

# Cryptography

Michele Laurenti

January 14, 2017

## Abstract

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

## Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Perfect secrecy . . . . .	1
1.2	Authentication . . . . .	4
1.3	Randomness Extraction . . . . .	6
<b>2</b>	<b>Computational Cryptography</b>	<b>7</b>
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
2.8	Domain Extension . . . . .	18
2.9	Chosen Cyphertext Attacks and Authenticated Encryption . . .	22
2.10	Pseudo Random Permutations . . . . .	24
2.11	Collision Resistant Hash Functions . . . . .	27
2.12	Claw-Free Permutations . . . . .	29
2.13	Waters Signature Scheme . . . . .	30
	<b>Acronyms</b>	<b>34</b>

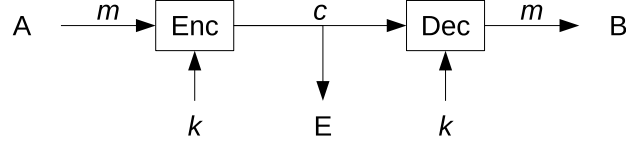


Figure 1: Message exchange between  $A$  and  $B$  using symmetric encryption.  $E$  is the eavesdropper.

## 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.  $\text{Gen}$  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);

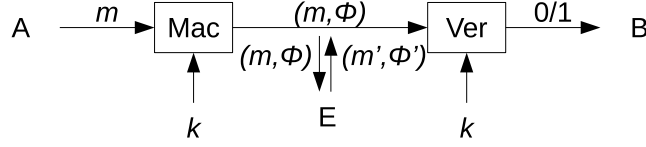


Figure 2: Message exchange between  $A$  and  $B$  using symmetric authentication.  $E$  is the eavesdropper.

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  $\Phi$ ;
- 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  $\varepsilon$ -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  $(m', \phi')$  that verifies with key  $k$  with low probability, even if it knows a valid pair  $(m, \phi)$ .  $\diamond$

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  $m, m', \phi, \phi'$ , 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  $\phi$ , *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  $\phi, \phi'$ . 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  $1^\lambda$  given to the adversary  $\mathcal{A}$  is there to highlight the fact that  $\mathcal{A}$  is polynomial in the length of the input ( $\lambda$ ).

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  $\mathcal{X}_\lambda$  and  $\mathcal{Y}_\lambda$  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  $\lambda$ 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  $\mathcal{X}_\lambda$  and  $\mathcal{Y}_\lambda$  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;

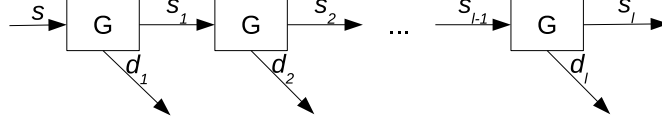


Figure 3: Extending a PRG with 1 bit stretch to a PRG with  $l$  bit stretch.

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  $\mathcal{H}_i^\lambda$  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.

$\mathcal{H}_l^\lambda$  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  $\mathcal{H}_i^\lambda$ , 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  $\Pi$  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  $\varepsilon$  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.  $\xrightarrow{\$} \text{Enc}(k, m_b)$  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  $\varepsilon$  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.  $b' \leftarrow \mathcal{A}^{\text{Enc}(k, \cdot)}(1^\lambda, c)$ .

$\Pi$  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  $\text{Enc}$  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  $\mathcal{A}_{\text{PRF}}$  that breaks the PRF.

$\mathcal{A}_{\text{PRF}}$  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  $\mathcal{A}_{\text{PRF}}$  some message  $m$ .  $\mathcal{A}_{\text{PRF}}$  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  $\mathcal{A}_{\text{PRF}}$   $(m_0, m_1)$ , 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  $c^*$ .  $\mathcal{A}$ ,



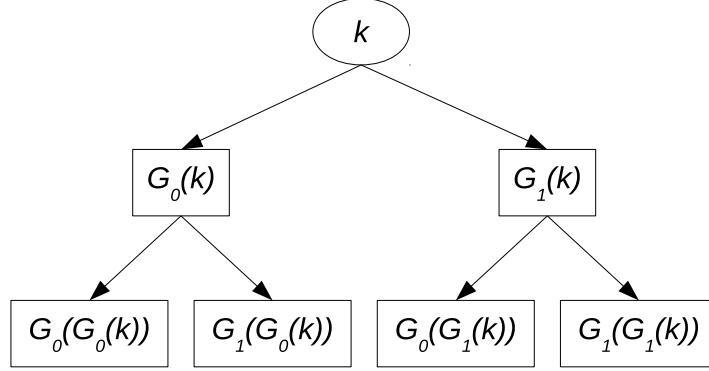


Figure 4: First two levels of a GGM tree.

after receiving  $c^*$ , asks some more encryption queries, and then outputs  $b'$ . If  $b' = 1$ ,  $\mathcal{A}_{\text{PRF}}$  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  $\lambda$  bits from the second  $\lambda$  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  $k_1, \dots, k_{t-(i+1)} \leftarrow \mathcal{S}\{0, 1\}^\lambda$ , 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  $k_{\bar{x}} \leftarrow \mathcal{S}\{0, 1\}^\lambda$  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  $k_{\perp} \leftarrow \mathcal{S}\{0, 1\}^\lambda$ , 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  $k \leftarrow \mathcal{S}\{0, 1\}^\lambda$ ;
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  $\Pi$  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.

**Construction 5.** [MAC from PRF] Let  $\mathcal{F} = \{F_k : \{0, 1\}^n \rightarrow \{0, 1\}^l\}_{k \in \{0, 1\}^\lambda}$  be a PRF family, and let  $\mathcal{K} = \{0, 1\}^\lambda$ . Define  $\text{Mac}(k, m) = F_k(m)$ . ◇

**Theorem 13.** *If  $\mathcal{F}$  is a PRF, the MAC shown in construction 5 is UFCMA.* ◇

*Proof of theorem 13.* Consider the game  $\mathcal{H}(\lambda)$  where:

1.  $R \leftarrow \mathcal{R}(n \rightarrow l)$  is a random function;
2.  $(m^*, \phi^*) \leftarrow \mathcal{A}^{R(\cdot)}(1^\lambda)$ ;
3. output 1 if  $R(m^*) = \phi^*$  and  $m^*$  is “fresh”.

Our first claim is that  $\mathcal{H}(\lambda) \approx_c \mathcal{G}_{\Pi, \mathcal{A}}^{\text{ufcma}}(\lambda)$  for all PPT adversaries  $\mathcal{A}$ . Assume not, then  $\exists$  a distinguisher  $\mathcal{D}$  for  $\mathcal{H}(\lambda)$  and  $\mathcal{G}_{\Pi, \mathcal{A}}^{\text{ufcma}}(\lambda)$ , and we can construct a distinguisher  $\mathcal{D}'$  for the PRF.  $\mathcal{D}'$  has access to an oracle  $O(\cdot)$  which is either  $F_k(\cdot)$  for some random  $k$ , or  $R(\cdot)$  for some random function  $R$ .  $\mathcal{D}'$  feeds a game to  $\mathcal{D}$  using  $O(\cdot)$ .

Our second claim is that  $\Pr [\mathcal{H}(\lambda) = 1] \leq 2^{-\lambda}$ , since  $R(\cdot)$  is random and the only way to predict it is by guessing. □

Up to this point we have shown that OWF, PRG, PRF and MAC are all in Minicrypt.

