Cryptography

Damien STEHLÉ

M1IF
2014–2015
Contents

1 Introduction ................................. 3
   1.1 Historic ciphers .......................... 4
       1.1.1 Shift cipher (CAESAR) ............... 5
       1.1.2 VIGÈNÈRE ............................ 5
       1.1.3 Substitution cipher .................. 5
       1.1.4 Enigma ............................... 5
       1.1.5 Permutation cipher ................. 5
   1.2 Perfect secrecy and the one-time pad .......... 6

2 Pseudo random number generators and stream ciphers .......... 9
   2.1 Pseudo-randomness ....................... 9
   2.2 Building PRG is not easy ................ 12

3 Semantic security ......................... 13

4 Block ciphers ............................. 15
   4.1 DES ..................................... 15
   4.2 AES ..................................... 15

5 Formalization of primitives .................. 16
   5.1 PRFs and PRPs ............................ 16
   5.2 Constructing PRPs from PRFs ............. 17
       5.2.1 Constructing PRF form PRG and vice-versa .... 18
   5.3 How to encrypt with a PRF/PRP? ............ 18
       5.3.1 Chosen plaintext security .......... 18
   5.4 MACs and CCA-secure encryption .......... 20
       5.4.1 MACs ............................... 20
       5.4.2 Constructing MACs .................. 21
       5.4.3 CCA-secure encryption .......... 23
Cryptography is everywhere! In credit cards, secured websites (https), Wi-Fi connections (WEP, WPA protocols) ...

The most classical form of cryptography is encryption: Alice and Bob communicate on an open channel and the conversation is modified in order to guarantee the confidentiality. Only Alice and Bob must be able to understand.

There are basically two phases in a communication:

Phase 1 (handshake) Alice and Bob agree on a secure key over the public channel
Phase 2 (communication) Alice and Bob use the shared secret to communicate securely.

Cryptography goes beyond encryption (confidentiality): science behind information security.

There is crypto without communication (encrypt hard drive)

Other properties we may want to have:

- integrity
- authenticity (I want to be certain to who I am talking).
- anonymity: TOR network, freenet
- Bitcoin: cryptography currency
- electronic voting
- confidential computations on the cloud (homomorphic encryption)
- Confidentiality preserving spam filters (functional encryption).

Cryptography is a science:

- define the functionality of the protocol under scope.
• define the power of the attacker
• propose a realization of the protocol
• prove that the existence of an efficient attacker leads to an efficient algorithm against a supposedly hard computational problem.

No public description ⇒ it’s insecure: protocols should be public: only the key is secret ⇒ KERCKHOFFS’s principles∗

No proof of security ⇒ no security (more controversial).

Cryptography has limitations

• very few unconditional result (e.g. factoring integers could turn out to be easy after all),
• To obtain information in real life, the simplest way is social engineering (e.g. put ink on buttons of a digicode, to know which ones are pressed),
• there can be attacks outside of the model (e.g. power attacks, NSA in your computer),
• there are bugs, sometimes devastating.

1.1 Historic ciphers

Alice and Bob want to discuss confidentially. We assume they already share the key.

We consider a symmetric encryption scheme over $K \times P \times C$ where $K$ is the set of keys, $P$ the set of plaintext messages and $C$ the set of ciphertexts.

An encryption scheme is a triple of efficient algorithms.

• KeyGen → produces $k \in K$,
• Enc : $(k, m) \in K \times P \mapsto c \in C$,
• Dec : $(k, c) \in K \times C \mapsto m' \in P$,

Correctness: $\forall k$ output by KeyGen, $\forall m \in P$, $\text{Dec}(k, \text{Enc}(k, m)) = m$.

Remark 1.

• Efficient for practice: we can do many bytes per second, when parameters are set so that all known attacks cost at least some amount of time/money.
• Efficient for theory: KeyGen, Enc, Dec should be probabilistic polynomial time (PPT). $(K, P, C)$ is in fact $(K_n, P_n, C_n) n \geq 1$ with bitsize of elements of $K_n, P_n, C_n \leq$ some polynomial function of $n$. We want KeyGen, Enc, Dec be polynomial in $n$.
• Probabilistic: most often KeyGen and Enc are probabilistic and Dec is deterministic.

∗https://en.wikipedia.org/wiki/Kerckhoffs%27s_principle
1.1.1  **Shift cipher (CAESAR)**

Take a letter, shift is by the secret value (e.g. for Cesar, it’s 3).

\[
\begin{align*}
A & \rightarrow D \\
B & \rightarrow E \\
C & \rightarrow F \\
& \vdots \\
& \vdots
\end{align*}
\]

Can be attacked by frequency analysis. In English and French ‘e’ is the most frequent. The most frequent letter in the ciphertext reveals the shift.

Scheme should be secure not just for random plaintexts, but for arbitrary plaintexts.

26 keys is too small. Usual recommendation: \(2^{80}\) keys.

1.1.2  **VIGÉNÈRE**

Fix a block size \(b\). The key is a \(b\)-letter long word \(\rightarrow 26^b\) keys.

Cut the text in blocks of length \(b\).

For each block, add the key mod 26.

- Frequency analysis still works.
- You can easily detect if the plaintext contains twice the same block.
- Easily broken by a chosen plaintext attack (Eve request an encryption to Alice) (active attack).

1.1.3  **Substitution cipher**

We choose a secret substitution table for the whole alphabet.

\[
\begin{align*}
A & \rightarrow S \\
B & \rightarrow U \\
C & \rightarrow A
\end{align*}
\]

Number of keys = 26! \(\approx 2^{88}\) but the frequency analysis still works.

1.1.4  **Enigma**

See the Wikipedia article because what the prof says is incorrect.

1.1.5  **Permutation cipher**

Fix a block length \(b\). The key is a permutation \(\sigma \in \mathfrak{S}_b\). Cut the test into blocks of length \(b\). A blocks \(m_1 \ldots m_b\) is mapped to \(m_{\sigma(1)} \ldots m_{\sigma(b)}\).

- Frequency analysis on pairs of letters.
- Some problem as for VIGÉNÈRE. We can see easily if the plaintext contains twice the same block.
1.2 Perfect secrecy and the one-time pad

**Definition 1.**

An encryption scheme $K \times P \times C$ is a triple of PPT algorithms.
- KeyGen: we choose some $k \in K$.
- Enc : $K \times P \rightarrow C$
- Dec : $K \times C \rightarrow P$

such that $\forall k$ output by Keygen, $\forall m \in P$, $Dec(k, Enc(k, m)) = m$ (correctness).

**Definition 2 (Perfect secrecy (SHANNON 49)).**

An encryption scheme is said perfectly secure if

$$\forall m \in P, \quad P(c = \overline{c}|m = \overline{m}) = P(c = \overline{c})$$

Probabilities are for uniform choice of $m \in P$, and possibly uniform choice of $k$.

**Remark 2.**

- Intuition: the choice of $\overline{m}$ has no impact on the ciphertext distribution.
- The random variables $c$ and $m$ are independent.
- Uniform $m$ is unrealistic, but we’re going to see that even this is hard to achieve.
- Assuming $P(c = \overline{c}) \neq 0$,

$$P(c = \overline{c}|m = \overline{m}) = \frac{P(c = \overline{c} \land m = \overline{m})}{P(m = \overline{m})} = \frac{P(m = \overline{m}|c = \overline{c}) P(c = \overline{c})}{P(m = \overline{m})}$$

$$\rightarrow P(m = \overline{m}|c = \overline{c}) = P(m = \overline{m}) \quad \text{(to determine } m, \text{ the cipher is useless).}$$

**Definition 3 (VERNAM’s cipher (1917) = one-time pad).**

$K = P = C = \{0, 1\}^l$ for some $l$.
- KeyGen: Sample $k$ uniformly in $\{0, 1\}^l$
- Enc : $(k, m) \mapsto m \oplus k$ (bit by bit)
- Dec : $(k, c) \mapsto c \oplus k$ (bit by bit)
Theorem 1.

Vernam’s cipher is correct and perfectly secure.

Proof. Correctness: \( \text{Dec}(k, \text{Enc}(k, m)) = k \oplus (k \oplus m) = (k \oplus k) \oplus m = m \)

Perfect secrecy: Let \( \overline{m} \in \mathcal{P} \).

\[
\mathbb{P}(c = \overline{c}|m = \overline{m}) = \mathbb{P}(k \oplus m = \overline{c}|m = \overline{m}) \\
= \mathbb{P}(c = \overline{c} \oplus \overline{m}) \\
= 2^l
\]

This is independent of \( \overline{m} \).

Two major problems:

- key is huge,
- communicating key as hard as communicating message,
- security against eavesdropping only.

Lemma 2.

Perfect secrecy \( \Rightarrow \forall \overline{m}, \forall \overline{c} \text{ reachable }, \exists \text{ key } k : \overline{c} = \text{Enc}(k, \overline{m}) \)

\[
\Rightarrow |\mathcal{K}| \geq |\mathcal{C}'| \geq |\mathcal{P}|
\]

set of reachable ciphertexts

Proof. \( |\mathcal{C}'| \geq |\mathcal{P}| \) is obvious because we are able to decrypt.

Fix \( \overline{m} \in \mathcal{P}, \overline{c} \in \mathcal{C}' \).

\[
\mathbb{P}(c = \overline{c}|m = \overline{m}) \mathbb{P}(c = \overline{c}) > 0 \text{ perfect secrecy reachable}
\]

So there exists \( k \) such that \( \overline{c} = \text{Enc}(k, \overline{m}) \).

I have proved that we have an injective map from \( \mathcal{C}' \) to \( \mathcal{K} \Rightarrow |\mathcal{K}| \geq |\mathcal{C}'| \).

Theorem 3 (Shannon).

Let \( (\text{KeyGen}, \text{Enc}, \text{Dec}) \) be an encryption scheme. Assumer that \( |\mathcal{K}| = |\mathcal{C}| = |\mathcal{P}| \geq 1 \).

It enjoys perfect secrecy iff

1. each \( k \in \mathcal{K} \) is chosen with probability \( \frac{1}{|\mathcal{K}|} \)
2. \( \forall m, \overline{m} \in \mathcal{P}, \forall \overline{c} \in \mathcal{C}, \exists k \in \mathcal{K} : \overline{c} = \text{Enc}(k, \overline{m}) \)
Proof. By lemma 2, we must have $C = C'$.

Let $\overline{m} \in \mathcal{P}$. Consider the map

$$\mathcal{K} \to C$$

$$k \mapsto \text{Enc}(k, \overline{m})$$

Goal: show that this is a bijection. (Lemma 2 $\Rightarrow$ it is surjective, and $|\mathcal{K}| = |C|$). This proves (2). This unique $k$ of $\overline{m}, \overline{\tau}$ is denoted $k(\overline{m}, \overline{\tau})$.

Fix $\overline{\tau} \in \mathcal{C}, \overline{m} \in \mathcal{P}$.

$$\mathbb{P}(c = \overline{\tau}) = \mathbb{P}(c = \overline{\tau} | m = \overline{m})$$

$$= \mathbb{P}(k = \overline{\tau})$$

$\Rightarrow$ $\mathbb{P}(k = k(\overline{m}, \overline{\tau}))$ is constant over all $k(\overline{m}, \overline{\tau})$'s. We have $\mathcal{K} = \{k(\overline{m}, \overline{\tau}) | (\overline{m}, \overline{\tau}) \in \mathcal{P} \times \mathcal{C}\}$ (to prove this, show that $\overline{m} \mapsto k(\overline{m}, \overline{\tau})$ is a bijection.)

$\Rightarrow$ $\mathbb{P}(k = \overline{\tau})$ is the same for all $\overline{k} \in \mathcal{K} \to$ it is $\frac{1}{|\mathcal{K}|}$.

$\Leftarrow$ Let $\overline{m} \in \mathcal{P}, \overline{\tau} \in \mathcal{C}$. We want to show (using (1) and (2)) that $\mathbb{P}(c = \overline{\tau} | m = \overline{m}) = \mathbb{P}(c = \overline{\tau})$.

$$\mathbb{P}(c = \overline{\tau} | m = \overline{m}) = \mathbb{P}(k = k(\overline{m}, \overline{\tau}))$$

$$= \frac{1}{|\mathcal{K}|}$$

$$\mathbb{P}(c = \overline{\tau}) = \sum_{\overline{m}} \mathbb{P}(c = \overline{\tau} \land m = \overline{m})$$

$$= \sum_{\overline{m}} \mathbb{P}(m = \overline{m}) \cdot \mathbb{P}(c = \overline{\tau} | m = \overline{m})$$

$$= \frac{1}{|\mathcal{K}|} \sum_{\overline{m}} \mathbb{P}(m = \overline{m})$$

$\Box$

**Remark 3.**

I didn’t use the assumption that $m$ is chosen uniformly (superfluous assumption).
Chapter 2

Pseudo random number generators and stream ciphers

2.1 Pseudo-randomness

Main issue with OTP (one time pad): the key is too long. In practice, we want a small key. We cannot cut the message in blocks and use the same OTP key for each block (VIGÉNÈRE).
We would like to have a pseudo random string in place of the OTP key.

\[ G : \{0, 1\}^s \rightarrow \{0, 1\}^n \]
with \( n \gg s \). The input of \( G \) is the seed. \( \frac{n}{s} \) is called the expansion factor. We impose \( G \) to be deterministic the randomness is coming from the seed only.

