Contents

Lesson 1

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

1.1 Secret communication

Modern confidentiality/authentication systems are designed following *Kerckhoffs's principle*, which states that a secure system shall only rely on the encryption keys and not on the algorithm's secrecy. The principle is often condensed as *'no security by obscurity'*.

In symmetric-key encryption (SKE), the same cryptographic key is used for both encryption of plaintext and decryption of ciphertext. However, sharing the key between two parties without the risk of eavesdropping is a costly operation.

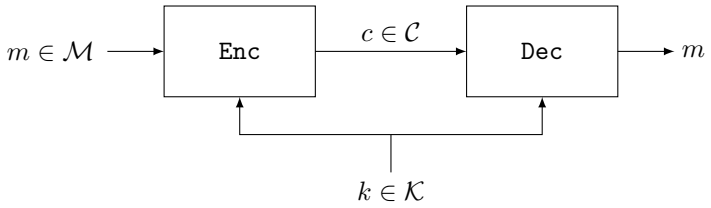


Figure 1.1: Symmetric-key encryption

1.1.1 Syntax

The typical objects defined and used throughout cryptography discourses are:

- the key space $\mathcal{K}$
- the message space $\mathcal{M}$
- the ciphertext space $\mathcal{C}$
- the encryption function $\texttt{Enc} : \mathcal{K} \times \mathcal{M} \to \mathcal{C}$
  $(k, m) \mapsto c$
- the decryption function $\texttt{Dec} : \mathcal{K} \times \mathcal{C} \to \mathcal{M}$
  $(k, c) \mapsto m$.

Enc and Dec form a cryptographic scheme, which must be correct:

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

1.1.2 Perfect secrecy

Definition 1 (Shannon's 'Perfect secrecy')**.** Let M be any distribution over the message space $\mathcal{M}$, C be any distribution over the cyphertext space $\mathcal{C}$ and K be a uniform distribution over the key space $\mathcal{K}$. Then, the cryptosystem ($\texttt{Enc}$, $\texttt{Dec}$) is deemed *perfectly secret* iff $\forall M, m \in \mathcal{M}\ \forall c \in C$

$$\Pr[M = m] = \Pr[M = m | C = c]\,. \tag{1.1}$$

◇

The cyphertext reveals nothing about the message, thus making the encryption method uncrackable even by adversaries with infinite computational power. This definition does not involve the encryption key because it must hold for all keys in the key space. It can be rephrased in different ways, bringing more details to light:

1. equation 1.1
2. $M \perp C$ (M and C are independent) (1.2)
3. $k \xleftarrow{\$} {}^{1}\mathcal{K} \implies \Pr[\texttt{Enc}(k, m_1) = c] = \Pr[\texttt{Enc}(k, m_2) = c]$. (1.3)

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

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

- (1) $\implies$ (2)

$$\begin{aligned}\Pr[C = c \wedge M = m] &= \Pr[C = c]\Pr[M = m | C = c] && \text{(Cond. prob.)}\\ &= \Pr[C = c]\Pr[M = m] && \text{(Using 1.1)}\end{aligned}$$

  Which implies that M and C are independent.

- (2) $\implies$ (3) fix M. Then,

$$\begin{aligned}\Pr[\texttt{Enc}(K, m_1) = c] &= \Pr[\texttt{Enc}(K, M) = c | M = m_1]\\ &= \Pr[C = c | M = m_1] && (C \text{ def.})\\ &= \Pr[C = c] && \text{(Using 1.2)}\\ &= \Pr[C = c | M = m_2] && \text{(Reverse steps with } m_2 \text{ instead of } m_1)\\ &= \Pr[\texttt{Enc}(K, M) = c | M = m_2]\\ &= \Pr[\texttt{Enc}(K, m_2) = c]\end{aligned}$$

[1] What does this dollar sign over arrow in function mapping mean?

- (3) $\implies$ (1)

$$\begin{aligned}
\Pr[C = c] &= \sum_m \Pr[C = c \wedge M = m] && \text{(Total prob.)} \\
&= \sum_m \Pr[\texttt{Enc}(K, M) = c \wedge M = m] && (C \text{ def.}) \\
&= \sum_m \Pr[\texttt{Enc}(K, M) = c | M = m] \Pr[M = m] && \text{(Cond. prob.)} \\
&= \sum_m \Pr[\texttt{Enc}(K, m) = c] \Pr[M = m] && \text{(Prob. collapse)} \\
&= \Pr[\texttt{Enc}(K, m') = c] \sum_m \Pr[M = m] && \text{(Using 1.3)} \\
&= \Pr[\texttt{Enc}(K, m') = c] && \text{(Total prob.)} \\
&= \Pr[\texttt{Enc}(K, M) = c | M = m'] && \text{(Cond. prob.)} \\
&= \Pr[C = c | M = m'] && (C \text{ def.})
\end{aligned}$$

Then,

$$\begin{aligned}
&\Pr[C = c] = \Pr[C = c | M = m] \\
&\implies \Pr[C = c] = \Pr[M = m | C = c] \frac{\Pr[C = c]}{\Pr[M = m]} && \text{(Bayes's th.)} \\
&\implies \Pr[M = m] = \Pr[M = m | C = c]
\end{aligned}$$

All three definitions for perfect secrecy are equivalent. □

1.1.3 One-time pad (OTP)

The one-time pad (OTP) is a cryptographic scheme such that

- $\mathcal{K} = \mathcal{M} = \mathcal{C} = \{0,1\}^l$
- $K \sim \text{unif}\{\mathcal{K}\}$
- $\texttt{Enc}(k, m) = k \oplus m$
- $\texttt{Dec}(k, c) = k \oplus c$.

The scheme is correct because $\texttt{Dec}(k, \texttt{Enc}(k, m)) = \texttt{Dec}(k, k \oplus m) = k \oplus k \oplus m = m$.

Theorem 1. *OTP is perfectly secret.* ◇

Proof. We have that $\forall m_1, m_2 \in \mathcal{M}$ and $\forall c \in \mathcal{C}$

$$\begin{aligned}
\Pr[\texttt{Enc}(K, m_1) = c] &= \Pr[K \oplus m_1 = c] \\
&= \Pr[K = c \oplus m_1] \\
&= |\mathcal{K}|^{-1} && (K \text{ is uniform}) \\
&= \ldots && \text{(Reverse steps with } m_1 \text{ instead of } m_2) \\
&= \Pr[\texttt{Enc}(K, m_2) = c]
\end{aligned}$$

This satisfies equation 1.3. □

By observing the proof of theorem 1, we can gain a deeper insight of OTP:

1. $|k| = |m|$ (key and message sizes match)
2. if a key is used multiple two or more times, an attacker may exploit XOR's idempotency to extract valuable information from the ciphertexts[2]:

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

Combined with the fact that keys must be pre-emptively shared in a secure fashion, these problems make OTP impractical.

1.1.4 Size of the key space

It is possible to generalise the analysis on OTP to all perfectly secret schemes.

Theorem 2. *In any perfectly secret SKE, the size of the key space must be greater than or equal to the size of the message space:*

$$|\mathcal{K}| \geq |\mathcal{M}|\,.$$

$\diamond$

Proof. Perfect secrecy will be disproved by breaking the first definition.
Define $M \sim \text{unif}\{\mathcal{M}\}$ and $c \in \mathcal{C} : \Pr[C = c] > 0$. Let $\mathcal{D} = \{\texttt{Dec}(k, c) : k \in \mathcal{K}\}$ be the set of all images of the decryption routine with all keys in $\mathcal{K}$ and assume $|\mathcal{K}| < |\mathcal{M}|$ (negation of the thesis). Then, using the definition of $\mathcal{D}$ we have

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

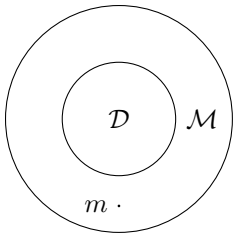


Figure 1.2: Visual representation of $\mathcal{D}$, $\mathcal{M}$ and m

Fix m. Using M's definition, we have $\Pr[M = m] = |\mathcal{M}|^{-1}$. Since $m \notin \mathcal{D}$, there can be no key in $\mathcal{K}$ such that $\texttt{Dec}(k, c) = m$. We can prove $\Pr[M = m|C = c] = 0$ by observing that C strictly distributes over $\mathcal{D}$, in which m is not present.
In conclusion, we have

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

that clearly violates the first definition of perfect secrecy. $\square$

[2] A cryptographic algorithm is said to be *malleable* if there exists a function f that transforms a ciphertext into another ciphertext which decrypts to a related plaintext (i.e. $f(c) = \texttt{Enc}(k, f(m))$). This concept will be explored further.

Lesson 2

2.1 Message authentication codes (MAC)

A message authentication code (MAC) allows to verify the integrity and the source of a message.

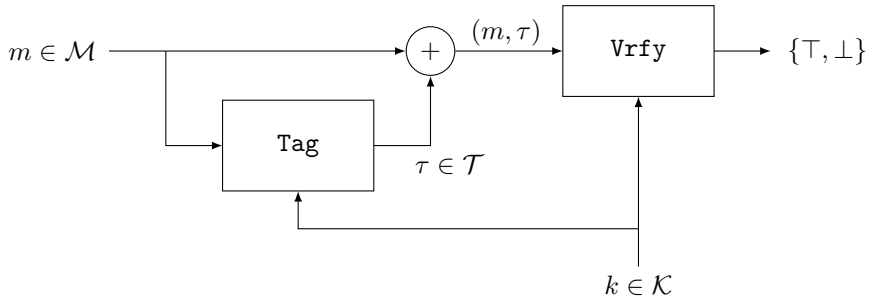


Figure 2.1: Message authentication code

2.1.1 Syntax

The typical objects defined and used throughout message-authentication discourses are:

- the key space $\mathcal{K}$
- the message space $\mathcal{M}$
- the tag space $\mathcal{T}$
- the random variable $K \sim \text{unif}\{\mathcal{K}\}$
- the tag function $\texttt{Tag} : \mathcal{K} \times \mathcal{M} \to \mathcal{T}$
  $$(k, m) \mapsto \tau$$
- the verify function $\texttt{Vrfy} : \mathcal{K} \times \mathcal{M} \times \mathcal{T} \to \{\top, \bot\}$
  $$(k, m, \tau) \mapsto \texttt{Tag}(k, m) = \tau \,.$$

Tag and Vrfy form a message-authentication scheme, which must be correct:

$$\forall m \in \mathcal{M}\ \forall k \in \mathcal{K}\ \forall \tau \in \mathcal{T}\ \texttt{Vrfy}(k, m, \tau) = \top \iff \texttt{Tag}(k, m) = \tau \,.$$

2.1.2 Properties

Since `Tag` is deterministic, there exists a canonical `Vrfy` that always produce the correct output. A MAC is unforgeable if it is computationally infeasible to find a valid tag of the given message without knowledge of the key. However, unforgeability does not imply that an adversary cannot forge tags: in the worst case, we assume the adversary can forge the tag of any message except the given one.

Definition 2 (ε-statistical one-time security)**.** We say that a MAC has ε-statistical one-time security if $\forall m, m' \in \mathcal{M} : m \neq m' \ \forall \tau, \tau' \in \mathcal{T}$

$$\Pr[\mathtt{Tag}(K, m') = \tau' | \mathtt{Tag}(K, m) = \tau] \leq \varepsilon \,. \tag{2.1}$$

$\diamond$

Suppose a computationally unbounded adversary chooses $m \in \mathcal{M}$ and obtains $\tau = \mathtt{Tag}(k, m)$. His goal is to forge a tag on a new message $m' \neq m$, i.e. finding a tag τ' such that $\mathtt{Tag}(k, m') = \tau'$, with no knowledge of k.

2.1.3 Pairwise independent hashing

Definition 3 (Pairwise independence)**.** A family of (hash) functions $H = \{h_s : \mathcal{M} \to \mathcal{T}\}_{s \in \mathcal{S}}$ is said to be pairwise independent iff, for any two distinct messages m and m', we have that $(h_s(m), h_s(m'))$ is uniform over $\mathcal{T}^2$ for a random choice of seed $s \in \mathcal{S}$:

$$\forall m, m' : m \neq m' \ \forall \tau, \tau' \in \mathcal{T} \ \Pr[h_s(m) = \tau \wedge h_s(m') = \tau'] = \frac{1}{|\mathcal{T}|^2} \,.$$

$\diamond$

Theorem 3. *Let p be a prime number and let $\mathcal{M} = \mathcal{T} = \mathcal{Z}_p = \{0, 1, \ldots, p-1\}$. Then, we can define the group $(\mathcal{Z}_p, +)$. Consider the function $h_{a,b}(x) = ax + b \mod p$ where $a, b \in \mathcal{Z}_p$. The family $H = \{h_{a,b}\}$ is pairwise independent.* $\diamond$

Proof. Fix $m, m' \in \mathcal{Z}_p : m \neq m'$ and $\tau, \tau' \in \mathcal{Z}_p$. Assume that a and b are random in $\mathcal{Z}_p$. Then,

$$\begin{aligned}
\Pr_{a,b}[h_{a,b}(m) = \tau \wedge h_{a,b}(m') = \tau'] &= \Pr_{a,b}[am + b = \tau \wedge am' + b = \tau'] \\
&= \Pr_{a,b}\left[\begin{pmatrix} m & 1 \\ m' & 1 \end{pmatrix} \cdot \begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} \tau \\ \tau' \end{pmatrix}\right] \\
&= \Pr_{a,b}\left[\begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} m & 1 \\ m' & 1 \end{pmatrix}^{-1} \cdot \begin{pmatrix} \tau \\ \tau' \end{pmatrix}\right] \\
&= \frac{1}{|\mathcal{Z}_p|^2} \\
&= \frac{1}{p^2}
\end{aligned}$$

where $\Pr_{a,b}$ denotes the probability over a and b random in $\mathcal{Z}_p$. $\square$

Theorem 4. *Let $\mathtt{Tag}(K, m) = h_k(m)$ for $\mathcal{H} = \{h_k : \mathcal{M} \to \mathcal{T}\}_k$ be pairwise independent. Then, the MAC is $\frac{1}{|\mathcal{T}|}$-statistical one-time secure.* $\diamond$

Proof.

$$\forall m \in \mathcal{M}\ \Pr[\mathtt{Tag}(K,m) = \tau] = \Pr_k[h_k(m) = \tau] = \frac{1}{|\mathcal{T}|}$$

$$\forall m, m' \in \mathcal{M} : m \neq m'\ \forall \tau \in \mathcal{T}\ \Pr[\mathtt{Tag}(K,m) = \tau \wedge \mathtt{Tag}(K,m') = \tau'] = \frac{1}{|\mathcal{T}|^2}$$

$$\begin{aligned}\Pr[\mathtt{Tag}(K,m') = \tau' | \mathtt{Tag}(K,m) = \tau] &= \frac{\Pr[Mac(K,m) = \tau \wedge Mac(K,m') = \tau']}{Pr[Mac(K,m) = \tau]} \\ &= \frac{|\mathcal{T}|^{-2}}{|\mathcal{T}|^{-1}} \\ &= \frac{1}{|\mathcal{T}|}\end{aligned}$$

□

Lemma 1. *A $2^{-\lambda}$-statistical t-times secure* MAC *has a key of size $(t+1)\lambda$ for all $\lambda \in \mathbb{N}^+$.* ◇

Lesson 3

3.1 Randomness Extraction

Randomness extraction is the process of extracting real randomness from imperfect randomness.

3.1.1 Von Neumann extractor

Take a biased coin B such that $\Pr[B = 0] = p \neq 1/2$ and $\Pr[B = 1] = 1 - p$. The Von Neumann extractor is the algorithm

Algorithm 1 Von Neumann extractor

do
  sample $b_1 \leftarrow B$
  sample $b_2 \leftarrow B$
while $b_1 \neq b_2$
return b_2

b_1 and b_2 are i.i.d, thus the probability of both outputs is $\Pr[b_1 = 0 \wedge b_2 = 1] = \Pr[b_1 = 1 \wedge b_2 = 0] = p(p-1)$. The probability of outputting something in a given iteration is $2p(p-1)$, therefore the probability of failing after n iterations is $\Pr[\text{fail } n \text{ times}] = [1-2p(1-p)]^n$, which is very small (it follows the geometric distribution).

3.1.2 Unseeded extraction

In real life there are sources of unpredictability and their randomness is measured through min-entropy. The min-entropy depends on the probability of the most frequent event, a sort of worst-case scenario for randomness.

Definition 4 (Min-entropy)**.** The min-entropy H_∞ of a random variable X is defined as $H_\infty(X) = -\log_2 \max \Pr[X = x]$. ◇

Example 1. Take $X = \text{unif}\{\{0,1\}^n\}$, then $H_\infty = -\log_2 \frac{1}{2^n} = n$. Now, take X such that $\Pr[X = 0^n] = 1$ and $\Pr[X \neq 0^n] = 0$ (it is always 0^n), then $H_\infty = -\log_2 1 = 0$.

Definition 5. An (n, k)-source is a random variable $X \in \{0,1\}^n$ such that $H_\infty(X) \geq k$. ◇

We want to create a function $\texttt{Ext}$ that extracts perfect randomness from a (n,k)-source that works for all possible (n,k)-sources. However, this is not possible even if $\texttt{Ext} : \{0,1\}^n \to \{0,1\}$ and $k = n-1$.

Proof. Let $\texttt{Ext} : \{0,1\}^n \to \{0,1\}$ be any candidate extractor. Let $b \in \{0,1\}$ such that $|\texttt{Ext}^{-1}(b)| \geq 2^{n-1}$ (the most frequent case).

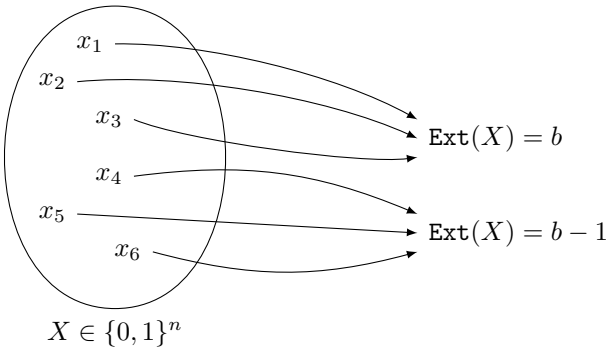


Figure 3.1: The candidate extractor $\texttt{Ext}$.