## 2.8 Domain Extension

We look now at domain extension. Suppose we have a PRF family  $\mathcal{F} = \{F_k : \{0, 1\}^n \rightarrow \{0, 1\}^l\}$  as above, and we have a message  $m = m_1 || \dots || m_t$ , with  $m_i \in \{0, 1\}^n$ , and with  $t$  being the number of blocks of  $m$ .

Let's look at some constructions that won't work.

1.  $\phi = \text{Mac}\left(k, \bigoplus_{i=1}^t m_i\right)$  does not work, since with  $m = m_1 || m_2$  we could swap the bits in position  $i$  of  $m_1$  and  $m_2$  and have the same authenticator;
2.  $\phi_i = \text{Mac}(k, m_i)$  and  $\phi = \phi_1 || \dots || \phi_t$  does not work, since we could rearrange the blocks of the authenticator and of the message and still get a valid couple. *i.e.*, take  $m' = m_1 || m_3 || m_2$  and  $\phi' = \phi_1 || \phi_3 || \phi_2$ ;

3.  $\phi_i = \text{Mac}(k, \langle i \rangle || m_i)$  and  $\phi = \phi_1 || \dots || \phi_t$ , where  $\langle i \rangle$  is the binary representation of integer  $i$ , does not work, since we could cut and paste blocks from different message/authenticator couples and to get a fresh valid couple.

Now, for the real one. To extend the domain of a PRF  $\mathcal{F}$  we need a function  $h : \{0, 1\}^{nt} \rightarrow \{0, 1\}^n$  for which is hard to find a collision, *i.e.*, two distinct messages  $m', m''$  such that  $h(m') = h(m'')$ . To do this, we introduce Collision Resistant Hash Functions (CRHs), an object found in Cryptomania. We add a key to the hash function.

**Definition 15.** [Universal Hash Function] The family of functions  $\mathcal{H} = \{h_s : \{0, 1\}^N \rightarrow \{0, 1\}^n\}_{s \in \{0, 1\}^\lambda}$  is universal (as in Universal Hash Function (UHF)) if for all distinct  $x, x'$  we have that

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

Two cases are possible, depending on what  $\varepsilon$  is:

- if  $\varepsilon = 2^{-n}$ , then  $\mathcal{H}$  is said to be Perfect Universal (PU);
- if  $\varepsilon = \text{negl}(\lambda)$ , with  $\lambda = |s|$ , then  $\mathcal{H}$  is said to be Almost Universal (AU).  $\diamond$

With UHF we can extend the domain of a PRF.

**Theorem 14.** If  $\mathcal{F}$  is a PRF and  $\mathcal{H}$  is a AU family of hash functions, then  $\mathcal{F}(\mathcal{H})$ , defined as

$$\mathcal{F}(\mathcal{H}) = \{F_k(h_s(\cdot)) : \{0, 1\}^N \rightarrow \{0, 1\}^l\}_{k'=(k,s)}$$

is a PRF.  $\diamond$

*Proof of theorem 14.* Consider the following games:

- $\mathcal{G}_{\mathcal{F}(\mathcal{H}), \mathcal{A}}^{\text{real}}(\lambda)$ , defined as:
  1.  $k \leftarrow \mathcal{S}\{0, 1\}^\lambda, s \leftarrow \mathcal{S}\{0, 1\}^\lambda$ ;
  2.  $b' \leftarrow \mathcal{A}^{F_k(h_s(\cdot))}(1^\lambda)$ .
- $\mathcal{G}_{\mathcal{S}, \mathcal{A}}^{\text{rand}}(\lambda)$ , defined as:
  1.  $\bar{R} \leftarrow \mathcal{R}(N \rightarrow l)$ ;
  2.  $b \leftarrow \mathcal{A}^{\bar{R}(\cdot)}(1^\lambda)$ .

Consider also the hybrid  $H_{\mathcal{S}, \mathcal{H}, \mathcal{A}}(\lambda)$ :

1.  $s \leftarrow \mathcal{S}\{0, 1\}^\lambda$ ;
2.  $R \leftarrow \mathcal{R}(n \rightarrow l)$ ;

3.  $b \leftarrow \mathcal{A}^{R(h_s(\cdot))}(1^\lambda)$ .

The first claim, *i.e.*, that  $\mathcal{G}_{\mathcal{F}(\mathcal{H}), \mathcal{A}}^{\text{real}}(\lambda) \approx_c H_{\mathbb{S}, \mathcal{H}, \mathcal{A}}(\lambda)$ , is left as exercise.

The second claim is that  $H_{\mathbb{S}, \mathcal{H}, \mathcal{A}}(\lambda) \approx_c \mathcal{G}_{\mathbb{S}, \mathcal{A}}^{\text{rand}}(\lambda)$ . Assume the adversary asks  $q$  distinct queries. Consider the event  $E = \exists i, j \text{ such that } h_s(x_i) = h_s(x_j) \text{ with } i \neq j$ , and with  $i, j \leq q$ . If  $E$  doesn't happen,  $H_{\mathbb{S}, \mathcal{H}, \mathcal{A}}(\lambda)$  and  $\mathcal{G}_{\mathbb{S}, \mathcal{A}}^{\text{rand}}(\lambda)$  are the same. This event is the same as getting first all the inputs that the adversary wants to try, and then sampling  $s \leftarrow \mathbb{S}\{0, 1\}^\lambda$ , so its probability can be bounded as

$$\Pr[E] = \Pr[\exists i, j : h_s(x_i) = h_s(x_j)] \leq \binom{q}{2} \varepsilon \leq q^2 \varepsilon. \quad \square$$

Now, let's look at a construction.

**Construction 6.** [UHF with Galois field] Let  $\mathbb{F}$  be a finite field, such as the Galois field over  $2^n$ . In the Galois field, a bit string represents the coefficients of a polynomial of degree  $n - 1$ . Addition is the usual, while for multiplication an irreducible polynomial  $p(x)$  of degree  $n$  is fixed, and the operation is carried out modulo  $p(x)$ .

We pick  $s \in \mathbb{F}$ , and  $x = x_1 || \dots || x_t$  with  $x_i \in \mathbb{F}$  for all  $i$ . The hash function is defined as

$$h_s(x) = h_s(x_1 || \dots || x_t) = \sum_{i=1}^t x_i \cdot s^{i-1} = Q_x(s).$$

A collision is two distinct  $x, x'$  such that

$$Q_x(s) = Q_{x'}(s) \iff Q_{x-x'}(s) = 0 \iff \sum_{i=1}^t (x_i - x'_i) s^{i-1} = 0.$$

This means that  $s$  is a root of  $Q_{x-x'}$ . So the probability of a collision is:

$$\Pr[h_s(x) = h_s(x')] = \frac{t-1}{|\mathbb{F}|} = \frac{t-1}{2^n}. \quad (\text{negligible})$$

◇

We now look at a computational variant of hash functions. We want hash functions for which collisions are difficult to find for any PPT adversary  $\mathcal{A}$ , *i.e.*, families of functions such that

$$\Pr_s [h_s(x) = h_s(x') : (x, x') \leftarrow \mathcal{A}(1^\lambda)] \leq \varepsilon.$$

We want to use some PRF family  $\mathcal{F}$  to define  $\mathcal{H}$ . Enter Cypher Block Chain (CBC)-MAC (fig. 5). CBC-MAC is defined as

$$h_s(x_1, \dots, x_t) = F_s(x_t \oplus F_s(x_{t-1} \oplus \dots \oplus F_s(x_1)) \dots).$$

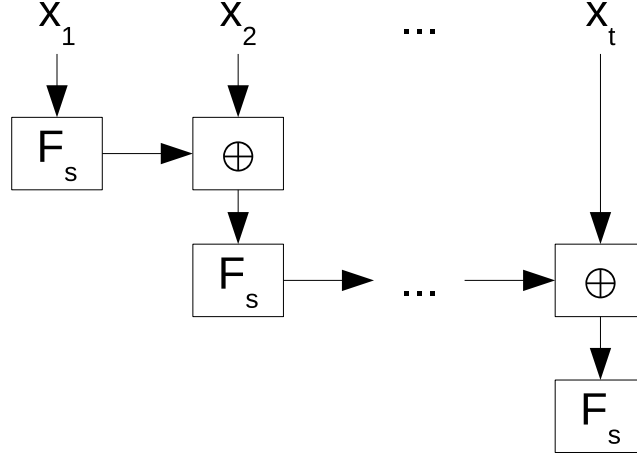


Figure 5: Construction of the CBC-MAC.