To encrypt we would like to proceed as follows:

- **KeyGen**: take a uniform \( k \in \{0, 1\}^s \),
- **Enc**: \( (k, m) \in \{0, 1\}^s \times \{0, 1\}^n \mapsto G(k) \oplus m \in \{0, 1\}^n \),
- **Dec**: \( (k, c) \in \{0, 1\}^s \times \{0, 1\}^n \mapsto G(k) \oplus c \),

**Definition 4 (Unpredictability).**

A PRG (pseudo random generator) \( G : \{0, 1\}^s \rightarrow \{0, 1\}^n \) is predictable if there exists a PPT (probabilistic polynomial time algorithm) \( A \) and \( i < n \) such that

\[
P(A(G(k)_{1\ldots i})) = G(k)_{i+1} \geq \frac{1}{2} + \varepsilon_{\text{non-negligible function of } s}
\]

\( G \) is unpredictable if it is not predictable.

In practice, non negligible can be \( \geq 2^{-80} \). In theory: non negligible is \( \geq \frac{1}{2} \) for some constant \( c > 0 \).
This definition is very strong, because it stands for all PPT algorithms. The existence of an unpredictable function is an open problem (it is thought to be a stronger claim than $P \neq NP$).

Remark 4.

$G(\{0,1\}^s)$, is tiny in $\{0,1\}^n$, so $G(U(\{0,1\}^s))$ is bounded to be very far from uniform in $\{0,1\}$.

Remark 5.

Take two distribution $D_1$ and $D_2$ over $\{0,1\}^n$. A PPT $A : \{0,1\}^n \to \{0,1\}$ is a distinguisher for $(D_1, D_2)$ if

$$\text{Adv}_A(D_1, D_2) := \left| \Pr_{x \leftarrow D_1} (A(x) = 1) - \Pr_{x \leftarrow D_2} (A(x) = 1) \right| \geq \epsilon$$

for some non negligible $\epsilon$.

$D_1$ and $D_2$ are said indistinguishable if for all PPT $A$, $\text{Adv}_A(D_1, D_2)$ is negligible.

Definition 5.

A PRG $G : \{0,1\}^s \to \{0,1\}^n$ is secure if $G(U(\{0,1\}^s))$ and $U(\{0,1\}^n)$ are indistinguishable.

Definition 6.

A PRG $G : \{0,1\}^s \to \{0,1\}^n$ is secure if $G(U(\{0,1\}^s))$ and $U(\{0,1\}^n)$ are indistinguishable.

There will always be statistical distinguisher (e.g. try all seeds) ($s \ll n$) but we want them to be too costly.

Theorem 4.

These two definitions are equivalent.

The next-bit test is a universal test.

Proof. (Existence of a next-bit prediction $\Rightarrow$ existence of a distinguisher): Let $i < n$, $\epsilon$ non negligible, $A$ PPT such that $\Pr (A(G(k)_{1..i}) = G(k)_{i+1}) \geq \frac{1}{2} + \epsilon$.

Define $A' : \{0,1\} \to \{0,1\}$ as follows: for input $x_1, \ldots, x_n$ if $A(x_1, \ldots, x_i) = x_{i+1}$ then output 1 (guess: it is $G(k)$ for some $k$) else output 0 (guess: it is uniform).

$$\text{Adv}_{A'}(G(U(\{0,1\}^s))), U(\{0,1\}^n)) = \left| \Pr_{x \leftarrow G(U)} (A'(x) = 1) - \Pr_{x \leftarrow U} (A'(x) = 1) \right|$$

$$= \left| \Pr_{k \leftarrow U(\{0,1\}^s)} (A(G(k)_{1..i}) = G(k)_{i+1}) - \frac{1}{2} \right|$$

$$= \frac{1}{2} + \epsilon - \frac{1}{2}$$

$$= \epsilon$$
(Existence of a distinguisher $\Rightarrow$ existence of a next-bit prediction) (Andrew Yao 1982): Proof base on the hybrid argument.

Let $\epsilon$ non negligible and $\mathcal{A}$ PPT such that $\text{Adv}_{\mathcal{A}}(G(U(\{0,1\}^k)), U(\{0,1\}^n)) \geq \epsilon$. Where is the "i" of the predictor? Let $D_i$ be the distribution over $\{0,1\}^k$ obtained as follows:

- $k \leftarrow U(\{0,1\}^k)$
- $z_1, \ldots, z_n = G(k)$
- Define $x_1, \ldots, x_i = z_1, \ldots, z_i$
- Sample $x_{i+1}, \ldots, x_n$ uniformly
- Output $x_1, \ldots, x_n$.

We have $D_0 = U(\{0,1\}^n)$ and $D_n = G(U(\{0,1\}^k))$.

As

$$|\mathbb{P}_{x \leftarrow D_0}(\mathcal{A}(x) = 1) - \mathbb{P}_{x \leftarrow D_n}(\mathcal{A}(x) = 1)| \geq \epsilon$$

there exists $i < n$ such that

$$|\mathbb{P}_{x \leftarrow D_i}(\mathcal{A}(x) = 1) - \mathbb{P}_{x \leftarrow D_{i+1}}(\mathcal{A}(x) = 1)| \geq \frac{\epsilon}{n}$$

This comes from triangular inequality:

$$|\mathbb{P}_{x \leftarrow D_0}(\mathcal{A}(x) = 1) - \mathbb{P}_{x \leftarrow D_n}(\mathcal{A}(x) = 1)| = \left| \sum_i \mathbb{P}_{x \leftarrow D_i}(\mathcal{A}(x) = 1) - \mathbb{P}_{x \leftarrow D_{i+1}}(\mathcal{A}(x) = 1) \right|$$

$$\leq \sum_i |\mathbb{P}_{x \leftarrow D_i}(\mathcal{A}(x) = 1) - \mathbb{P}_{x \leftarrow D_{i+1}}(\mathcal{A}(x) = 1)|$$

$D_i$ and $D_{i+1}$ differ only in their $i + 1$-th bit. How do we know which $i$? See the exercise session.

Let us assume that we know $i$. We also assume that $|\mathbb{P}_{D_i} - \mathbb{P}_{D_{i+1}}| = \mathbb{P}_{D_{i+1}} - \mathbb{P}_{D_i}$. (The probability that $\mathcal{A}$ output $1$ is bigger for $D_{i+1}$ than for $D_i$. If it’s not the case, change $\mathcal{A}$ into $1 - \mathcal{A}$.)

$\mathbb{P}_{x \leftarrow D_{i+1}}(\mathcal{A}(x) = 1) - \mathbb{P}_{x \leftarrow D_i}(\mathcal{A}(x) = 1) \geq \frac{\epsilon}{n}$. We want to build $\mathcal{A}': \{0,1\}^i \rightarrow \{0,1\}$ which predicts $G(k)_{i+1}$ given $G(k)_i$. Let $x_1, \ldots, x_i = G(k)_{1,\ldots,i}$ be the input of $\mathcal{A}'$

$\mathcal{A}'$ samples $x_{i+1}, \ldots, x_n$ uniformly and gives $x_1, \ldots, x_n$ as input to $\mathcal{A}$. If $\mathcal{A}$ outputs 1, it "thinks" it is given a sample from $D_{i+1}$ as input Then $\mathcal{A}'$ outputs $x_{i+1}$. If $\mathcal{A}$ outputs 0, it "thinks" it is given a sample from $D_i$ as input Then $\mathcal{A}'$ outputs $\overline{x}_{i+1}$.

We want to prove that $\mathbb{P}(\mathcal{A}'(G(k)_{1,\ldots,i}) = G(k)_{i+1})$ is non-negligible.

Let $\overline{D_{i+1}}$ be the distribution obtained by sampling $x_1, \ldots, x_n$ from $D_{i+1}$ and replacing $x_{i+1}$ by $\overline{x}_{i+1}$. We have $D_i = \frac{1}{2} D_{i+1} + \frac{1}{2} \overline{D}_{i+1}$.

$$\mathbb{P}(\mathcal{A}'(G(k)_{1,\ldots,i}) = G(k)_{i+1}) = \frac{1}{2} \mathbb{P}_{x \leftarrow D_{i+1}}(\mathcal{A}(x) = 1) + \frac{1}{2} \mathbb{P}_{x \leftarrow \overline{D}_{i+1}}(\mathcal{A}(x) = 0)$$

$$= \frac{1}{2} \mathbb{P}_{x \leftarrow D_{i+1}}(\mathcal{A}(x) = 1) + \frac{1}{2} \mathbb{P}_{x \leftarrow \overline{D}_{i+1}}(\mathcal{A}(x) = 1)$$

$$= \mathbb{P}_{x \leftarrow D_{i+1}}(\mathcal{A}(x) = 1)$$

$$= \mathbb{P}(\mathcal{A}'(G(k)_{1,i}) = G(k)_{i+1})$$

$$= \frac{1}{2} + \frac{1}{2} \mathbb{P}_{x \leftarrow D_{i+1}}(\mathcal{A}(x) = 1)$$

$$= \frac{1}{2} + \text{Adv}_{\mathcal{A}}(D_i, D_{i+1})$$

$$\geq \frac{1}{2} + \frac{\epsilon}{n}$$
2.2 Building PRG is not easy

(Truncated) linear congruential generators.

\[ n[i] = a \times r[i-1] + b \mod p \]

with \( a, b, p \) parameters. \( n[0] = \text{seed} \).

These are very bad crypto PRG’s. Even if \( a, b, p \) are hidden, even if only the most significant bits of \( r[i] \) are output at every iteration.

Example: \texttt{rand()} in GLIBC:

\[ r[i] = r[i-3] + r[i-31] \mod 2^{32} \]

output 31 most significant bits. Completely predictable, never use this for cryptography.

LFSR (Linear Feedback Shift Registers)
Super efficient in hardware (CSS (DVD), GSM, Bluetooth)

RC4 (1987, RIVEST) used in HTTPS and WEP.
One of the classical default option. Not too bad... but

- never use it with correlated seeds,
- throw away the first output bits (there is a bias).

estream (Salsa20)

\[ \{0,1\}^s \times \{0,1\}^f \rightarrow \{0,1\}^w \]

With some seed but different randomness, you can get distinct outputs.
Chapter 3

Semantic security

**Definition 7.**

An encryption scheme \((\text{Gen}, \text{Enc}, \text{Dec})\) over a message space \(M\) is perfectly secure if for every adversary \(A\), the experiment \(\text{Exp}^{\text{ind}}\) outputs 1 with probability \(\frac{1}{2}\).

\(\text{Exp}^{\text{ind}}:\)

1. Adversary outputs a pair \(m_0, m_1 \in M\).
2. A random key \(k = \text{Gen}\). An random bit \(b\) is chosen to compute \(c = \text{Enc}_k(m_b)\) which is given to \(A\).
3. Adversary \(A\) outputs \(b' \in \{0, 1\}\). Return 1 iff \(b' = b\)

Perfect secrecy:

- can be achieved (VERMAN’s cipher).
- But requires keys as long as messages.
- Shorter keys can be used if we only want computational secrecy.

**Definition 8.**

A function \(f : \mathbb{N} \rightarrow \mathbb{N}\) is negligible if, for every polynomial \(P\), there exists an \(N \in \mathbb{N}\) such that, for all integers \(n > N, f(n) < \frac{1}{P(n)}\).

**Example 1.**

- \(n \mapsto 2^{-n}\)
- \(n \mapsto 2^{-\sqrt{n}}\)
- \(n \mapsto n^{-\log n}\)
Indistinguishability against passive adversary. Consider the following experiment between challenger and adversary.

Experiment $\text{Exp}^{ss-b}(n)$

1. Adversary is given a security parameter $1^n$ and chooses messages $m_0, m_1$ of the same length $|m_0| = |m_1|$.
2. Challenger generates a key $k$ and computes $c' = \text{Enc}_k(m_b)$ which is given as a challenge to $A$.
3. Adversary $A$ outputs a bit $b' \in \{0, 1\}$ and $\text{Exp}^{ss-b}(n)$ outputs $b' \in \{0, 1\}$.

$$\text{Adv}(n) := |\Pr (A'(c') = 1 | b = 1) - \Pr (A'(c') = 1 | b = 0)|$$

**Definition 9.**

A scheme $(\text{Gen}, \text{Enc}, \text{Dec})$ is semantically secure against passive (eavesdropping) if, for any PPT adversary $A$, there exists a negligible function $\text{negl} : \mathbb{N} \to \mathbb{N}$ such that $\text{Adv}_A(n) < \text{negl}(n)$.

**Remark 6.**

- The key $k$ is only used once
- In a stronger definition, we do everything with $n$-tuple.

Let $G : \{0, 1\}^n \to \{0, 1\}^l$ be a pseudo random generator ($n \ll l$). The following scheme encrypts $l$-bit messages:

- $\text{Gen}(1^n)$ given a security parameter $1^n$, choose $k \in \{0, 1\}^n$ and outputs it as a key.
- $\text{Enc}_k(m)$ to encrypt $m \in \{0, 1\}^{l(n)}$ under $k \in \{0, 1\}^n$, output $c = m \oplus G(k) \in \{0, 1\}^{l(n)}$.
- $\text{Dec}_k(c)$ to decrypt $c \in \{0, 1\}^{l(n)}$ using $k \in \{0, 1\}^k$, outputs $m = c \oplus G(k)$.