Let the random variable X_{bad} be uniform over $\texttt{Ext}^{-1}(b)$. Then, $\texttt{Ext}(X_{\text{bad}}) = b$ (always the same) and $H_\infty(X_{\text{bad}}) \geq n-1$. □

3.1.3 Seeded extraction

Definition 6 (Statistical distance)**.** Let X and X' be two random variables over the set $\mathcal{X}$. Their statistical distance SD is defined as

$$\text{SD}(X,X') = \frac{1}{2}\sum_{x\in\mathcal{X}} |\Pr[X=x] - \Pr[X'=x]| \,.$$

◇

It is 0 when X and X' follow the same distribution, it is 1 when they have no possible events in common.

Definition 7 (Seeded extractor)**.** Let $U_l \sim \text{unif}\{\{0,1\}^l\}$ and $S \sim \text{unif}\{\{0,1\}^d\}$. A function

$$\texttt{Ext} : \underbrace{\{0,1\}^d}_{\text{seed (public)}} \times \underbrace{\{0,1\}^n}_{\text{input}} \to \underbrace{\{0,1\}^l}_{\text{output}}$$

is a (k,ε)-extractor if $\forall X \in \{0,1\}^n : H_\infty(X) \geq k$

$$\text{SD}((S,\texttt{Ext}(S,X)),(S,U_l)) \leq \varepsilon \,.$$

◇

Even though the seed is public, the output of the extractor cannot be distinguished from a random string.

Lemma 2 (Leftover hash lemma)**.** *Let $\mathcal{H} = \{h_S : \{0,1\}^n \to \{0,1\}^l\}_{S\in\{0,1\}}$ be a pairwise independent family (that includes an extractor for each seed). Then, $\texttt{Ext}(s,x) = h_s(x)$ is a (k,ε)-extractor for $k \geq l + 2\log\frac{2}{\varepsilon} - 2$.* ◇

Informally, the leftover hash lemma states that the length of the output must be less than k. In order to prove it, we need several other elements.

Definition 8 (Collision probability)**.** Let Y and Y' two i.i.d random variables over a set $\mathcal{Y}$. Then, the collision probability of Y is defined as

$$\mathrm{Col}(Y) = \Pr[Y = Y'] = \sum_{y \in \mathcal{Y}} \Pr[Y = y]^2$$

and is the probability that both variables assume the same value. ◇

Exercise 5. Let X be a random variable such that $H_\infty(X) \geq k$. Then,

$$\mathrm{Col}(X) \leq 2^{-k}\,. \tag{3.1}$$

T0D0 1: Solution

Exercise 6. Let $\mathcal{H}$ be a pairwise-independent family, then

$$\Pr[h_S(X) = h_S[X'] \wedge X \neq X'] \leq 2^{-l}\,. \tag{3.2}$$

T0D0 2: Solution

Lemma 3 (Technical lemma for the leftover hash)**.** *Let Y be a random variable over a set $\mathcal{Y}$ such that its collision probability is at most $|\mathcal{Y}|^{-1} + 1 + 4\varepsilon^2$. Then,*

$$\mathrm{SD}(Y, \mathrm{unif}\{\mathcal{Y}\}) \leq \varepsilon\,.$$

◇

Proof. Define q_y as

$$q_y = \Pr[Y = y] - \frac{1}{|\mathcal{Y}|}$$

and let s_t be its sign:

$$s_y = \begin{cases} 1 & \text{if } q_y \geq 0 \\ 0 & \text{otherwise.} \end{cases}$$

Then, we can define the vectors $\overline{q_y} = (q_y)_{y \in \mathcal{Y}}$ and $\overline{s_y} = (s_y)_{y \in \mathcal{Y}}$, so that

$$\begin{aligned}
\mathrm{SD}(Y, \mathrm{unif}\{\mathcal{Y}\}) &= \frac{1}{2} \sum_{y \in \mathcal{Y}} \left| \Pr[Y = y] - \frac{1}{|\mathcal{Y}|} \right| \\
&= \frac{1}{2} \sum_{y \in \mathcal{Y}} q_y \cdot s_y \\
&= \frac{1}{2} \langle \overline{q_y}, \overline{s_y} \rangle \\
&\leq \frac{1}{2} \sqrt{\langle \overline{q_y}, \overline{q_y} \rangle \cdot \langle \overline{s_y}, \overline{s_y} \rangle} \qquad \text{(Law of cosines)} \\
&= \frac{1}{2} \sqrt{\sum_{y \in \mathcal{Y}} q_y \cdot |\mathcal{Y}|}
\end{aligned} \tag{3.3}$$

Let us focus only on q_y:

$$\begin{aligned}
\sum_{y\in\mathcal{Y}} q_y &= \sum_{y\in\mathcal{Y}} (\Pr[Y=y]^2 + \frac{1}{|\mathcal{Y}|^2} - 2\frac{\Pr[Y=y]}{|\mathcal{Y}|}) \\
&= \mathrm{Col}(Y) + \frac{1}{|\mathcal{Y}|} - \frac{2}{|\mathcal{Y}|} \\
&= \mathrm{Col}(Y) - \frac{1}{|\mathcal{Y}|} \\
&= \frac{(1+e\varepsilon^2)}{|\mathcal{Y}|} - \frac{1}{|\mathcal{Y}|} && \text{(By assumption)} \\
&= \frac{4\varepsilon^2}{|\mathcal{Y}|} && (3.4)
\end{aligned}$$

Substituting (3.4) in (3.3) we have

$$\begin{aligned}
\mathrm{SD}(Y, \mathrm{unif}\{\mathcal{Y}\}) &\leq \frac{1}{2}\sqrt{\frac{4\varepsilon^2}{|\mathcal{Y}|}|\mathcal{Y}|} \\
&= \varepsilon
\end{aligned}$$

□

Finally, we can prove lemma 2.

Proof of leftover hash lemma. Let $y = (S, h_S(x))$, $\mathcal{Y} = \{0,1\}^{d+l}$ and $y' = (S', h_{S'}(X'))$. Then,

$$\begin{aligned}
\mathrm{Col}(Y) &= \Pr[Y = Y'] \\
&= \Pr[S = S' \wedge h_S(x) = h_S(x')] \\
&= \Pr[S = S']\Pr[h_S(X) = h_S(X')] \\
&= 2^{-d}\Pr[h_S(X) = h_S(X')] && \text{(Using 3.1)} \\
&= 2^{-d}\Pr[X = X' \vee (X = X' \wedge h_S(x) = h_S(x'))] \\
&\leq 2^{-d}(\Pr[h_S(x) = h_S(x') \wedge X = X'] + \Pr[X = X']) && \text{(Boole's ineq.)} \\
&\leq 2^{-d}(2^{-k} + 2^{-l}) && \text{(By 3.1 and 3.2)} \\
&= \frac{1}{2^{d+l}}(2^{l-k} + 1) \\
&\leq \frac{1}{2^{d+l}}(2^{2-2\log\varepsilon^{-1}} + 1) \\
&= \frac{1}{|\mathcal{Y}|}(1 + 4\varepsilon^2)
\end{aligned}$$

By lemma 3 we have that $\mathrm{SD}(Y, \mathrm{unif}\{\mathcal{Y}\}) \leq \varepsilon$. □

Lesson 4

4.1 Negligible function

What is exactly a negligible function? Below here there is a possible interpretation of this notion:

> "In real life, we can just consider adversaries with limited computational power; even if every non-perfectly secure authentication scheme can resist to unbounded computational power, the true unbounded computational power doesn't exist at all. So, it's reasonable to consider just bounded adversaries.
>
> So consider a scheme Π where the only attack against it is brute-force attack. We consider Π to be secure if it cannot be broken by a brute-force attack in polynomial time.
>
> The idea of **negligible probability** encompasses this exact notion. In Π, let's say that we have a polynomial-bounded adversary. Brute force attack is not an option.
>
> But instead of brute force, the adversary can try (a polynomial number of) random values and hope to guess the right one. In this case, we define security using negligible functions: The probability of success has to be smaller than the reciprocal of any polynomial function.
>
> And this makes a lot of sense: if the success probability for an individual guess is a reciprocal of a polynomial function, then the adversary can try a polynomial amount of guesses and succeed with high probability. If the overall success rate is $\frac{1}{poly(\lambda)}$ then we consider this attempt a feasible attack to the scheme, which makes the latter insecure.
>
> So, we require that the success probability must be less than the reciprocal of every polynomial function. This way, even if the adversary tries $poly(\lambda)$ guesses, it will not be significant since it will only have tried: $\frac{poly(\lambda)}{superpoly(\lambda)}$[3]
>
> As λ grows, the denominator grows far faster than the numerator and the success probability will not be significant[4]."

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

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

Definition

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

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

Exercise 7. Let $p(\lambda), p'(\lambda) \in poly(\lambda)$ and $\nu(\lambda), \nu'(\lambda) \in \text{negl}(\lambda)$. Then prove the following:

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

Solution 1 (1.1)**.** T0D0 3: Questa soluzione usa disuguaglianze deboli; per essere negligibile una funzione dev'essere strettamente minore di un polinomiale inverso. Da approfondire

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

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

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

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

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

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

4.2 One-Way Functions

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

Definition 9. Let $f : \{0,1\}^{n(\lambda)} \to 0, 1^{n(\lambda)}$ be a function. Then it is a OWF iff:

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

◇

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

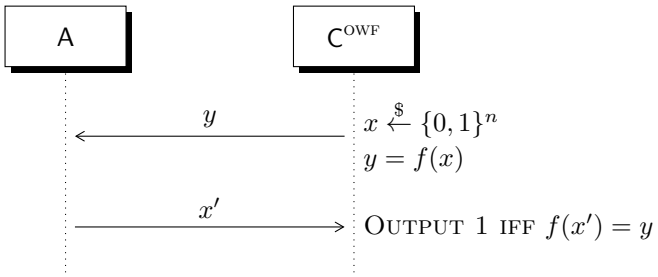


Figure 4.1: One-Way Function hardness

Exercise 8.

1. Show that there exists an inefficient adversary that wins GAME^{OWF} with probability 1
2. Show that there exists an efficient adversary that wins GAME^{OWF} with probability 2^{-n}

One-Way puzzles

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

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

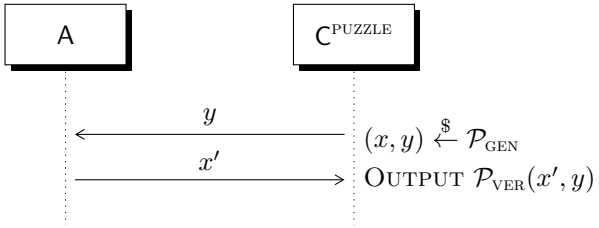


Figure 4.2: The puzzle game

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

4.2.1 Impagliazzo's Worlds

Suppose to have Gauss, a genius child, and his professor. The professor gives to Gauss some mathematical problems, and Gauss wants to solve them all.

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

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

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

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

  The professor, with some effort, can create a game difficult enough, but Gauss will solve it anyway; here there are some problems that the professor cannot find a solution to

- *Pessiland*: NP problems are hard on average but no one-way functions exist

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

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

4.3 Computational Indistinguishability

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

Definition 10. X and Y are **computationally indistinguishable** ($X \approx_c Y$) if $\forall$ PPT $D, \exists\nu(\lambda) \in \text{negl}(\lambda)$ such that:

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

$\diamond$

T0D0 4: AP181129-2344: There may be room for improvement, but I like how it's worded: it puts some unusual perspective into the cryptographic game, and it could be a good thing since it closely precedes our first reduction, and the whole hybrid argument mish-mash.

Suppose we have this mental game: a Distinguisher D receives the value z. This value has been chosen by me, the Challenger, among X_λ and Y_λ, and the Distinguisher has to *distinguish* which was the source of z. What does this formula mean?

This formula means that, fixed 1 as one of the sources, the *probability* that D says "1!" when I pick z from X_λ is **not so far** from the *probability* that D says "1!" when I pick z from Y_λ.

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

What's the deep meaning of this formula? This is something to do.

Lemma 4. $\forall$ PPT f, $X \approx_c Y \implies f(x) \approx_c f(y)$ $\diamond$

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

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

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

.

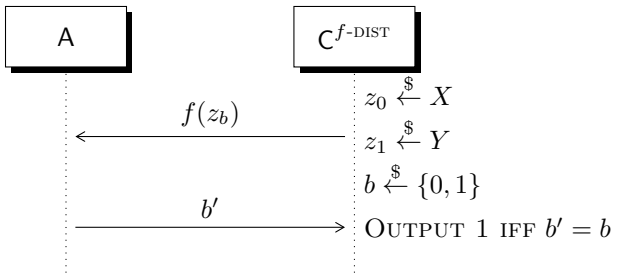


Figure 4.3: A distinguisher for f

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

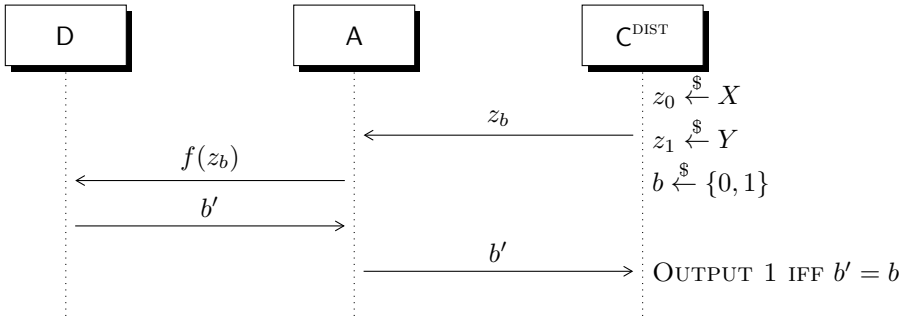


Figure 4.4: Distinguisher reduction

□

4.4 Pseudorandom Generators

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

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

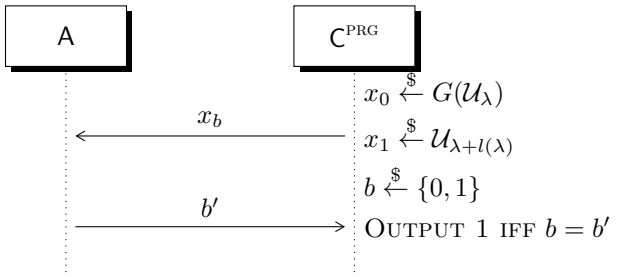


Figure 4.5: The pseudorandom game

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

Lesson 5

5.1 Stretching a PRG

Consider this algorithm that uses G to construct G^l, as depicted in figure 5.1:

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

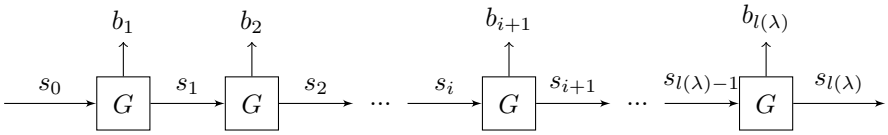


Figure 5.1: Constructing $G^{l(\lambda)}$ form $G(\lambda)$

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

Lemma 5. *(**Hybrid Argument**) Let $X = \{X_n\}, Y = \{Y_n\}, Z = \{Z_n\} : X \approx_c Y \wedge Y \approx_c Z$. Then $X \approx_c Z$.* ◇

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

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

□

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

Theorem 9. *If there exists a PRG $G(\lambda)$ with one bit stretch, then there exists a PRG $G^{l(\lambda)}$ with polynomial stretch relative to its input length:*

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

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

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

Focusing on two subsequent generic hybrids, as shown in figures 5.2 and 5.3, it can be observed that the only difference between the two resides in how b_{i+1} is generated: in H^i it comes from an instance of G, whereas in H^{i+1} is chosen at random. H^0_λ is the starting point where all bits are pseudorandom, which coincides with the $G^{l(\lambda)}$, and $H^{l(\lambda)}_\lambda$ will generate a totally random string.

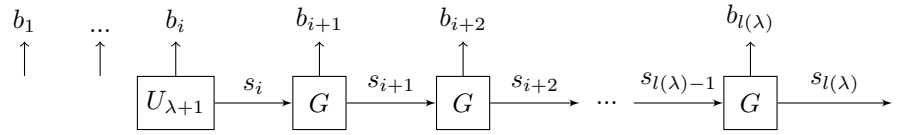


Figure 5.2: H^i_λ

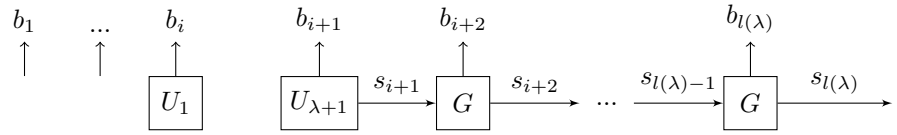


Figure 5.3: H^{i+1}_λ

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

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

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

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

Since by PRG definition $G(U_\lambda) \approx_c U_{\lambda+1}$, by lemma 4, we can deduce that $f_i(G(U_\lambda)) \approx_c f_i(U_{\lambda+1})$, which in turn, by how f is defined, implies $H^i \approx_c H^{i+1}$. This holds for an arbitrary choice of i, so by extension:

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

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

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

Suppose G^l is not a PRG; then there must be a point in the hybrid chain $H_\lambda^0 \approx_c \ldots \approx_c H_\lambda^l$ where $H_\lambda^i \not\approx_c H_\lambda^{i+1}$. Thus there exists a distinguisher D' able to tell apart H_λ^i from H_λ^{i+1}, as shown in figure 5.4:

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

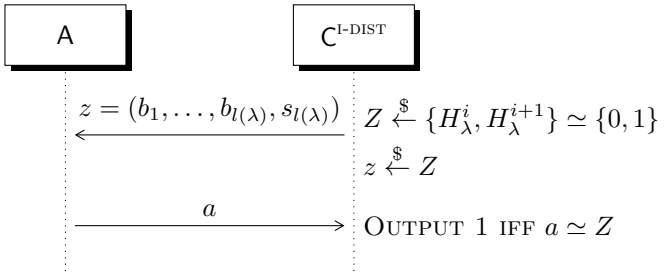


Figure 5.4: Distinguisher for H_λ^i and H_λ^{i+1}

If such a distinguisher exists, it can be also used to distinguish an output of G from a $\lambda + 1$ uniform string by "crafting" a suitable bit sequence, which will distribute exactly as the hybrids in question, as shown in the reduction in figure 5.5. This contradicts the hypothesis of f being a PRG, which by definition is to be indistinguishable from a truly random distribution. Therefore, G^l is indeed a PRG.