**Theorem 15.** *CBC-MAC is a computationally secure AU hash function if  $\mathcal{F}$  is a PRF.*  $\diamond$

There's also the encrypted CBC-MAC, *i.e.*,  $F_k(\text{CBC-MAC}(s, x))$ .

**Theorem 16.** *CBC-MAC is a PRF.*  $\diamond$

**Theorem 17.** *CBC-MAC is AU.*  $\diamond$

CBC-MAC is insecure with variable length messages.

XOR-MAC is defined as follows: take  $\eta$ , a random value (nonce), and output  $(\eta, F_k(\eta) \oplus h_s(x))$ . Note that here the input is shrunk to the output size of the PRF, while before we shrunk to the input size of the PRF.

Suppose the adversary is given a pair  $(m, (\eta, v))$  from a XOR-MAC. She could try to output  $(m', (\eta, v \oplus a))$ , trying to guess an  $a$  such that  $h_s(m) \oplus a = h_s(m')$ , so that this is still a valid tag. If  $a$  is hard to find (as should be), we have “almost xor universality”. Almost universality is the special case where  $a = 0$ .

From a PRF family we can get a MAC for Fixed Input Length (FIL) messages (a FIL-MAC). Table 1 compares the constructions we have seen earlier for FIL-MACs and Variable Input Length (VIL)-MACs.

CBC-MAC cannot be extended securely to VIL. As an example, take

$$\text{CBC-MAC}(m_1 || \dots || m_t) = F_k(m_t \oplus \dots \oplus F_k(m_1) \dots). \quad (1)$$

If we have  $(m_1, \phi_1)$ , with  $\phi_1 = F_k(m_1)$ . We could then take  $m_2 = m_1 || \phi_1 \oplus m_1$ , and  $\phi_1$  would be a valid authenticator for  $m_2$ :

$$\text{CBC-MAC}(m_2) = F_k(m_1 \oplus \phi_1 \oplus F_k(m_1)) = F_k(m_1 \oplus \cancel{\phi_1} \oplus \cancel{\phi_1}) = F_k(m_1) = \phi_1.$$

	FIL-PRF	FIL-MAC	VIL-MAC
$\mathcal{F}(\mathcal{H})$	✓	✓	
CBC-MAC		✓	
E-CBC-MAC	✓	✓	✓
XOR-MAC		✓	✓

Table 1: Constructions for FIL-PRF, FIL-MAC, and VIL-MAC.

## 2.9 Chosen Cyphertext Attacks and Authenticated Encryption

In Chosen Cyphertext Attack (CCA) security, the adversary is allowed to choose the cyphertext, and to see its decryption.

**Definition 16.** [CCA-security] Let  $\Pi = (\text{Gen}, \text{Enc}, \text{Dec})$  be a SKE scheme, and consider the following game  $\mathcal{G}_{\Pi, \mathcal{A}}^{\text{cca}}(\lambda, b)$ :

1.  $k \leftarrow \mathcal{K}$ ;  $\mathcal{K} = \{0, 1\}^\lambda$ ;
2.  $(m_0, m_1) \leftarrow A^{\text{Enc}(k, \cdot), \text{Dec}(k, \cdot)}(1^\lambda)$ ;
3.  $c \leftarrow \text{Enc}(k, m_b)$ ;
4.  $b' \leftarrow A^{\text{Enc}(k, \cdot), \text{Dec}^*(k, \cdot)}(1^\lambda, c)$  where  $\text{Dec}^*$  does not accept  $c$ .

$\Pi$  is CCA-secure if for all PPT adversaries  $\mathcal{A}$  we have that

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

CCA-security implies a property called *malleability*: if you change a bit the cyphertext you don't get similar messages.

**Claim 1.** The SKE scheme consisting of  $\text{Enc}(k, m) = (r, F_k(r) \oplus m)$  (for random  $r$ ) and  $\text{Dec}(k, (c_1, c_2)) = F_k(c_1) \oplus c_2 = m$  is not CCA-secure.  $\diamond$

*Proof of claim 1.* 1. Output  $m_0 = 0^n$  and  $m_1 = 1^n$ ;

2. get  $c = (c_1, c_2) = (r, F_k(r) \oplus m_b)$ ;
3. let  $c'_2 = c_2 \oplus 10^{n-1}$ ;
4. query  $\text{Dec}(k, (c_1, c'_2))$  (which is different from  $c$ );
5. if you get  $10^{n-1}$ , output 0, else output 1.

This always works:

$$\begin{aligned} \text{Dec}(k, (c_1, c'_2)) &= F_k(c_1) \oplus c'_2 = \overbrace{F_k(c_1) \oplus c_2}^{m_b} \oplus 10^{n-1} \\ &= m_b \oplus 10^{n-1} = 10^{n-1} \iff m_b = 0^n. \end{aligned}$$

□

We'll build now Authenticated Encryption. It's both CPA and Integrity (of cyphertext) (INT), *i.e.*, it's hard for the adversary to generate a valid cyphertext not queried to the encryption oracle.

As an exercise, formalise the fact that CPA and INT imply CCA, *i.e.*, reduce CCA to CPA.

Any CPA-secure encryption scheme, together with a MAC, gives you CCA security. This is called an encrypted MAC.

**Construction 7.** [Encrypted MAC] Consider the encryption scheme  $\Pi_1 = (\text{Gen}, \text{Enc}, \text{Dec})$ , with key space  $\mathcal{K}_1$ , and the MAC  $\Pi_2 = (\text{Gen}, \text{Mac}, \text{Vrfy})$ , with key space  $\mathcal{K}_2$ . We build the encryption scheme  $\Pi' = (\text{Gen}', \text{Enc}', \text{Dec}')$ , with key space  $\mathcal{K}' = \mathcal{K}_1 \times \mathcal{K}_2$  as follows:

1.  $\text{Enc}'(k', m) = (c, \phi) = c'$ , with  $c \leftarrow \text{Enc}(k_1, m)$  and  $\phi \leftarrow \text{Mac}(k_2, c)$ ;
2.  $\text{Dec}'(k', (c, \phi))$  checks if  $\text{Mac}(k_2, c) = \phi$ : if not, it outputs  $\perp$ , else it outputs  $\text{Dec}(k_1, c)$ .  $\diamond$

**Theorem 18.** *If  $\Pi_1$  is CPA-secure and  $\Pi_2$  is strongly UFCMA-secure, then  $\Pi'$  is CPA and INT.*  $\diamond$

**I don't know what I wrote here?**

Strong UFCMA security means you output  $(m^*, \phi^*)$  where the couple was never asked. So if you know  $(m, \phi)$ , you can output  $(m, \phi')$ .

