Perfect secrecy

Shannon defined “perfect secrecy”, *i.e.*, the fact that the cyphertext carries no information about the plaintext.

Definition 1. [Perfect secrecy] Let M be a random variable (RV) over \mathcal{M} , and K be a uniform distribution over \mathcal{K} .

(Enc, Dec) has perfect secrecy if

$$\forall M, \forall m \in \mathcal{M}, c \in \mathcal{C}. \Pr [M = m] = \Pr [M = m | C = c]$$

where $C = \text{Enc}(k, m)$ is a third RV. \diamond

We have equivalent definitions for perfect secrecy.

Theorem 1. *The following definitions are equivalent:*

1. *definition 1;*
2. *M and C are independent;*
3. *$\forall m, m' \in \mathcal{M}, \forall c \in \mathcal{C}$*

$$\Pr [\text{Enc}(k, m) = c] = \Pr [\text{Enc}(k, m') = c]$$

where k is a random key in \mathcal{K} chosen with uniform probability. \diamond

Proof of theorem 1. First, we show that 1 implies 2.

$$\begin{aligned} \Pr [M = m] &= \Pr [M = m | C = c] \\ &= \frac{\Pr [M = m \wedge C = c]}{\Pr [C = c]} && \text{(by Bayes)} \\ &\implies \\ \Pr [M = m] \Pr [C = c] &= \Pr [M = m \wedge C = c] \end{aligned}$$

which is the definition of independence.

Now we show that 2 implies 3. Fix $m \in \mathcal{M}$ and $c \in \mathcal{C}$.

$$\begin{aligned} \Pr [\text{Enc}(k, m) = c] &= \Pr [\text{Enc}(k, M) = c | M = m] && \text{(we fixed } m) \\ &= \Pr [C = c | M = m] && \text{(definition of the RV } C) \\ &= \Pr [C = c]. && \text{(by 2)} \end{aligned}$$

Since m is arbitrary, we can do the same for m' , and obtain

$$\Pr [\text{Enc}(k, m') = c] = \Pr [C = c]$$

which gives us 3.

Now we want to show that 3 implies 1. Take any $c \in \mathcal{C}$.

$$\begin{aligned}
\Pr[C = c] &= \sum_{m' \in \mathcal{M}} \Pr[C = c \wedge M = m'] \\
&= \sum_{m' \in \mathcal{M}} \Pr[C = c | M = m'] \Pr[M = m'] && \text{(by Bayes)} \\
&= \sum_{m' \in \mathcal{M}} \Pr[\text{Enc}(k, M) = c | M = m'] \Pr[M = m'] \\
&= \sum_{m' \in \mathcal{M}} \Pr[\text{Enc}(k, m') = c] \Pr[M = m'] \\
&= \Pr[\text{Enc}(k, m) = c] \underbrace{\sum_{m' \in \mathcal{M}} \Pr[M = m']}_1 \\
&\quad \text{(Enc is independent of } M, \text{ so we take it out)} \\
&= \Pr[\text{Enc}(k, M) = c | M = m] = \Pr[C = c | M = m].
\end{aligned}$$

We are left to show that $\Pr[M = m] = \Pr[M = m | C = c]$, but this is easy with Bayes. \square

One Time Pad

Now we'll see a perfect encryption scheme, the One Time Pad (OTP). The message space, the cyphertext space, and the key space are all the same, *i.e.*, $\mathcal{M} = \mathcal{K} = \mathcal{C} = \{0, 1\}^l$, with $l \in \mathbb{N}^+$.

Encryption and decryption use the xor operation:

- $\text{Enc}(k, m) = k \oplus m = c$;
- $\text{Dec}(k, c) = c \oplus k = (k \oplus m) \oplus k = m$.

Seeing that this is correct is immediate.

This can actually be done in any finite abelian group $(\mathbb{G}, +)$, where you just do $k + m$ to encode and $c - k$ to decode.

Theorem 2. *OTP is perfectly secure.* \diamond

Proof of theorem 2. Fix $m \in \mathcal{M}, c \in \mathcal{C}$, and choose a random key.

$$\Pr[\text{Enc}(k, m) = c] = \Pr[k = c - m] = \frac{1}{|\mathbb{G}|}.$$

This is true for any m , so we are done. \square

OTP has two problems:

1. the key is long (as long as the message);

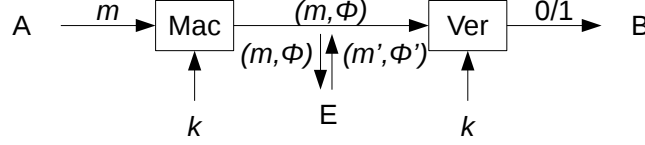


Figure 2: Message exchange between A and B using symmetric authentication. E is the eavesdropper.

2. we can't reuse the key:

$$\begin{aligned} c &= k + m \\ c' &= k + m' \end{aligned} \implies c - c' = m - m' \implies m' = m - (c - c').$$

Theorem 3. [Shannon, 1949] In any perfectly secure encryption scheme the size of the key space is at least as large as the size of the message space, i.e., $|\mathcal{K}| \geq |\mathcal{M}|$. \diamond

Proof of theorem 3. Assume, for the sake of contradiction, that $|\mathcal{K}| < |\mathcal{M}|$. Fix M to be the uniform distribution over \mathcal{M} , which we can do as perfect secrecy works for any distribution. Take a cyphertext $c \in \mathcal{C}$ such that $\Pr[C = c] > 0$, i.e., $\exists m, k$ such that $\text{Enc}(k, m) = c$.

Consider $\mathcal{M}' = \{\text{Dec}(k, c) : k \in \mathcal{K}\}$, the set of all messages decrypted from c using any key. Clearly, $|\mathcal{M}'| \leq |\mathcal{K}| < |\mathcal{M}|$, so $\exists m' \in \mathcal{M}$ such that $m' \notin \mathcal{M}'$. This means that

$$\Pr[M = m'] = \frac{1}{|\mathcal{M}|} \neq \Pr[M = m' | C = c] = 0$$

in contradiction with perfect secrecy. \square

In the rest of the course we will forget about perfect secrecy, and strive for computational security, i.e., bound the computational power of the adversary.

1.2 Authentication

The aim of authentication is to avoid tampering of E with the messages exchanged between A and B (fig. 2).