□

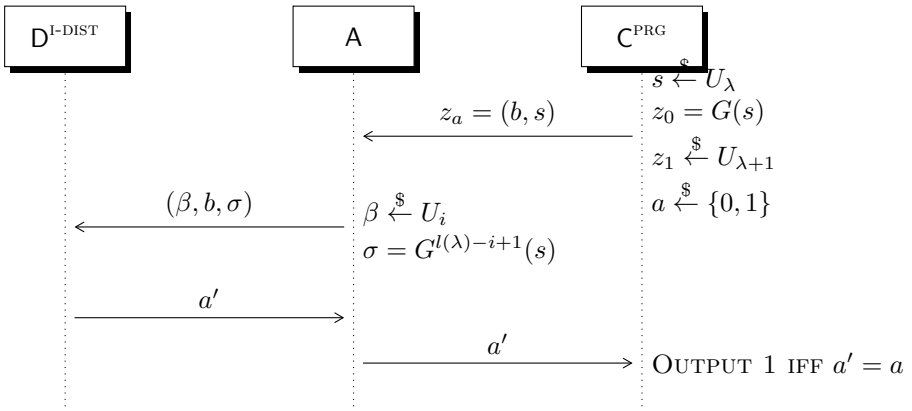


Figure 5.5: Reducing to a distinguisher for G, where $\beta = (b_1, \ldots, b_{i-1})$ and $\sigma = (b_{i+1}, \ldots, b_{l(\lambda)}, s_{l(\lambda)})$

5.2 Hardcore predicate

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

Consider a typical one-way function f, s.t. $f(x) = y$.

Question 1. Which bits of the input x are hard to compute given $y = f(x)$?

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

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

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

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

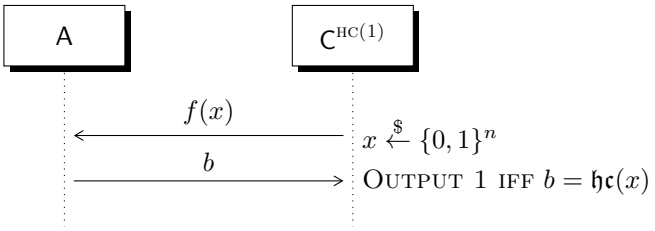


Figure 5.6: The hardcore game, f and $\mathfrak{hc}$ are known

There is also an alternative definition:

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

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

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

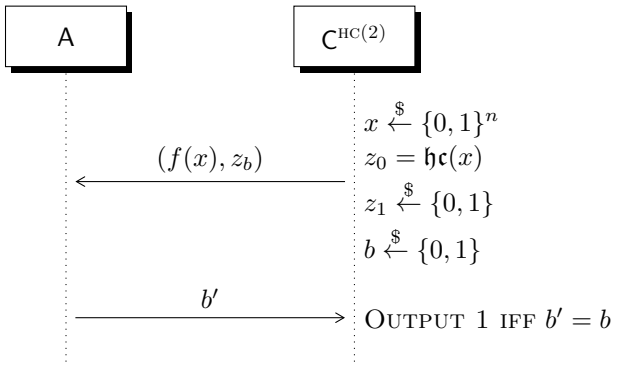


Figure 5.7: Another hardcore game, f and $\mathfrak{hc}$ are known

Having made this definition, some observations are in order:

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

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

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

Theorem 10 (Goldreich-Levin, '99)**.** *Let f be a OWF and consider $g(x,r) = (f(x), r)$ for $r \in \{0,1\}^n$. Then g is a OWF and:*

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

is hard core for g. ◇

Proof. T0D0 5: TO BE COMPLETED (...did we actually do this? è una bella menata dimostrare questo)

□

Exercise 11. Prove that g is a OWF if f is a OWF (Hint: do a reduction).

5.3 One Way Permutation

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

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

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

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

is a PRG.

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

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

□

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

Lesson 6

6.1 Computationally secure encryption

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

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

This scheme should have these properties:

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

.

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

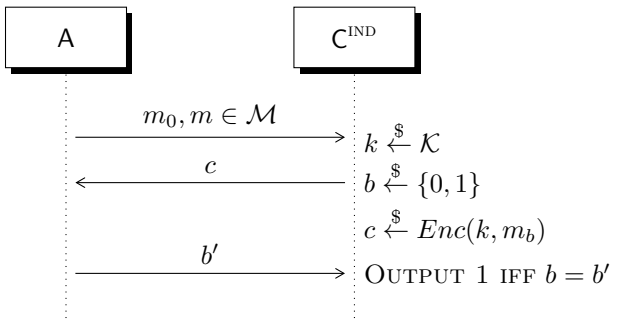


Figure 6.1: $\text{Game}^{\text{IND}}_{\Pi,\mathsf{A}}(\lambda, b)$

Definition 11. Π is said to be computationally **one time secure** iff:

$$\text{Game}^{\text{IND}}_{\Pi,\mathsf{A}}(\lambda, 0) \approx_{\text{C}} \text{Game}^{\text{IND}}_{\Pi,\mathsf{A}}(\lambda, 1)^{5}$$

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

or, alternatively $\forall$ PPT A:

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

$\diamond$

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

T0D0 6: TO BE REVIEWED

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

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

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

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

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

and thus, by XORing the result with $\bar{m}$, obtain the entire unknown message[6]. Nevertheless, this scheme is still **one time secure**:

Theorem 12. *If G is a PRG, then $\Pi_\oplus$ is computationally **one time secure*** $\diamond$

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

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

Consider the hybrid game in figure 6.2, where the original encryption routine is changed to use a random value instead of $G(k)$[7]. Compare with the original **one time secure** definition in figure 6.1 to observe that it perfectly matches.

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

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

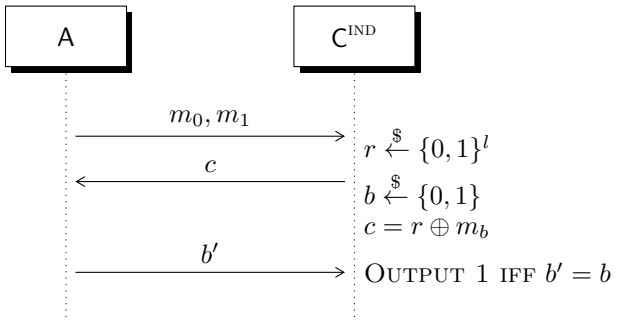


Figure 6.2: $\text{HYB}_{\Pi_\oplus,\mathsf{A}}(\lambda, b)$

Lemma 6. $\text{HYB}_{\Pi_\oplus,\mathsf{A}}(\lambda, 0) \equiv \text{HYB}_{\Pi_\oplus,\mathsf{A}}(\lambda, 1)$ ◇

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

T0D0 7: Eh?

□

Lemma 7. $\forall b \in \{0,1\}, HYB_{\Pi_\oplus,A}(\lambda, b) \approx_c GAME^{IND}_{\Pi_\oplus,A}(\lambda, b)$ ◇

Proof. The proof proceeds by reduction as depicted in figure 6.3, by assuming there exists a distinguisher D^{IND} for $c = G(k) \oplus m_b$ and $c = r \oplus m_b$, and using it to break the PRG itself.

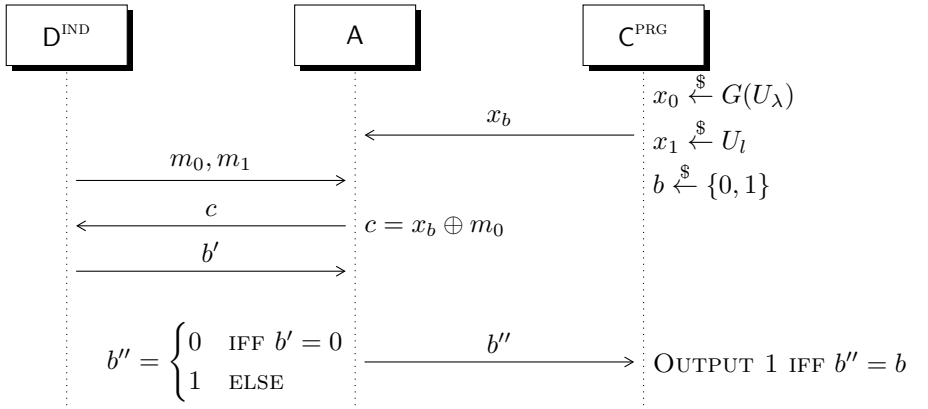


Figure 6.3: Reducing to breaking a PRG

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

- if D^{IND} is wrong, then A can deduce with high probability that the value given by C^{PRG} is in fact truly random;

- if D^{IND} is right, then A has a sensibly greater probability of having received a pseudorandom value from C^{PRG}.

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

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

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

□

6.2 Pseudorandom functions

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

Definition 12. A random function $R : \{0,1\}^n \to \{0,1\}^l$ is a function that, depending on what is known about its previous applications:

- if x is "fresh" (in formal terms, R has never been applied to x beforehand), then a value y is chosen UAR from R's codomain, and it is permanently associated as the image of x in R[8];
- if x is not fresh, then $R(x)$ is directly returned instead.

◇

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

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

Usually, PRFs come as function families F_k, where k is a key that identifies a single function inside the family itself[9].

With this in mind, let $\mathcal{F} = \{F_k : \{0,1\}^{n(\lambda)} \to \{0,1\}^{l(\lambda)}\}_{k \in \{0,1\}^\lambda}$ be the *keyed* function family that defines a PRF. Consider the two games in figure 6.4, each one involving respectively $\mathcal{F}$ and a random function R, where $\mathcal{R} = \{R : \{0,1\}^n \to \{0,1\}^l\}$, also written as $\mathcal{R}(\lambda, n, l)$ is the set of all random functions from n-bit strings to l-bit strings.

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

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

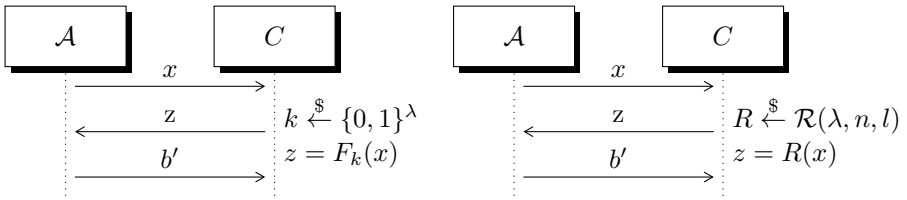


Figure 6.4: $Real_{\mathcal{F},\mathcal{A}}(\lambda)$ vs $Rand_{\mathcal{R},\mathcal{A}}(\lambda)$

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

Definition 13. F_k is a PRF iff:

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

$\diamond$

Exercise 13. Show that no PRG is secure against **unbounded attackers**.

Exercise 14. Show the same as above, but for any PRF instead.

6.2.1 GGM-tree

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

Construction 1. Let $G : \{0,1\}^\lambda \to \{0,1\}^{2\lambda}$ be a PRG such that it doubles the length of its input, and splits ithe images in halves, effectively returning string couples:

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

Consider the GGM-tree depicted in figure 6.5, which describes how G will be used to craft a PRF. To lighten up the notation, a function composition $G_a(G_b(G_c(k)))$ will be contracted to $G_{abc}(k)$. So let $\mathcal{F} = \{F_k : \{0,1\}^n \to \{0,1\}^\lambda\}$ be a family of functions such that:

$$F_k(r) = G_{r_n}(G_{r_{n-1}}(\dots G_{r_2}(G_{r_1}(k))\dots))$$

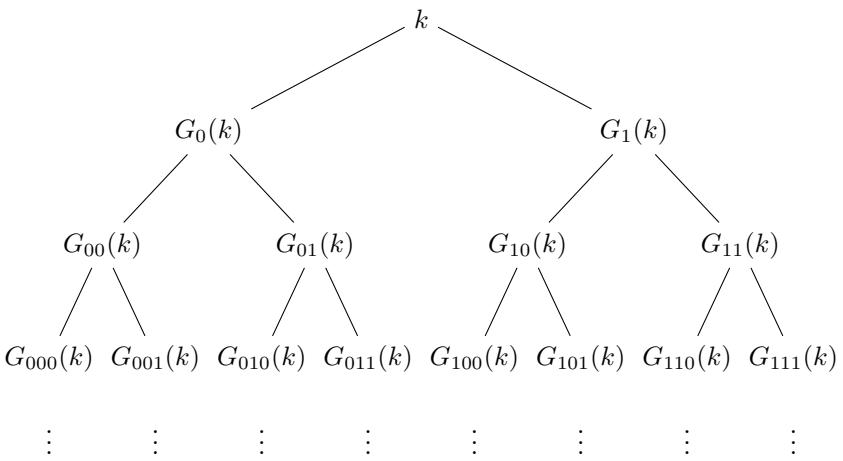


Figure 6.5: The GGM-tree for G

◇

An example, with a string value of 001 for r would evaluate as $F_k(001) = G_0(G_0(G_1(k)))$.

Lesson 7

Recall the GGM-tree construct used to define a function family $\mathcal{F} := \{F_k\}$ from a PRG G. From now on we will refer to this function family as F_k, implicitly stating that k represents a uniform probability distribution. Our wish is that F_k is indeed a PRF.

Theorem 15. *If G is a PRG, then F_k is a PRF.* ◇

Proof. Concerning F_k's efficiency, one can observe that it involves computing G a polynomial number of times, and G itself is efficient, thus F_k is indeed easy to compute.

The rest of the proof, regarding F_k's hardness, will proceed by induction over F_k's domain length n, which indirectly defines its GGM-tree's height.

Base case ($n = 1$): F_k's domain is restricted to $\{0,1\}$, meaning that its images will be respectively the two halves on a single iteration of $G(k)$, which are pseudorandom by G's definition:

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

therefore, in this case, F_k is pseudorandom.

Inductive step: Let $F'_k : \{0,1\}^{n-1} \to \{0,1\}^\lambda$ be a PRF. Define $F_k(x,y)$ as $G_x(F'_k(y))$, where $F_k : \{0,1\}^n \to \{0,1\}^\lambda$ and $x \in \{0,1\}$. It must be proven that if F'_k is a PRF, then so is F_k. We will proceed by hybrid games, depicted in figures 7.1, 7.2 and 7.3:

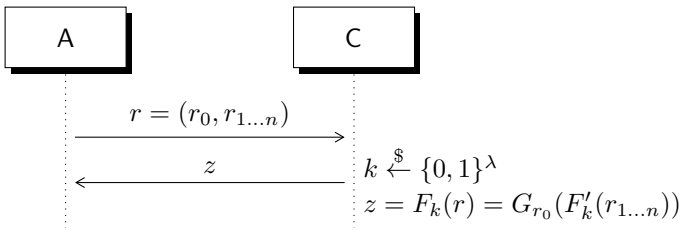


Figure 7.1: $\text{HYB}^0_{\mathcal{F},\mathsf{A}}(\lambda) = \text{GAME}^{\text{PRF}}_{\mathcal{F},\mathsf{A}}(\lambda)$

Lemma 8. $HYB^0_{\mathcal{F},A}(\lambda) \approx_c HYB^1_{\mathcal{R},A}(\lambda)$ ◇

Proof. Assume $\exists$ PPT D that can distinguish F_k from H; then an adversary A can use D as in figure 7.4 to break the induction hypothesis (i.e. can distinguish F'_k from $\overline{R}$).

□

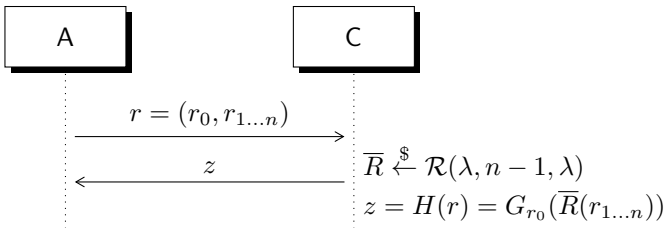


Figure 7.2: $\text{HYB}^{1}_{\overline{\mathcal{R}},\mathsf{A}}(\lambda)$

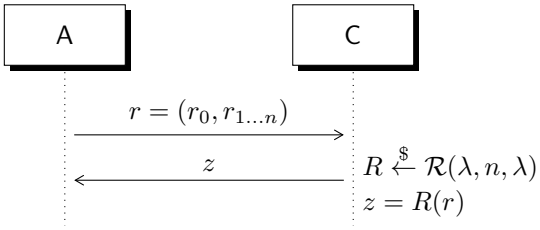


Figure 7.3: $\text{HYB}^{2}_{\mathcal{R},\mathsf{A}}(\lambda)$

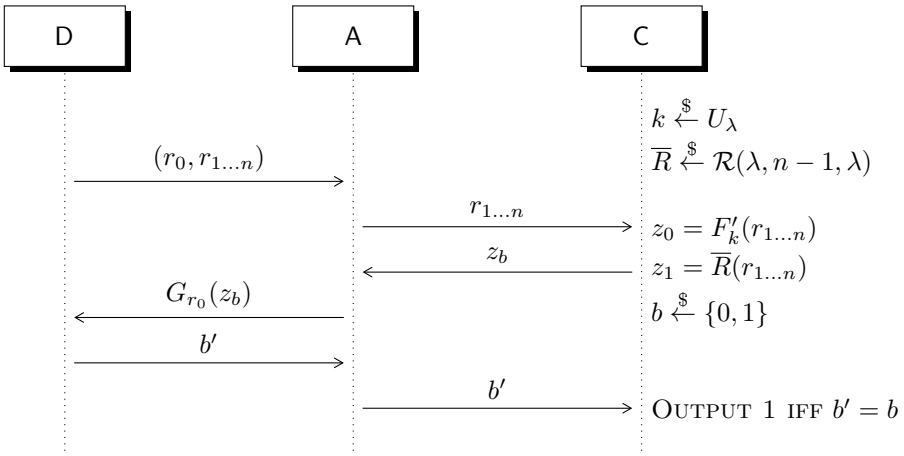


Figure 7.4

Before tackling the reduction from the last two hybrids, it is best to introduce another lemma:

Lemma 9. *If* $G : \{0,1\}^\lambda \to \{0,1\}^{2\lambda}$ *is a PRG, then for any* $t(\lambda) \in \text{poly}\lambda$*:*

$$(G(k_1), \ldots, G(k_t)) \approx_c (U_{2\lambda}, \ldots, U_{2\lambda}) \quad \forall k_i \xleftarrow{\$} U_\lambda$$

◇

Proof. T0D0 8: Idea: all values are independent and pseudorandom on their own, hybridize progressively...