**Theorem 5.**

If $G$ is a PRG, the scheme is semantically secure against passive adversaries.

**Proof.** Let us assume a PPT adversary with non negligible advantage $\epsilon$ against the semantic secrecy, we build a PRG distinguisher $D$ with advantage $\epsilon/2$. 

---

14
Chapter 4

Block ciphers

A block cipher is an efficient, keyed permutation.

\[ F : \{0,1\}^n \times \{0,1\}^l \rightarrow \{0,1\}^l \]

where \( n \) is the key length and \( l \) is the block length such that for each \( k \in \{0,1\}^n \), \( F_k(x) = F(k,x) \) is a bijection over \( \{0,1\}^l \). Moreover \( F_k \) and \( F_k^{-1} \) are efficiently computable given \( k \).

Definition 10.

- \( n \) and \( l \) are fixed constants; security is assessed in terms of concrete security rather than asymptotically.
- Should be viewed as building blocks for encryption schemes.
- Should be indistinguishable from random functions over \( \{0,1\}^l \) for any adversary running in any reasonable amount of time.
- A block cipher is good if the best known attack is essentially as costly as exhaustive search over the key space.
- Should behave like a random permutation over \( \{0,1\}^l \) although a random permutation with block length \( l \) requires \( \log(2^{2^l}) \approx l2^l \) bits in its representation.

4.1 DES

Key length 56.
Base length \( l = 64 \).
Only the 56-bits key is not public.
Despite 30 years of attacks, best attack is still exhaustive search over the key space.

4.2 AES

AES replace DES in 2001. Key size is 128, 192 or 256. Number of rounds : 10, 12 or 14. Block size is 128 bits. AES is not a Feistel network but substitution-permutation networks.
Chapter 5

Formalization of primitives

5.1 PRFs and PRPs

A PRF is a deterministic function $F : \{0, 1\}^k \times \{0, 1\}^n \rightarrow \{0, 1\}^m$ that cannot be distinguished from a random function $R : \{0, 1\}^n \rightarrow \{0, 1\}^m$. Consider two experiments:

$\text{Exp}_0$:
1. Choose $k \leftarrow \{0, 1\}^t$.
2. Adversary repeatedly chooses $x \in \{0, 1\}^n$ and asks for $f(x)$. Challenger returns $F(k, x)$.
3. Adversary outputs a bit $b \in \{0, 1\}$.

$\text{Exp}_1$:
1. Choose a random function $F : \{0, 1\}^n \rightarrow \{0, 1\}^m$.
2. Adversary repeatedly chooses $x$ and asks for $f(x)$. Challenger returns $F(x)$.
3. Adversary outputs a bit $b \in \{0, 1\}$.

$\text{Adv}_{\text{PRF}}^A(1^t) = |\mathbb{P}(A^{\text{Exp}_0}(1^t) = 1) - \mathbb{P}(A^{\text{Exp}_1}(1^t) = 1)|$

$F$ is a PRF family if for any PPT adversary, $\text{Adv}_{\text{PRF}}^A(1^t) \leq f(t)$ for some negligible function $f$.

A PRP is a deterministic function $F : \{0, 1\}^n \times \{0, 1\}^k \rightarrow \{0, 1\}^n$.

- For $K \in \{0, 1\}^k$, $x \mapsto F(x, K)(= F_K(x))$ is a bijection from $\{0, 1\}^n$ to $\{0, 1\}^n$.

- For $K \in \{0, 1\}^k$, $x \mapsto F(x, K)$ is polynomial

- For all PPT distinguishers $D : |\mathbb{P}(D^{F_K}(1^n) = 1) - \mathbb{P}(D^{f_n}(1^n) = 1)| < \epsilon(k)$, where $K \leftarrow \{0, 1\}^k$ is chosen uniformly at random and $f_n$ is chosen uniformly at random from the set of permutations on $n$-bit strings.
5.2 Constructing PRPs from PRFs

Feistel networks allow constructing invertible functions from non-invertible ones. The $i$-th round takes as input $L_{i-1}||R_{i-1} \in \{0, 1\}^n$ where $L_{i-1}, R_{i-1} \in \{0, 1\}^{\frac{n}{2}}$ and outputs $L_i||R_i$ where $L_i = R_{i-1}$ and $R_i = L_{i-1} \oplus f_i(R_{i-1})$ where $f_i$ is not necessarily invertible.

For each $r > 0$, we call, Feistel $f_1, \ldots, f_r$ the Feistel network obtained from $f_1, \ldots, f_r$. Feistel $f_1, \ldots, f_r$ is invertible even if $f_1, \ldots, f_r$ are not.

If $F$ is a PRF, a 1-round Feistel networks $F_k : \{0, 1\}^n \rightarrow \{0, 1\}^n$ is a permutation but not a PRP since the first $\frac{n}{2}$ bits of $F_k^n(x)$ are the last $\frac{n}{2}$ bits of $x$.

Two rounds $F_{k_1,k_2} : \{0, 1\}^n \rightarrow \{0, 1\}^n$ do not suffice to give a PRP either.

### Theorem 6.

3-round Feistel networks with $F : \{0, 1\}^{\frac{n}{2}} \rightarrow \{0, 1\}^{\frac{n}{2}}$ a PRF using independent random keys at each round, gives a weak PRP.

Strong PRP : Let $F : \{0, 1\}^{\frac{n}{2}} \rightarrow \{0, 1\}^{\frac{n}{2}}$ be a keyed function. Define the keyed permutation $F^{(4)}$ as follows.

- Parse the key $k \in \{0, 1\}^t$ as $k = (k_1, \ldots, k_t)$ with $|k_i| = t$ for each $i$.
- Parse the input $x \in \{0, 1\}^n$ as $x = (L_0, R_0)$ with $L_0, R_0 \in \{0, 1\}^{\frac{n}{2}}$.
- Computes
  1. $L_1 = R_0$ and $R_1 = L_0 \oplus F_{k_1}(R_0)$
  2. $L_2 = R_1$ and $R_2 = L_1 \oplus F_{k_2}(R_1)$
3. \( L_3 = R_2 \) and \( R_3 = L_2 \oplus F_k(R_2) \)
4. \( L_4 = R_3 \) and \( R_4 = L_3 \oplus F_k(R_3) \)

**Theorem 7 (LULY-RACHOFF, 1988).**

The above function is a strong PRP if \( R \) is a PRF.

In the opposite direction, any PRP is also a PRF when the domain is sufficiently large.

**Lemma 8 (PRP/PRF switching lemma).**

If \( E : \{0, 1\}^t \times \{0, 1\}^n \rightarrow \{0, 1\}^m \) is a PRP, then if \( A \) makes at most \( q \) queries, then

\[
\left| \text{Adv}^{\text{PRF}}(E) - \text{Adv}^{\text{PRP}}(E) \right| < \frac{q^2}{2^{n+2}}
\]

### 5.2.1 Constructing PRF form PRG and vice-versa

**Easy direction:** PRF ⇒ PRG. Let \( F : \{0, 1\}^t \times \{0, 1\}^n \rightarrow \{0, 1\}^m \).

Define \( G(k) = (F_k(0), F_k(1), \ldots, F_k(d-1)) \) to get a function \( G : \{0, 1\}^t \rightarrow \{0, 1\}^{dm} \).

If \( G \) can be distinguished from a true random generator, then \( F \) can be distinguished from a random function family.

**Harder direction:** PRG ⇒ PRF. Let us assume a PRG \( G : \{0, 1\}^t \rightarrow \{0, 1\}^{2t} \) (for each \( s \in \{0, 1\}^t \), we write \( G(s) = (G_0(s), G_1(s)) \) where \( G_0(s), G_1(s) \in \{0, 1\}^t \).

Define \( F_k(0) = G_0(k) \) and \( F_k(1) = G_1(k) \).

If \( G \) is a secure PRG, \( F \) is a sequence PRF over \( \{0, 1\} \). If a distinguisher can tell the difference with a true random permutation, we can break the PRG.

### 5.3 How to encrypt with a PRF/PRP?

One thing not to do, use the naïve Electronic Code Book (ECB) mode of operation which encrypts blocks messages \( m = (m_1, \ldots, m_l) \) as \( C = (F_k(m_1), \ldots, F_k(m_l)) \) using a PRP \( F \).

### 5.3.1 Chosen plaintext security (CPA)

Adversary obtains multiple encryption under \( k \) for message of its choice.

Consider 2 experiments:

- **Exp_0:** Adversary makes multiple encryption queries \( \{(m_i, 0), m_i, 1\}_{i \in [1, n]} \). Eventually, \( A \) outputs \( b \in \{0, 1\} \).

- **Exp_1:** \( Adv \) makes encryption queries \( \{(m_{i,0}, m_{i,1})\} \) and gets \( \{e_i\} \). \( Adv \) outputs a bit.
$$\text{Adv}_{A}^{\text{CPA}} = \left| P \left( A^{\text{Exp}_{0}}(1^t) = 1 \right) - P \left( A^{\text{Exp}_{1}}(1^t) = 1 \right) \right|$$

All previous constructions fail to provide CPA security because they are deterministic.

A first example of CPA secure scheme: let $F : \{0,1\}^k \times \{0,1\}^n \rightarrow \{0,1\}^m$. To encrypt $m \in \{0,1\}^n$ under the key $k \in \{0,1\}^k$, choose $r \in \{0,1\}^m$ at random and compute $c = (r, m \oplus F_k(r))$.

Problem: $\text{bit_length(ciphertext)} > \text{bit_length(plaintext)}$.

Better solution: CBC (cipher blocks chaining).

Cut message $m$ into blocks of length 256 output of $E$.

ciphertext length = plaintext length + 1 block (IV: Initiation Vector).

**Remark 7.**

- If $\text{bit_length(m)}$ is not a multiple of 256 use padding PKCS#5 (and more) (not a proper standard, a recommendation of the RSA company). Last block: incomplete. Let $d$ be the number of remaining bytes. Fill the block with $d || d || \cdots || d$ where $d$ is encoded on 8 bits. If the last block is complete, add a “dummy block” $32 || 32 || \cdots || 32$.

- Sequential encryption, parallel decryption.

**Theorem 9.**

If there exists a PPT attacher $A$ against multiple message IND-CPA security against CBC, there exists a PPT $B$ against the strong PRP $E$ such that

$$\text{Adv}_{A}^{\text{CPA}}[\text{CBC}] \leq 2 \cdot \text{Adv}_{B}^{\text{PRP}}[E] + \frac{2q^2d^2}{2^n}$$


If LHS is non negligible, then $\text{Adv}_{B}$ is non negligible.
Even better solution: counter mode (CTR)

If there exists a PPT $A$ against multiple messages IND-CPA security of CTR, then there exists a PPT $B$ against the PRF $F$, such that:

$$Adv_{C}^{CPA}[CTR] \leq 2Adv_{B}^{PRF}(F) + \frac{2q^2d}{2^n}$$

List of advantages

- lighter security proof.
- use a PRF instead of a strong PRP.
- Encryption is parallel.
- No need for padding (if $F_k(IV + d)$ is too long, cut is and xor what remains with $m[d]$).
- no error propagation.

5.4 MACs and CCA-secure encryption

MAC: message authentication code.

Goal: provide integrity and authenticity of a message (no confidentiality requirement).

This is different from error detection code! Random error vs active adversary.
5.4.1 MACs

**Definition 13.**

A MAC over \((K, M, T)\) is a triple \((KeyGen, Sign, Verify)\) of PPT algorithms

\[
\begin{align*}
KeyGen & \rightarrow k \in K \\
Sign & : (k, m) \rightarrow t \\
Verify & : (k, m, t) \rightarrow \begin{cases} 
0 \\
1 \text{ means "OK"}
\end{cases}
\end{align*}
\]

Correctness \(\forall k, \forall m\), we have \(Verify(k, m, Sign(k, m)) = 1\).

**Definition 14.**

Existential unforgeability under chosen message attack (or CMA).

1. \(C: k \leftarrow KeyGen\)
2. \(A \rightarrow C: m^{(i)}\)
3. \(C: t^{(i)} = Sign(k, m^{(i)})\)
4. \(C \rightarrow A: t^{(i)}\) (repeat 2, 3 and 4 arbitrary many times).
5. \(A \rightarrow C: (m, t)\)
6. \(C: \text{If } (m, t) \notin \{(m^{(i)}, t^{(i)})\} \text{ and } Verify(m, t) = 1 \text{ then outputs 1, otherwise outputs 0.}\)

\[
Succ^{MAC}_{A} = \mathbb{P}(C \text{ outputs 1})
\]

The attacker \(A\) wins if \(Succ^{MAC}_{A}\) is non negligible.

5.4.2 Constructing MACs

MAC for fixed length messages with small length: use a PRF \(F: K \times X \rightarrow Y\). \(KeyGen(\text{Mac}) = KeyGen(\text{PRF})\) (sample \(f \leftarrow K \) uniformly). \(Sign_{MAC}(k, m): t = F(k, m)\). \(Verify_{MAC}(k, m, t)\) : outputs 1 iff \(t = F(k, m)\).