A Message Authentication Code (MAC) is defined as a tuple $\Pi = (\text{Gen}, \text{Mac}, \text{Vrfy})$, where:

- Gen, as usual, outputs a random key from some key space \mathcal{K} ;
- Mac : $\mathcal{K} \times \mathcal{M} \rightarrow \Phi$ maps a key and a message to an authenticator in some authenticator space Φ ;
- Vrfy : $\mathcal{K} \times \mathcal{M} \times \Phi \rightarrow \{0, 1\}$ verifies the authenticator.

As usual, we expect a MAC to be correct, *i.e.*,

$$\forall m \in \mathcal{M}, \forall k \in \mathcal{K}. \text{Vrfy}(k, m, \text{Mac}(k, m)) = 1.$$

If the Mac function is deterministic, then it must be that $\text{Vrfy}(k, m, \phi) = 1$ if and only if $\text{Mac}(k, m) = \phi$.

Security for MACs is that *forgery* must be hard: you can't come up with an authenticator for a message if you don't know the key.

Definition 2. [Information theoretic MAC] $(\text{Mac}, \text{Vrfy})$ has ε -statistical security if for all (possibly unbounded) adversary \mathcal{A} , for all $m \in \mathcal{M}$,

$$\Pr \left[\begin{array}{l} \text{Vrfy}(k, m', \phi') = 1 \wedge m' \neq m : \\ \begin{array}{l} k \leftarrow \text{sKeyGen}; \\ \phi = \text{Mac}(k, m); \\ (m', \phi') \leftarrow \mathcal{A}(m, \phi) \end{array} \end{array} \right] \leq \varepsilon$$

i.e., the adversary forges a (m', ϕ') that verifies with key k with low probability, even if it knows a valid pair (m, ϕ) . \diamond

As an exercise, prove that the above is impossible if $\varepsilon = 0$.

Information theoretic security is also called unconditional security. Later we'll see *conditional* security, based on computational assumptions.

Definition 3. [Pairwise independence] Given a family $\mathcal{H} = \{h_k : \mathcal{M} \rightarrow \Phi\}_{k \in \mathcal{K}}$ of functions, we say that \mathcal{H} is pairwise independent if for all distinct m, m' we have that $(h_k(m), h_k(m')) \in \Phi^2$ is uniform over the choice of $k \leftarrow \mathcal{K}$. \diamond

We show straight away a construction of a pairwise independent family of function.

Construction 1. [Pairwise independent function] Let p be a prime, the functions in our family \mathcal{H} are defined as

$$h_{a,b}(m) = am + b \pmod{p}$$

with $\mathcal{K} = \mathbb{Z}_p^2$, and with $\mathcal{M} = \Phi = \mathbb{Z}_p$. \diamond

Theorem 4. *Construction 1 is pairwise independent.* \diamond

Proof of theorem 4. For any m, m', ϕ, ϕ' , we want to find the value of

$$\Pr [am + b = \phi \wedge am' + b = \phi']$$

for $a, b \leftarrow \mathbb{Z}_p^2$. This is the same as

$$\Pr_{a,b} \left[\begin{pmatrix} m & 1 \\ m' & 1 \end{pmatrix} \begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} \phi \\ \phi' \end{pmatrix} \right] = \Pr_{a,b} \left[\begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} m & 1 \\ m' & 1 \end{pmatrix}^{-1} \begin{pmatrix} \phi \\ \phi' \end{pmatrix} \right] = \frac{1}{|\Phi|^2}.$$

This is true since $\begin{pmatrix} m & 1 \\ m' & 1 \end{pmatrix}^{-1} \begin{pmatrix} \phi \\ \phi' \end{pmatrix}$ is just a couple of (constant) numbers, so the probability of choosing (a, b) such that they equal the constant is just $\frac{1}{|\Phi|^2}$. \square

If h_k is part of a pairwise independent family of functions, then $\text{Mac}(k, m) = h_k(m)$, and $\text{Vrfy}(k, m, \phi)$ is simply computing $h_k(m)$ and comparing it with ϕ , *i.e.*,

$$\text{Vrfy}(k, m, \phi) = 1 \iff h_k(m) = \phi.$$

We now prove that this is an information theoretic MAC.

Theorem 5. *Any pairwise independent function is $\frac{1}{|\Phi|}$ -statistical secure.* \diamond

Proof of theorem 5. Take any two distinct m, m' , and two ϕ, ϕ' . We show that the probability that $\text{Mac}(k, m') = \phi'$ is exponentially small.

$$\Pr_k [\text{Mac}(k, m) = \phi] = \Pr_k [h_k(m) = \phi] = \frac{1}{|\Phi|}.$$

Now look at the joint probabilities:

$$\begin{aligned} \Pr_k [\text{Mac}(k, m) = \phi \wedge \text{Mac}(k, m') = \phi'] &= \Pr_k [h_k(m) = \phi \wedge h_k(m') = \phi'] \\ &\quad \text{(by definition)} \\ &= \frac{1}{|\Phi|^2} = \frac{1}{|\Phi|} \cdot \frac{1}{|\Phi|}. \end{aligned}$$

The last steps come from the fact that h_k is pairwise independent. To see that the construction is $\frac{1}{|\Phi|}$ -statistical secure:

$$\begin{aligned} \Pr_k [\text{Mac}(k, m') = \phi' | \text{Mac}(k, m) = \phi] &= \Pr_k [h_k(m') = \phi' | h_k(m) = \phi] \\ &= \frac{\Pr_k [h_k(m) = \phi \wedge h_k(m') = \phi']}{\Pr_k [h_k(m) = \phi]} \\ &= \frac{1}{|\Phi|}. \end{aligned}$$

□

Note that construction 1 ($h_k(m) = am + b \pmod{p}$) is insecure if the same key $k = (a, b)$ is used for two messages.

Theorem 6. *Any t -time $2^{-\lambda}$ -statistically secure MAC has key of size $(t + 1)\lambda$.* \diamond

1.3 Randomness Extraction

X is a random source (possibly not uniform). $\text{Ext}(X) = Y$ is a uniform RV.

First, let's see a construction for a binary RV. Let B be a RV such that $\Pr[B = 1] = p$ and $\Pr[B = 0] = 1 - p$, with $p \neq 1 - p$. We take two samples, B_1 and B_2 from B , and we want to obtain an unbiased RV B' .