□

Now for the final lemma:

Lemma 10. $HYB^1_{\overline{\mathcal{R}},A}(\lambda) \approx_c HYB^2_{\mathcal{R},A}(\lambda)$ ◇

Proof. T0D0 9: Bad proof: the idea is to reduce to a distinguishing game for the previous lemma)

~~Consider the distinguishing game for H and R in figure 7.5. The random functions $\overline{R}$ and R are entirely independant, and G transforms randomness in pseudorandomness, so this game boils down into distinguishing pseudorandom values from random ones, which is possible only with negligible probability.~~

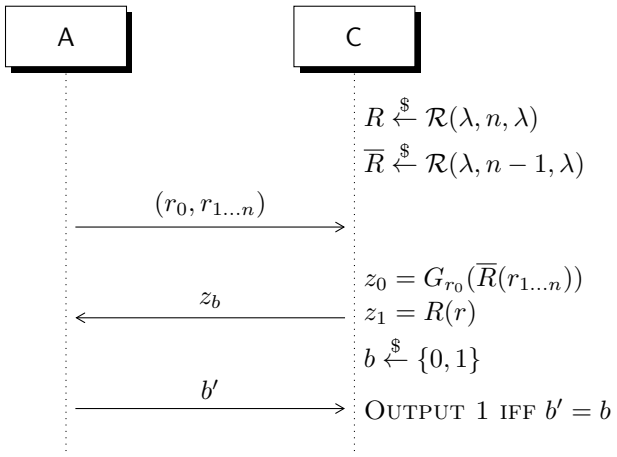


Figure 7.5: The distinguishing game between H in $\text{HYB}^1_{\Pi,A}$ and R in $\text{HYB}^2_{\Pi,A}$

□

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

□

7.1 CPA-security

Now it's time to define a stronger notion of security, which is widely used in cryptology for first assessments on cryptographic schemes. Let $\Pi := (Enc, Dec)$ be a SKE scheme, and consider the game depicted in figure 7.6. Observe that this time, the adversary can "query" the challenger for the ciphertexts of any messages of his choice, with the only reasonable restriction that the query amount must be polynomially bound by λ. This kind of game/attack is called the *Chosen Plaintext Attack*, because of the adversary's capability of obtaining ciphertexts from messages. The usual victory conditions found in n-time security games, which are based on ciphertext distinguishability, apply.

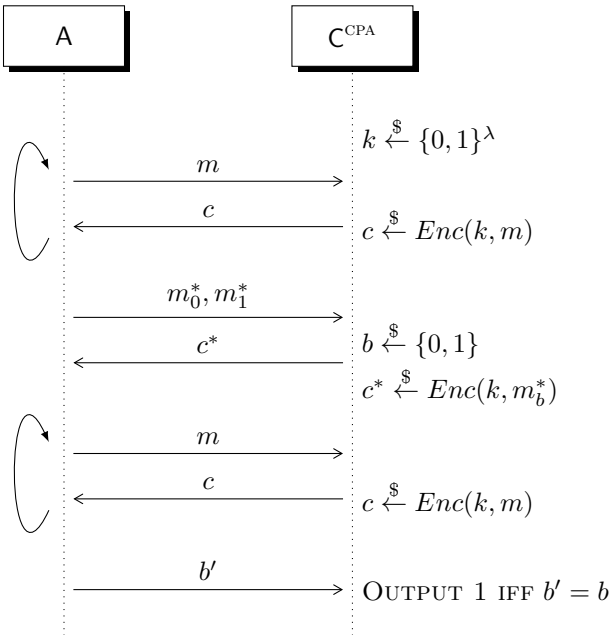


Figure 7.6: The CPA-security game: $\text{GAME}^{\text{CPA}}_{\Pi,\mathsf{A}}(\lambda, b)$

Definition 14. A scheme is CPA-secure if $\text{GAME}^{\text{CPA}}_{\Pi,\mathsf{A}}(\lambda, 0) \approx_c \text{GAME}^{\text{CPA}}_{\Pi,\mathsf{A}}(\lambda, 1)$ ◇

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

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

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

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

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

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

Theorem 16. *If $\mathcal{F}$ is a PRF, then $\Pi_{\mathcal{F}}$ is CPA-secure.* ◇

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

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

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

Lemma 11. $\forall b \in \{0,1\} \implies HYB^0_{\Pi,\mathcal{A}}(\lambda, b) \approx_c HYB^1_{\Pi,\mathcal{A}}(\lambda, b)$. ◇

Proof. As usual, the proof is by reduction: suppose there exists a distinguisher D capable of telling the two hybrids apart; then D can be used to break $\mathcal{F}$'s property of being a PRF. The way to use D is to make it play a CPA-like game, as shown in figure 7.7[10], where the adversary attempting to break $\mathcal{F}$ decides which message to encrypt between m_0 and m_1 beforehand, and checks whether D guesses which message has been encrypted. Either way, the adversary can get a sensible probability gain in guessing if the received values from the challenger were random, or generated by $\mathcal{F}$. Thus, assuming such D exists, A can efficiently break $\mathcal{F}$, which is absurd.

□

[10] An observant student may notice a striking similarity with a previously exposed reduction in figure 6.3

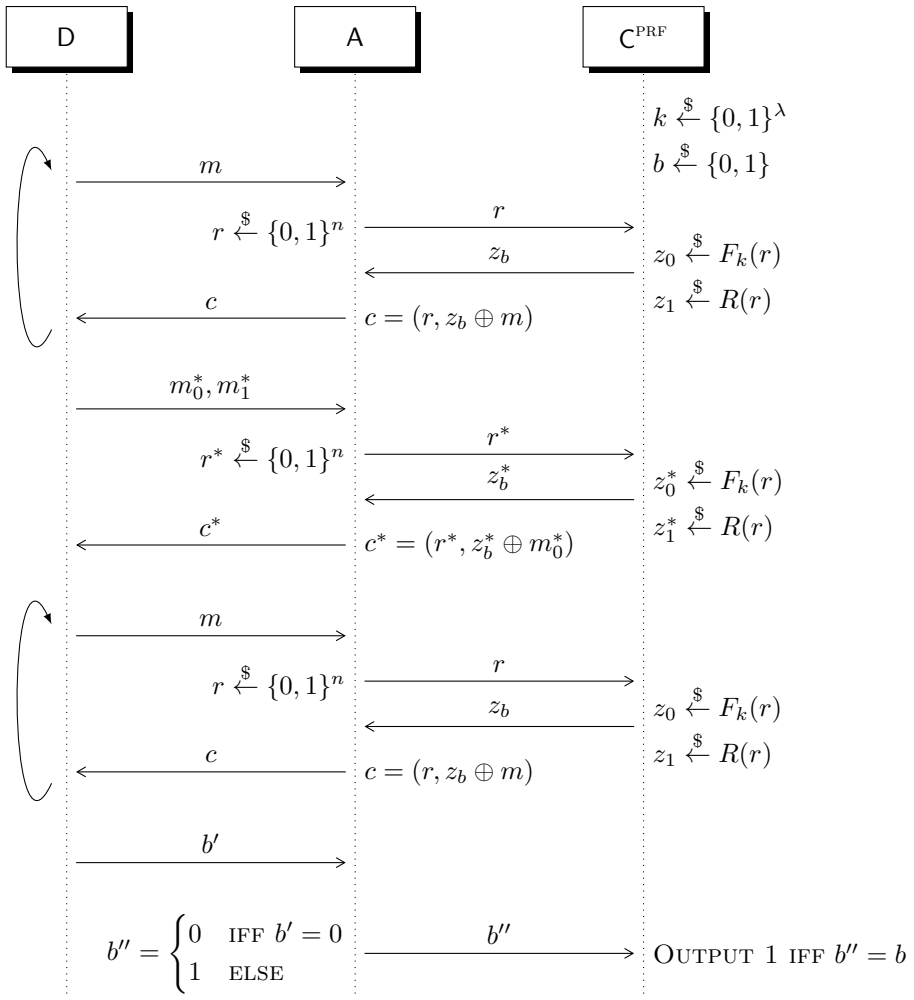


Figure 7.7: Breaking a PRF, for fixed message choice of m_0

Lemma 12. $\forall b \in \{0,1\} \implies HYB^{1}_{\Pi,\mathcal{A}}(\lambda, b) \approx_c HYB^{2}_{\Pi,\mathcal{A}}(\lambda, b)$. $\diamond$

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

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

Call REPEAT this collision event on $\overline{r}$ between 2 consecutive games. Then:

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

which proves that the REPEAT influences negligibly on the two hybrids' equivalence. Thus $\text{HYB}^{1}_{\Pi,\mathsf{A}}(\lambda, b) \approx_{\text{C}} \text{HYB}^{2}_{\Pi,\mathsf{A}}(\lambda, b)$[11].

□

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

□

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

Lesson 8

8.1 Domain extension

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

8.1.1 Electronic Codebook mode

The operation of ECB-mode is straightforward: Given a message split into blocks $(m_1, \ldots, m_t)$, apply the scheme's encryption routine to each block, as shown in figure 8.1:

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

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

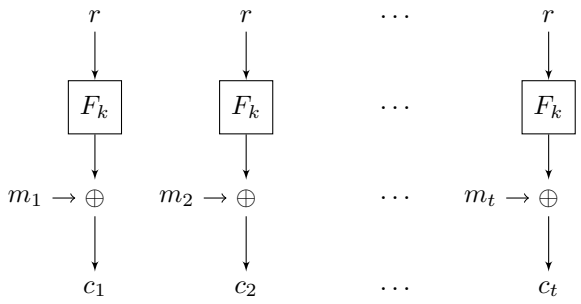


Figure 8.1: ECB-mode block-cipher in action, using a PRF as the encryption routine

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

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

8.1.2 Cipher block chaining mode (CBC)

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

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

This time, a *pseudorandom permutation*(PRP) is used instead of a PRF; they will be discussed later on. The diagram in figure 8.2 shows a general view of CBC-mode's operation. The decryption process is analogous but in a reversed fashion, by computing the preimage of a ciphered block and XOR-ing it with the preceding ciphered block:

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

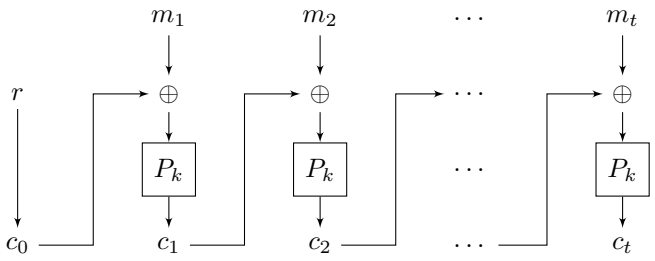


Figure 8.2: CBC-mode block-cipher in action, using a PRP as the encryption routine

8.1.3 Counter mode

This mode closely resembles ECB-mode but uses a "rolling" nonce instead of a static one, as shown in figure 8.3. At each successive block, the nonce is incremented by 1 and then used in a single block encryption. Since the nonce is in $\{0,1\}^n$, the increment is done modulo 2^n so that the value will wrap around to 0 it it ever overflows. Decryption is analogous.

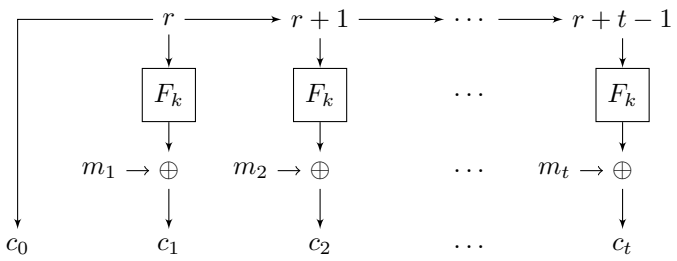


Figure 8.3: Counter-mode block-cipher in action, using a PRF as the encryption routine

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

Theorem 17. *Assume F_k is a PRF, then the counter-mode block cipher (CTR) is CPA-secure for variable length messages*[12]. ◇

Proof. Figure 8.4 models a CPA attack to a counter-mode block-cipher. The proof will proceed by hybrid argument starting from this game, therefore the statement to verify will be $\text{GAME}^{\text{CPA}}_{\text{CTR},\mathsf{A}}(\lambda, 0) \approx_{\text{C}} \text{GAME}^{\text{CPA}}_{\text{CTR},\mathsf{A}}(\lambda, 1)$.

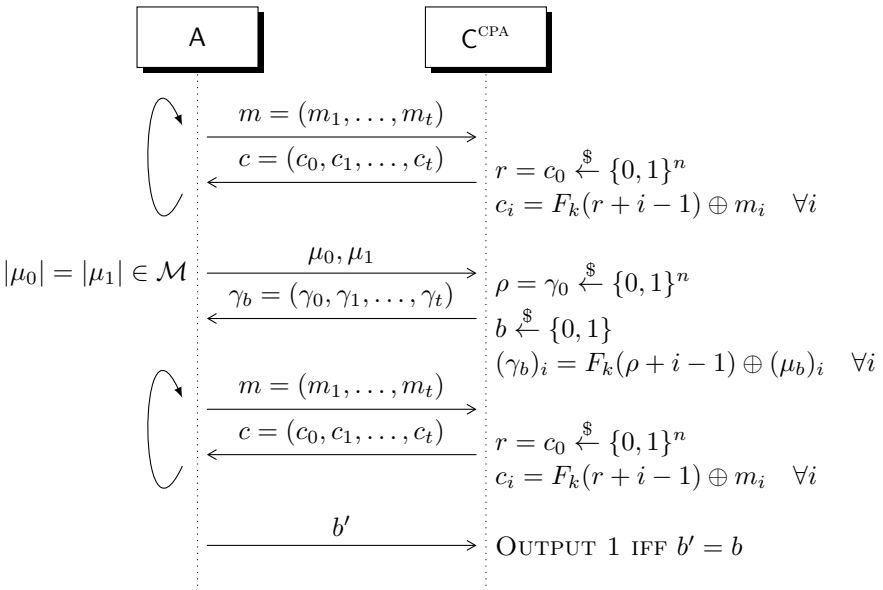


Figure 8.4: A chosen plaintext attack to counter-mode block-cipher

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

- $\text{HYB}^1_{\text{CTR},\mathsf{A}}(\lambda, b)$: A random function R is chosen UAR from $\mathcal{R}(\lambda, n, n)$ at the beginning of the game, and is used in place of F_k in all block encryptions;
- $\text{HYB}^2_{\text{CTR},\mathsf{A}}(\lambda, b)$: The challenger will pick random values from $\{0,1\}^n$ as ciphered blocks, disregarding any encryption routine.

Lemma 13. $\mathit{GAME}^{\mathit{CPA}}_{\mathit{CTR},\mathsf{A}}(\lambda, b) \approx_{\mathit{C}} \mathit{HYB}^1_{\mathit{CTR},\mathsf{A}}(\lambda, b) \quad \forall b \in \{0,1\}$ ◇

Proof. The proof is left as exercise.

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

Lemma 14. $\mathit{HYB}^1_{\mathit{CTR},\mathsf{A}}(\lambda, b) \approx_{\mathit{C}} \mathit{HYB}^2_{\mathit{CTR},\mathsf{A}}(\lambda, b) \quad \forall b \in \{0,1\}$ ◇

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

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

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

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

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

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

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

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

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

Then:

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

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

□

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

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

□

Lesson 9

9.1 Message Authentication Codes and unforgeability

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

The desirable property that a MAC scheme should hold is to prevent any attacker from generating a valid couple (m^*, ϕ^*), even after querying a tagging oracle polynomially many times[13]. The act of generating a valid couple from scratch is called *forging*, and the aforementioned propery is defined as *unforgeability against chosen-message attacks* (or UFCMA, in short); its game diagram is shown in figure 9.1. Do note that m^* is stated to be outside the query set M, expressing the "freshness" of the forged couple[14]. In formal terms:

Definition 15. A MAC scheme Π is UFCMA-secure iff:

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

◇

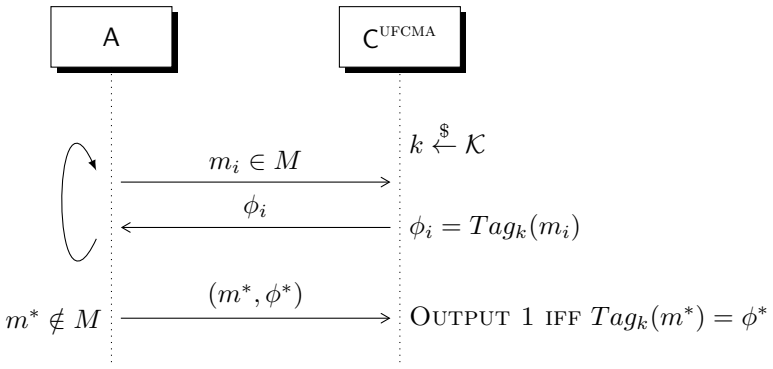


Figure 9.1: $\text{GAME}^{\text{UFCMA}}_{\Pi,\mathsf{A}}(\lambda)$

[13] Do note how this property resembles CPA-security in the encryption setting
[14] Observe how this setup resembles the original OWF game

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

Theorem 18. *Let* Π *be a* MAC *scheme such that* $Tag_k = Verify_k = F_k$*, where* F_k *is a* PRF*. Then* Π *is* UFCMA*-secure.* ◇

Proof. The usual proof by randomic hybridization entails. The original game is identical to the UFCMA game, where the tagging function is the PRF, whereas the hybrid game will have it replaced with a truly random function, as shown in figure 9.2.