**Theorem 11.**

If there is a PPT \(A\) against e.u.CMA security of this MAC, then there is a PPT \(B\) breaking PRF \(F\) with

\[
Adv^{PRF}_{B}[F] \geq Succ^{MAC}_{A}[MAC_F] - \frac{1}{|Y|}
\]

Breaking PRF: \(\text{Exp}_{\text{Real}}\)
1. $C: k \leftarrow \text{KeyGen}$
2. $B \rightarrow C: m^{(i)}$
3. $C: c^{(i)} = F(k, m^{(i)})$
4. $C \rightarrow B: c^{(i)}$
5. $B: b \in \{0, 1\}$

$Exp_{\text{Ideal}}$:
1. $C: f \leftarrow \mathcal{U}(X \rightarrow Y)$
2. $B \rightarrow C: m^{(i)}$
3. $C: c^{(i)} = f(m^{(i)})$
4. $C \rightarrow B: c^{(i)}$
5. $B: b \in \{0, 1\}$

\[
\text{Adv}^\text{PRF}_B = \left| \mathbb{P}(B \xrightarrow{\text{Exp}_{\text{Real}}} 1) - \mathbb{P}(B \xrightarrow{\text{Exp}_{\text{Ideal}}} 1) \right|
\]

If in the MAC, we replace $F$ by a uniform $f : X \rightarrow Y$. If $(m, t)$ is the "forgery", then $m$ has not been queried before and $t = f(m)$. Then $A$ succeeds with probability $\frac{1}{|Y|}$.

1. $A \rightarrow B: m^{(i)}$
2. $B \rightarrow C: m^{(i)}$
3. $C \rightarrow B: c^{(i)}$
4. $B \rightarrow A: c^{(i)}$
5. $A \rightarrow B: (m, t)$
6. $B \rightarrow C: m$
7. $C \rightarrow B: c$
8. $B: (m, t) \notin \left\{ (m^{(i)}, c^{(i)}) \right\}$ and $c = t$? Reply 1 (we are in $Exp_{\text{Real}}$). Else reply 0.

If $Exp_{\text{Real}}$, then $c^{(i)}$ is the correct tag.
In $Exp_{\text{Real}}$, this is $t = F(k, m)$ (valid forgery).
In $Exp_{\text{Real}}$, $B$ outputs 1 $\iff A$ provides a forgery.
In $Exp_{\text{Ideal}}$, $B$ outputs 1 with probability $\leq \frac{1}{|Y|}$.

\[
\text{Adv}^\text{PRF}_B = |\text{Succ}_A^{\text{MAC}}| \\
\geq \text{Succ}_A^{\text{MAC}} - \frac{1}{|Y|}
\]

Larger length MAC, but still fixed. CBC-MAC.
Only 1 output, $F$ only PRF, IV=0.

Exercise: show that if all outputs of $F$ are given as part of the signature then this leads to an insecure PRF.
For variable length messages: ECBC MAC (encrypted CBC)

\[ \text{Sign}_{\text{ECBC}} = F(k', \text{Sign}_{\text{CBC}}(k, m)) \]

where \( k \) and \( k' \) are uniformly and independently chosen.

**Theorem 12.**

If there exists a PPT \( A \) against e.u. CMA security of ECBC(F), then there exists a PPT \( B \) against PRF \( F \), with

\[ \text{Adv}^\text{MAC}_{A}[\text{ECBC}] \leq \text{Adv}^\text{PRF}_{B}[P] + \frac{2q^2}{2^n} \]

### 5.4.3 CCA-secure encryption

\((\text{KeyGen}, \text{Enc}, \text{Dec})\) over \( K \times P \times C \) is a CCA-secure encryption scheme if

- \( \text{KeyGen} : \rightarrow k \in K \)
- \( \text{Enc} : (k, m) \mapsto c \in C \)
- \( \text{Dec} : (k, c) \mapsto m \in P \) or \( \perp \) (decryption fail).

Correctness property: \( \forall k, \forall m, \text{Dec}(k, \text{Enc}(k, m)) = m \).

CCA-security \( \text{Exp}_b \) with \( b \in \{0, 1\} \).

1. \( C : k \leftarrow \text{KeyGen} \)
2. \( A \to C : m^{(i)}_0, m^{(i)}_1 \)
3. \( C : c^{(i)} = \text{Enc}(k, m^{(i)}_b) \)
4. \( C \to A : c^{(i)} \)
5. \( A \to C : c^{(i)} \)
6. \( C : m^{(i)} = \text{Dec}(k, c^{(i)}) \)
7. \( C \to A : m^{(i)} \)
8. $A$: $b' \in \{0, 1\}$ (guess of $b$).

$\forall j, c^{(i)} \notin \{c^{(i)}\}$ (else the definition is not interesting).

$$Adv_A = \left| P\left( A^{Exp_0} 1 \right) - P\left( A^{Exp_1} 1 \right) \right|$$

$Adv_A$ should be negligible for all PPT $A$.

It seems off for $C$ to reply to decryption queries.

$A$ (encrypts price proposal) and $E$ (same) bidding with $B$ for a car. $B$ decrypt and replies. $A$ gets the car for $price_A$ (if $price_A > price_E$).

Maybe after several iterations $E$ extracts Alice's tray.

BLEICHENBACHER 1998 $\rightarrow$ CCA attacks on RSA secret keys / PKCS was changes because of this.

None of the CPA-secure encryption schemes we have seen in CCA-secure.

Let $(Enc, Dec)$ be a CPA secure encryption scheme. Let $(Sign, Verify)$ be a secure MAC.

Consider $(Enc', Dec')$ defined as follows. $k = (k_1, k_2)$ where $k_1$ is the key for $(Enc, Dec)$ and $k_2$ is the key for the MAC.

$$Enc'(M) : c_1 \leftarrow Enc(k, M)$$
$$c_2 \leftarrow Sign(k_2, c_1)$$
$$Send(c_1, c_2)$$

$$Dec'(M) : If \ Verify(k_1, c_2) = 0, output \bot else output Dec(k_1, c_1)$$

---

**Theorem 13.**

$(Enc', Dec')$ is CCA secure

---

**Proof.** Admitted.

**Intuition.** If attackers makes a decryption query, $(c_1, c_2)$, we have

- challenger replies $\bot$
- this is the ciphertext output by a plaintext query $\rightarrow$ forbidden
- $c_2$ is a valid tag for $c_1 \rightarrow A$ has broken the MAC.

If $A$ doesn't break the MAC, decryption queries are useless and we are looking at CPA security.
Chapter 6

Cryptographic hash functions

Central tool in crypto used in many contexts in symmetric crypto and public key crypto.
Until 10-12 years, only two main hash functions: MD5 (RIVEST) and SHA-1 (NSA).
2004-05: Xiaoyun Wang broke MD5, SHA1 and MD4 SHAO, Haval-128, RIPEMD
Only one hash function left: SHA-32 (also called SHA 256).
NIST organized a competition. It lasted until 2012. Then Kaccack was chosen to be SHA-3.

6.1 Definition

Function from $D$ to $R$ with $|D| \gg |R|$. By the pigeon hole principle, there exists many collisions.
Goal: finding a collisions is computationally difficult.

A hash function is a pair $(Gen, Hash)$ of PPT algorithms:

- $Gen$ takes as input a security parameter $n$ in unary, and outputs a key $s$.
- $Hash$ is deterministic.

$$H: \{0,1\}^{key\text{-length}} \times \{0,1\}^{l'(n)} \rightarrow \{0,1\}^{l(n)}$$

$$(s, m) \mapsto H_s(m)$$

If $l'(n) < \infty$, we speak of fixed length input. If $\{0,1\}^{l'(n)}$ is $\{0,1\}^*$ we speak of variable length input. If $l'(n) < \infty$, $\frac{l'(n)}{l(n)}$ is called compression factor.

6.1.1 Collision-Resistance

$(Gen, H)$ is said collision-resistant if no PPT $A$ wins the following game with non negligible probability.
The key of the hash function is public!

1. $C: s \leftarrow Gen$
2. $C \rightarrow A: s$
3. $A \rightarrow C: m, m'$
4. If $|m| = |m'| = l'(n) \land H_s(m) = H_s(m') \land m \neq m'$, then wins.

Other notions of security:

- pre-image resistance: given $s, y$ find $m$ such that $y = H_s(m)$.
- second pre-image resistance: given $s, m$ find $m' \neq m$ such that $H_s(m) = H_s(m')$.

Exercise: $H$ collision resistant $\rightarrow H$ 2nd pre-image resistant. $H$ collision resistant $\rightarrow H$ pre-image resistant.

Random Oracle Model: $H_s$ is modelled as a truly random function into the range. For any $x$ not queried so far, $H_s(x)$ is uniform.

This is only a model. We can do security proofs in this model (as opposed to the standard model).

### 6.2 Birthday paradox

Best generic attack against collision-resistance.

**Theorem 14.**

Fix $R$ with cardinality $N, q > 0$. Let $y_1, \ldots, y_q$ be independent samples in $R$. Then the probability that there exists $i \neq j$ such that $y_i = y_j$ is $\geq 1 - \exp\left(-\frac{q(q-1)}{2N}\right)$

**Remark 8.**

For $q \leq \sqrt{2N}$, this is $\geq \frac{q(q-1)}{4N}$. For $q \geq 1.1\sqrt{N} + 1$, this is $\geq 0.7$.

Proof for the uniform distribution.

$$P(\text{no coll}) = \left(1 - \frac{1}{N}\right) \left(1 - \frac{2}{N}\right) \cdots \left(1 - \frac{q-1}{N}\right) \geq \prod_{i=1}^{q-1} \exp\left(-\frac{i}{N}\right) = \exp\left(-\frac{1}{N} \frac{q(q-1)}{2}\right)$$

If $H$ output is $n$ bits long, the birthday attack cost $2^{\frac{n}{2}}$.

### 6.3 Building hash functions

From fixed length hashing to variable length hashing: MERKLE-DAMGÅRD.

Assume we have $(Gen, F)$ collision-resistant with $l'(m) = 2l(m)$.

We define $H$ by:
where the last block is the length of the message.

**Theorem 15.**

\((\text{Gen}, F)\) collision resistant \(\Rightarrow (\text{Gen}, H)\) collision resistant.

**Remark 9.**

The good measure of efficiency of a hash function is compression per time unit.

**Proof.** We assume we have a collision for \(H\). We want to derive a collision for \(h\).

As \(H(m) = H(m')\), we have \(F(z_B||L) = F(z'_B||L')\)

- Case 1: \(z_B||L \neq z'_B||L'\) \(\Rightarrow\) Ok because we can recover \(z_B, L, z'_B, L'\)
- Case 2: \(z_B||L = z'_B||L'\) \(\Rightarrow\) \(L = L', B = B'\) and \(z_B = z'_B\).

By definition of collision for \(H\) there exists \(i \leq B\) maximal such that \(z_{i-1}||m_i \neq z'_{i-1}||m'\).

Then \(F(z_{i-1}||m_i) = z_i = z'_i = F(z'_{i-1}||m'_{i-1})\)

There exist constructions of hash functions from block-ciphers.

**DOVIES-MEYER:** \(h(m_0||m_1) = \text{Enc}(m_0, m_1) \oplus m_1\)

**MIYEGUCHI-PRENEEL:** \(h(m_0||m_1) = \text{Enc}(m_0, m_1) \oplus m_1 \oplus m_0\)

No proof of collision resistance in standard model. \(\exists\) proofs in the ideal cipher model\(^*\).

### 6.4 MACs based on hash functions

\(\text{Sign}(k, m) = H(k||m)\).

Exercise: show that if \(H\) is derived from MD construction, than this MAC is insecure.

**HMAC:**
If $h$ is collision-resistant, if $M \rightarrow h_s(k \parallel M)$ is a secure PRF and if $k \rightarrow h_s(k \oplus ipad) \parallel h_s(k \oplus opad)$ is a secure PRG then $HMAC(h)$ is a secure MAC.

Theorem 16.

Why use HMAC rather than ECBC-MAC?
Chapter 7

Public key encryption

Secret key crypto assume that any two communicating users share a common key ⇒ 3 problems.
Problem 1 Key distribution is costly
Problem 2 Users have to meet on real life to agree on a key.
Problem 3 Have to meet again to update the key.
One solution: involve a Trusted third party (TTP) = key distribution center.
Let \( (k_A, k_B) \) the key shared between \((A, B)\) and the TTP.
To obtain a \( k_{AB} \) shared by A and B:
- A contacts TTP tells he wants to chat with B.
- TTP sends \( c_A = \text{Ens}_{k_A}(A, b\|k_{AB}) \) and \( c_B = \text{Ens}_{k_B}(A, b\|k_{AB}) \)
- A decrypts \( c_A \) and sends \( c_B \) to B.

⇒ security against eavesdropping adversary due to the CPE security of \((\text{Enc}, \text{Dec})\).
But:
- TTP must be fully trusted
- Single point of security: if TTP is compromised security is.
- There are active attacks to worry about

Example 2.
Kerberos (sued by MS Windows for securing local networks).

Question: can we avoid TTPs?
Breakthrough result 1976: "New directions in Crypto" by DIFFIE and HELLMAN introduced:
- Public-key encryption (PKE),
- Digital signature,
- Interactive key exchange.