1. Take two samples, $b_1 \leftarrow \mathcal{B}_1$ and $b_2 \leftarrow \mathcal{B}_2$;
2. if $b_1 = b_2$, sample again;
3. if $(b_1 = 1 \wedge b_2 = 0)$, output 1; if $(b_1 = 0 \wedge b_2 = 1)$, output 0.

It's easy to verify that B' is uniform:

$$\begin{aligned}\Pr[B' = 1] &= \Pr[B_1 = 1 \wedge B_2 = 0] = p(1 - p) \\ \Pr[B' = 0] &= \Pr[B_1 = 0 \wedge B_2 = 1] = (1 - p)p.\end{aligned}$$

How many trials do we have to make before outputting something? $2(1 - p)p$ is the probability that we output something. The probability that we don't output anything for n steps is thus $(1 - 2(1 - p)p)^n$.

Something is missing here on randomness extraction and min-entropy.

2 Computational Cryptography

Some details should be added on negligible functions.

To introduce computational cryptography we first have to define a computational model. We assume the adversary is efficient, *i.e.*, it is a Probabilistic Polynomial Time (PPT) adversary.

We want that the probability of success of the adversary is tiny, *i.e.*, negligible for some $\lambda \in \mathbb{N}$. A function $\varepsilon : \mathbb{N} \rightarrow \mathbb{R}$ is negligible if $\forall c > 0. \exists n_0$ such that $\forall n > n_0. \varepsilon(n) < n^{-c}$.

We rely on computational assumptions, *i.e.*, in tasks believed to be hard for any efficient adversary. In this setting we make conditional statements, *i.e.*, if a certain assumption holds then a certain crypto-scheme is secure.

2.1 One Way Functions

A simple computational assumption is the existence of One Way Functions (OWFs), *i.e.*, functions for which is hard to compute the inverse.

Definition 4. [One Way Function] A function $f : \{0, 1\}^* \rightarrow \{0, 1\}^*$ is a OWF if $f(x)$ can be computed in polynomial time for all x and for all PPT adversaries \mathcal{A} it holds that

$$\Pr[f(x') = y : x \leftarrow \mathcal{S}\{0, 1\}^*; y = f(x); x' \leftarrow \mathcal{A}(1^\lambda, y)] \leq \varepsilon(\lambda). \quad \diamond$$

The 1^λ given to the adversary \mathcal{A} is there to highlight the fact that \mathcal{A} is polynomial in the length of the input (λ).

Russel Impagliazzo proved that OWFs are equivalent to One Way Puzzles, *i.e.*, couples (Pgen, Pver) where $\text{Pgen}(1^\lambda) \rightarrow (y, x)$ gives us a puzzle (y) and a solution to it (x), while $\text{Pver}(x, y) \rightarrow 0/1$ verifies if x is a solution of y .

Another object of interest in this classification are average hard NP-puzzles, for which you can only get an instance, *i.e.*, $\text{Pgen}(1^\lambda) \rightarrow y$.

Impagliazzo says we live in one of five worlds:

1. Algorithmica, where $P = NP$;
2. Heuristica, where there are no average hard NP-puzzles, *i.e.*, problems without solution;
3. Pessiland, where you have average hard NP-puzzles;
4. Minicrypt, where you have OWF, one-way NP-puzzles, but no Public Key Cryptography (PKC);
5. Cryptomania, where you have both OWF and PKC.

We'll stay in Minicrypt for now.

OWF are hard to invert on average. Two examples:

- factoring the product of two large prime numbers;
- compute the discrete logarithm, *i.e.*, take a finite group (\mathbb{G}, \cdot) , and compute $y = g^x$ for some $g \in \mathbb{G}$. The find $x = \log_g(y)$. This is hard to compute in some groups, *e.g.*, \mathbb{Z}_p^* .

2.2 Computational Indistinguishability

Definition 5. [Distribution Ensemble] A distribution ensemble $\mathcal{X} = \{X_n\}_{n \in \mathbb{N}}$ is a sequence of distributions X_i over some space $\{0, 1\}^\lambda$. \diamond

Definition 6. [Computational Indistinguishability] Two distribution ensembles \mathcal{X}_λ and \mathcal{Y}_λ are computationally indistinguishable, written as $\mathcal{X}_\lambda \approx_c \mathcal{Y}_\lambda$, if for all PPT distinguishers \mathcal{D} it holds that

$$\left| \Pr[\mathcal{D}(\mathcal{X}_\lambda) = 1] - \Pr[\mathcal{D}(\mathcal{Y}_\lambda) = 1] \right| \leq \varepsilon(\lambda).$$

\diamond

Lemma 1. [Reduction] If $\mathcal{X} \approx_c \mathcal{Y}$, then for all PPT functions f , $f(\mathcal{X}) \approx_c f(\mathcal{Y})$. \diamond

Proof of lemma 1. Assume, for the sake of contradiction, that $\exists f$ such that $f(\mathcal{X}) \not\approx_c f(\mathcal{Y})$: then we can distinguish \mathcal{X} and \mathcal{Y} . Since $f(\mathcal{X}) \not\approx_c f(\mathcal{Y})$, then $\exists p = \text{poly}(\lambda), \mathcal{D}$ such that, for infinitely many λ s

$$\left| \Pr[\mathcal{D}(f(\mathcal{X}_\lambda)) = 1] - \Pr[\mathcal{D}(f(\mathcal{Y}_\lambda)) = 1] \right| \geq \frac{1}{p(\lambda)}.$$

\mathcal{D} distinguishes \mathcal{X}_λ and \mathcal{Y}_λ with non-negligible probability. Consider the following \mathcal{D}' , which is given

$$z = \begin{cases} x \leftarrow \mathcal{X}_\lambda; \\ y \leftarrow \mathcal{Y}_\lambda. \end{cases}$$

\mathcal{D}' runs $\mathcal{D}(f(z))$ and outputs whatever it outputs, and has the same probability of distinguishing \mathcal{X} and \mathcal{Y} of \mathcal{D} , in contradiction with the fact that $\mathcal{X} \approx_c \mathcal{Y}$. \square

Now we show that computational indistinguishability is transitive.