*Proof of theorem 18.* We need to show that  $\Pi'$  is both CPA and INT.

1. The proof for CPA is just a reduction to the CPA-security of  $\Pi_1$ . Assume  $\mathcal{A}'$  breaks CPA-security of  $\Pi'$ , we can construct  $\mathcal{A}_1$  which breaks CPA of  $\Pi_1$ .

$\mathcal{A}_1$  picks a key to impersonate the MAC, then for each message  $m$  gets its encryption  $c$  from  $\Pi_1$ , and then does Mac of  $c$  to get the authenticator  $\phi$ . Then it returns  $(c, \phi)$  to  $\mathcal{A}'$ . When it receives  $m_0, m_1$  from  $\mathcal{A}'$ , it receives  $c^*$  from  $\Pi_1$ , computes its Mac, and gives the result to  $\mathcal{A}'$ . Then it outputs whatever  $\mathcal{A}'$  outputs.

2. For INT, assume  $\mathcal{A}''$  breaking INT of  $\Pi'$ , we can build  $\mathcal{A}_2$  which breaks INT of  $\Pi_2$ .

We ask  $\mathcal{A}_2$  the encryption of  $m$ .  $\mathcal{A}_2$  picks a key  $k$ , computes  $\text{Enc}(k, m) = c$ , and gives  $c$  to the  $\text{Mac}(\cdot)$  oracle. Then it gives  $(c, \phi)$  to  $\mathcal{A}''$ . Later on,  $\mathcal{A}''$  gives  $\mathcal{A}_2$  some  $(c^*, \phi^*)$ , which is a valid validator if  $\Pi'$  is not INT, so  $\mathcal{A}_2$  has broken  $\Pi_2$ .  $\square$

The approach to CCA consisting of Encrypt-then-MAC works. Other approaches don't work in general:



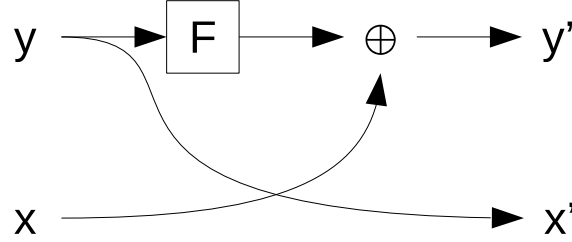


Figure 6: The Feistel permutation.

- Encrypt-and-MAC, which is what Secure Shell (SSH) does, *i.e.*,  $c \leftarrow \text{Enc}(k_c, m)$  and  $\phi \leftarrow \text{Mac}(k_s, m)$ ;
- MAC-then-Encrypt, which is what Transport Layer Security (TLS) does, *i.e.*,  $\phi \leftarrow \text{Mac}(k_s, m)$  and  $c \leftarrow \text{Enc}(k_c, m || \phi)$ .

## 2.10 Pseudo Random Permutations

Block cyphers are Pseudo Random Permutations (PRPs), a function family that is a PRF but also a permutation. A PRP family cannot be distinguished from a true permutation. For a *strong* PRP family, the adversary has access to the inverse of the permutation. From PRFs we can build both PRPs and strong PRPs.

**Definition 17.** [Feistel] Let  $F : \{0, 1\}^n \rightarrow \{0, 1\}^n$ , then the Feistel function (fig. 6) is defined as

$$\psi_F(\underbrace{x, y}_{2n}) = (y, x \oplus F(y)) = (\underbrace{x', y'}_{2n}). \quad \diamond$$

It's easy to see that the Feistel function is invertible:

$$\psi_F^{-1}(x', y') = (F(x') \oplus y', x') = (\cancel{F(y)} \oplus \cancel{F(y)} \oplus x, y) = (x, y).$$

We can “cascade” several Feistel functions, to create a Feistel network. Take  $F_1, \dots, F_l$ , and define the following function:

$$\psi_{\mathcal{F}}[l](x, y) = \psi_{F_l}(\psi_{F_{l-1}}(\dots \psi_{F_1}(x, y) \dots))$$

and its inverse:

$$\psi_{\mathcal{F}}^{-1}[l](x', y') = \psi_{F_1}^{-1}(\dots \psi_{F_{l-1}}^{-1}(\psi_{F_l}^{-1}(x', y')) \dots).$$

**Theorem 19.** [Luby-Rackoff] If  $\mathcal{F} = \{F_k : \{0,1\}^n \rightarrow \{0,1\}^n\}_{k \in \{0,1\}^\lambda}$  is a PRF, then  $\psi_{\mathcal{F}}[3]$  is a PRP and  $\psi_{\mathcal{F}}[4]$  is a strong PRP.  $\diamond$

We will prove only that  $\psi_{\mathcal{F}}[3]$  is a PRP.

**Theorem 20.** If  $\mathcal{F}$  is a PRF,  $\psi_{\mathcal{F}}[3]$  is a PRP.  $\diamond$

Recall that

$$\psi_{\mathcal{F}}[3] = \psi_{F_{k_3}}(\psi_{F_{k_2}}(\psi_{F_{k_1}}(x, y))).$$

*Proof of theorem 20.* Consider these experiments:

$$\begin{aligned} H_0 : (x, y) &\xrightarrow{\psi_{F_{k_1}}} (x_1, y_1) \xrightarrow{\psi_{F_{k_2}}} (x_2, y_2) \xrightarrow{\psi_{F_{k_3}}} (x_3, y_3) \\ H_1 : (x, y) &\xrightarrow{\psi_{R_1}} (x_1, y_1) \xrightarrow{\psi_{R_2}} (x_2, y_2) \xrightarrow{\psi_{R_3}} (x_3, y_3). \end{aligned}$$

$H_2$  is just like  $H_1$ , but we stop if  $y_1$  “collides”. A collision happens when we have  $(x, y) \rightarrow (x_1 = y, y_1 = x \oplus R_1(y))$ , and  $y_1 = y$ .

In  $H_3$  we replace  $y_2 = R_2(y_1) \oplus x_1$  with  $y_2 \leftarrow \mathcal{R}\{0,1\}^n$ , and set  $x_2 = y_1$ .

In  $H_4$  we replace  $y_3 = R_3(y_2) \oplus x_2$  with  $y_3 \leftarrow \mathcal{R}\{0,1\}^n$ , and set  $x_3 = y_2$ .

In  $H_5$  we directly map  $(x, y) \xrightarrow{\bar{R}} (x_3, y_3)$ , with  $\bar{R} \leftarrow \mathcal{R}(2n \rightarrow 2n)$  being a random permutation.

First claim:  $H_0 \approx_c H_1$ , since we replaced the PRF with truly random functions. The proof is the usual proof by hybrids.

Second claim:  $H_1 \approx_c H_2$ . Consider the event  $E$  of a collision, defined as “ $\exists (x, y) \neq (x', y')$  such that  $x \oplus R_1(y) = x' \oplus R_1(y')$ , i.e.,  $x \oplus x' = R_1(y) \oplus R_1(y')$ ”. If  $y = y'$ , there can’t be a collision, so we can assume that  $y \neq y'$ . So the probability of a collision is:

$$\Pr[E] = \Pr[R_1(y) \oplus R_1(y') = x \oplus x'] \leq 2^{-n}.$$

Third claim:  $H_2 \approx_c H_3$ . In  $H_2$ ,  $y_1$  is just a stream of independent values. Since  $y_1$  never collides and  $R_2$  is random, all  $R_2(y_1)$  are uniform and independent, and so is  $y_2 = R_2(y_1) \oplus x_1$ .