First proposals of PKE: RSA, McEliece, ElGamal...
7.1 Public-key encryption

**Definition 16.**

A PKE scheme is a tuple of PPT algorithms \((\text{KeyGen}, \text{Enc}, \text{Dec})\)

- \(\text{KeyGen}(1^n)\): given a security parameter \(1^n\) outputs \((p_k, s_k)\). mode of a public key \(p_k\) and a private key \(s_k\).
- \(\text{Enc}(p_k, M)\) given \(p_k\) and a message \(M\), outputs a ciphertext \(c\).
- \(\text{Dec}(s_k, c)\) given \(s_k\) and a ciphertext \(c\) output a plaintext \(M\).

Correctness: for all \((p_k, s_k) \leftarrow \text{KeyGen}(1^n)\) and all \(M\), we have \(\text{Dec}(s_k, \text{Enc}(p_k, M)) = M\).

**Remark 10.**

- \(\text{Enc}\) may be probabilistic; \(\text{Dec}\) is deterministic.
- \(p_k\) is public by available; \(s_k\) is kept secret.
- Several advantages over secret key crypto: only one \((p_k, s_k)\) for everyone interacting with the receiver. no complex key distribution phase (just put \(p_k\) in a public directory).
- Issues:
  - How do we know that \(p_k\) is the genuine public key?
  - Most often, much slower the secret key crypto (by a factor \(10^3\)): usual solution is hybrid encryption.

Security notion: Semantic (aka indistinguishably under chose-plaintext attacks or IND-CPA security). Consider the experiment \(\text{Exp}^{\text{IND-CPA}}(1^n)\):

1. Challenger generates \((p_k, s_k) \leftarrow \text{KeyGen}(1^n)\) and gives \(p_k\) to the adversary \(A\).
2. A chooses two equals-length messages \(M_0, M_1\) challenger. Picks \(\beta \in \mathcal{U}\{0, 1\}\) and returns \(c \leftarrow \text{Enc}(p_k, M_\beta)\) to \(A\).
3. A outputs \(\beta' \in \{0, 1\}\) and wins if \(\beta = \beta'\) in which case, the challenger outputs 1. Otherwise challenger outputs 0.

\[
\text{Adv}_A^{\text{IND-CPA}}(1^n) = \left| \Pr \left( \text{Exp}^{\text{IND-CPA}}(1^n) = 1 \right) - \frac{1}{2} \right|
\]
\[
= \frac{1}{2} \left| \Pr \left( A(p_k, c) = 1 | \beta = 1 \right) - \Pr \left( A(p_k, c) = 1 | \beta = 0 \right) \right|
\]

We say that \((\text{KeyGen}, \text{Enc}, \text{Dec})\) is IND-CPA secure if no PPT adversary \(A\) has non negligible advantage.
Remark 11.

- no need for encryption queries since \( p_k \) is public.
- Encryption must be probabilistic to enable IND-CPE security.
- Unlike secret key crypto, one time security implies many times security.

Hybrid encryption; Alice public key \( p_k \) on her web page. Sender picks a random secret key \( K \), encrypts it as \( c \leftarrow \text{Enc}(p_k, K) \) and sends \( c \) to Alice. Then they both share \( K \).

IND-CPA security of the PKE scheme ensures the security of the process since the distribution

\[
\{ (p_k, \text{Enc}(p_k, K)) \}
\]

is indistinguishable from

\[
\{ (k, \text{Enc}(p_k, K), U) \}
\]

for a random string \( U \).

Otherwise, we could build an IND-CPA adversary against PKE.

7.2 The discrete logarithm, DIFFIE-HELLMAN and Decision DIFFIE-HELLMAN problem

Definition 17.

Discrete logarithm problem (DLP): given a cyclic group \( G \) of prime order \( q \), a generator \( g \in G \) and \( g^a \) for some random \( a \in \mathbb{Z}/q\mathbb{Z} \).
Find \( a \in \mathbb{Z}/q\mathbb{Z} \).

Definition 18.

Computational DIFFIE-HELLMAN (CDH) problem: given a cyclic group \( G \) of order \( q \) with a generator \( g \in G \) and \( (g, g^a, g^b) \) for random \( a, b \in \mathbb{Z}/q\mathbb{Z} \), compute \( g^{ab} \).

Remark 12.

If we can solve DLP we can also solve CDH
Definition 19.

Decision Diffie-Hellman (DDH) problem. In a cyclic group $G$ of order $q$ with a generator $g \in G$, distinguish the distributions

$$D_0 = \{(g, g^a, g^b, g^{ab}) | a, b \in \mathbb{Z}/q\mathbb{Z}\}$$

and

$$D_1 = \{(g, g^a, g^b, g^c) | a, b, c \in \mathbb{Z}/q\mathbb{Z}\}$$

For a distinguisher $D$, we define

$$\text{Adv}^\text{DDH}_D (1^n) = \left| P \left( D(g, g^a, g^b, g^{ab} | a, b \in \mathbb{Z}/q\mathbb{Z} \right) - P \left( D(g, g^a, g^b, g^c | a, b, c \in \mathbb{Z}/q\mathbb{Z} \right) \right|$$

How can we choose $G$?

- Bad choice is $(\mathbb{Z}/p\mathbb{Z}, +)$ with a generator $k$ coprime with $p$. DLP is easy given $ak \mod p$ computing $a$ is easy.
- Historical choice $(\mathbb{Z}/p\mathbb{Z}^*, \cdot)$ is a cyclic group of order $p - 1$. If $p = 2q + 1$ where $p, q$ are both prime. $(\mathbb{Z}/p\mathbb{Z})^*$ contains a subgroup of order $q$.

Question: How to choose $p, q$? How to find an element $g \in (\mathbb{Z}/p\mathbb{Z})^*$ of order $q$? Choose $a \in \mathbb{Z}/p\mathbb{Z}$ and test if $a^q = 1 \mod p$.

7.3 The El Gamal encryption scheme (1984)

Used in PGP.

**Keygen**(1^n): choose a cyclic group $G$ of prime order $q$ (with $|q| = n$) with a generator $g \in G$. Choose $x \in \mathbb{Z}/q\mathbb{Z}$ and compute $X = g^x \in G$. Define $p_k := (G, g, W = g^x)$ and $s_k := x \in \mathbb{Z}/q\mathbb{Z}$.

**Enc**(pk, M) To encrypt $M \in G$, choose $r \in \mathbb{Z}/q\mathbb{Z}$ and compute $c = (c_0, c_1) = (MX^r, g^r)$.

**Dec**(sk, c): to decrypt $c = (c_0, c_1)$ using $s_k = x$, compute $M = c_0 / c_1^*$.  

Correctness: $(c_0, c_1) = (MX^r, g^r) = (M(g^x)^r, g^r) = (M^2 c_1^*, c_1)$.

**Theorem 17.**

ElGamal is IND-CPA secure iff DDH is hard in $G$.

**Proof.** Let $A$ be an IND-CPA adversary with advantage $\epsilon$. We build DDH distinguisher with advantage $\epsilon$ using $A$. $B$ takes as input a DDH instance $(g, g^a, g^b, T)$ where either $Ta^{ab}$ or $T \in G = \langle g \rangle$.

$B$ builds $p_k = (G = \langle g \rangle, g, X) = (G, g, g^a)$ (implicitly $s_k = a$).

$A$ chooses $M_0, M_1 \in G$. Then $B$ picks a random $\beta \in \{0, 1\}$ and sets $c = (c_0, c_1) = (M, T, g^b)$.

$A$ outputs $\beta \in \{0, 1\}$.

If $\beta = \beta'$, then $B$ outputs 1 (meaning that $T = g^{ab}$)

If $\beta \neq \beta'$, then $B$ outputs 0 (meaning that $T \in G$)
If \( T = g^{ab} \), \( c = (c_0, c_1) = (M_\beta(g^a)^b, g^b) = (M_\beta X^b, g^b) \) which is a valid encryption of \( M_\beta \).

\[
\mathbb{P} \left( \beta = 1 \mid T = g^{ab} \right) = \mathbb{P} \left( \beta' = \beta \mid T = g^{ab} \right) = \varepsilon + \frac{1}{2}
\]

If \( R \in G \) we can write \( T = g^{ab+c} \) where \( c \in \mathbb{Z}/q\mathbb{Z} \).

\[
c = (c_0, c_1) = (M_\beta(g^a)^b g^c, g^b) = ((M_\beta g^c) X^b, g^b)
\]

\[
\mathbb{P} \left( \beta = 1 \mid T \in G \right) = \mathbb{P} \left( \beta' = \beta \mid T \in G \right) = \frac{1}{2}
\]

\[\square\]

**Remark 13.**

- Security is only guaranteed if \( M \in G \).
- What is we want to encrypt a string \( M \in \{0, 1\}^l \)?

### 7.4 The DIFFIE-HELMAN protocol

A and B exchange messages over a public channel and want to agree on a random key \( k_{AB} \). They run a protocol as follows. Let \( G \) be a cyclic group of order \( q \) such that \( |q| = n \).

1. \( A \) picks a random exponent \( a \in \mathbb{Z}/q\mathbb{Z} \) and computes \( g^a \) which is sent to \( B \).
2. \( B \) picks a random exponent \( b \in \mathbb{Z}/q\mathbb{Z} \) and computes \( g^b \) which is sent to \( A \).
3. They compute \( k_A = (g^b)^a = (g^a)^b = k_B \). They both know \( k_{AB} = g^{ab} \).

In \( \text{Exp}_{\text{real}} \), adversary is given a transcript \( \text{trans} \) and \( k_{AB} \).

In \( \text{Exp}_{\text{ideal}} \), adversary is given a transcript \( \text{trans} \) and \( k \in \{0, 1\}^* \).

Protocol is secure against passive attacks if

\[
\text{Adv}_A(1^n) = \left| \mathbb{P} \left( A^{\text{Exp}_{\text{real}}(1^n)=1} \right) - \mathbb{P} \left( A^{\text{Exp}_{\text{ideal}}(1^n)=1} \right) \right| = \text{negligible}
\]

### 7.4.1 Min-in-the-middle attack

Charlie is between and speak to Bob pretending he is Alice and to Alice pretending he’s Bob.

1. \( A \to C: g^a \)
2. \( B \to C: g^b \)
3. \( C: c \leftarrow G \)
4. \( C \to A: g^c \)
5. \( C \to B: g^c \)
6. \( A: g^{ac} \)
7. \( B: g^{bc} \)
8. \( C: g^{ac} \) and \( g^{bc} \)

The problem here is **authentication**. To make the key exchange, we would need Authenticated Key Exchange.

### 7.5 RSA encryption

#### 7.5.1 Textbook RSA (Rivest Shamir Adleman, 1977)

**KeyGen:**
- \( p, q \) prime integers (pick \( p \) uniformly and random with some bit length). Check primality. If not prime, restart.
- \( n = pq \)
- \( f = (p - 1)(q - 1) = \varphi(n) \)
- Take \( e \) coprime with \( f \)
- And \( d \) such that \( ed = 1 \mod f \).

\( p_k = (n, e) \) and \( s_k = (n, d) \).

- \( e \): encryption exponent
- \( d \): decryption exponent

**Enc** (\( M \in \mathbb{Z}/n\mathbb{Z} \)) = \( M^e \mod n \)

**Dec** (\( C \in \mathbb{Z}/n\mathbb{Z} \)) = \( C^d \mod n \)

\( \rightarrow \) square and multiply modulo \( N \) (for ever operation, else number are too big for polytime).

**Correctness.**

\[
M^{ed} = M \mod p \\
M^{ed} = M \mod q
\]
and, by the Chinese remainder theorem

\[
M^{ed} = M \mod pq
\]

Not CPA secure: **Enc** is deterministic encryption. Never use Textbook RSA!

**PKLS#1 v.1.5: Padded-RSA**

**Enc:** for \( M \) on \( l < \log_2 n \) bits.

Let \( M' = (0 \ldots 0||0\ldots 0|r||0\ldots 0\|M) \) with \( r, k - l - 3 \) uniform bytes. Send \( c = (M')^e \mod n \).

Maybe in a couple of weeks. RSA-OAEP is not recommended instead.

**Efficiency:** cost of encryption: \( O \left( \log_2 e \log_2^2 N \right) \); cost of decryption: \( O \left( \log_2 d \log_2^2 n \right) \).
and \(d\) cannot be small together. Small \(d\) is insecure. If \(d \leq N^{1 - \frac{1}{2} \sqrt{2}} \approx N^{0.292}\), it can be recovered in poly-time (Boneh-Durfee, 1997*).

Small \(e\) is used a lot. \(e = 1\) insecure; \(e = 2\) never invertible mod \(\phi\). \(e = 3\), used a lot. Other classical choice: \(2^{16} + 1\). In that case, \(\log_2 d \approx \log_2 n\).

Use CRT to compute \(C^d \mod n\).

\[
\begin{align*}
C^d \mod p & \} \text{CRT} : C^d \mod pq \\
C^d \mod q & 
\end{align*}
\]

**Remark 14** (Security remarks):

- Factoring \(N\) ⇒ breaking RSA (best known attack). Opposite direction unknown.
- Best known algorithm for factoring: number field sieve:
  \[
  \exp \left( \mathcal{O} \left( (\log N)^{\frac{1}{3}} (\log \log N)^{\frac{2}{3}} \right) \right)
  \]
  (same as DLP over \((\mathbb{Z}/p\mathbb{Z})^\times\))

RSA assumption: for all ppt \(A\)

\[
\Pr \left( A(n, e, y) = x \text{ such that } x^e = y \mod n \right) \leq \text{negl}(\log n)
\]

\(p, q \leftarrow \text{uniform} \frac{n}{2}\)-bit primes. \(N = pq\). \(y \leftarrow \text{uniform} \mathbb{Z}/n\mathbb{Z}\).