Lemma 2. [Hybrid Argument] Let $\mathcal{X} = \{X_\lambda\}$, $\mathcal{Y} = \{Y_\lambda\}$, $\mathcal{Z} = \{Z_\lambda\}$ be distribution ensembles. If $\mathcal{X}_\lambda \approx_c \mathcal{Y}_\lambda$ and $\mathcal{Y}_\lambda \approx_c \mathcal{Z}_\lambda$, then $\mathcal{X}_\lambda \approx_c \mathcal{Z}_\lambda$. \diamond

Proof of lemma 2. This follows from the triangular inequality.

$$\begin{aligned} \left| \Pr[\mathcal{D}(\mathcal{X}_\lambda) = 1] - \Pr[\mathcal{D}(\mathcal{Z}_\lambda) = 1] \right| &= \left| \Pr[\mathcal{D}(\mathcal{X}_\lambda) = 1] - \Pr[\mathcal{D}(\mathcal{Y}_\lambda) = 1] \right. \\ &\quad \left. + \Pr[\mathcal{D}(\mathcal{Y}_\lambda) = 1] - \Pr[\mathcal{D}(\mathcal{Z}_\lambda) = 1] \right| \\ &\leq \left| \Pr[\mathcal{D}(\mathcal{X}_\lambda) = 1] - \Pr[\mathcal{D}(\mathcal{Y}_\lambda) = 1] \right| \\ &\quad + \left| \Pr[\mathcal{D}(\mathcal{Y}_\lambda) = 1] - \Pr[\mathcal{D}(\mathcal{Z}_\lambda) = 1] \right| \\ &\leq 2\varepsilon(\lambda). \end{aligned} \quad (\text{negligible})$$

\square

We often prove $\mathcal{X} \approx_c \mathcal{Y}$ by defining a sequence $\mathcal{H}_0, \mathcal{H}_1, \dots, \mathcal{H}_t$ of distributions ensembles such that $\mathcal{H}_0 \equiv \mathcal{X}$ and $\mathcal{H}_t \equiv \mathcal{Y}$, and that for all i , $\mathcal{H}_i \approx_c \mathcal{H}_{i+1}$.

2.3 Pseudo Random Generators

Let's see our first cryptographic primitive. Pseudo Random Generators (PRGs) take in input a random seed and generate pseudo random sequences with some stretch, *i.e.*, output longer than input, and indistinguishable from a true random sequence.

Definition 7. [Pseudo Random Generator] A function $\mathcal{G} : \{0, 1\}^\lambda \rightarrow \{0, 1\}^{\lambda+l(\lambda)}$ is a PRG if and only if

1. \mathcal{G} is computable in polynomial time;

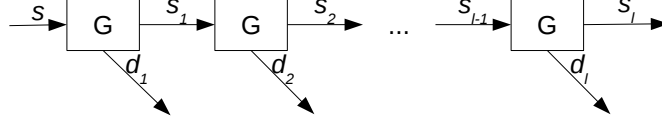


Figure 3: Extending a PRG with 1 bit stretch to a PRG with l bit stretch.

2. $|\mathcal{G}(s)| = \lambda + l(\lambda)$ for all $s \in \{0, 1\}^\lambda$;
3. $\mathcal{G}(\mathcal{U}_\lambda) \approx_c \mathcal{U}_{\lambda+l(\lambda)}$.

◇

Theorem 7. *If \exists PRG with 1 bit of stretch, then \exists PRG with $l(\lambda)$ bits of stretch, with $l(\lambda) = \text{poly}(\lambda)$.*

◇

Proof of theorem 7. We'll prove this just for some fixed constant $l(\lambda) = l \in \mathbb{N}$.

First, let's look at the construction (fig. 3). We replicate our PRG \mathcal{G} with 1 bit stretch l times. The PRG \mathcal{G}^l that we define takes in input $s \in \{0, 1\}^\lambda$, computes $(s_1, b_1) = \mathcal{G}(s)$, where $s_1 \in \{0, 1\}^l$ and $b_1 \in \{0, 1\}$, outputs b_1 and feeds s_1 to the second copy of PRG \mathcal{G} , and so on until the l -th PRG.

To show that our construction is a PRG, we define l hybrids, with $\mathcal{H}_0^\lambda \equiv \mathcal{G}^l(\mathcal{U}_\lambda)$, where $\mathcal{G}^l : \{0, 1\}^\lambda \rightarrow \{0, 1\}^{\lambda+l}$ is our proposed construction, and \mathcal{H}_i^λ takes $b_1, \dots, b_i \leftarrow \mathcal{S}\{0, 1\}$, $s_i \leftarrow \mathcal{S}\{0, 1\}^\lambda$, and outputs (b_1, \dots, b_i, s_l) , where $s_l \in \{0, 1\}^{\lambda+l-i}$ is $s_l = \mathcal{G}^{l-i}(s_i)$, i.e., the output of our construction restricted to $l-i$ units.

\mathcal{H}_l^λ takes $b_1, \dots, b_l \leftarrow \mathcal{S}\{0, 1\}$ and $s_l \leftarrow \mathcal{S}\{0, 1\}^l$ and outputs (b_1, \dots, b_l, s_l) directly.

We need to show that $\mathcal{H}_i^\lambda \approx_c \mathcal{H}_{i+1}^\lambda$. To do so, fix some i . The only difference between the two hybrids is that s_{i+1}, b_{i+1} are pseudo random in \mathcal{H}_i^λ , and are truly random in $\mathcal{H}_{i+1}^\lambda$. All bits before them are truly random, all bits after are pseudo random.

Assume these two hybrids are distinguishable, then we can break the PRG. Consider the PPT function f_i defined by $f(s_{i+1}, b_{i+1}) = (b_1, \dots, b_l, s_l)$ such that $b_1, \dots, b_i \leftarrow \mathcal{S}\{0, 1\}$ and, for all $j \in [i+1, l]$ $(b_j, s_j) = \mathcal{G}(s_{j-1})$.

By the security of PRGs we have that $\mathcal{G}(\mathcal{U}_\lambda) \approx_c \mathcal{U}_{\lambda+1}$. By reduction, we also have that $f(\mathcal{G}(\mathcal{U}_\lambda)) \approx_c f(\mathcal{U}_{\lambda+1})$. Thus, $\mathcal{H}_i^\lambda \approx_c \mathcal{H}_{i+1}^\lambda$. □

2.4 Hard Core Predicates

Definition 8. [Hard Core Predicate - I] A polynomial time function $h : \{0, 1\}^n \rightarrow \{0, 1\}$ is *hard core* for $f : \{0, 1\}^n \rightarrow \{0, 1\}^n$ if for all PPT adversaries \mathcal{A}

$$\Pr[\mathcal{A}(f(x)) = h(x) : x \leftarrow \mathcal{S}\{0, 1\}^n] \leq \frac{1}{2} + \varepsilon(\lambda).$$

◇

The $\frac{1}{2}$ in the upper bound tells us that the adversary can't do better than guessing.