Fourth claim:  $H_3 \approx_c H_4$ . Now  $y_3$  is random, and  $x_3 = y_2$ . It suffices that  $y_2$  never collides, since  $R_3(y_2)$  is a sequence of one time pad keys.

$$\Pr[y_2 \text{ collides}] \leq \binom{q}{2} 2^{-n}.$$

Fifth claim:  $H_4 \approx_c H_5$ . Just notice that  $H_4$  is simply a random function from  $2n$  bits to  $2n$  bits, and  $H_5$  is a random permutation. They can only be distinguished if there is a collision in  $H_4$ , which again has negligible probability.  $\square$

A strong PRP  $P$  leads to CCA security with the following construction:

- $\text{Enc}(k, m) = P(k, m || r)$ , with  $m, r \in \{0,1\}^n$ ;
- $\text{Dec}(k, c) = P^{-1}(k, c)$  and take the first  $n$  bits.

## Domain Extension for Pseudo Random Permutations

Assume we have a message  $m = m_1 || \dots || m_t$ , with  $m_i \in \{0, 1\}^n$ , and you are given a PRP from  $n$  bits to  $n$  bits.

- One natural thing to do is Electronic Code Book (ECB). Let  $c = c_1 || \dots || c_t$  with  $c_i = P_k(m_i)$ . This is very fast, parallelisable, but insecure (since it's deterministic).
- Cypher Feed Back (CFB): sample  $c_0$  at random, then let  $c_i = P_k(c_{i-1}) \oplus m_i$ . This is CPA secure and parallelisable for decryption (but not for encryption).
- CBC: sample  $c_0$  at random, then let  $c_i = P_k(c_{i-1} \oplus m_i)$ . This is also CPA secure. Note that if you output all  $c_i$  this does not work as a MAC (recall that CBC-MAC outputs just  $c_t$ ).
- Counter (CTR)-mode: sample  $r \leftarrow \mathcal{S}[N]$ , with  $N = 2^n$ . Let  $c_i = P_k(r + i - 1 \bmod N) \oplus m_i$ , and output  $c_0 || c_1 || \dots || c_t$  with  $c_0 = r$ . For decryption, compute  $m_i = P_k(c_0 + i - 1 \bmod N) \oplus c_i$ .

**Theorem 21.** *If  $\mathcal{F}$  is a PRF family, then CTR-mode is CPA-secure for VIL.*  $\diamond$

*Proof of theorem 21.* Consider the game  $H_0(\lambda, b)$ , defined as:

1.  $k \leftarrow \mathcal{S}\{0, 1\}^\lambda$ ;
2. the adversary asks encryption queries, for messages  $m = m_1 || \dots || m_t$ :
  - $c_0 = r \leftarrow \mathcal{S}[N]$  (with  $N = 2^n$ );
  - $c_i = F_k(r + i - 1 \bmod N) \oplus m_i$ ;
  - output  $c_0 || c_1 || \dots || c_t$ .
3. challenge: the adversary gives  $(m_0^*, m_1^*)$ , and take  $m_b^* = m_{b_1}^* || \dots || m_{b_t}^*$  (both messages have the same length);
4. compute  $c^*$  from  $m_b^*$  and output to adversary;
5. adversary asks more encryption queries, then it outputs  $b'$ .

We want to show that  $H_0(\lambda, 0) \approx_c H_1(\lambda, 1)$ .

First, we define the hybrid  $H_1(\lambda, b)$  which samples  $R \leftarrow \mathcal{S}\mathcal{R}(n \rightarrow n)$  and uses  $R(\cdot)$  in place of  $F_k(\cdot)$ . The proof that  $H_0(\lambda, b) \approx_c H_1(\lambda, b)$  is a usual proof by reduction to security of the PRF. Assume there exists a distinguisher  $\mathcal{D}$  for  $H_0$  and  $H_1$ , we build a distinguisher  $\mathcal{D}'$  for the PRF, which has access to some oracle that is either  $F_k(\cdot)$  or  $R(\cdot)$  for random  $R$ .  $\mathcal{D}'$  plays the game defined above with  $\mathcal{D}$  using the oracle, and outputs whatever it outputs, thus distinguishing the PRF.

Now consider the hybrid  $H_2(\lambda)$ , which outputs a uniformly random challenge ciphertext  $c^* \leftarrow \{0,1\}^{n(t+1)}$ . We claim that  $H_1(\lambda, b) \approx_s H_2(\lambda)$  for  $b \in \{0,1\}$ , *i.e.*, they are statistically indistinguishable: we accept unbounded adversaries that can only ask a polynomial number of queries.

Consider  $c^*$ , and the values of  $R(r^*), R(r^*+1), \dots, R(r^*+t^*-1)$ . Then, consider the  $i$ -th encryption query,  $c^i$ , and the values of  $R(r_i), R(r_i+1), \dots, R(r_i+t_i-1)$ . If  $R(r_i+j)$  is always different from  $R(r^*+j')$  for any  $j'$ , this is basically OTP. So, calling  $E$  the event “ $\exists j_1, j_2$  such that  $R(r_i+j_1) = R(r^*+j_2)$ ”, it suffices to show that  $\Pr[E] \leq \text{negl}(\lambda)$ .

Assume there are  $q$  queries, and fix a maximum length  $t$  of the queried messages, and let  $E_i$ , for  $i \in [q]$ , be the event of an overlap happening at query  $i$ . Clearly  $\Pr[E] \leq \sum_{i=1}^q \Pr[E_i]$ . For  $E_i$  to happen, we must have that  $r^* - t + 1 \leq r_i \leq r^* + t - 1$  (this is not tight!). So the size of the interval in which  $r_i$  must be is  $2t - 1$ . Then,  $\Pr[E_i] = \frac{2t-1}{2^n}$ , and  $\Pr[E] \leq q \frac{2t-1}{2^n}$ , which is negligible.  $\square$

## 2.11 Collision Resistant Hash Functions

When we saw domain extensions for PRFs, we showed that  $\mathcal{F}(\mathcal{H})$  is a PRF if  $\mathcal{F}$  is a PRF and  $\mathcal{H}$  is AU. This doesn't work for MACs: to extend the domain of a MAC we need CRHs.

A hash function is used to compress  $l$  bits into  $n$  bits, with  $n \ll l$ . A family of hash functions is defined as  $\mathcal{H} = \{H_s : \{0,1\}^l \rightarrow \{0,1\}^n\}_{s \in \{0,1\}^\lambda}$ . AU means that it's hard to find a collision for an adversary that does not know  $s$ . Collision resistance means that it's hard to find a collision even if the adversary knows  $s$ .

**Definition 18.** [Collision Resistant Hash Function] Let  $\mathcal{H} = \{h_s : \{0,1\}^l \rightarrow \{0,1\}^n\}_{s \in \{0,1\}^\lambda}$  be a family of functions, and consider the game  $\mathcal{G}_{\mathcal{H}, \mathcal{A}}^{\text{CR}}(\lambda)$ , defined as

1.  $s \leftarrow \{0,1\}^\lambda$ ;
2.  $(x, x') \leftarrow \mathcal{A}(1^\lambda, s)$ ;
3. output 1 if  $x \neq x' \wedge h_s(x) = h_s(x')$ .

$\mathcal{H}$  is a CRH if for all PPT adversaries  $\mathcal{A}$  there is a negligible function  $\varepsilon(\lambda)$  such that

$$\Pr[\mathcal{G}_{\mathcal{H}, \mathcal{A}}^{\text{CR}}(\lambda) = 1] \leq \varepsilon(\lambda). \quad \diamond$$

We'll show how to construct CRHs in two steps.