**Remark 15.** There exists schemes that are provably CPA-secure, under the assumption that factoring is hard: Rabin’s encryption scheme (too costly to be used in practice).

### 7.5.2 The random oracle methodology

Assumption: there exists an efficient random function \(H\) taking inputs in \(\{0, 1\}^*\) outputting elements in a prescribed finite set, which can be accessed via an oracle.

\(H(x)\) unif. is \(x\) has not been queried before. \(H(x)\) consistent with prior query any is such a query has been made before.

Model for security proof. In practice, \(H = \text{SHA3}\) or any other crypto hash function in the proof, we model \(H\) as a random oracle.

Even though we know no RO exists, we hope that crypto functions faithfully emulate it.

Drawbacks: RO does not exist, once a description of \(H\) is made public, there is no randomness left.

Advantages: schemes proves secure in the ROM are typically much more efficient than those proved in the standard model. (SM is the “classical reduction framework”). A proof in the ROM is better than no proof (and can be a first step fir a proof in the standard model).

*http://crypto.stanford.edu/~dabo/papers/lowRSAexp.ps
How to make a proof in the ROM? Probabilities of success of attackers also hold over the randomness used by the RO. The simulator can program the RO (programmability).

RO provides a collision resistant hash function.

\[
\text{Gen } : \emptyset \\
\text{Hash } : x \mapsto H(x) \in \{0,1\}^n
\]

If \(H\) is modelled as a RO, this is collision resistant.

Security game: attacker \(A\) is given query access to \(H\), and has to find \(x \neq x'\) such that \(H(x) = H(x')\).

Security proof. Wlog., we assume that \(A\) had queried \(x\) and \(x'\) before outputting them.

Let \(x_1, \ldots, x_q\) be the queries of \(A\).

\[
\Pr(\text{\(A\) succeeds}) \leq \Pr(\exists i \neq j : H(x_i) = H(x_j))
\]

By the RO, all \(H(x_i)\)'s are independent uniform. By the birthday paradox, \(\Pr(\text{\(A\) succeeds}) \leq \frac{q^2}{n^2}\). \(\square\)

### 7.5.3 IND-CPA encryption from the RSA assumption (in the ROM)

\text{KeyGen}: \(p_k = (n,e), s_k = (n,d)\).

\text{Enc}(m \in \{0,1\}^n): \text{Sample } r \leftarrow U(\mathbb{Z}/n\mathbb{Z}). (c_1, c_2) = (r^e \mod nnH(r) \oplus m) \in \mathbb{Z}/n\mathbb{Z} \times \{0,1\}^n.

\text{Dec}((c_1, c_2)) : c_1^d \mod n \rightarrow \text{recover } r. \text{Output } c_2 \oplus H(r).

Correctness: easy.

Intuition of CPA-security.

Given \(r^e \mod n, r\) is still computationally unknown (because of RSA assumption) \(\rightarrow H(r)\) is uniform, and hence, statistically hides \(m\).

#### Remark 16.

Construction can be extended to any trapdoor one-way function instead of RSA

#### Theorem 18.

If RSA assumption holds and if \(H\) is modelled as a RO, then this encryption scheme is CPA secure.

RSA problem: Given \(N, e, r^e\), find \(r'\) such that \(r'^e = r^e \mod n\).

\text{Exp}_b

1. \(C: n, d, e \leftarrow \text{KeyGen}\)
2. \(C \rightarrow A: n, e\)
3. \(A \rightarrow C: x_i\)
4. \(C \rightarrow A: H(x_i)\)
5. \( A \to C: M_0, M_1 \)
6. \( C \to A: c = (c_1, c_2) \leftarrow Enc(M_b) \)
7. \( C \to A: c \)
8. \( A \to C: x_j \)
9. \( C \to A: H(x_j) \)
10. \( A: b' \in \{0, 1\} \), guess for \( b \).

\( A \) is ppt such that \( Adv = \left| P \left( A \xrightarrow{Exp_0} 1 \right) - P \left( A \xrightarrow{Exp_1} 1 \right) \right| \geq \) non-negl.

We want to use \( A \) to solve RSA problem.

"Query": \( A \) queries \( H \) on \( r \) during its execution (the "\( r \)" of the challenge ciphertext).

\[
Adv = \left| P \left( A \xrightarrow{Exp_0} 1 \land Query \right) + P \left( A \xrightarrow{Exp_0} 1 \land \neg Query \right) - P \left( A \xrightarrow{Exp_1} 1 \land Query \right) - P \left( A \xrightarrow{Exp_1} 1 \land \neg Query \right) \right| \\
\leq \left| P \left( A \xrightarrow{Exp_0} 1 \mid Query \right) - P \left( A \xrightarrow{Exp_1} 1 \mid \neg Query \right) \right| + P \left( \text{Query in Exp}_0 \right) + P \left( \text{Query in Exp}_1 \right)
\]

**Proposition 19.**

\[
P \left( A \xrightarrow{Exp_0} 1 \mid Query \right) - P \left( A \xrightarrow{Exp_1} 1 \mid \neg Query \right) = 0
\]

**Proof.** As \( r \) not queried, \( H(r) \) is uniform to \( A \), and the transcripts of \( Game_0 \) and \( Game_1 \) follow the same distribution. Same event, same distribution \( \Rightarrow \) equal probabilities.

We want to build \( A' \) breaking RSA assumption using an \( A \) such that \( Query \) has non-negl probability.

1. \( C: (n, e, d) \leftarrow \text{KeyRSA} \)
2. \( C: \text{sample } r \)
3. \( C \to A': (n, e, r^e) \)
4. \( A': \text{Prepare } c_1 = r^e. \text{Sample } k \text{ uniformly Sample } b \text{ in } \{0, 1\} \text{ uniformly. Initiate a RO table} \)
5. \( A' \to A: (n, e) \)
6. \( A \to A': \text{RO queries } x_i \)
7. \( A' \to A: (1) \)
8. \( A \to A': m_0, m_1 \)
9. \( A' \to A: (2) \)
10. \( A \to A': b' \)
11. \( A' \to C: r \)

37
$k$ is to be used as $H(r_i)$, $b$ is for Game$_b$.

(1) if $x_i, k_i$ in the table reply $k_i$. If $x_i^r = r^r$, then reply $k$ and store $(x_i, k)$ in the table. Else, sample $k_i$ uniformly, reply $k_i$, and store $(x_i, k_i)$ in the table.

(2) $(c_1, k \oplus m_b)$

(3) If $(x, k)$ is in the table, return $x_i$.

Simulation is valid to $A \Rightarrow A$ dehaves as in attack $\Rightarrow P(\text{Query in Exp}_0) + P(\text{Query in Exp}_1) \geq$ non-negl. When "Query", $A'$ succeeds.
Chapter 8

Digital signatures

Goal: provide authenticity and integrity (no confidentiality).
Concept invented by Diffie & Hellman (1976).
First candidate construction: Rivest, Shamir & Adleman (a couple of year later).
Signatures are used extensively in practice. Digital signature is the asymmetric version of MAC.

8.1 Definition

A digital signature is a triple of ppt algorithms \((\text{KeyGen}, \text{Sign}, \text{Verify})\) such that
- \text{Keygen} takes as input the security parameter \(1^n\). Returns two keys: secret key \(s_k\) and verification key \(v_k\).
- \text{Sign}(s_k, M) returns a signature \(\sigma\).
- \text{Verify}(v_k, M, \sigma) returns 0 (reject) or 1 (accept).

Correctness: \(\forall (s_k, v_k)\) output by \(\text{KeyGen}, \forall M, \text{Verify}(v_k, M, \text{Sign}(s_k, M)) = 1\)

Remark 17.
- \(\sigma \ « \text{provides} » \) that the signer knows \(s_k\).
- Some signature schemes have a verification algorithm that does not take \(M\) as input and recovers the message from the signature. (signature scheme with message recovery, interesting for space efficiency).

Digital signature vs real life signature:
- Signer may not be human, message van be anything (program, public key, signature, ciphertext...)
- Not transferable from one message to another.
• The document cannot be modified once it is signed.

It binds the knowledge of \( s_k \) to \( M \).

**Definition 21 (Security euCMA).**

(existential unforgeability under Chosen Message Attacks)

1. \( C: (s_k, v_k) \leftarrow \text{KeyGen} \)
2. \( C \rightarrow A: v_k \)
3. Signature queries
   - \( A \rightarrow C: M_i \)
   - \( C: \sigma_i = \text{Sign}(s_k, M_i) \)
   - \( C \rightarrow A: \sigma_i \)
4. \( A: \text{Forgery} \)
5. \( A \rightarrow C: (M^*, \sigma^*) \)
6. \( C: \text{If } \text{Verify}(v_k, M^*, \sigma^*) = 1 \text{ and } M^* \notin \{M_i\}_i \text{ return 1 else return 0.} \)

**Remark 18.**

Strong euCMA (seuCMA). Same but \((M^*, \sigma^*) \notin \{(M_i, \sigma_i)\}_i\).

The scheme is euCMA secure is no ppt \( A \) is such that \( C \) returns 1 with non negligible probability.

**Remark 19.**

It’s easy to construct MAC. It seems much more difficult to construct digital signatures. We will see proofs of security only in the ROM.

### 8.2 Public key infrastructure (PKI)

In **DIFFIE-HELLMAN**, the main problem is man-in-the-middle.

It’s due to the lack of authentication in **DIFFIE-HELLMAN**.

Exact same problem with hybrid encryption.

1. \( A: (s_{kA}, v_{kA}) \)
2. \( B: (s_{kB}, v_{kB}) \)
3. \( A: \sigma_A = \text{Sign}(s_{kA}, g^a) \)
4. \( B: \sigma_B = \text{Sign}(s_{kB}, g^a) \)
5. $A \rightarrow B: g^a, \sigma_A$
6. $B \rightarrow A: g^b, \sigma_B$
7. $A$: If $\text{Verify}(v_{kB}, g^b, \sigma_B) = 0$, ten abort.
8. $B$: If $\text{Verify}(v_{kA}, g^a, \sigma_A) = 0$, ten abort.

MIM attack seems to be prevented... But how does know that $v_{kA}$ is Alice’s? Main issue in public key crypto: establish authenticated communication.

Solution 1 Rely on a certification authority (CA). CA is trusted, and signs verification keys. $v_{kA}$ comes with a signature from CA $\sigma_A = \text{Sign}(s_{kCA}, v_{kA})$. Bob checks validity of $v_{kA}$ by checking $\sigma_A$, for this, you need $v_{kCA}$...

Difficulty 1 cf supra
Difficulty 2 How to manage key revocation? CA maintains and publishes lists of revoked keys.

CA’s are used for SSL-TLS to provide authenticity of webpages. Symantec and Docomo are two largest CAs owns. Verisign: cdiscount; geotrust: Google webpages.

Solution 2 PGP and GPG (for emails) → web of trust. A given user has its own key signed by other users.

8.3 RSA signature

$\text{KeyGen}$: $N = pq$, $ed = 1[\varphi(N)]$, $s_k = (N, d)$, $v_k = (N, e)$.

$\text{Sign}$: $M \in \mathbb{Z}/N\mathbb{Z} \mapsto \sigma = M^d[N]
\text{Verify}$: $(M, \sigma) \mapsto [M = \sigma^e[N]]$

Signatures are not the dual of public-key encryption.

It is not euCMA secure.

$\sigma_1 = \text{Sign}(M_1)$ et $\sigma_2 = \text{Sign}(M_2)$, then $\text{Verify}(M_1 M_2, \sigma_1, \sigma_2) = 1$.

One could think of making this a sign with message recovery... but then any $\sigma$ is a valid signature for $M = \sigma^e[N]$.

One way to break multiplicativity of RSA: padding.

$m \ll \log_2 N$

$m \rightarrow (\underbrace{\cdots || m}^d \underbrace{[N]}_{<\log_2 N \text{ bits}})$. Where pad is constant or a function of $m$.

$\text{Verify}(m, \sigma)$ compute $m’ = \sigma^e[N]$ and check whether LHS is a valid pad for RHS and that RHS = $m$. Hopefully, breaks multiplicativity.

$\rightsquigarrow$ RSA-PSS, standardize in pkcs #1. There exists a variant of RSA-pad enjoying a proof in the ROM (under the RSA hardness assumption).

Another way to break multiplicity (and to sign longer messages) is RSA-FDH (full domain hash).

$\text{Sign}(m) : \sigma = (H(m))^d[N].$

$\text{Verify}(m, \sigma) : H(m) \equiv \sigma^e[N]$ Hash function from $\{0, 1\}^*$ to $\mathbb{Z}/N\mathbb{Z}$.  

*https://en.wikipedia.org/wiki/Iverson_bracket
8.3.1 Security of RSA-FDH in the ROM

(assuming the RSA problem is hard)

RSA problem for $e$:

\[ \left[ \begin{array}{c} N \\ \hline \end{array} \right], y \leftarrow \text{Uniform}(N) \mapsto x \text{ such that } x^e = y[N] \] (with non negligible problem).