Definition 9. [Hard Core Predicate - II] A polynomial time function $h : \{0, 1\}^n \rightarrow \{0, 1\}$ is *hard core* for $f : \{0, 1\}^n \rightarrow \{0, 1\}^n$ if for all PPT adversaries \mathcal{A}

$$\left| \Pr \left[\begin{array}{c} \mathcal{A}(f(x), h(x)) = 1 : \\ x \leftarrow \{0, 1\}^n \end{array} \right] - \Pr \left[\begin{array}{c} \mathcal{A}(f(x), b) = 1 : \\ x \leftarrow \{0, 1\}^n; \\ b \leftarrow \{0, 1\} \end{array} \right] \right| \leq \varepsilon(\lambda).$$

◇

Theorem 8. *Definition 8 and definition 9 are equivalent.*

◇

Proof of this theorem is left as exercise.

Luckily for us, every OWF has a Hard Core Predicate (HCP). There isn't a single HCP h for all OWFs f . Suppose \exists such h , then take f and let $f'(x) = h(x) || f(x)$. Then, if $f'(x) = y || b$ for some x , it will always be that $h(x) = b$.

But, given a OWF, we can create a new OWF for which h is hard core.

Theorem 9. [Goldreich-Levin (GL), 1983] Let $f : \{0, 1\}^n \rightarrow \{0, 1\}^n$ be a OWF, and define $g(x, r) = f(x) || r$ for $r \leftarrow \{0, 1\}^n$. Then g is a OWF, and

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

is *hardcore* for g .

◇

Definition 10. [One Way Permutation] We say that $f : \{0, 1\}^n \rightarrow \{0, 1\}^n$ is a One Way Permutation (OWP) if f is a OWF, $\forall x. |x| = |f(x)|$, and for all distinct $x, x'. f(x) \neq f(x')$.

◇

Corollary 1. Let f be a OWP, and consider $g : \{0, 1\}^n \rightarrow \{0, 1\}^n$ from the GL theorem. Then $\mathcal{G}(s) = (g(s), h(s))$ is a PRG with stretch 1.

◇

Proof of corollary 1.

$$\begin{aligned} \mathcal{G}(\mathcal{U}_{2n}) &= (g(x, r), h(x, r)) \\ &= (f(x) || r, \langle x, r \rangle) \\ &\approx_c (f(x) || r, b) \\ &\approx_c \mathcal{U}_{2n+1}. \end{aligned} \tag{GL}$$

□

UNCLEAR

Assume instead f is a OWF, and that is 1-to-1 (injective). Consider $\mathcal{X} = g^m(\bar{x}) = (g(x_1), h(x_1), \dots, g(x_m), h(x_m))$, where $x_1, \dots, x_m \in \{0, 1\}^n$ (i.e., $\bar{x} \in \{0, 1\}^{nm}$). You can construct a PRG from a OWF as shown by H.I.L.L.

Fact 1. \mathcal{X} is indistinguishable from \mathcal{X}' such that $\mathcal{H}_\infty(\mathcal{X}') \geq k = n \cdot m + m$, since f is injective. \diamond

Now $\mathcal{G}(s, \bar{x}) = (s, \text{Ext}(s, g^m(\bar{x})))$ where $\text{Ext} : \{0, 1\}^d \times \{0, 1\}^{nm} \rightarrow \{0, 1\}^l$, and $l = nm + 1$. This works for $m = \omega(\log(n))$. You get extraction error $\varepsilon \approx 2^{-m}$.

2.5 Symmetric Key Encryption Schemes

We call $\Pi = (\text{Gen}, \text{Enc}, \text{Dec})$ a Symmetric Key Encryption (SKE) scheme.

- Gen outputs a key $k \leftarrow \mathcal{K}$;
- $\text{Enc}(k, m) = c$ for some $m \in \mathcal{M}$, $c \in \mathcal{C}$;
- $\text{Dec}(k, c) = m$.

As usual, we want Π to be correct.

We want to introduce computational security: a bounded adversary can not gain information on the message given the cyphertext.

Definition 11. [One time security] A SKE scheme $\Pi = (\text{Gen}, \text{Enc}, \text{Dec})$ has one time computational security if for all PPT adversaries $\mathcal{A} \ni$ a negligible function ε such that

$$\left| \Pr [\mathcal{G}_{\Pi, \mathcal{A}}^{\text{one time}}(\lambda, 0) = 1] - \Pr [\mathcal{G}_{\Pi, \mathcal{A}}^{\text{one time}}(\lambda, 1) = 1] \right| \leq \varepsilon(\lambda)$$

where $\mathcal{G}_{\Pi, \mathcal{A}}^{\text{one time}}(\lambda, b)$ is the following “game” (or experiment):

1. pick $k \leftarrow \mathcal{K}$;
2. \mathcal{A} outputs two messages $(m_0, m_1) \leftarrow \mathcal{A}(1^\lambda)$ where $m_0, m_1 \in \mathcal{M}$ and $|m_0| = |m_1|$;
3. $\xrightarrow{\text{Enc}} \text{Enc}(k, m_b)$ with b input of the experiment;
4. output $b' \leftarrow \mathcal{A}(1^\lambda, c)$, i.e., the adversary tries to guess which message was encrypted. \diamond

Let's look at a construction.