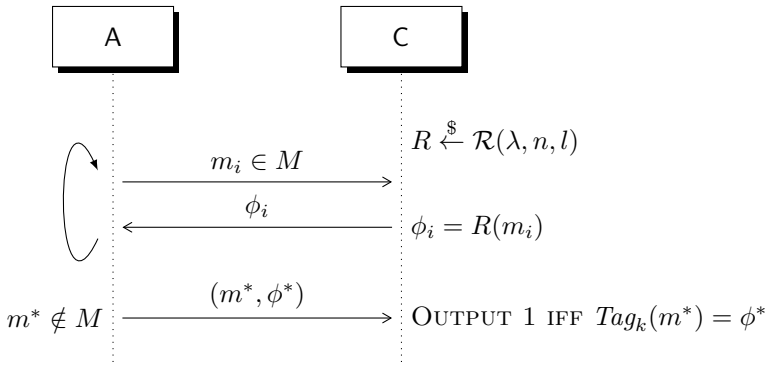


Figure 9.2: $\text{HYB}^1_{\Pi,\mathsf{A}}(\lambda)$

Lemma 15. $GAME^{UFCMA}_{\Pi,\mathcal{A}}(\lambda) \approx_c HYB^1_{\Pi,\mathcal{A}}(\lambda)$ ◇

Proof. By assuming there is a distinguisher $\mathsf{D}^{\text{UFCMA}}$ capable of disproving the lemma, it can be used to distinguish the PRF itself, as depicted by the reduction in figure 9.3 □

Lemma 16. $\forall$ *PPT* $\mathcal{A} \implies \Pr[HYB^1_{\Pi,\mathcal{A}}(\lambda) = 1] \leq 2^{-l}$ ◇

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

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

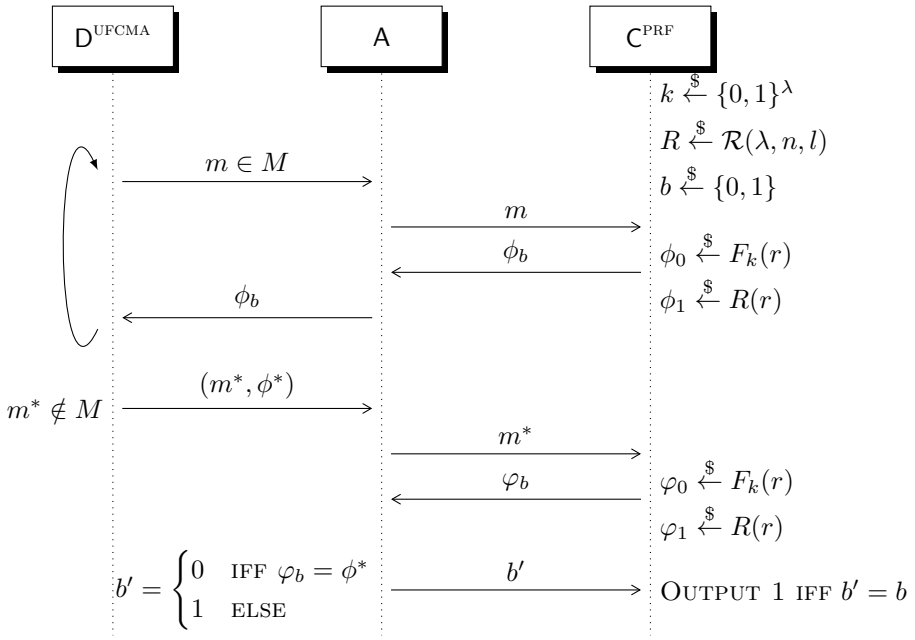


Figure 9.3: Distinguishing a PRF by using $\mathsf{D}^{\text{UFCMA}}$

9.2 Domain extension for MAC schemes

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

Assume the message $m = (m_1, \ldots, m_t) \in \{0,1\}^{n \cdot t}$ for some $t \geq 1$. Given the tagging function $Tag_k : \{0,1\}^n \to \{0,1\}^l$, an attempt to tag the whole message may be to:

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

9.2.1 Universal hash functions

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

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

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

So what are the properties of the induced family $\mathcal{F}(\mathcal{H}) = \{F_k(h_s(.))\}$? The main problem are *collisions*, since for each $m \in \{0, v1\}^{n \cdot t}$ it should be hard to find $m' \neq m$ such that $h_s(m) = h_s(m')$. But collisions do exist for functions in $\mathcal{F}(\mathcal{H})$, because they map elements from $\{0,1\}^{n \cdot t}$ to $\{0,1\}^t$, and since the codomain is smaller than the domain, the functions cannot be injective in any way.

To overcome this problem, we can consider two options:

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

Definition 16. A function family $\mathcal{H}$ is deemed *ε-universal* iff:

$$\forall x \neq x' \in \{0,1\}^{n \cdot t} \implies \Pr_{s \xleftarrow{\$} \{0,1\}^\lambda} [h_s(x) = h_s(x')] \leq \epsilon$$

$\diamond$

If $\epsilon = 2^{-n}$, meaning the collision probability is minimized, then the family is also called *perfectly universal*; in the case where $\epsilon \in \text{negl}(\lambda)$ isntead, it is defined as *almost universal* (AU). Take care abotu telling the difference between universality and pairwise independence, which states:

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

Lemma 17. *Show that any pairwise independent hash function is perfectly universal.* $\diamond$

Proof. The proof is left as exercise. (should I use Col for solving this? What is the difference and when I should use Col instead of one-shot-probability?) **ASK FOR SOLVING PROPERLY** (Thoughts: when I ask *what's the probability that, chosen 2 distinct x-es, their hashes are the same* ***on a certain value****?*, maybe I have to use one-shot, because one-shot refers to the prob. that the two inputs collide on a specific value, even if not specified.

Instead, if I consider *what's the prob. that , chosen 2 distinct x-es, their hashes are the same?*, maybe I have to calculate all the possible collisions, because I want to know if the 2 inputs can collide in general.) $\square$

Theorem 19. *Assuming $\mathcal{F}$ is a PRF with n-bit domain and $\mathcal{H}$ is AU, then $\mathcal{F}' = \mathcal{F}(\mathcal{H})$ is a PRF on $(n \cdot t)-$ bit domain, for $t \geq 1$.* $\diamond$

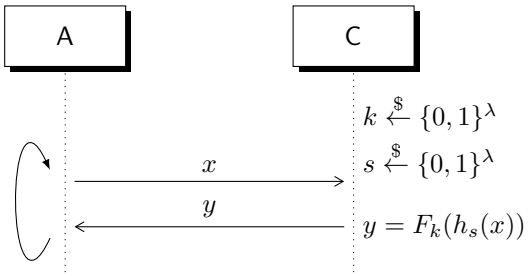


Figure 9.4: $Real_{\mathcal{F},\mathcal{A}}(\lambda)$

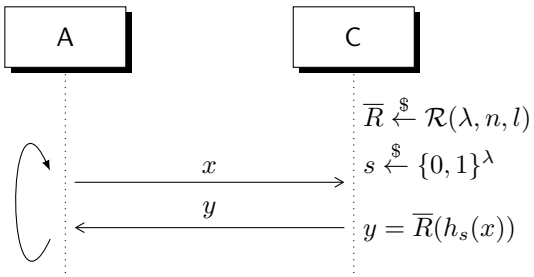


Figure 9.5: $\mathcal{HYB}_{\mathcal{R},\mathcal{A}}(\lambda)$

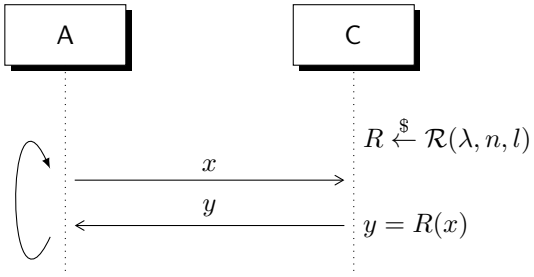


Figure 9.6: $Rand_{\mathcal{R}',\mathcal{A}}(\lambda)$

Proof. This proof too will proceed by hybridizing the original game up to the ideal random one. Consider the three sequences depicted in figures 9.4, 9.5 and 9.6:

Lemma 18.

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

◇

Proof. The proof is left as exercise. □

Lemma 19.

$$\mathcal{HYB}_{\mathcal{R},\mathcal{A}}(\lambda) \approx_c Rand_{\mathcal{R}',\mathcal{A}}(\lambda)$$

◇

Proof. Again, collisions come into play there, but in a much sneakier way. Given two queries with arguments x_1, x_2 returning the same image y, the random game can model two scenarios:

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

while the hybrid can model three of them:

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

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

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

where q denotes the adversary's query count. It suffices to show that $\Pr[\text{BAD}] \in \text{negl}(\lambda)$.

Since we don't care what happens after a collision, we can alternatively consider a mental experiment where we answer all queries at random, and only at the end sample $s \xleftarrow{\$} \{0,1\}^\lambda$ and check ofr collisions: this does not change the value of $\Pr(\text{BAD})$. Now queries are independent of s, and this eases our proof:

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

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

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

Corollary 1. *Let $\Pi = (Tag, Verify)$ be a variable length message MAC scheme where, given a PRF F_k and an AU hash funcion family h_s, the tagging function is defined as $F_k(h_s)$. Then Π is UFCMA-secure.* ◇

9.2.2 Hash function families from finite fields

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

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

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

◇

Lemma 20. *The above function family h_s is almost universal.* ◇

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

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

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

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

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

$$\sum_{i=1}^{t} \delta_i s_i = \delta_1 s_1 + \sum_{i=2}^{t} \delta_i s_i = 0$$

$$\delta_1 s_1 = -\sum_{i=2}^{t} \delta_i s_i$$

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

which means when a collision happens, s_1 must be exactly equal to the sum of the other blocks, which is another element of $\mathbb{F}$. But since every seed is chosen at random among $\mathbb{F}$, the probability of picking the element s_1 satisfying the above equation is just $|\mathbb{F}|^{-1} = 2^{-n} \in \text{negl}(\lambda)$. By repeating this reasoning for every difference-block, a sum of negligible probabilities is obtained, which is in turn negligible; therefore the hash function family h_s is almost universal.

□

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

Construction 3. Take $\mathbb{F} = GF(2^n)$, a *Galois field* of 2^n elements.

Let $m = (m_1, ..., m_t) \in \mathbb{F}^t$ and $s \xleftarrow{\$} \mathbb{F}^t$. We state that

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

$\diamond$

Exercise 20. Prove that this construction is **almost universal**.

(possible proof: to be almost universal, looking at the definition, collisions with $m \neq m'$ must be negligible.

So consider a collision as above: it must be true that

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

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

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

)

Lesson 10

10.1 Domain extension for PRF-based MAC schemes

10.1.1 Hash function families from PRFs

Another way to obtain domain extension for a MAC scheme, using the $\mathcal{F}_k(h_s)$ approach, is to construct the hash function family from another PRF. We expect to have:

- $\Pr[h_s(m) = h_s(m'), s \xleftarrow{\$} \{0,1\}^\lambda, (m, m') \xleftarrow{\$} A(1^\lambda)] \in \text{negl}(\lambda)$;
- We need two PRFs: one is F_k, and the other is F_s

10.1.2 XOR-mode

Assume that we have this function

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

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

Lemma 21. *Above $\mathcal{H}$ is computational AU if F_s is a PRF.* ◇

Proof. The proof is left as exercise.

(Hint: The pseudorandom functions can be defined as $F_s = F'_k(0, \dots)$ and $F_k = F'_k(1 \dots)$).

Possible proof: we have to show that

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

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

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

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

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

□

10.1.3 CBC-mode MAC scheme

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

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

Lemma 22. *CBC MAC defines completely an AU family.* ◇

Proof. (not proven) □

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

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

Theorem 21. *Actually if F_k is a PRF, CBC-MAC is already a MAC with domain $n \cdot t$ for arbitrarily fixed $t \in \mathbb{N}$.* ◇

Proof. (not proven) □

10.1.4 XOR MAC

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

Authentication is done as:

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

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

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

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

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

so now he could send the authenticated message

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

which is a valid message.

T0D0 10: AXU property definition is missing

Lemma 23. *XOR mode gives computational AXU (Almost Xor Universal)* ◇

Proof. (not proven) □

Theorem 22. *If $\mathcal{F}$ is a PRF and $\mathcal{H}$ is computational AXU, then XOR-MAC is a MAC.* ◇

Proof. (not proven) □

Summary

T0D0 11: not sure what to do with this bullet list...

With variable input lenght:

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

10.2 CCA-security

Going back to the encryption realm, a new definition of attack to a SKE scheme will be introduced. Now the adversary can query a decryption oracle, along with the CPA-related encryption oracle, for polynomially many queries. This attack is called the *Chosen Ciphertext Attack*[15], and schemes that are proven to be CCA-secure are also defined as *non-malleable*, on the reasoning that an attacker cannot craft fresh valid ciphertexts from other valid ones.

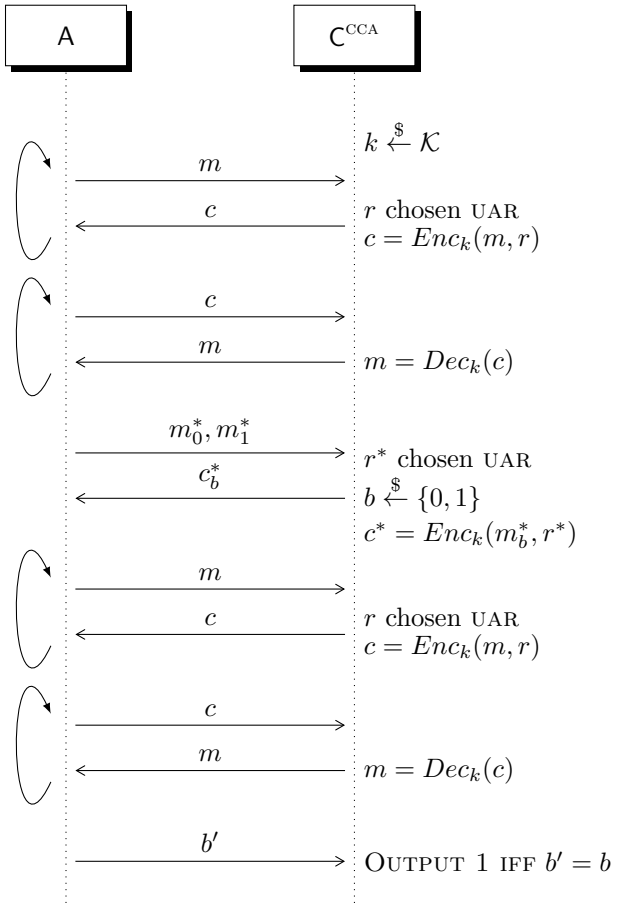


Figure 10.1: The chosen ciphertext attack, on top of CPA

Exercise 23. Show that the scheme $\Pi_{\mathcal{F}}$ defined in theorem 16, while CPA-secure, is not CCA-secure.

Proof. Let m_0 and m_1 be the messages the adversary sends to the challenger as the challenge plaintexts; on receiving the ciphertext $c_b = (r, F_k(r \oplus m_b) : b \xleftarrow{\$} \{0,1\}$, the adversary crafts another ciphertext with an arbitrary value α:

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

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

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

$$\begin{aligned} Dec_k(\widehat{c_b}) &= Dec_k(r, F_k(r) \oplus m_b \oplus \alpha) \\ &= F_k(r) \oplus F_k(r) \oplus m_b \oplus \alpha \\ &= m_b \oplus \alpha \end{aligned}$$

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

□

10.3 Authenticated encryption

Instead of tackling the CCA-security problem upfront, it might be useful to consider a construction that achieves both secrecy and authentication: that is, a scheme that encrypts messages and authenticates their respective senders at the same time. In the encryption setting, such a scheme is defined as being CPA-secure with an additional AUTH property denoting the scheme's resistance to forgeries, much like its MAC cousins. The game shown in figure 10.2 models this AUTH property of a scheme $\Pi = (Enc, Dec)$, with an additional quirk to the decryption routine:

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

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

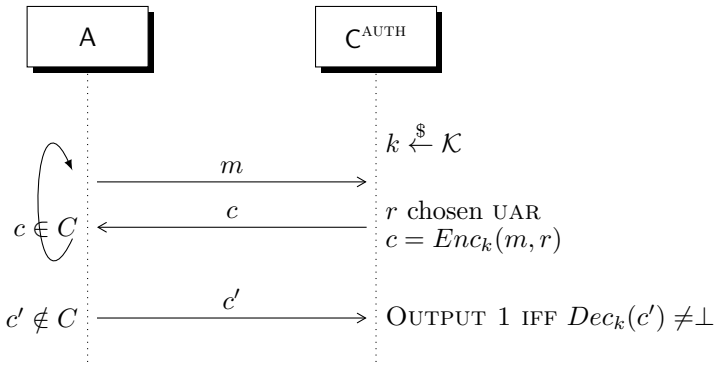


Figure 10.2: $\text{GAME}^{\text{AUTH}}_{\Pi,\mathsf{A}}(\lambda)$

Theorem 24. *Let Π be a SKE scheme. If it is CPA-secure, and has the AUTH property, then it is also CCA-secure.* ◇

Proof. The proof is left as an exercise.