GOAL: break the RSA problem using an algorithm $A$ (allowed to make sign and hash queries) such that can produce a forgery for RSA-FDH. (H is modeled as a random oracle).

We want to embed the RSA problem instance in the forgery. Hope $H(m^*) = y$ and $\sigma^*$ = solution.

If no $H$ query, $P(H(m^*) = y) = \frac{1}{N}$

probability of success is $\frac{\epsilon}{N}$ $\rightarrow$ success probability of $A$.

If one query to $H$?

- If $H$ query not an $m^*$, same as above
- If $H$ query is on $m^*$. The challenger replies “$H(m^*) = y$”. This looks like a valid RO-reply. We can use $\sigma^*$ as solution to RSA problem.

If more than 1 query, we bet which query correspond to the forgery and we reply ”$y$” to that one.

This bet is correct with probability $\frac{1}{q}$. $\rightarrow$ success probability in solving RSA = \[
\begin{cases}
\frac{\epsilon}{q} & \text{non negligible} \\
\text{polynomial} & \text{non-negl.}
\end{cases}
\]

Problems

1. How do we know $q$?
2. There is no polynomial $q$ that works for all possible $pp$ $A$.

here we assume $q$ is polynomial and known.

Proof of security. $\bullet$ $C$: $(N, e, d) \leftarrow \text{Keygen}$

- $C$: $y \leftarrow \text{Uniform}$
- $C \rightarrow A'$: $(N, e, y)$
- $A' \rightarrow A$: $(N, e)$
- $A'$: $k^* := y$
- $A'$: $i^* \leftarrow \text{Uniform}([1, \ldots, q])$
- $A'$: initiate a RO table $T (m, H(m), \text{Sign}(m))$.
- $A \rightarrow A'$: ”$H$", $m_i$
- $A'$: If $(m_i, k_i, \sigma_i) \in T$, $A' \rightarrow A$: $(k_i)$. Else if $(i = i^*) \in T$, $A' \rightarrow A$: $k^*$. Else sample $\sigma_i$, set $k_i = e_i^*[N]$, put $(m_i, k_i, \sigma_i)$ in $T$ and $A' \rightarrow A$: $k_i$
- $A \rightarrow A'$: ”$\text{Sign}$", $m_i$.
- $A'$: If $(m_i, k_i, \sigma_i) \in T$, $A' \rightarrow A$: $\sigma_i$. Else is $i = i^*$, Abort. Else sample $\sigma_i$, set $k_i = e_i^*[N]$, put $(m_i, k_i, \sigma_i)$ in $T$ and $A' \rightarrow A$: $\sigma$
• $A \rightarrow A': (m^*, \sigma^*)$
• $A' \rightarrow C: \sigma^*$

$(m, k, \sigma) \in T, k = H(m)$ and $\sigma^c = H(m)[N]$.

Fact 1 we can assume that $A$ makes a $H$-query on $m^*$ (success probability of $A$ would be $\exp$ small). If $A$ makes a $H$-query on $m^*$, then $A'$ correctly guesses which one is this $H$-query, with probability $\frac{1}{q}$.

In that situation, $A$'s view is the same as in the real attack.

$A'$ replies to $H$-queries are uniform if fresh and consistent otherwise.

$A'$ replies to signature queries are correct and consistent to the replies to the $H$-queries. $\rightarrow A$ produces a valid forgery $(m^*, \sigma^*)$ with probability $\epsilon$. In this case: $(\sigma^*)^c = k^* = y[N] \Rightarrow A'$ solves RSA instance with probability $\frac{1}{q}$.

8.4 Signatures schemes based on the discrete logarithm problem

8.4.1 SCHRÖR's signature

https://en.wikipedia.org/wiki/Claus_P._Schnorr
https://fr.wikipedia.org/wiki/Protocole_d%27authentification_de_Schnorr

Keygen($1^n$)

1. Choose a cyclic group $G$ of prime order $q > 2^n$ with a generator $g \leftarrow G$.
2. Choose $x \leftarrow \mathbb{Z}_q$ and compute $Y = g^x \in G$.
3. Choose a hash function $H : \{0, 1\}^* \rightarrow \mathbb{Z}_q$. Set $PK := (G, g, Y = g^x, H)$.

Sign($SK, M$): To sign $M \in \{0, 1\}^*$.

1. Pick $r \leftarrow \mathbb{Z}_q$ and compute $A := g^r \in G$.
2. Compute $c = H(M, A) \in \mathbb{Z}_q$.
3. Compute $z = r + cx \mod q$ and output $\sigma = (c, z) \in \mathbb{Z}_q^2$.

Verify($PK, M, \sigma$): given $\sigma = (c, z) \in \mathbb{Z}_q^2$, compute $A' = g^{2Y^{-c}} \in G$ and output 1 iff $c = H(M, A')$.

Correctness: $A' = g^{zY^{-c}} = g^{z+cxY^{-c}} = g^{x+\sigma^c(g^x)} = g^c = A$.

Theorem 20 (PAINTERHAL-STERN 1996).

SCHRÖR's signature is EUF-CMA secure in the random oracle model if the discrete logarithm problem is hard in $G$. 43
A \equiv g^r can be precomputed in an off-line phase (before knowing M) to have a very short response time at step 3.

- Used in challenge response authentication by smart cards since step 3 of the signature algorithm is cheap.
- A different A = g^r should be used in each signature.
- It’s a different paradigm than RDA-FDH ("challenge-response" instead of "hash-and-sign")
- Works in any discrete log hard group G.

Remark 20.

8.4.2 The digital signature algorithm (DSA) (aka. DSS)

Developed by KRANTZ (NSA) in 1993.

Keygen(1^n):
1. Choose a large-prime p such that p > 2^{l(n)} for some function l : \mathbb{N} \rightarrow \mathbb{N} and there exists a prime q > 2^n for which p - 1 = qz for some z \in \mathbb{N}.
2. Choose h \in Z_p^* and compute g = h^{2^k} \mod p which has order q.
3. Pick x \leftarrow Z_q and compute Y = g^x \mod p.
4. Choose a hash function H : \{0, 1\}^* \rightarrow Z_q. Set PK := (p, q, g, Y = g^x \mod l, H).

Sign(SK, M):
1. Pick k \leftarrow Z_q.
2. Compute r = (g^k \mod p) \mod q.
3. Compute s = (H(M) + rx) / k \mod q. Output \sigma = (r, s) \in Z_q^2.

Verify(PK, M, \sigma): Given \sigma = (r, s), compute M_1 = H(M) / s \mod q, M_2 = r / s \mod q. Return 1 iff 
\[ r = (g^{M_1} Y^{M_2}) \mod q. \]

Correctness
\[ g^{M_1 Y^{M_2}} \equiv (g^{H(M)Y^r})^{\frac{1}{k}} \]
\[ s^{M_1 Y^{M_2}} \mod p = (g^{H(M)Y^r})^{\frac{k}{H(M)Y^{r} \mod q}} \mod p \]
\[ = g^k \mod p \]

Remark 21.

- k should be chosen uniformly in Z_q at each signature and never be re-used.
- In 2010, Sony \textsuperscript{TM} used (EC)DSA to sign PlayStation 3\textsuperscript{©} software but recycled k in several signatures, leaking SK.
- Modified versions of DSA have a security proof in the random oracle model.
- As in SCHORR, signatures have 320-bit signatures for recommended parameters.
8.4.3 The KATZ-WONG signature (2003)

KeyGen(1^n):

1. Choose a cyclic group $G$ of prime order $q > 2^n$ with generators $g, h \in G$.
2. Pick $x \leftarrow \mathbb{Z}_q$ and compute $X = g^x$ and $Y = h^x$.
3. Choose a hash function $H : \{0,1\}^* \rightarrow \mathbb{Z}_q$. Set $PK := (g, h, X = g^x, Y = h^x, H)$.

Sign(SK, M) To sign $M \in \{0,1\}^*$.

1. Choose $r \leftarrow \mathbb{Z}_q$ and $A := g^r$.
2. Compute $c = H(PK, A, B, M) \in \mathbb{Z}_q$.
3. Compute $z = r + cx \mod q$ and output $\sigma = (c, z) \in \mathbb{Z}_q^2$.

Verify(PK, M, \sigma): compute $A' = g^2 X^{-c}$ and $B' = h^2 Y^{-c}$ and return 1 iff $C = H(PK, A', B', M)$.

**Remark 22.**

- Similarity with SCHNORR (same signature length), but much simpler security proof.
- Each signature is a proof that $\log_g(X) = \log_h(Y)$ (also a proof of knowledge of $x$, but the security proof does not use this).

**Theorem 21 (KATZ-WONG, 2003).**

The scheme is EUF-CMA secure in the random oracle model under the Decision DIFFIE-HELLMAN assumption.

**Proof.** Suppose a PPT adversary $A$ breaks the EUF CMA-security with advantage $\epsilon$. We build a DDH distinguisher $B$ with advantage $\epsilon' \approx \epsilon$. $B$ takes as input a DDH instance $(g, h, X = g^x, Y) \in G^4$ and uses $A$ to decide if $Y = h^x$ or $Y \in_R G$. $B$ runs $A$ on input of $PK = (G, g, h, X, Y)$, $B$ initializes on empty list $L_H = \emptyset$.

$B$ interacts with $A$ as follows.

Hash queries: when $A$ queries the has value $H(PK, A, B, M)$, $B$ checks if $L_H$ contains an entry $(PK, A, B, M, C)$. If it does, $B$ returns $C$. Otherwise, $B$ outputs a random $c \in_R \mathbb{Z}_q$ and stores $(PK, A, B, M, C)$ in $L_H$.

Signing queries: when $A$ queries a signature on $M$, $B$ chooses, $c, z \leftarrow \mathbb{Z}_q$ computes $A' = g^2 X^{-c}$, $B' = h^2 Y^{-c}$. If $H(PK, A', B', M)$ is already defined.

If it does, $B$ fails and abort. Otherwise, $B$ defines, $H(PK, A', B', M) = c$, stores $(PK, A', B', M, C)$ in $L_H$ and returns $\sigma = (c, z) \in \mathbb{Z}_q^2$.

Output: when $A$ outputs $(M^*, \sigma^* = (c^*, z^*))$, $B$ computes $A^* = g^{c^*} X^{-c^*}$, $B^* = h^{c^*} Y^{-c^*}$ and outputs 1 iff $\sigma^*$ correctly verifies ie $B$ checks if $H(PK, A^*, B^*, M^*) = c^*$, $B$ outputs 1 (meaning $Y = h^x$), otherwise, $B$ outputs 0.
Let abort be the event that \( B \) fails in a signing query. We know that, if \( Y = h^x \) and \( B \) does not fail, \( A \)'s new is the same as in the real attack.

\[
\Pr \left( B = 1 \mid \text{abort} \land Y = h^x \right) = \epsilon
\]

Let us assume \( Y \in \mathbb{R}_G \), we claim that \( \Pr \left( B = 1 \mid \text{abort} \land Y \in \mathbb{R}_G \right) < \frac{q_H}{2^n} \) if \( q_H \) is the number of hash queries.

Indeed, for each \((A, B) \in G\), there is a unique \((c, z) \in \mathbb{Z}_q^2\) such that

\[
\begin{align*}
A &= g^{2X^{-c}} \\
B &= h^2Y^{-c}
\end{align*}
\]

since

\[
\log_g(A) = z - c \log_g(X) \\
\log_g(A) = z - c \log_g(X)
\]

are independent linear equations. For each hash query \( H(PK, A^*, B^*, M) \), the reduction \( B \) returns the corresponding unique \( e^* \) determined \((A^*, B^*)\) with probability \( \frac{1}{q} \Rightarrow \Pr \left( \sigma^* \text{ is valid} \mid \text{abort} \land Y \in \mathbb{R}_G \right) \leq \frac{q_H}{2^n} < \frac{q_H}{2^n} \).

Since \( B \) aborts with probability \( \leq \frac{q_s(q_s + q_H)}{2^n} \) if \( q_s \) is the number of signing queries. We have \( \Pr \left( \text{abort} \right) \geq 1 - \frac{q_s(q_s + q_H)}{2^n} \).

\[
\Rightarrow \quad \text{Adv}_{\text{DDH}}^D(n) := \left| \Pr \left( B = 1 \mid Y = h^x \right) - \Pr \left( B = 1 \mid Y \in \mathbb{R}_G \right) \right| \\
\geq \epsilon - \frac{q_h}{2^n} - \frac{q_s(q_s + q_H)}{2^n}
\]

**Remark 23.**

- **SCHNORR** has a very different security proof.
- Run \( A \) once and get \((M^*, \sigma^* = (c^*, z^*))\).
- Look at the index of the hash query \( H(M^*, A^*) \).
- Run \( A \) a second time: in the second run, \( H \) is replaced by a different random oracle after the \( j \)th query \( H(M^*, A^*) \).
- Obtain \((M^*, \sigma' = (c', z'))\) such that \( A' = g^{2X^{-c}} = g^{2X^{-c}} \Rightarrow \log_g(X) = \frac{c' - c}{e - e} \).


Chapter 9

About CCA

9.1 Security under Chosen-Ciphertext Attacks

**Definition 22.**

A public key encryption (PKE) scheme is secure against adaptive chosen-ciphertext attacks (IND-CCA 2) if no PPT adversary has non negligible advantage in this game.