Construction 2. [SKE scheme from PRG] Let $\mathcal{G} : \{0,1\}^n \rightarrow \{0,1\}^l$ be a PRG. Set $\mathcal{K} = \{0,1\}^n$, and $\mathcal{M} = \mathcal{C} = \{0,1\}^l$. Define $\text{Enc}(k, m) = \mathcal{G}(k) \oplus m$ and $\text{Dec}(k, c) = \mathcal{G}(k) \oplus c$. \diamond

Theorem 10. *If \mathcal{G} is a PRG, the SKE in construction 2 is one-time computationally secure.* \diamond

Proof of theorem 10. Consider the following experiments:

- $\mathcal{H}_0(\lambda, b)$ is like $\mathcal{G}_{\Pi, \mathcal{A}}^{\text{one time}}$:
 1. $k \leftarrow \mathcal{K}$;
 2. $(m_0, m_1) \leftarrow \mathcal{A}(1^\lambda)$;
 3. $c = \mathcal{G}(k) \oplus m_b$;
 4. $b' \leftarrow \mathcal{A}(1^\lambda, c)$.
- $\mathcal{H}_1(\lambda, b)$ replaces \mathcal{G} with something truly random:
 1. $(m_0, m_1) \leftarrow \mathcal{A}(1^\lambda)$;
 2. $r \leftarrow \mathcal{C}$;
 3. $c = r \oplus m_b$, basically like One Time Pad (OTP);
 4. $b' \leftarrow \mathcal{A}(1^\lambda, c)$.
- $\mathcal{H}_2(\lambda)$ is just randomness:
 1. $(m_0, m_1) \leftarrow \mathcal{A}(1^\lambda)$;
 2. $c \leftarrow \mathcal{C}$;
 3. $b' \leftarrow \mathcal{A}(1^\lambda, c)$.

First, we show that $\mathcal{H}_0(\lambda, b) \approx_c \mathcal{H}_1(\lambda, b)$, for $b \in \{0,1\}$. Fix some value for b , and assume exists a PPT distinguisher \mathcal{D} between $\mathcal{H}_0(\lambda, b)$ and $\mathcal{H}_1(\lambda, b)$: we then can construct a distinguisher \mathcal{D}' for the PRG.

\mathcal{D}' , on input z , which can be either $\mathcal{G}(k)$ for some $k \leftarrow \mathcal{K}$, or directly $z \leftarrow \mathcal{C}$, does the following:

- get $(m_0, m_1) \leftarrow \mathcal{D}(1^\lambda)$;
- feed $z \oplus m_b$ to \mathcal{D} ;
- output the result of \mathcal{D} .

Now, we show that $\mathcal{H}_1(\lambda, b) \approx_c \mathcal{H}_2(\lambda, b)$, for $b \in \{0,1\}$. By perfect secrecy of OTP we have that $(m_0 \oplus r) \approx z \approx (m_1 \oplus r)$, so $\mathcal{H}_1(\lambda, 0) \approx_c \mathcal{H}_2(\lambda) \approx_c \mathcal{H}_1(\lambda, 1)$. \square

Corollary 2. *One-time computationally secure SKE schemes are in Minicrypt.* \diamond

This scheme is not secure if the adversary knows a (m_1, c_1) pair, and we reuse the key. Take any m, c , then $c \oplus c_1 = m \oplus m_1$, and you can find m . This is called a Chosen Plaintext Attack (CPA), something we will defined shortly using a Pseudo Random Function (PRF).

2.6 Chosen Plaintext Attacks and Pseudo Random Functions

Definition 12. [Pseudo Random Function] Let $\mathcal{F} = \{F_k : \{0, 1\}^n \rightarrow \{0, 1\}^l\}$ be a family of functions, for $k \in \{0, 1\}^\lambda$. Consider the following two experiments:

- $\mathcal{G}_{\mathcal{F}, \mathcal{A}}^{\text{real}}(\lambda)$, defined as:
 1. $k \leftarrow \mathcal{S}\{0, 1\}^\lambda$;
 2. $b' \leftarrow \mathcal{A}^{F_k(\cdot)}(1^\lambda)$, where \mathcal{A} can query an oracle for values of $F_k(\cdot)$, without knowing k .
- $\mathcal{G}_{\mathcal{F}, \mathcal{A}}^{\text{rand}}(\lambda)$, defined as:
 1. $R \leftarrow \mathcal{SR}(n \rightarrow l)$, *i.e.*, a function R is chosen at random from all functions from $\{0, 1\}^n$ to $\{0, 1\}^l$;
 2. $b' \leftarrow \mathcal{A}^{R(\cdot)}(1^\lambda)$, where \mathcal{A} can query an oracle for values of $R(\cdot)$.

The family \mathcal{F} of functions is a PRF family if for all PPT adversaries $\mathcal{A} \exists$ a negligible function ε such that

$$\left| \Pr [\mathcal{G}_{\mathcal{F}, \mathcal{A}}^{\text{real}}(\lambda) = 1] - \Pr [\mathcal{G}_{\mathcal{F}, \mathcal{A}}^{\text{rand}}(\lambda) = 1] \right| \leq \varepsilon(\lambda). \quad \diamond$$

To introduce CPAs and CPA-secure SKE schemes, we first introduce the game of CPA. As usual, a SKE scheme is a tuple $\Pi = (\text{Gen}, \text{Enc}, \text{Dec})$.

Definition 13. [CPA-secure SKE scheme] Let $\Pi = (\text{Gen}, \text{Enc}, \text{Dec})$ be a SKE scheme, and consider the game $\mathcal{G}_{\Pi, \mathcal{A}}^{\text{cpa}}(\lambda, b)$, defined as:

1. $k \leftarrow \mathcal{S}\{0, 1\}^\lambda$;
2. $(m_0, m_1) \leftarrow \mathcal{A}^{\text{Enc}(k, \cdot)}(1^\lambda)$. \mathcal{A} is given access to an oracle for $\text{Enc}(k, \cdot)$, so she knows some (m, c) couples, with $c = \text{Enc}(k, m)$;
3. $c \leftarrow \text{Enc}(k, m_b)$;
4. $b' \leftarrow \mathcal{A}^{\text{Enc}(k, \cdot)}(1^\lambda, c)$.

Π is CPA-secure if for all PPT adversaries \mathcal{A}

$$\mathcal{G}_{\Pi, \mathcal{A}}^{\text{cpa}}(\lambda, 0) \approx_c \mathcal{G}_{\Pi, \mathcal{A}}^{\text{cpa}}(\lambda, 1). \quad \diamond$$

Deterministic schemes cannot achieve this, *i.e.*, when Enc is deterministic the adversary could cipher m_0 and then compare c to $\text{Enc}(k, m_0)$, and output 0 if and only if $c = \text{Enc}(k, m_0)$.

Let's construct a CPA-secure SKE scheme using PRFs.