Hint: consider the experiment where $Dec(k, c)$:

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

- else output $\perp$

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

□

10.3.1 Combining SKE & MAC schemes

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

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

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

Theorem 25. *If an authenticated encryption scheme* Π *is made by combining a* CPA*-secure* SKE *scheme* Π_1 *with a* strongly unforgeable MAC *scheme* Π_2 *in the* Encrypt-then-tag *method. Then* Π *is* CPA*-secure and auth-secure.* ◇

Proof. Assume that Π is not CPA-secure; then an adversary can use the resulting distinguisher D^{CPA} to direct a succesful CPA against Π_1, as shown in figure 10.3: the point is to run the two components of Π separately.

T0D0 12: Professor says that we have to show that $Game^{cpa}(\lambda, 0) \approx_c Game^{cpa}(\lambda, 1)$, but why??? Isn't this proof enough?

□

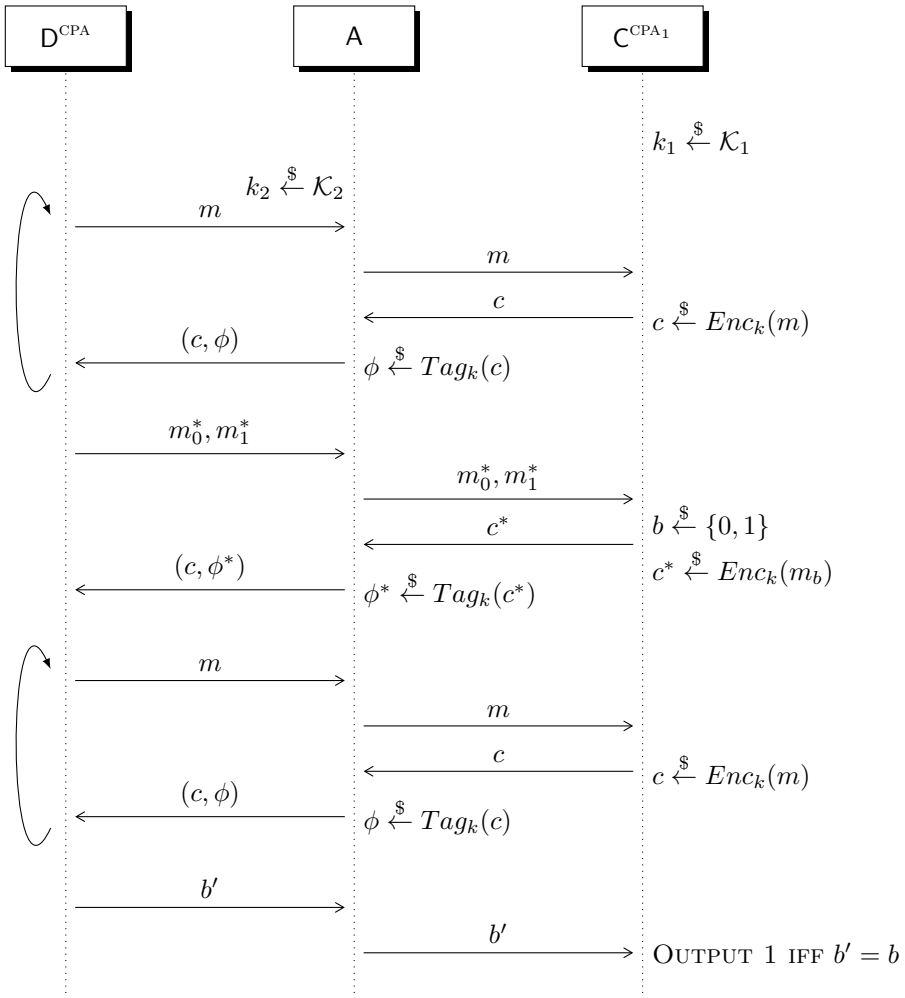

$\mathsf{D}^{\mathrm{CPA}}$
A
$\mathsf{C}^{\mathrm{CPA}_1}$
$k_1 \xleftarrow{\$} \mathcal{K}_1$
$k_2 \xleftarrow{\$} \mathcal{K}_2$
m
m
c
$c \xleftarrow{\$} Enc_k(m)$
(c, ϕ)
$\phi \xleftarrow{\$} Tag_k(c)$
m_0^*, m_1^*
m_0^*, m_1^*
$b \xleftarrow{\$} \{0,1\}$
c^*
(c, ϕ^*)
$c^* \xleftarrow{\$} Enc_k(m_b)$
$\phi^* \xleftarrow{\$} Tag_k(c^*)$
m
m
c
$c \xleftarrow{\$} Enc_k(m)$
(c, ϕ)
$\phi \xleftarrow{\$} Tag_k(c)$
b'
b'
Output 1 iff $b' = b$

Figure 10.3

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

Exercise 26. Prove it!

Similar to the cpa-security proof.

Lesson 11

11.1 Authenticated encryption (continued)

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

Definition 17. Let $\Pi = (Tag, Verify)$ be a MAC scheme. Then Π is EUFCMA-secure iff it is UFCMA-secure, that is:

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

with the additional restriction that the tag ϕ^* of the forged message must be "fresh" itself. ◇

Note the small difference in security between UFCMA and EUFCMA.

Theorem 27. *Let $\Pi = (Enc, Dec, Tag, Verify)$ be an authenticated encryption scheme, composed by a SKE scheme Π_1 and a MAC scheme Π_2. If Π_2 is EUFCMA, then Π has the AUTH property.* ◇

Proof. The proof is analogous to the previous proof regarding the scheme's CPA security. Suppose that Π has not the AUTH property; then an adversary can use the distinguisher D^{AUTH} to successfully forge authenticated messages with fresh signatures against Π_2, as depicted in figure 11.1.

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

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

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

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

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

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

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

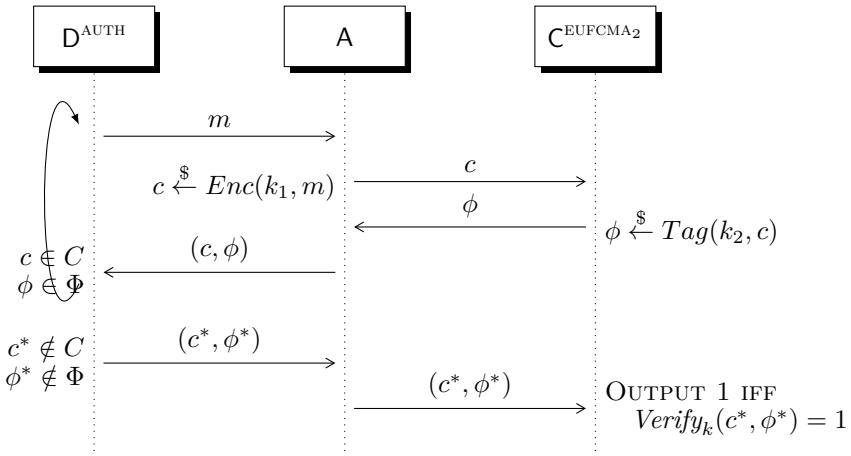


Figure 11.1: Breaking authenticity of Π_2

11.2 Pseudorandom permutations

T0D0 13: Luby-Rackoff is cited here, but it's related to both PRFs-PRPs and the Feistel network analysis

Nothing prevents a PRF F_k to be bijective; in this case, it is referred to as a *pseudorandom permutation*, or PRP in short. Their definition is analogous to a generic PRF: as shown in figures 11.2 and 11.3, PRPs are computationally indistinguishable from a random permutation:

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

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

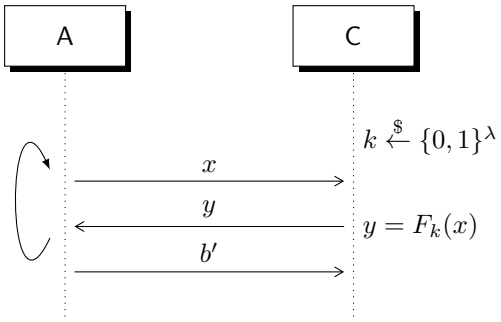


Figure 11.2: $Real_{\mathcal{F},\mathcal{A}}(\lambda)$

11.2.1 Feistel network

PRPs have been successfully constructed by using existing PRFs into what is called a *Feistel network*. As a starting point, let $F : \{0,1\}^n \to \{0,1\}^n$ be a PRF,

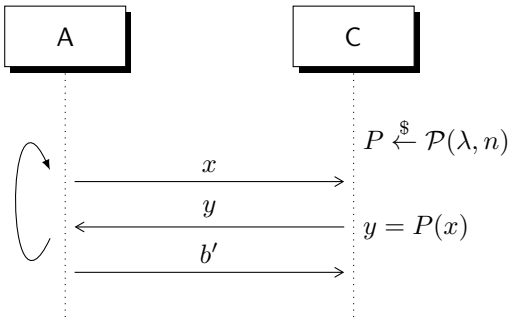


Figure 11.3: $Ideal_{\mathcal{P},\mathcal{A}}(\lambda)$

and define the function ψ_F as follows:

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

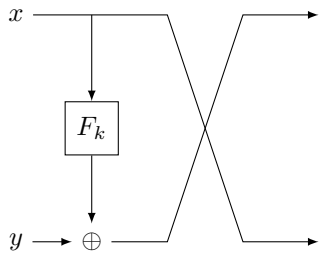


Figure 11.4: A single-round Feistel network

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

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

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

Lemma 24. *For any unbounded adversary making $q \in \text{poly}(\lambda)$ queries, the following games in figure 11.5 are statistically close as long as $y_1, \dots, y_q$ are mutually distinct.* ◇

Proof. T0D0 14: Idea: Hybridize over the queries before the challenge, from PR to random; prove that the stat distance between i and i+1 is negligible

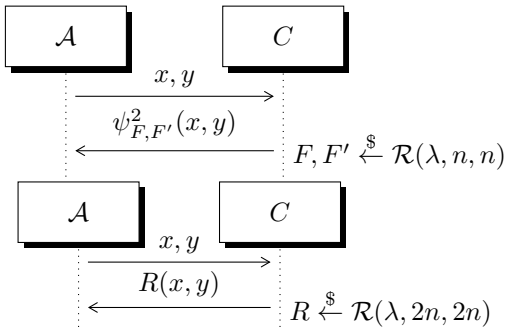


Figure 11.5

□

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

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

where Φ is the sequence of PRFs used at each single step. It can be shown that the rounds neeed to obtain a network that is indeed pseudorandom is just 3; also, the same PRF can be used, it is sufficient to change the seed on each iteration[16]:

Theorem 28. *Let F_i, F_j, F_k be a PRF over three seeds. Then $\psi^3_{i,j,k}$ is a PRP.* ◇

Proof. T0D0 15: Idea: Four total games: original, swap prfs with random functions, swap the three functions with a single one (use previous lemma), swap random function with random permutation (avoid bad events generated by injection property)

□

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

Lesson 12

12.1 Hashing

Remember one solution to domain-exetension for PRFs, as a composition of a prf into an (almost) universal hash function. Hash functions compress their arguments to some fingerprint which is asssumed to be unique. But since this compression in this context inherently introduces information loss, it is not guarantedd that every message gets its own unique fingerprint. On the contrary, there will be some instances, which we all *collisions*, where two messages yield the same hash value. So it is desirable for a hash function to be *resistant* to these events, meaning that they are hard to reproduce.

Definition 18. A function family $\mathcal{H}$ is deemed *collision-resistant* iff the probability of finding a collision is negligible, even when given a fixed key s. Formally:

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

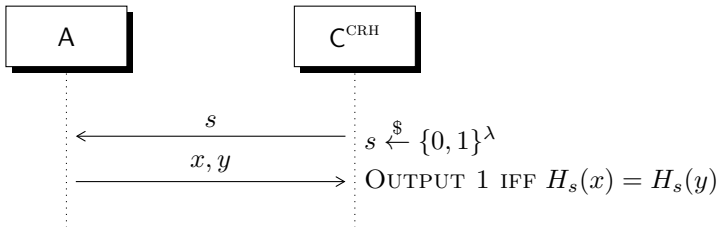


Figure 12.1: The corresponding *collision-resistance* game

◇

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

Exercise: Let Π be a UFCMAauthentication scheme over the message space $\{0,1\}^n$. Show that $\Pi' = (Tag', Verify') : Tag'_{k,s}(m) = Tag_k(H_s(m))$ is UFCMA-secure over $\{0l, 1\}^l$, where $l \in \text{poly}(n)$, as long as H is resistant to collisions.

12.1.1 Merkle-Damgård construction

First step: Compress the original message (assuming fixed size) by one bit

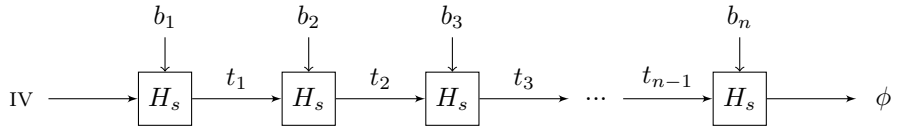


Figure 12.2: Basic outline of a Merkle-Damgård construction

Let H be a one-bit shrinking function. Then, it can be used to construct a hash function H' that splits an arbitrary-size message into fixed-size blocks, apply H onto them, and return a digest of fixed length. This is exemplified by the diagram in figure 12.2

Theorem 29. *Let H be a CRH function from n+1 to n bits, then the construction H' obtained from using merkle-damgard is a crh function* ◇

Proof. Assume H' can be broken efficiently by a distinguisher D^{CRH}, meaning that finding two distinct block sequences that give the same hash is easy.

Ignore same blocks: find largest j such that:

$$(b_j, y_{j-1}) \neq (b'_j, y'_{j-1}) \wedge H_s(b_j, y_{j-1}) = H_s(b'_j, y'_{j-1})$$

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

□

Not secure for VLM,

Lemma: MD strengthening (suffix freeness): Let $H_s \in 2^{n+l} \to 2^n$, then:

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

Theorem: strengthened md is crh

Proof: similar as above, case by case

12.1.2 Merkle trees

12.1.3 Compression functions

Let (gen, f, g) be a PKEscheme, where the functions are PRPs. A *claw* is a couple of values (x, x') such that $f_{pk}(x) = g_{pk}(x')$

Theorem: Assuming $/F$ is claw-free, H is a crh function from n+l to n bits.

$H(k, x) = E_k(x) \oplus x$, maps n+$\lambda$ to n. E is AES

Lesson 13

13.1 Number theory

Theorem 30. Fermat's last theorem

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

◇

Lemma 25.

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

◇

Proof. Assume there exists b in group such that $ab \equiv_n 1$. Then, there exisits a quotient for the division between a and b with remainder 1. Observe that $\gcd(a, n)$ divides $ab + qn$, which is equal to 1. It entails that $\gcd(a, n) = 1$, which is a contradiction. □

Lemma 26.

$$\forall a, b \in \mathbb{Z} : a \geq b > 0 \implies \gcd(a, b) = \gcd(b, REM_b(a))$$

◇

Proof. □

Theorem 31. *Given a, b integers, their greatest common divisor can be computed efficiently wrt a and b's lengths. Additionally, two other numbers u and v can be computed in order to satisfy bezout's identity:* $\gcd(a, b) = au + bv$ ◇

Proof. Hint: Use previous lemma recursively... □

Claim: $r_{i+2} \leq r_i/2 \forall o \leq i \leq t - 2 \implies \#steps = \lambda - 1$ if $|b| \in 2^\lambda$
Proof: ...

Definition 19. Exponentiation mod n: Square and multiply
Let $bin2^l$, where by writing b_i we denote b's i-th bit. Then:

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

◇

Theorem 32. *The number of primes lesser than or equal to x is a number greater than or equal to $x/3\log_2 x$* ◇

Theorem 33. *(Milner-Rabin - AKS) We can est in polytime if a random λ-bit number is prime* ◇

Conjecture: Integer multiplication of two lambda-bit primes is a OWF. (is this the factorization assumption?)

13.2 Standard model assumptions

Given a group G, its order is the least i such that $a^i \equiv_n 1$
Corollary: $\forall a \in \mathbb{Z}_m^\times \implies a^{\phi(n)} \equiv_n 1 \wedge a^b \equiv_n a^{rem_{\phi(n)}(b)}$

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

Discrete logarithm

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

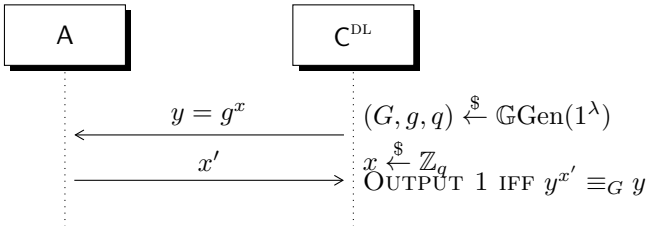


Figure 13.1

Meaning, DL for a generic group G is a OWF, whereas in a multiplicative group $\mathbb{Z}_p^\times$, DL is a OWP.

Computational Diffie-Hellman

Statement: given a group, and two elements of it g^x, g^y, it is impractical to compute g^{xy} without knowing both x and y.

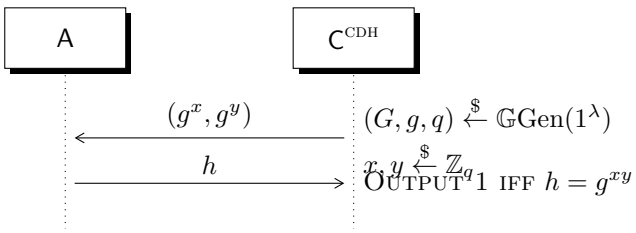


Figure 13.2

Decisional Diffie-Hellman

Statement: see game

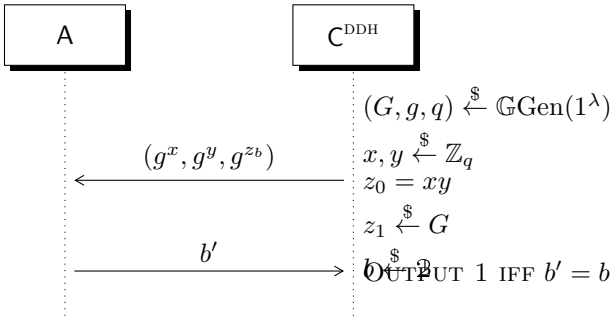


Figure 13.3

All these assumptions arose in definining what is the Diffie Hellman key exchange, which is a way to establish a SKE channel from an unsafe channel, with any adversary unable to efficiently break the channel's secrecy. Party authentication is not considered here.