1. Start with a compression function  $h_s : \{0,1\}^{2n} \rightarrow \{0,1\}^n$  and obtain a domain extension, *i.e.*, a function  $h'_s : \{0,1\}^* \rightarrow \{0,1\}^n$ .
2. Build a Collision Resistant (CR) compression function.

There are two famous constructions.

**Construction 8.** [Merkle-Damgard] It goes from  $tn$  bits to  $n$  bits, for fixed  $t$ . Let  $m = x_1 || \dots || x_t$ , and define intermediate outputs  $y_i$  for  $i \in [0, t]$ . We define  $y_0 = 0^n$ , and  $y_i = h_s(x_i || y_{i-1})$ . The output is  $y = y_t$ .  $\diamond$

**Construction 9.** [Merkle tree] It goes from  $2^d n$  bits to  $n$  bits, with  $d$  being the height of the tree.  $\diamond$

With the Merkle tree one could give a partially hashed version of a file.

**Theorem 22.** *Merkle-Damgard (MD) (construction 8) is a CRH  $\mathcal{H}' = \{h'_s : \{0, 1\}^{tn} \rightarrow \{0, 1\}^n\}$  if the function  $\mathcal{H} = \{h_s : \{0, 1\}^{2n} \rightarrow \{0, 1\}^n\}$  is CR.*  $\diamond$

*Proof of theorem 22.* Let  $\mathcal{A}'$  be an adversary capable of outputting  $x \neq x'$  such that  $h'_s(x) = h'_s(x')$ . Then we can construct  $\mathcal{A}$  breaking  $\mathcal{H}$  (which is CR).

If  $h'_s(x) = h'_s(x')$  but  $x \neq x'$ , there must be some  $j \in [1, t]$  (with  $x = x_1 || \dots || x_t$  and  $x' = x'_1 || \dots || x'_t$ ) such that  $(x_j, y_{j-1}) \neq (x'_j, y'_{j-1})$ , but after that they are equal. Then  $h_s(x_j, y_{j-1}) = h_s(x'_j, y'_{j-1})$ , which is a collision.  $\square$

**Construction 10.** [Strengthened Merkle-Damgard] Strengthened MD is defined as

$$h'_s(x_1 || \dots || x_t) = h_s(\langle t \rangle || h_s(x_t || \dots || h_s(x_1 || 0^n) \dots)). \quad \diamond$$

With the strengthened MD we have suffix-free messages, which give us a VIL MD.

**Theorem 23.** *Strengthened MD (construction 10) is CR.*  $\diamond$

*Proof of theorem 23.* Assume there is a collision  $x = x_1 || \dots || x_t \neq x' = x'_1 || \dots || x'_t$ , i.e., they are such that  $h'_s(x) = h'_s(x')$ . Two cases are possible:

1. if  $t = t'$ , we have already shown this in the proof of theorem 22;
2. if  $t \neq t'$ , the collision is on  $h_s(\langle t \rangle || y_t)$  and  $h_s(\langle t' \rangle || y'_{t'})$ .  $\square$

**Construction 11.** [Davies-Meyer]  $\mathcal{H}_E(s, x) = E_s(x) \oplus x$  where  $E_{(\cdot)}(\cdot)$  is a block cypher.  $\diamond$

For this construction to be CR, it should be hard to find  $(x, s) \neq (x', s')$  such that  $E_s(x) \oplus x = E_{s'}(x') \oplus x'$ .

There's something strange going on: a block cypher is a PRP, and OWF give us PRP, but we know that it's impossible to get CRH from OWF. Furthermore, this is actually insecure for concrete PRPs. Let  $E$  be a PRP, and define  $E'$  such that  $E'(0^n, 0^n) = 0^n$  and  $E'(1^n, 1^n) = 1^n$ , and otherwise  $E'(x) = E(x)$ . This is still a PRP, but  $E'(0^n, 0^n) \oplus 0^n = E'(1^n, 1^n) \oplus 1^n$ .

We must assume that we have no bad keys. This is called the Ideal Cypher Model (ICM).

In the ICM, one has a truly random permutation  $E_{(\cdot)}(\cdot)$  for all keys. When the adversary asks for  $(s_1, x_1)$ , choose random  $y_1 \leftarrow \{0, 1\}^n$  and set  $E_{s_1}(x_1) = y_1$ . On later query  $(s_1, x_i)$ , pick  $y_i \leftarrow \{0, 1\}^n \setminus \bigcup_{j=1}^i \{y_j\}$ , and set  $E_{s_1}(x_i) = y_i$ .

**Theorem 24.** *Davies-Meyer is CR in ICM.*  $\diamond$

*Proof of theorem 24.* As usual, we consider a series of experiments.

$$H_0(\lambda) \equiv \mathcal{G}_{\text{DM}, \mathcal{A}}^{\text{CR}}(\lambda).$$

$\mathcal{A}$  makes two types of queries: forward queries, in the form  $(s, x)$ , to get  $E_s(x)$ , and backward queries, in the form  $(s, y)$ , to get  $E_s^{-1}(y)$ . Imagine of keeping the value of  $z = x \oplus y$ .

Now, we define  $H_1(\lambda)$ : after picking  $y$  for some  $(s, x)$ , we don't remove  $y$  from the range (and the same for backward queries).

The first claim is that, assuming  $\mathcal{A}$  asks  $q$  queries,

$$\left| \Pr[H_0(\lambda) = 1] - \Pr[H_1(\lambda) = 1] \right| \leq \frac{q^2}{2} 2^{-n}.$$

To verify this, note that  $H_0$  and  $H_1$  are distinguishable if and only if a collision happens in  $E_s(\cdot)$  or in  $E_s^{-1}(\cdot)$ .

$H_2(\lambda)$ : check for collisions in  $z = x \oplus y$ . If not all  $z$  values are distinct, abort.

The second claim is that

$$\left| \Pr[H_1(\lambda) = 1] - \Pr[H_2(\lambda) = 1] \right| \leq \frac{q^2}{2} 2^{-n}.$$

Verification is as usual.

$H_3(\lambda)$ : when  $\mathcal{A}$  outputs  $(x, s), (x', s')$  collision, check that  $(x, s, y, z)$  and  $(x', s', y', z')$  are in the table. If not, define the entries.

Third claim:

$$\left| \Pr[H_2(\lambda) = 1] - \Pr[H_3(\lambda) = 1] \right| \leq \frac{4q}{2^n}.$$

$y$  can collide with  $q$  values, with probability  $\frac{q}{2^n}$ . The same goes for  $z, y', z'$ , so we get  $\frac{4q}{2^n}$ .

$H_4(\lambda)$ : there are no collisions on  $z$ .  $\Pr[H_4(\lambda) = 1] = 0$  since  $(x, s, y, z), (x', s', y', z') \implies z \neq z'$ .

$H_3 \approx_c H_4$ , and we're done.  $\square$

## 2.12 Claw-Free Permutations

We introduce a new assumption: Claw-Free Permutations (CFPs). A claw is a pair of functions  $(f_1, f_0)$  and two values  $(x_1, x_0)$  such that  $f_1(x_1) = f_0(x_0)$ , with  $f_1 \neq f_0$ .