Construction 3. [SKE using PRFs] Let \mathcal{F} be a PRF, we define the following SKE scheme $\Pi = (\text{Gen}, \text{Enc}, \text{Dec})$:

- Gen takes $k \leftarrow \mathcal{S}\{0, 1\}^\lambda$;
- $\text{Enc}(k, m) = (r, F_k(r) \oplus m)$, with $r \leftarrow \mathcal{S}\{0, 1\}^n$. Note that, since $F_k : \{0, 1\}^n \rightarrow \{0, 1\}^l$, we have that $\mathcal{M} = \{0, 1\}^l$ and $\mathcal{C} = \{0, 1\}^{n+l}$;
- $\text{Dec}(k, (c_1, c_2)) = F_k(c_1) \oplus c_2$. \diamond

Our construction is both one time computationally secure, and secure against CPAs.

Theorem 11. *If \mathcal{F} is a PRF, construction 3 is CPA-secure.* \diamond

Proof of theorem 11. First, we define the experiment $\mathcal{H}_0(\lambda, b) \equiv \mathcal{G}_{\Pi, \mathcal{A}}^{\text{cpa}}(\lambda, b)$ as follows:

1. $k \leftarrow \mathcal{S}\{0, 1\}^\lambda$;
2. $(m_0, m_1) \leftarrow \mathcal{A}^{\text{Enc}(k, \cdot)}(1^\lambda)$;
3. $c^* \leftarrow (r^*, F_k(r^*) \oplus m_b)$, where $r^* \leftarrow \mathcal{S}\{0, 1\}^n$;
4. output $b' \leftarrow \mathcal{A}^{\text{Enc}(k, \cdot)}(1^\lambda, c^*)$.

Note that in the CPA game the adversary has access to an encryption oracle using the chosen key.

Now, for the first hybrid $\mathcal{H}_1(\lambda, b)$, where we sample a random function R in place of F_k :

1. $R \leftarrow \mathcal{SR}(n \rightarrow l)$;
2. $(m_0, m_1) \leftarrow \mathcal{A}^{\text{Enc}(R, \cdot)}(1^\lambda)$, where now $\text{Enc}(R, m) = (r, R(r) \oplus m)$ for some random r ;
3. $c^* \leftarrow (r^*, R(r^*) \oplus m_b)$, where $r^* \leftarrow \mathcal{S}\{0, 1\}^n$;
4. output $b' \leftarrow \mathcal{A}^{\text{Enc}(R, \cdot)}(1^\lambda, c^*)$.

Our first claim is that $\mathcal{H}_0(\lambda, b) \approx_c \mathcal{H}_1(\lambda, b)$ for $b \in \{0, 1\}$. As usual, we assume that exists an adversary \mathcal{A} which can distinguish the experiments, *i.e.*, that can distinguish the oracles, and use \mathcal{A} to create \mathcal{A}_{PRF} that breaks the PRF.

\mathcal{A}_{PRF} has access to some oracle $O(\cdot)$, with is one of two possibilities:

$$O(x) = \begin{cases} F_k(x) & \text{for } k \leftarrow \mathcal{S}\{0, 1\}^\lambda \\ R(x) & \text{for } R \leftarrow \mathcal{SR}(n \rightarrow l). \end{cases}$$

\mathcal{A} gives \mathcal{A}_{PRF} some message m . \mathcal{A}_{PRF} picks $r \leftarrow \mathcal{S}\{0, 1\}^n$, and queries $O(r)$ to get $z \in \{0, 1\}^l$. Then it gives $(r, z \oplus m)$ to \mathcal{A} . This is repeated as long as \mathcal{A} asks for encryption queries.

Then \mathcal{A} gives to \mathcal{A}_{PRF} (m_0, m_1) , which repeats the same procedure using m_0 as a message (to distinguish $\mathcal{H}_0(\lambda, 0)$ from $\mathcal{H}_1(\lambda, 0)$) to compute c^* . \mathcal{A} ,

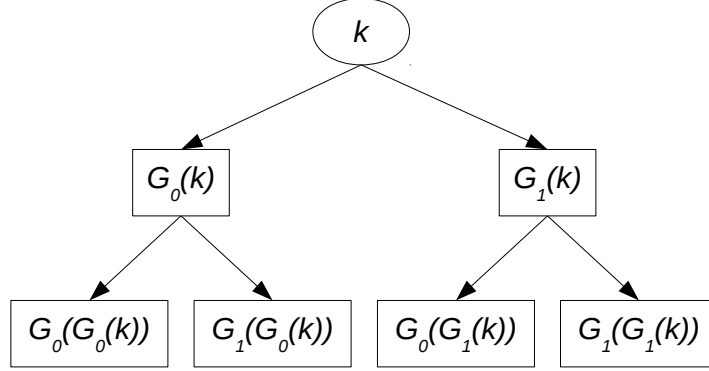


Figure 4: First two levels of a GGM tree.

after receiving c^* , asks some more encryption queries, and then outputs b' . If $b' = 1$, \mathcal{A}_{PRF} says $R(\cdot)$, otherwise it says $F_k(\cdot)$.

Now for the third experiment, $\mathcal{H}_2(\lambda)$, which uses $\text{Enc}(m) = (r_1, r_2)$ with $(r_1, r_2) \leftarrow \{0, 1\}^{n+l}$, i.e., it outputs just randomness.

Our second claim is that $\mathcal{H}_1(\lambda, b) \approx_c \mathcal{H}_2(\lambda)$ for $b \in \{0, 1\}$. To see this, note that \mathcal{H}_1 and \mathcal{H}_2 are identical as long as collisions don't happen when choosing the r s. It suffices for us to show that collisions happen with small probability.

Call $E_{i,j}$ the event “random r_i collides with random r_j ”. The event of a collision is thus $E = \bigvee_{i,j} E_{i,j}$, and its probability can be upper bounded as follows:

$$\Pr[E] = \sum_{i,j} \Pr[E_{i,j}] = \sum_{i,j} \text{Coll}(\mathcal{U}_n) \leq \binom{q}{2} 2^{-n} \leq \frac{q^2}{2^n}$$

where q is the (polynomial) number of queries that the adversary does, and $\text{Coll}(\mathcal{U}_n)$ is the probability of a collision when using a uniform distribution, which is 2^{-n} . \square

Theorem 12. [GGM, 1982] PRFs can be constructed from PRGs. \diamond

Corollary 3. PRFs are in Minicrypt. \diamond

Construction 4. [GGM tree] Assume we have a length doubling PRG $\mathcal{G} : \{0, 1\}^\lambda \rightarrow \{0, 1\}^{2\lambda}$. We say that $\mathcal{G}(x) \triangleq (\mathcal{G}_0(x), \mathcal{G}_1(x))$ to distinguish the first λ bits from the second λ bits.