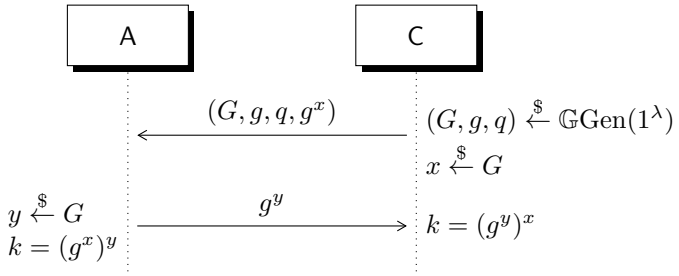


Figure 13.4: The Diffie-Hellman Key Exchange protocol

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

Lesson 14

14.0.1 Decisional Diffie-Hellman assumption

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

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

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

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

Otherwise:

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

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

???

Can have a distinguisher; end of Proof

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

Recall: DL is hard

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

prove this is hard by hybrid arg

14.0.2 Naor-Reingold encryption scheme

14.1 Public key encryption schemes

Lesson 15

15.1 Public key encryption recap

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

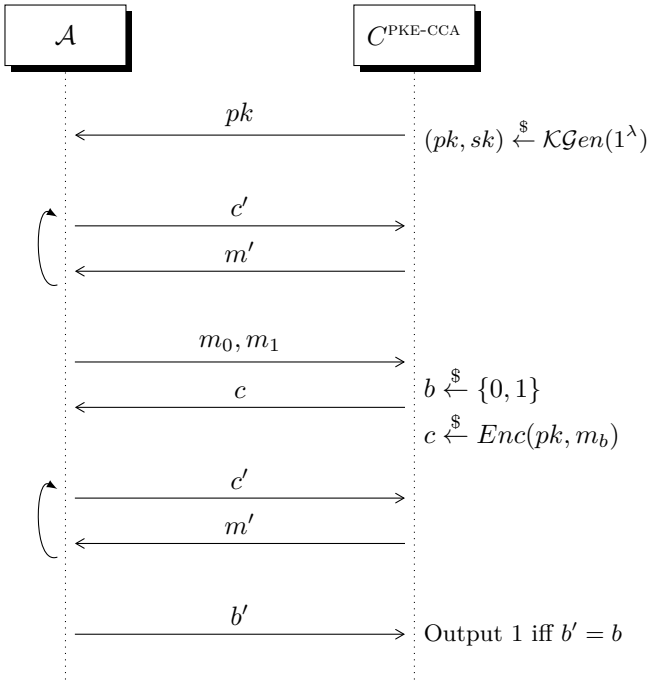


Figure 15.1: CCA on a PKE scheme

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

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

15.1.1 Trapdoor permutation

A *trapdoor permutation* (or TDP) is a OWP family structured has these features:

- A key pair is chosen UAR by a key generator algorithm:

$$(pk, sk) \xleftarrow{\$} \mathcal{KG}en(1^\lambda)$$

- There is a function family $f_{pk} \subseteq (V_{pk} \to V_{pk})$ such that:
  - Computing f_{pk} is efficient
  - Domain sampling ($x \xleftarrow{\$} V_{pk}$) is efficient
- There is an efficient function g_{sk} that "inverts" f_{pk} (sk is the "trapdoor"):

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

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

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

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

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

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

Theorem 35. *If $(\mathcal{KG}en, f, g)$ is a TDP, and $\mathfrak{hc}$ is hardcore for f, then the above scheme is CPA-secure.* ◇

Proof. The proof is left as exercise

T0D0 16: Apparently, the reduction here is not easy at all, some hints are needed.

□

15.1.2 TDP examples

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

Theorem 36. *(Chinese remainder, or CRT): The following isomorphisms to $\mathbb{Z}_n^\times$ are true:*

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

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

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

How to use this theorem for constructing a PKE scheme:
Reminder (Euler's theorem):

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

Reminder: $\forall p, q \in \mathbb{P} \implies \varphi(pq) = (p-1)(q-1)$
So let a be the public key such that $\gcd(a, \varphi(n)) = 1$, then $\exists! b \in \mathbb{Z}_n : ab = 1 \mod \varphi(n)$, b will be our private key.
Define encryption as $f(a, m) = m^a \mod n$, and then decryption as $g(b, c) = c^b \mod n$.
Observe that

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

because $ab = 1 \mod \varphi(n)$.
So we conjecture that the above is a valid TDP-based PKE scheme. This is actually referred to as the *RSA assumption*, and is depicted in figure 15.2

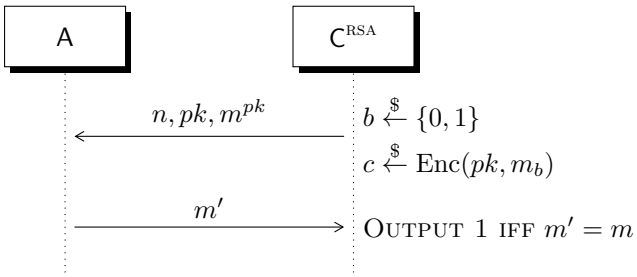


Figure 15.2: The RSA assumption

Relation to the factoring problem: RSA $\implies$ FACT
Proof: Given p, q, an adversary can compute $\varphi(n) = (p-1)(q-1)$, and then find the inverse of the public key in $\mathbb{Z}_{pq}^\times$.
It hasn't been proven that FACT $\implies$ RSA

15.2 Textbook RSA

This is an insecure toy example of the more complex RSA (Rivest Shamir Adleman) scheme.

- Setup: $pk \xleftarrow{\$} \mathbb{Z}_n^\times, sk \equiv_{\varphi(n)} pk^{-1}$
- Encrypt: $Enc_{pk}(m) = m^{pk} \pmod n$

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

Again, since the encryption routine is deterministic, the scheme is not CPA-secure. However, a hardcore predicate can be inserted to the routine: $\hat{m} = r||m$, where $r \xleftarrow{\$} \{0,1\}^l$. Now the encryption is pseudorandom.

Facts:

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

REVIEW:

2. If $m \in \{0,1\}$ then I can prove it CPA secure under RSA (just use standard TDP)

3. If m is "in the middle" ($\{0,1\} \leq m \leq \{0,1\}^l$) RSA is believed to be secure and is standardized (PKCS#1,5)

REVIEW:

4. Still not CCA secure! counterexample?

15.2.1 Trapdoor Permutation from Factoring

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

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

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

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

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

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

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

Assume $p \equiv 3 \pmod 4$ ($[*]p = 4t + 3 \Rightarrow t = \frac{p-3}{4}$), then squaring modulo p is a permutation because, given $\underline{y = x^2 \pmod p}$ if I compute:

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

$$\implies x = \pm y^{t+1}$$

But only 1 among the above $\pm y^{t+1}$ is a square, in particular only the positive one. Since we have that:

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

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

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

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

So, $y = g^{z'} \in \mathbb{QR}_p \Leftrightarrow z'$ is even. If z' is odd, then $y \notin \mathbb{QR}_p$.

Since $\frac{p-1}{2}$ is odd, then $g^{\frac{p-1}{2}} \notin \mathbb{QR}_p$, and since it is possible to generate all of the other numbers with odd exponents

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

and g powered to odd exponents will have this form.

From that, it's possible to state the following:

Lemma 27. $\forall z, z \in \mathbb{QR}_p \implies -z \notin \mathbb{QR}_p$ $\diamond$

15.2.2 Rabin's Trapdoor permutation

Now we study a one way function built on previous deductions about number theory and modular arithmetic.

The *Rabin trapdoor permutation* is defined as

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

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

We can observe that the image of this function is $\mathbb{QR}_n$, a subset of $\mathbb{Z}_n^\times$.

For **Chinese remainder theorem** it is possible to state that f maps as follows

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

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

So

$$y \in \mathbb{QR}_n \Leftrightarrow y_p \in \mathbb{QR}_p \wedge y_q \in \mathbb{QR}_q$$

As before, the image of f is exactly

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

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

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

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

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

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

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

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

◇

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

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

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

Theorem 37. *Squaring mod n, where n is a* Blum integer[17] *is a trapdoor permutation under the factoring assumption.* ◇

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

In other words

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

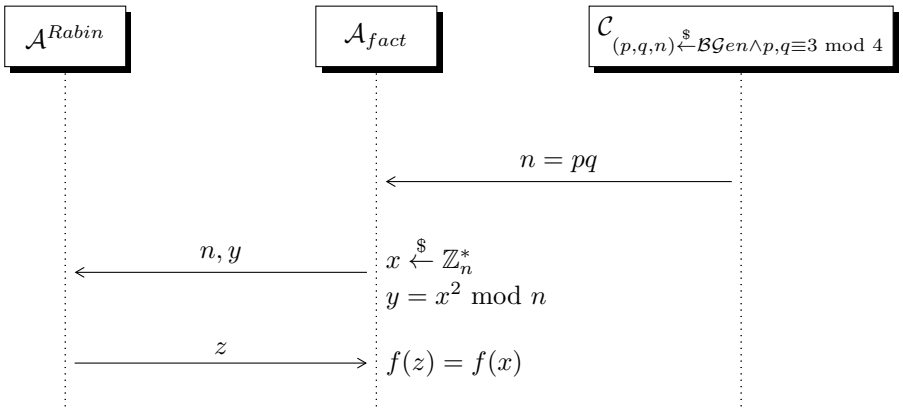
The following proof is by contraddiction.

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

We can build the following reduction to show that $\mathcal{A}$ choses x, here $\mathcal{BG}en$ is a sampler for Blum integers:

Once obtained $z \neq \pm x$ which $z^2 = y$ we can use **Claim 1**(just summing x and z and analyzing the result) to factorize n in polytime. But factorizing n in polytime is not possible. □

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


$\mathcal{A}^{Rabin}$
$\mathcal{A}_{fact}$
$\mathcal{C}_{(p,q,n)\xleftarrow{\$}\mathcal{BG}en\wedge p,q\equiv 3 \bmod 4}$
$n = pq$
n, y
$x \xleftarrow{\$} \mathbb{Z}_n^*$
$y = x^2 \bmod n$
z
$f(z) = f(x)$

Lesson 16

16.1 PKE schemes over DDH assumption

16.1.1 El Gamal scheme

Let's define a new Π = (KGen, Enc, Dec). Generate the needed (**public**) parameters $(G, g, q) \xleftarrow{\$} GroupGen(1^\lambda)$[18].

The KeyGen algorithm is defined as follows:

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

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

- Pick $r \xleftarrow{\$} \mathbb{Z}_q$
- Output $c = (c_1, c_2) = (g^r, pk^r \cdot m)$[19]

The decryption routine $\text{Dec}(sk, c)$ will:

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

Correctness of this scheme follows from some algebric steps:

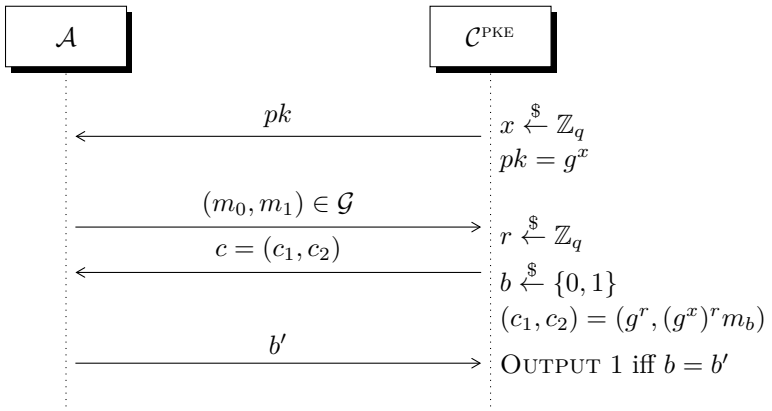
$$\begin{aligned}\hat{m} &= Dec(sk, Enc(pk, m)) \\ &= Dec(x, Enc(g^x, m)) \\ &= Dec(x, (g^r, (g^x)^r \cdot m)) \\ &= (g^r)^{-x} \cdot (g^x)^r \cdot m \\ &= m\end{aligned}$$

Theorem 38. *Assuming DDH, the El Gamal scheme is CPA-secure.* ◇

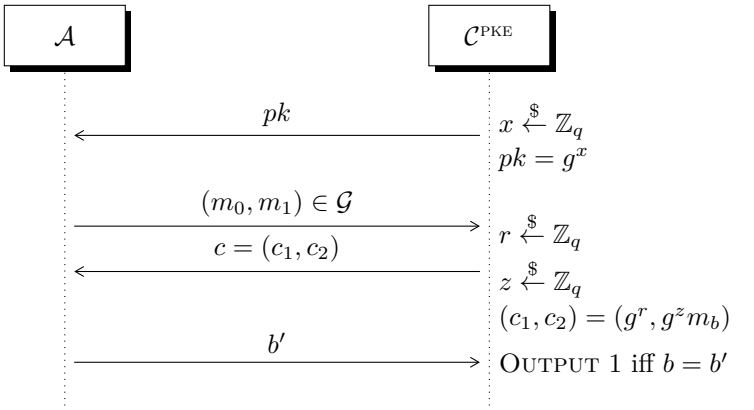
Proof. Consider the two following games $H_0(\lambda, b)$ and $H_1(\lambda, b)$ defined as follows. Observe that b can be fixed without loss of generality.

[18] G could be any "valid" group such as $\mathbb{QR}_p$ or an Elliptic Curve
[19] We need r because we want to re-randomize c

$H_0(\lambda, b)$:



$H_1(\lambda, b)$:



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

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

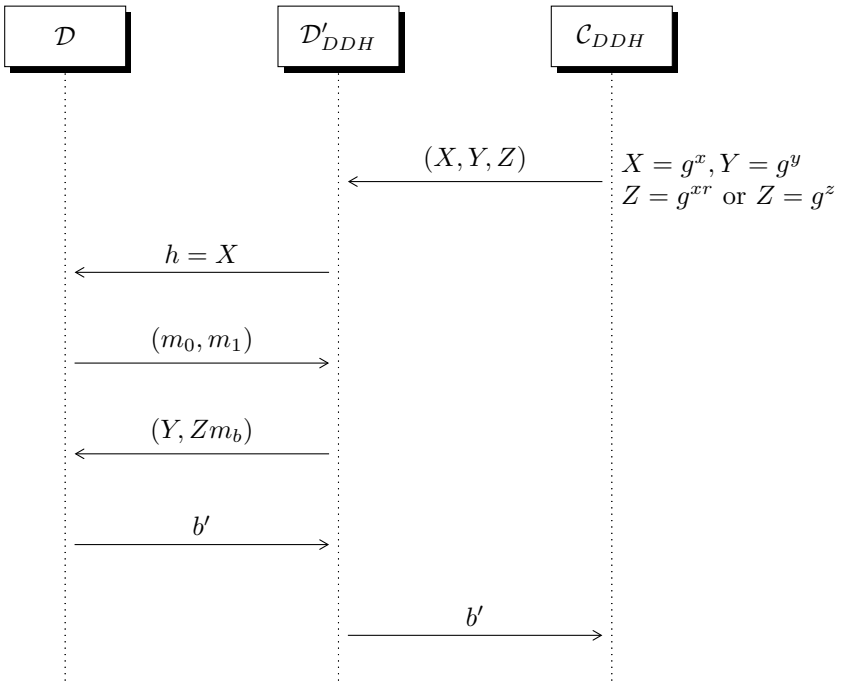
$$\implies H_0(\lambda, 0) \approx_c H_1(\lambda, 1)$$

Lemma 28. $\forall b \in \{0,1\}, H_0(\lambda, 0) \approx_c H_0(\lambda, 1)$
Fix b. (Reduction to DDH)
Assume $\exists$ PPT D which is able to distinguish $H_0(\lambda, b)$ and $H_1(\lambda, b)$ with non negl. probability. $\diamond$

Proof. Consider the following Game:

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

$\square$



Lemma 29. $H_1(\lambda, 0) \equiv H_1(\lambda, 1)$ ◇

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

Lemma 30. $H_1(\lambda, 1) \equiv H_0(\lambda, 1)$ ◇

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

□

Properties of of El Gamal PKE scheme

Some useful observations can be made about this scheme:

- It is **homomorphic**: Given two ciphertexts (c_1, c_2) and (c'_1, c'_2), then doing the product between them yields another valid ciphertext:

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

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

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

So this is not CCA-Secure!

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

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

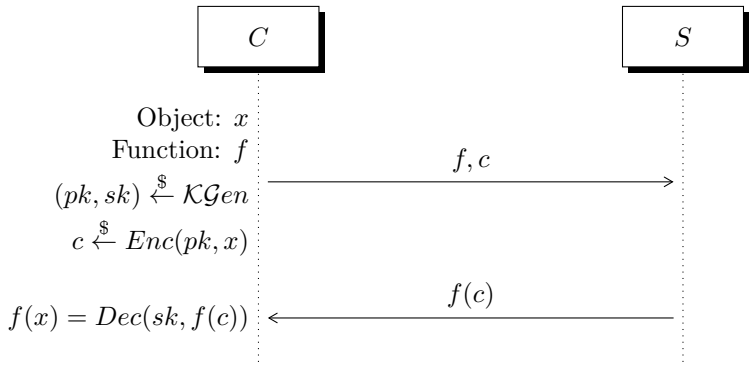


Figure 16.1: Delegated secret computation

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

16.1.2 Cramer-Shoup PKE scheme

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

Proof systems

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

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

where x is called a "witness" of y.