**Definition 19.** [Claw-Free Permutation] A CFP is a tuple  $\Pi = (\text{Gen}, f_0, f_1)$  where  $pk \leftarrow \text{sGen}(1^\lambda)$  defines a domain  $\mathcal{X}_{pk}$ , and  $f_0(pk, \cdot)$  and  $f_1(pk, \cdot)$  are permutations over  $\mathcal{X}_{pk}$ .

Consider the game  $\mathcal{G}_{\Pi, \mathcal{A}}^{\text{CF}}(\lambda)$ , defined as:

1.  $pk \leftarrow \text{\$Gen}(1^\lambda)$ ;
2.  $(x_0, x_1) \leftarrow \mathcal{A}(pk)$ ;
3. output 1  $\iff f_0(pk, x_0) = f_1(pk, x_1)$ .

$\Pi$  is claw-free if for all PPT adversaries  $\mathcal{A}$

$$\Pr [\mathcal{G}_{\Pi, \mathcal{A}}^{\text{CF}}(\lambda) = 1] \leq \varepsilon(\lambda). \quad \diamond$$

### 2.13 Waters Signature Scheme

Computational Diffie-Hellman (CDH) in bilinear groups.

**Definition 20.** [Bilinear Group]  $(\mathbb{G}, \mathbb{G}_T, q, g, \hat{e}) \leftarrow \text{Bilin}(1^\lambda)$ .

1.  $\mathbb{G}, \mathbb{G}_T$  are groups of order  $q$  with some efficient operation “ $\cdot$ ”.  $T$  stands for target;
2.  $g$  is a generator of  $\mathbb{G}$ ;
3.  $\hat{e} : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$  is a bilinear map, *i.e.*,

a) bilinearity:  $\forall g, h \in \mathbb{G}, \forall a, b \in \mathbb{Z}_q$ ,

$$\hat{e}(g^a, h^b) = \hat{e}(g, h)^{ab};$$

b) non-degeneracy:  $\hat{e}(g, g) \neq 1$  for generator  $g$ .  $\diamond$

**Observation 1.** *They are pairing on elliptic curves and CDH is believed to hold in such groups. Decisional Diffie-Hellman (DDH) comes easy in  $\mathbb{G}$  with a bilinear map.*  $\diamond$

**Construction 12.** [Waters Signature Scheme]

- $\text{KeyGen}(1^\lambda)$ :  $params = (\mathbb{G}, \mathbb{G}_T, q, g, \hat{e}) \leftarrow \text{\$Bilin}(1^\lambda)$ ,  $a \leftarrow \text{\$Z}_q$ ,  $g_1 = g^a$ ,  $g_2, \mu_1, \dots, \mu_k \leftarrow \text{\$G}$ .  $pk = (params, g_1, g_2, \mu_0, \dots, \mu_k)$ ,  $sk = g_2^a$ .
- $\text{Sign}(sk, m)$ : let  $m = (m[1], \dots, m[k]) \in \{0, 1\}^k$ , and  $\alpha(m) = \alpha_{\mu_0, \dots, \mu_k}(m) = \mu_0 \prod_{i=1}^k \mu_i^{m[i]}$ . Pick  $r \leftarrow \text{\$Z}_q$ , output  $\sigma = (g_2^a \cdot \alpha(m)^r, g^r) = (\sigma_1, \sigma_2)$ .
- $\text{Vrfy}(pk, (m, (\sigma_1, \sigma_2)))$ : check that  $\hat{e}(g, \sigma_1) = \hat{e}(\sigma_2, \alpha(m)) \cdot \hat{e}(g_1, g_2)$ .

Correctness:

$$\begin{aligned} \hat{e}(g, \sigma_1) &= \hat{e}(g, g_2^a \cdot \alpha(m)^r) = \hat{e}(g, g_2^a) \cdot \hat{e}(g, \alpha(m)^r) \\ &= \hat{e}(g^a, g_2) \cdot \hat{e}(g^r, \alpha(m)) = \hat{e}(g_1, g_2) \cdot \hat{e}(\sigma_2, \alpha(m)). \end{aligned}$$

Security: partitioning argument.  $\mathcal{M} = \{0, 1\}^k$ , and  $X$  is some auxiliary information. Given  $X$ , one can simulate signature on  $m \in \mathcal{M}_1^X$  and moreover with high probability all queries will belong to  $\mathcal{M}_1^X$ . Forgery will be in  $\mathcal{M}_2^X$  which will allow to break CDH.  $\diamond$

**Theorem 25.** *Under CDH, Waters Signature Scheme (WSS) is UFCMA.*  $\diamond$

*Proof of theorem 25.* Let  $\mathcal{A}$  be a PPT adversary capable of breaking WSS with probability greater than  $\frac{1}{p(\lambda)}$ , with  $p(\lambda) \in \text{poly}(\lambda)$ . Consider the adversary  $\mathcal{B}$  given  $\text{params} = (\mathbb{G}, \mathbb{G}_T, q, g, \hat{e})$ ,  $g_1 = g^a$ ,  $g_2 = g^b$ , and which wants to compute  $g^{ab}$ .  $\mathcal{B}(\text{params}, g_1, g_2)$ :

1. set  $l = kq_s$ , with  $q_s$  being the number of queries ( $<< q$ );
2. pick  $x_0 \leftarrow \mathbb{Z}_q$  and  $x_1, \dots, x_k \leftarrow \mathbb{Z}_q$  integers, and  $y_0, \dots, y_k \leftarrow \mathbb{Z}_q$ ;
3. set  $\mu_i = g_2^{x_i} g^{y_i}$  for all  $i \in [k]$ , and define the functions

$$\begin{aligned}\beta(m) &= x_0 + \sum_{i=1}^k m[i]x_i, \\ \gamma(m) &= y_0 + \sum_{i=1}^k m[i]y_i;\end{aligned}$$

4. run  $\mathcal{A}(\text{params}, g_1, g_2, \mu_0, \dots, \mu_k)$ ;
5. upon a query  $m \in \{0, 1\}^k$  from  $\mathcal{A}$ :
  - if  $\beta = \beta(m) = 0 \pmod q$ , abort;
  - else  $\gamma = \gamma(m)$ ,  $r \leftarrow \mathbb{Z}_q$ , and output

$$\sigma = (\sigma_1, \sigma_2) = (g_2^{\beta r} \cdot g^{\gamma r} \cdot g_1^{-\gamma \beta^{-1}}, g^r \cdot g_1^{-\beta^{-1}});$$

6. when  $\mathcal{A}$  outputs  $(m^*, (\sigma_1^*, \sigma_2^*))$ :

- if  $\beta(m^*) \neq 0 \pmod q$ , abort;
- else output

$$\frac{\sigma_1^*}{(\sigma_2^*)^{\gamma(m^*)}}.$$

If  $\beta(m) \neq 0$ ,  $\mathcal{B}$  aborts on output, but answers queries. If  $\beta(m) = 0$ ,  $\mathcal{B}$  aborts on queries, but outputs result.

Public key has random distribution as desired: perfect simulation of  $pk$ .

Conditioning on  $\mathcal{B}$  not aborting, the following two holds:

1. signature answers have the right distribution;
2.  $\mathcal{B}$  breaks CDH.