Now, to build the PRF we construct a Goldreich-Goldwasser-Micali (GGM) tree (fig. 4) starting with a key $k \in \{0, 1\}^\lambda$. On input $x = (x_1, \dots, x_n) \in \{0, 1\}^n$, with n being the height of the tree, the PRF picks a path in the tree:

$$F_k(x) = \mathcal{G}_{x_n}(\dots \mathcal{G}_{x_1}(k) \dots).$$

\diamond

Lemma 3. Let $\mathcal{G} : \{0, 1\}^\lambda \rightarrow \{0, 1\}^{2\lambda}$ be a PRG. Then for all $t(\lambda) = \text{poly}(\lambda)$ we have that

$$(\mathcal{G}(k_1), \dots, \mathcal{G}(k_t)) \approx_c \underbrace{(\mathcal{U}_{2\lambda}, \dots, \mathcal{U}_{2\lambda})}_{t \text{ times}}. \quad \diamond$$

Proof of lemma 3. We define t hybrids, where $\mathcal{H}_i(\lambda)$ is defined as

$$\mathcal{H}_i(\lambda) = (\mathcal{G}(k_1), \dots, \mathcal{G}(k_{t-i}), \underbrace{\mathcal{U}_{2\lambda}, \dots, \mathcal{U}_{2\lambda}}_{i \text{ times}})$$

thus $\mathcal{H}_0(\lambda) = (\mathcal{G}(k_1), \dots, \mathcal{G}(k_t))$ and $\mathcal{H}_t(\lambda) = (\mathcal{U}_{2\lambda}, \dots, \mathcal{U}_{2\lambda})$. To prove that $\mathcal{H}_1(\lambda) \approx_c \mathcal{H}_t(\lambda)$, we show that for any i it holds that $\mathcal{H}_i(\lambda) \approx_c \mathcal{H}_{i+1}(\lambda)$. This relies on the fact that $\mathcal{G}(k_{t-i}) \approx_c \mathcal{U}_{2\lambda}$: assume that exists a distinguisher \mathcal{D} for $\mathcal{H}_i(\lambda)$ and $\mathcal{H}_{i+1}(\lambda)$, we then break the PRG.

We build \mathcal{D}' , which takes in input some z from either $\mathcal{G}(k_{t-i})$ or $\mathcal{U}_{2\lambda}$. \mathcal{D}' takes $k_1, \dots, k_{t-(i+1)} \leftarrow \{0, 1\}^\lambda$, and feeds $(\mathcal{G}(k_1), \dots, \mathcal{G}(k_{t-(i+1)}), z, \mathcal{U}_{2\lambda}, \dots, \mathcal{U}_{2\lambda})$ to \mathcal{D} , and returns whatever it returns. \square

Proof that construction 4 is a PRF. We'll define a series of hybrids to show that the GGM tree is a PRF. $\mathcal{H}_0(\lambda) \equiv$ our GGM tree.

$\mathcal{H}_i(\lambda)$, for $i \in [1, n]$, will replace the tree up to depth i with a true random function. $\mathcal{H}_i(\lambda)$ initially has two empty arrays T_1 and T_2 . On input $x \in \{0, 1\}^n$, it checks if $\bar{x} = (x_1, \dots, x_i) \in T_1$. If not, $\mathcal{H}_i(\lambda)$ picks $k_{\bar{x}} \leftarrow \{0, 1\}^\lambda$ and adds \bar{x} to T_1 and $k_{\bar{x}}$ to T_2 . If $\bar{x} \in T_1$, it just retrieves $k_{\bar{x}}$ from T_2 . Then $\mathcal{H}_i(\lambda)$ outputs the following:

$$\mathcal{G}_{x_n}(\mathcal{G}_{x_{n-1}}(\dots \mathcal{G}_{x_{i+1}}(k_{\bar{x}}) \dots)).$$

If $i = 0$ we have that $\bar{x} = \perp$ and that $k_{\perp} \leftarrow \{0, 1\}^\lambda$, so $\mathcal{H}_0(\lambda) \equiv$ the GGM tree. On the other hand, if $i = n$, each input x leads to a random output, so $\mathcal{H}_n(\lambda)$ is just a true random function.

Assume now that exists an adversary \mathcal{A} capable of telling apart $\mathcal{H}_i(\lambda)$ from $\mathcal{H}_{i+1}(\lambda)$, we could break the PRG. \square

2.7 Computationally Secure MACs

A computationally secure Message Authentication Code (MAC) should be hard to forge, even if you see polynomially many authenticated messages.

Definition 14. [UFCMA MAC] Let $\Pi = (\text{Gen}, \text{Mac}, \text{Vrfy})$ be a MAC, and consider the game $\mathcal{G}_{\Pi, \mathcal{A}}^{\text{ufcma}}(\lambda)$ defined as:

1. pick $k \leftarrow \{0, 1\}^\lambda$;
2. $(m^*, \phi^*) \leftarrow \mathcal{A}^{\text{Mac}(k, \cdot)}(1^\lambda)$, where the adversary can query an authentication oracle;
3. output 1 if $\text{Vrfy}(k, (m^*, \phi^*)) = 1$ and m^* is “fresh”, *i.e.*, it was never queried to Mac.

We say that Π is Unforgeable Chosen Message Attack (UFCMA) if for all PPT adversaries \mathcal{A} it holds that

$$\Pr [\mathcal{G}_{\Pi, \mathcal{A}}^{\text{ufcma}}(\lambda) = 1] \leq \text{negl}(\lambda) .$$

◇

As a matter of fact, any PRF is a MAC.

Acronyms

DL	Discrete Log
CCA1	Chosen Cyphertext Attack 1
CCA2	Chosen Cyphertext Attack 2
CDH	Computational Diffie-Hellman
CPA	Chosen Plaintext Attack
DDH	Decisional Diffie-Hellman
GGM	Goldreich-Goldwasser-Micali
GL	Goldreich-Levin
HCP	Hard Core Predicate
MAC	Message Authentication Code
OTP	One Time Pad
OWF	One Way Function
OWP	One Way Permutation
PKC	Public Key Cryptography
PKE	Public Key Encryption
PPT	Probabilistic Polynomial Time
PRF	Pseudo Random Function
PRG	Pseudo Random Generator
RV	random variable
SKE	Symmetric Key Encryption
UFCMA	Unforgeable Chosen Message Attack