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

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

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

Additional notes:

- ω is public $(= pk)$

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

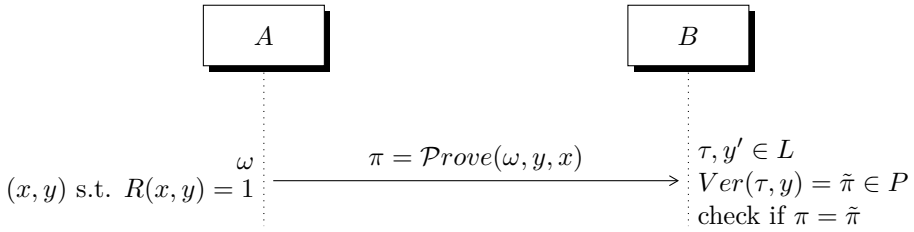


Figure 16.2: Overview of Cramer-Shoup operation

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

Properties

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

T0D0 17: to review and understand/better

t-universality

Definition 20. Let Π be DV-NIKZ[20].
We say it is *t-universal* if for any distinct

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

we have

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

where $(\omega, \tau) \xleftarrow{\$} Setup(1^\lambda)$ and $v_1, \ldots, v_t \xleftarrow{\$} \mathcal{P}$ where $\mathcal{P}$ should be the proofs' space. ◇

[20] Designated verifier non-interactive zero-knowledge

Enriching DV-NIKZ

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

Suppose to have the following:

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

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

Membership Hard Language (MH)

Definition 21. Language L is ***MH***if $\exists \bar{L}$ such that:

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

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

   (it's possible to say that $Samp(1^\lambda) \xleftarrow{\$} (y,x)$)

3. $\exists$ PPT $\bar{Samp}$ outputting $y \xleftarrow{\$} \bar{L}$
4. $\{y : (y,x) \xleftarrow{\$} Samp(1^\lambda)\} \approx_c \{y : y \xleftarrow{\$} \bar{Samp}(1^\lambda)\}$

$\diamond$

Lesson 17

17.1 Construction of a CCA-secure PKE

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

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

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

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

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

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

Some additional notes (may be incorrect):

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

Theorem 39. *Assuming π_1 is 1-universal, π_2 is 2-universal and L is a membership-hard language; then the above scheme is* CCA*-secure.* ◇

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

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

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

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

The games are defined as follows:

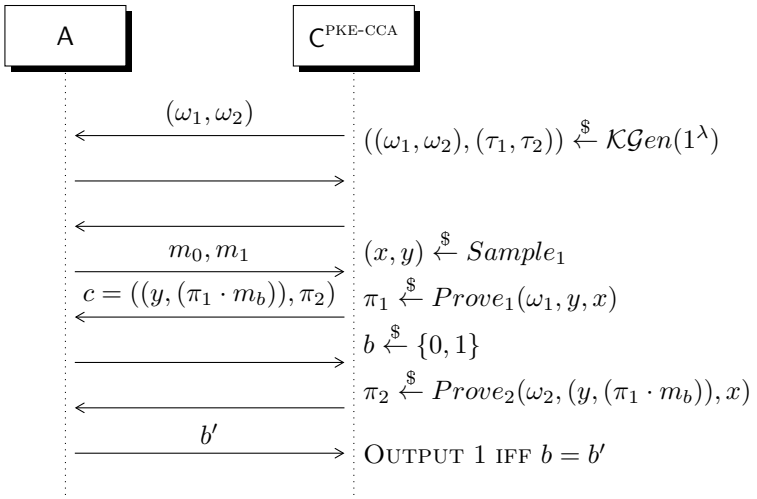


Figure 17.1: Original CCA game

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

□

Lemma 31.

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

◇

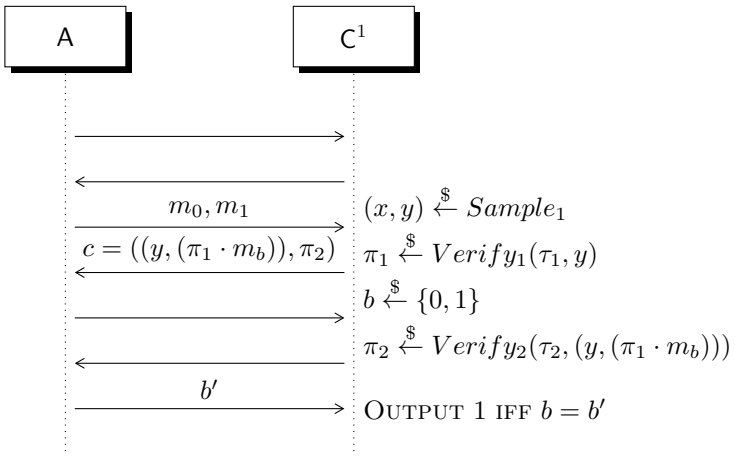


Figure 17.2: Use verifiers

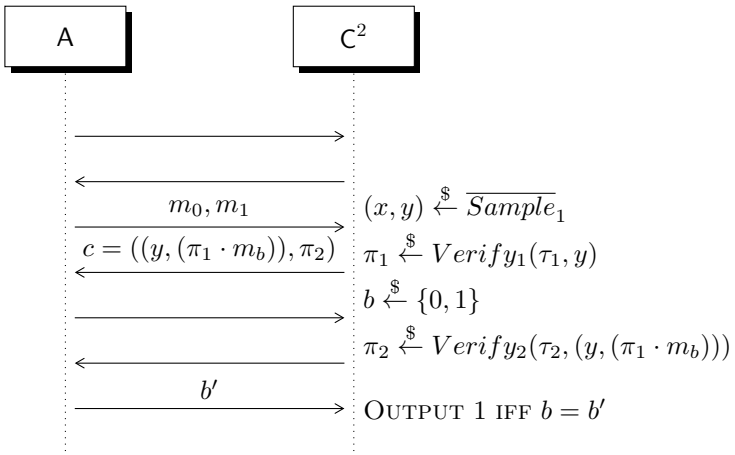


Figure 17.3: Sample statements outside the language

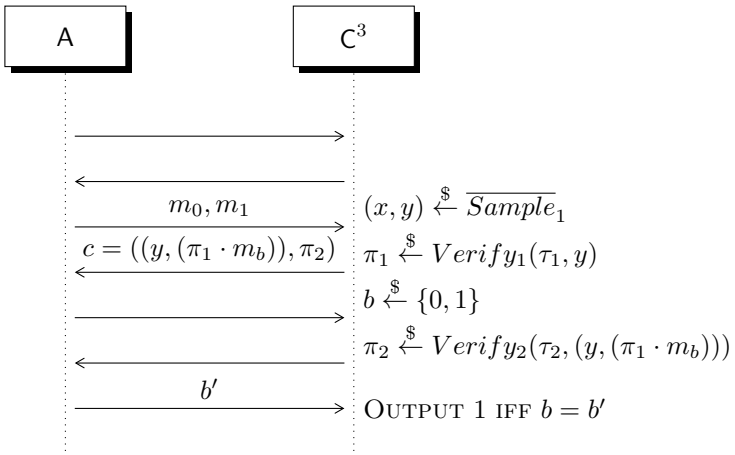


Figure 17.4: Modify decryption queries

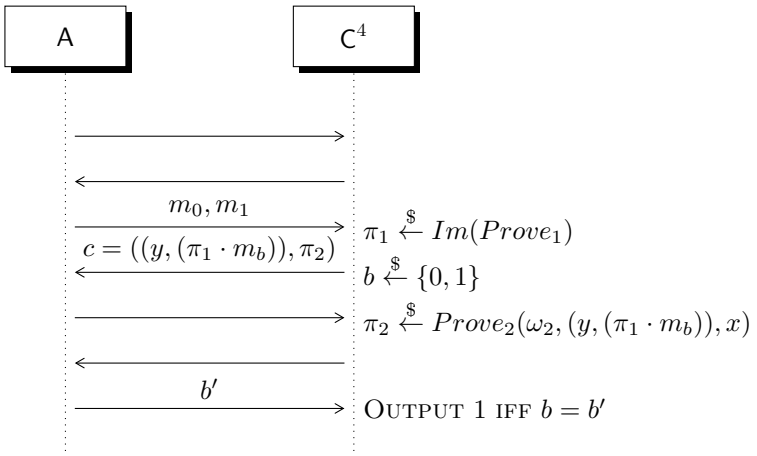


Figure 17.5: π_1 is chosen UAR

Proof. This follows by the correctness of Π_1 and Π_2

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

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

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

□

Lemma 32.

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

◇

Proof. Straight forward reduction from membership hardness. □

Lemma 33.

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

◇

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

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

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

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

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

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

1. if $(y^{(i)}, l^{(i)}) = (y, l)$ and $\tilde{\Pi}_2^{(i)} = \Pi_2^{(i)}$, it outputs $\perp$ if in the decryption scheme $(\Pi_2^{(i)} \neq \tilde{\Pi}_2^{(i)})$ it outputs $\perp$.

2. [21] otherwise $(y^{(i)}, l^{(i)}) \neq (y, l)$ if $y^{(i)} \notin L$ we want that $\Pi_2^{(i)}$ doesn't output exactly $Ver_2(\tau, (y^{(i)}, l^{(i)}))$, but it should output $\perp$.

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

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

for $y \notin L$.
The value

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

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

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

□

Lemma 34.

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

◇

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

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

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

Lemma 35.

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

◇

Proof. The challenge ciphertext is independent of b. □

REVIEW: referencing something from another part of lesson 17

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

MHL(Membership Hard Language) from DDH

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

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

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

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

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

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

Theorem 40. *Above construction defines a 1-universal DVNIZK for* L_{DDH} $\diamond$

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

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

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

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

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

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

□

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

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

Theorem 41. *The above construction define a 2-universal DVNIZK for* L_{DDH} $\diamond$

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

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

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

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

- Let's define a MAP

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

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

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

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

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

IFF:

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

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

Lesson 18

Cramer-Shoup scheme construction

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

T0D0 21: split definition from correctness

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

18.1 Digital signatures

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

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

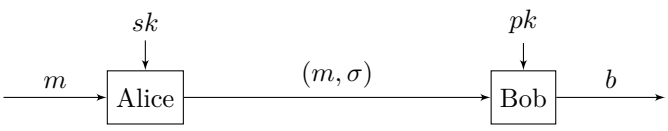


Figure 18.1: Asymmetric authentication

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

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

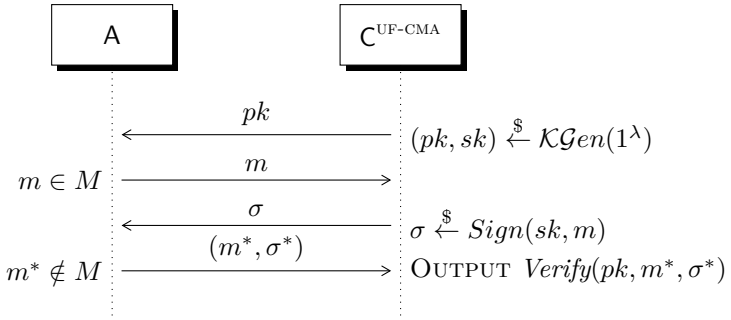


Figure 18.2: Unforgeable digital signatures

18.1.1 Public Key Infrastructure

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

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

The scheme works as follows:

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

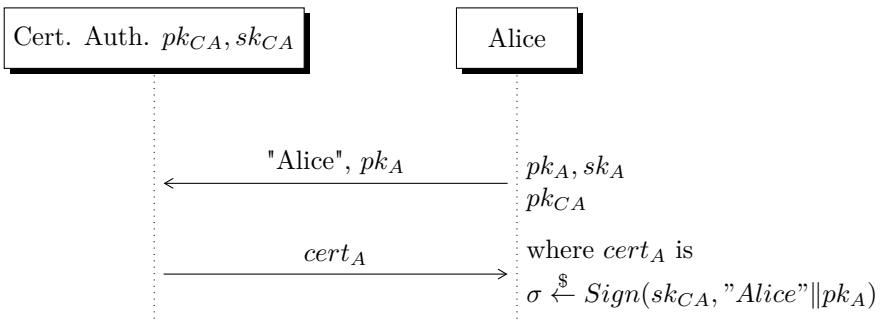
Now, when Bob wants to check the validity of the Alice's public key:

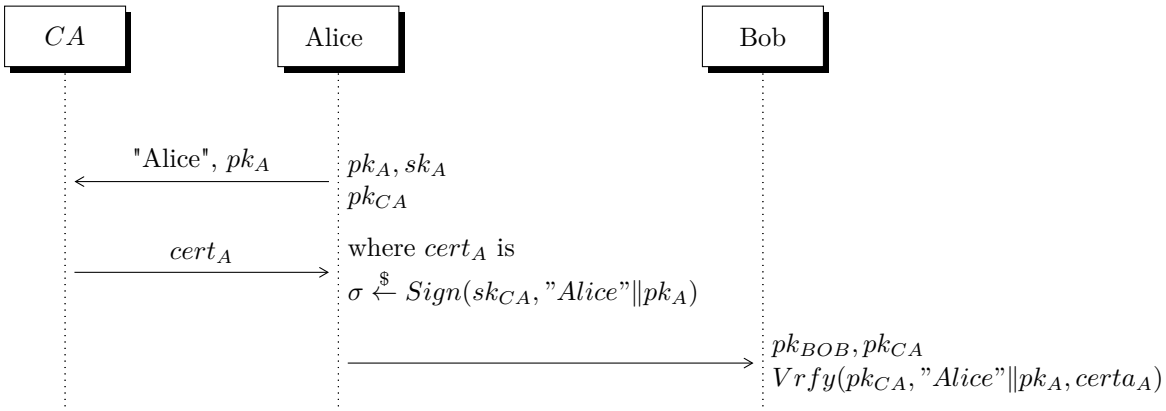
How can Bob recognize a valid certificate from an expired/invalid one?

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

Theorem 42. *Signatures are in* ***Minicrypt****.* ◇

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





In the next episodes:

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

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

Lesson 19

T0D0 22: Warining: e' venuto fuori un casino in questa lezione, ho cercato di riordinare le cose

19.1 Bilinear Map

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

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

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

"I can move the exponents"

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

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

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

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

– $S_k = g_2^a = g^{ab}$

Sign(S_k, m):

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

Vrf(P_k, m, (σ_1, σ_2))

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

Correctness: *"Just move the exponents"*

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

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

19.2 Waters signatures

Theorem 43. *The Waters' signature scheme is* UFCMA ◇

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

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

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

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

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

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

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

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

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

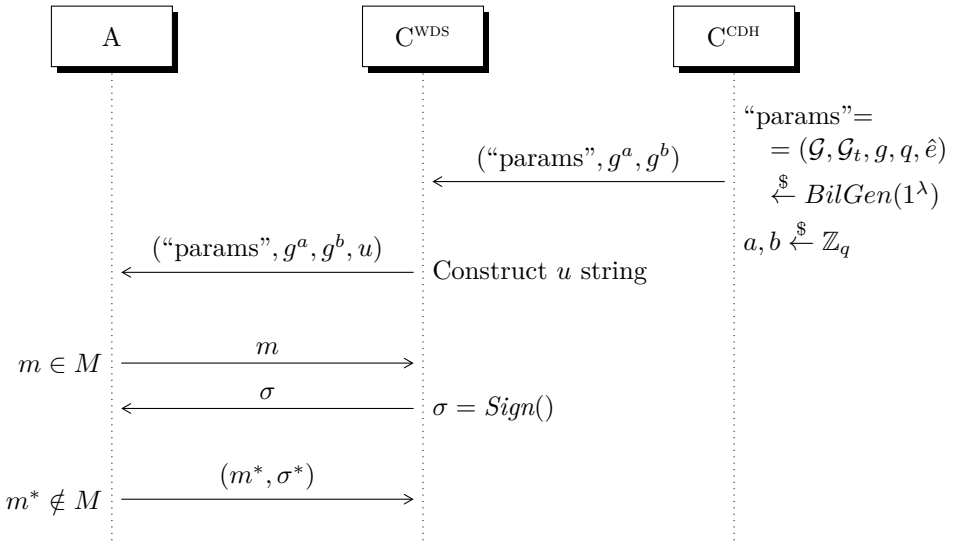


Figure 19.1: Reducing Waters' scheme to CDH

T0D0 25: Partizioni, doppi if.... qui non ci ho capito 'na mazza

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

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

□

Lesson 20

20.1 Random Oracle Model (ROM)

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

Recall: a truly random function $\mathcal{F}$ is defined to have a specific evaluation behaviour. Between subsequent evaluations:

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

They do act, in fact, as truth tables[22].

20.2 Full domain hashing

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

Take RSA:

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

Build a similar asymmetric-authentication scheme as such:

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

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

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

Proof: Reduce to TDP, program the random oracle.

Notes: this is a loose reduction

Some assumptions are made:

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

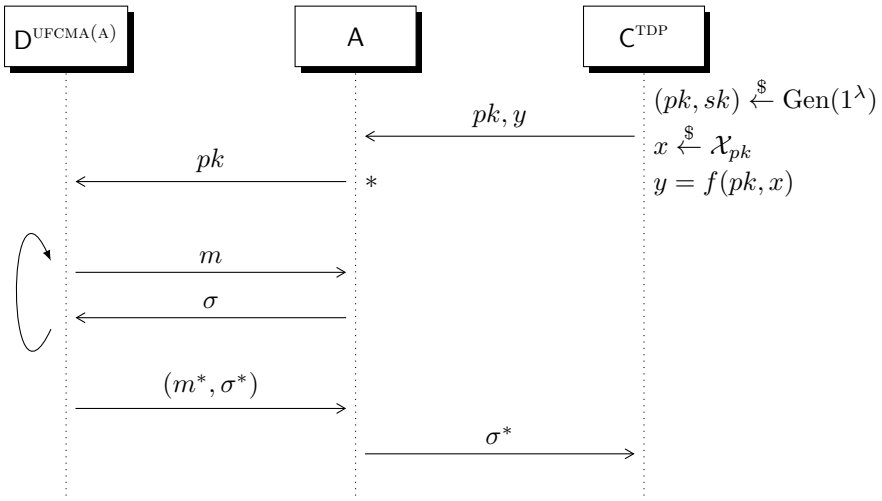


Figure 20.1

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

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

20.3 ID Scheme

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

20.4.1 Fiat-Shamir scheme

Lesson 21

21.1 Full domain hashing

Lesson 22

22.1 Examples of ID schemes

Lesson 23

23.1 Bilinear DDH assumption

Lesson 24

24.1 CCA proof for ???