To verify point 1:  $a = \log_g(g_1)$ ,  $b = \log_g(g_2)$ . A signature  $\sigma$  on  $m$  is  $\sigma = (\sigma_1, \sigma_2) = (g_2^a \cdot \alpha(m)^{\bar{r}}, g^{\bar{r}})$  for random  $\bar{r} \leftarrow \mathbb{Z}_q$ . Take now  $\bar{r} = r - \alpha\beta^{-1}$  and let  $\beta = \beta(m)$ ,  $\gamma = \gamma(m)$ .

$$\alpha(m) = \mu_0 \prod_{i=1}^k \mu_i^{m[i]} = g_2^{x_0} g^{y_0} \prod_{i=1}^k (g_2^{x_i} g^{y_i})^{m[i]} = g_2^\beta g^\gamma.$$

$$\begin{aligned} \sigma_1 &= g_2^a \cdot \alpha(m)^{\bar{r}} = g_2^a \cdot \alpha(m)^{r - \alpha\beta^{-1}} \\ &= g_2^a \cdot g_2^{\beta r} \cdot g^{\gamma r} \cdot g_2^{-\alpha} \cdot g^{-\alpha\gamma\beta^{-1}} = g_2^{\beta r} \cdot g^{\gamma r} \cdot g_1^{-\gamma\beta^{-1}} \\ \sigma_2 &= g^{\bar{r}} = g^r \cdot g^{-\alpha\beta^{-1}} = g^r \cdot g_1^{-\beta^{-1}} \end{aligned}$$

Since  $\bar{r} = r - \alpha\beta^{-1}$  is random, we get the same distribution.

To verify point 2: let  $(\sigma_1^*, \sigma_2^*)$  be the forgery such that  $\beta(m^*) = 0 \pmod q$ .

$$\frac{\hat{e}(g, \sigma_1^*)}{\hat{e}(\sigma_2^*, \alpha(m^*))} = \hat{e}(g_1, g_2)$$

because it's valid. Since  $\hat{e}(g_1, g_2) = \hat{e}(g, g)^{ab}$  we have

$$\begin{aligned} \hat{e}(g, g)^{ab} &= \frac{\hat{e}(g, \sigma_1^*)}{\hat{e}(\sigma_2^*, \alpha(m^*))} = \frac{\hat{e}(g, \sigma_1^*)}{\hat{e}(\sigma_2^*, \underbrace{g_1^{\beta(m^*)}}_1 \cdot g^{\gamma(m^*)})} = \frac{\hat{e}(g, \sigma_1^*)}{\hat{e}((\sigma_2^*)^{\gamma(m^*)}, g)} \\ \implies \hat{e}(g, g^{ab}) &= \hat{e}\left(g, \frac{\sigma_1^*}{(\sigma_2^*)^{\gamma(m^*)}}\right) \implies g^{ab} = \frac{\sigma_1^*}{(\sigma_2^*)^{\gamma(m^*)}} \end{aligned}$$

Next, we claim that  $\mathcal{B}$  aborts with negligible probability.

To verify it, we see that  $\mathcal{B}$  aborts if the following happens:

$$E = \beta(m^*) \neq 0 \vee (\beta(m^*) = 0 \wedge \beta(m_1) = 0) \vee \dots \vee (\beta(m^*) = 0 \wedge \beta(m_{q_s}) = 0).$$

Since  $|\beta(m)| \leq kl$  by definition of  $\beta(\cdot) \implies |\beta(m)| < q \implies$  we drop the modulus. By the union bound

$$\Pr[E] \leq \Pr[\beta(m^*) \neq 0] + \sum_{i=1}^{q_s} \Pr[\beta(m^*) = 0 \wedge \beta(m_i) = 0].$$

- Fix  $x_1, \dots, x_k$  and  $m^*$ , there is a single  $x_0$  such that  $\beta(m^*) = 0$ , thus

$$\Pr[\beta(m^*) \neq 0] = \frac{kl}{kl+1};$$

- fix  $i \in [q_s]$ , and let  $\bar{m} = m_i$ . Since  $m^* \neq m_i$ ,  $\exists j : m^*[j] \neq \bar{m}[j]$ . Assume without loss of generality that  $m^*[j] = 1 \wedge \bar{m}[j] = 0$ . Given arbitrary  $x_1, \dots, x_{j-1}, x_{j+1}, \dots, x_k$  then

$$(\beta(m^*) = \wedge \beta(\bar{m}) = 0) \iff \begin{pmatrix} x_0 + x_j = -\sum_{i \neq j} x_i m^*[i] \\ x_0 = -\sum_{i \neq j} x_i \bar{m}[i] \end{pmatrix}$$

$\exists!$  solution.

$$\Pr [\beta(m^\star) = 0 \wedge \beta(m_i) = 0] = \frac{1}{kl+1} \cdot \frac{1}{l+1}.$$

So the probability of aborting is:

$$\Pr [E] \leq \frac{kl}{kl+1} + q_s \cdot \frac{1}{kl+1} \cdot \frac{1}{l+1}$$

which gives us the probability of not aborting:

$$\begin{aligned} \Pr [\bar{E}] &\geq 1 - \frac{kl}{kl+1} - q_s \cdot \frac{1}{kl+1} \cdot \frac{1}{l+1} \\ &= \frac{1}{kl+1} \left( kl + 1 - kl - \frac{q_s}{l+1} \right) = \frac{1}{kl+1} \left( 1 - \frac{q_s}{l+1} \right) \quad (l = kq_s) \\ &= \frac{1}{kl+1} \left( 1 - \frac{l}{(l+1)k} \right) \geq \frac{1}{4kq_s+2} = \frac{1}{p'(\lambda)} \end{aligned}$$

with  $p'(\lambda) \in \text{poly}(\lambda)$ .

Finally, the probability that  $\mathcal{B}$  breaks CDH:

$$\Pr [\mathcal{B} \text{ breaks CDH}] = \underbrace{\frac{1}{p(\lambda)}}_{\text{breaks WSS}} \cdot \underbrace{\frac{1}{p'(\lambda)}}_{\text{not abort}} \in \text{poly}(\lambda)$$

□

## Acronyms

<b>AU</b>	Almost Universal
<b>DL</b>	Discrete Log
<b>CBC</b>	Cypher Block Chain
<b>CCA</b>	Chosen Cyphertext Attack
<b>CDH</b>	Computational Diffie-Hellman
<b>CFB</b>	Cypher Feed Back
<b>CFP</b>	Claw-Free Permutation
<b>CPA</b>	Chosen Plaintext Attack
<b>CR</b>	Collision Resistant
<b>CRH</b>	Collision Resistant Hash Function
<b>CTR</b>	Counter
<b>DDH</b>	Decisional Diffie-Hellman
<b>ECB</b>	Electronic Code Book
<b>FIL</b>	Fixed Input Length
<b>GGM</b>	Goldreich-Goldwasser-Micali
<b>GL</b>	Goldreich-Levin
<b>HCP</b>	Hard Core Predicate
<b>ICM</b>	Ideal Cypher Model
<b>INT</b>	Integrity (of cyphertext)
<b>MAC</b>	Message Authentication Code
<b>MD</b>	Merkle-Damgard
<b>OTP</b>	One Time Pad
<b>OWF</b>	One Way Function
<b>OWP</b>	One Way Permutation
<b>PKC</b>	Public Key Cryptography
<b>PKE</b>	Public Key Encryption
<b>PPT</b>	Probabilistic Polynomial Time

**PRF** Pseudo Random Function  
**PRG** Pseudo Random Generator  
**PRP** Pseudo Random Permutation  
**PU** Perfect Universal  
**RV** Random Variable  
**SKE** Symmetric Key Encryption  
**SS** Signature Scheme  
**SSH** Secure Shell  
**TLS** Transport Layer Security  
**UFCMA** Unforgeable Chosen Message Attack  
**UHF** Universal Hash Function  
**VIL** Variable Input Length  
**WSS** Waters Signature Scheme