- Challenger generates \((p_k, s_k) \leftarrow KeyGen(1^n)\) and gives \(p_k\) to \(A\).
- \(A\) adaptively queries the decryption oracle: \(A\) chooses a ciphertext and get \(M \leftarrow Decrypt(s_k, C)\) from the challenger (\(M\) may be \(\bot\) if \(C\) is invalid).
- \(A\) chooses \(M_0, M_1\) such that \(|M_0| = |M_1|\). Then challenger picks \(\beta \leftarrow \{0, 1\}\) and computes \(C^* \leftarrow Encrypt(p_k, M_\beta)\) which is sent to \(A\).
- \(A\) makes further decryption queries an arbitrary ciphertexts \(C \neq C^*\).
- \(A\) output \(\beta' \in \{0, 1\}\) and wins if \(\beta' = \beta\). We define

\[
\text{Adv}^{\text{CCA}}_A(n) := \left| \Pr(\beta = \beta') - \frac{1}{2} \right|
\]

**Remark 24.**

- Non adaptive CCA-security (IND-CCA1) is defined in a similar way without stage 4 (ie. no decryption query beyond the challenge phase).
- ElGamal is not IND-CCA2 secure.
- It is open whether ElGamal is IND-CCA1 under a reasonable assumption.
- CCA2 security matters: In 1998, Bleichenbacher gave a practical attack against an old version of SSL, exploiting a CCA attack against RSA-PKCS#1 (v1)
9.2 CCA2 secure encryption from RSA (in the ROM)

RSA assumption: given \((N = pq, y = x^e \mod N)\) where \(e\) is odd integer such that \(\gcd(e, \varphi(N)) = 1\) and \(x \leftarrow \mathbb{Z}_N^*\), no PPT algorithm can find \(x\).

KeyGen\((1^n)\)

1. Choose large primes \(p, q > 2^{l(n)}\) for some function \(l : \mathbb{N} \to \mathbb{N}\) and set \(N = pq\). Choose \(e\) such that \(\gcd(l, \varphi(N)) = 1\).
2. Choose a secret key encryption scheme \(M = Gen', Enc', Dec'\) for messages of length \(t(n)\) and key length \(n\).
3. Chooses a hash function \(H : \mathbb{Z}_N^* \to \{0, 1\}^n\).

Set \(K = (N = pq, e, M, H)\) and \(s_k := d\) such that \(ed \equiv 1 \mod \varphi(N)\).

Encrypt\((p_k, M), M \in \{0, 1\}^{t(n)}\)

1. Choose \(r \leftarrow \mathbb{Z}_N^*\) and compute \(c_1 = r^e \mod N\).
2. Compute \(K = H(r) \in \{0, 1\}^n\) and \(c_2 = Enc_K(M)\). Output \(C = (c_1, c_2)\).

Decrypt\((s_k, C)\):

1. Compute \(r = c_1^d \mod N\).
2. Compute \(K = H(r)\) and output \(M \leftarrow Dec_K(c_2)\).

**Theorem 22.**

In the random oracle model, the scheme is IND-CCA2 secure assuming that

- \(M\) is a chosen-ciphertext secure secret-key encryption scheme.
- The RSA assumption holds.

9.3 CCA2 secure encryption under DDH assumption (in the standard model)

Obstacles to achieve CCA2 security using ElGamal-like constructions.

1. The adversary can modify the components \(C^*\) and obtain encryptions of messages related to \(M_\beta\) (malleability).
2. The reduction does not know \(s_k = \log_g(X)\) in the security proof (hard to answer decryption queries).

Attempt 1: modify ElGamal so that the reduction knows \(s_k\) os the security proof.

KeyGen\((1^n)\)

1. Choose \(G\) of prime order \(q > 2^n\) with generators \(g, h \leftarrow G\).
2. Choose \(x, y \leftarrow \mathbb{Z}_q\) and compute \(X = g^x h^y\).

We define \(p_k = (G, g, h, X = g^x h^y)\) and \(s_k = (x, y) \in \mathbb{Z}_q^2\).

**Encrypt** \((p_k, M)\)

1. To encrypt \(M \in G\), choose \(r \leftarrow \mathbb{Z}_q\) and compute \(c = (c_0, c_1, c_2) = (MX^r, g^r, h^r)\).

**Decrypt** \((s_k, c)\)

1. Compute \(M = C_0 / (C_1 C_2^*)\).

**Theorem 23.**

The scheme is IND-CPA secure under the DDH assumption.

**Proof.** Let \(A\) be an IND-CPA adversary with advantage. We build a DDH distinguisher \(B\).

\(B\) takes as input a DDH instance \((g, y^a, g^b, T)\) where either \(T = g^{ab}\). \(B\) defines \(h = g^b\) and sets \(x = g^x h^y\) where \(x, y \leftarrow \mathbb{Z}_q^2\).

\(B\) gives \(p_k(g, h = g^b, X = g^x h^y)\) to \(A\) and knows \(s_k = (x, y) \in \mathbb{Z}_q^2\).

\(A\) chooses \(M_0, M_1 \in G\). Then \(B\) picks \(\beta \leftarrow \{0, 1\}\) and computes \(C^* = (C_0^*, C_1^*, C_2^*) = (M_\beta (g^c)^x T^y, g^a, T)\).

If \(A\) outputs \(\beta' \in \{0, 1\}\) such that \(\beta' = \beta\), then \(B\) outputs 1 (meaning \(T = g^{ab}\))

If \(A\) outputs \(\beta' \neq \beta\), \(B\) outputs 0 (meaning that \(T \in \mathbb{R} G\)).

If \(T = g^{ab}\), then \(C^* = (M_\beta X^a, g^a, h^a)\) so that \(C^*\) is a valid encryption of \(M_\beta \Rightarrow P(\beta = \beta' | T = g^{ab}) = P(1 | T = g^{ab}) = \frac{1}{2} + \epsilon\). \(\Box\)

If \(T \in \mathbb{R} G\), we can write \(T = g^{ab+c}\) where \(c \in \mathbb{R} \mathbb{Z}_q\).

\[
\Rightarrow C^* = (C_0^*, C_1^*, C_2^*) = (M_\beta X^a g^{cy}, g^a, h^a g^a) = (M_{rang} X^a, g^a, h^a, g^a)
\]

where \(M_{rang} = M_\beta g^{cy}\) completely hides \(M_\beta\) because \(y \in \mathbb{R} \mathbb{Z}_q\) is independent of \(A\)’s view.

\[
\Rightarrow P(B = 1 | T \in \mathbb{R} G) = P(\beta = \beta' | T \in \mathbb{R} G) = \frac{1}{2}
\]

\[
\Rightarrow \left| P(B = 1 | T = g^{ab}) - P(\beta = \beta' | T \in \mathbb{R} G) \right| \geq \epsilon
\]

**Remark 25.**

- Thus first attempt remains malleable for the same reason as ElGamal.
- Can we prove it IND-CCA1? Open question under DDH.
Problem: \( A \) can infer information about \( y \in \mathbb{Z}_q \) by making a decryption query on malformed ciphertext \( c = (c_0, g', h') \) since the oracle returns \( M = c_0 c_1^{-y} c_2^{-y} = c_0 x^{-y} h^{-y(r'-r)} \) which leaks \( y \in \mathbb{Z}_q \) to an unbounded \( A \).

Attempt 2: Append to \((c_0, c_1, c_2)\) a proof that \( \log_g(c_1) = \log_h(c_2) \).

The lite Cramer-Sharp cryptosystem (1998)

Keygen(1\(^n\))

1. Choose \( G \) of order \( q > 2^n \) with generators \( g, h \leftarrow G \).
2. Choose \( x, y \leftarrow \mathbb{Z}_q \), and \( u, v \leftarrow \mathbb{Z}_q \).

Define \( p_k = (G, g, hX = g^x h^y, Y = g^u h^v) \).
\( s_k = (x, y, u, v) \in \mathbb{Z}_q^4 \).

Encrypt(\( p_k, M \))

1. Choose \( r \leftarrow \mathbb{Z}_q \) and compute \( c = (c_0, c_1, c_2, c_3) = (MX', g', h', Y') \).

Decrypt(\( s_k, c \))

1. Check if \( c_3 = c_1^y c_2^y \). If not return \( \bot \).
2. Returns \( M = C_0 / (C_1^x c_2^y) \).

Correctness If \((c_1, c_2) = (g', h')\), then \( c_1^y c_2^y = (g')^u (h')^v = Y' \).

Remark 26.

Scheme remains malleable and thus vulnerability to CCA2 attacks.

Theorem 24.

The scheme is IND-CCA1 secure under the DDH assumption.

Proof. Similar to attempt 1, provided the decryption oracle rejects all invalid ciphertexts \((c_0, c_1, c_2, c_3)\) \((\log_g(c_1) \neq \log_h(c_2))\)

If \( c = (c_0, c_1, c_2, c_3) = (MX', g', h', c_3) \) is invalid \( r \neq r' \).

The reduction \( B \) rejects \( C \) unless \( c_3 = c_1^y c_2^y \). However \( c_1^u c_2^v = (g')^u (h')^v = (h'^{-r})^v = Y'h^v(r'-r) \).

\( c_1^u c_2^v \) is unpredictable since \( v \in \mathbb{Z}_q \) is independent of \( A \)'s view.

Each unsuccessful decryption query allows \( A \) to eliminate one candidate \( v \in \mathbb{Z}_q \).

\( \Rightarrow \) At the \( i \)-th decryption query, \( A \) can predict \( c_1^u c_2^v \) with probability \( P(F_i) \leq \frac{1}{q-1} \) where \( F_i \) is the event that \( B \) does not reject the \( i \)-th invalid ciphertext.

\( \Rightarrow \) The probability that an invalid ciphertext does not get rejected is at most

\[
P(F_1 \lor \ldots \lor F_{q_d}) \leq \frac{q_d}{q - q_d + 1} \in \text{negl}(n)
\]

where \( q_d \) is the number of decryption queries.
9.4 CCA1 security under DDH

Keygen\((1^n)\)

1. Choose \(G\) of prime order \(q > 2^n\), generators \(g, h \leftarrow G\).
2. Choose \(x, y, u, v \leftarrow \mathbb{Z}_q\) and compute \(X = g^x h^y\) and \(Y = g^u h^v\).

Set \(p_k = (G, g, h, X, Y)\) and \(s_k = (x, y, u, v)\).

Encrypt\((p_kM)\)

1. pick \(r \leftarrow \mathbb{Z}_q\) and compute \(c = (c_0, c_1, c_2, c_3) = (MX^r, g^r, h^r, Y^r)\).

Decrypt\((s_kc)\)

1. Return \(\bot\) is \(c_3 \neq c_1^x c_2^y\)
2. Return \(M = \frac{c_0}{c_1^x c_2^y}\).

Only provides CCA1 security (under DDH) since given \(c^* = (c_1^*, c_2^*, c_3^*, c_4^*)\) adversary can compute another encryption of \(M_\beta\) (or a related message).

Solution for CCA2 security: add an integrity check for \((c_0, c_1, c_2)\) and prevent the adversary from tempering with \(c_0\).

9.5 CCA2 security under DDH

Keygen\((1^n)\)

1. Choose \(G\) of prime order \(q > 2^n\), generators \(g, h \leftarrow G\).
2. Choose \(x, y, u_1, v_1, u_2, v_2 \leftarrow \mathbb{Z}_q\) and compute \(X = g^x h^y, Y_1 = g^{u_1} h^{v_1}\) and \(Y_2 = g^{u_2} h^{v_2}\).
3. Choose a collision resistant hash function \(H : \{0,1\}^* \rightarrow \mathbb{Z}_q\).

Set \(p_k = (G, g, h, X, Y_1, Y_2, H)\) and \(s_k = (x, y, u_1, v_1, u_2, v_2)\).

Encrypt\((p_kM)\)

1. pick \(r \leftarrow \mathbb{Z}_q\) and compute \(c = (c_0, c_1, c_2, c_3) = (MX^r, g^r, h^r, (Y_1^\alpha Y_2^\alpha)^r)\) where \(\alpha = H(c_0, c_1, c_2) \in \mathbb{Z}_q\).

Decrypt\((s_kc)\)

1. Compute \(\alpha = H(c_0, c_1, c_2)\).
2. Check if \(c_3 = c_1^{\alpha + u_2} c_2^{\alpha + v_2}\) and return \(\bot\) otherwise.
3. Output \(M = \frac{c_0}{c_1^x c_2^y}\).
The scheme is IND-CCA2 secure assuming that
1. DDH is hard in $G$,
2. $H$ is a collision-resistant hash function.

**Proof.** Let $A$ be a CCA2 with advantage $\varepsilon$. We build an algorithm $B$ that either breaks the DDH assumption or the collision-resistance of $H$.

$B$ takes as input $(g, g^a, g^b, T)$ where either $T = g^{ab}$ or $T \in_R G$ and a hash function $H_y$.

$B$ defines $h = g^b$ and picks $x, y, u_1, u_2, v_1, v_2 \leftarrow \mathbb{Z}_q$ compute $X = g^x h^y$, $Y_1 = g^{u_1 h^{v_1}}$ and $Y_2 = g^{u_2 h^{v_2}}$ and gives $p_k = (g, h = g^b, X = g^x, h^y, Y_1 = g^{u_1 h^{v_1}}, Y_2 = g^{u_2 h^{v_2}})$ to $A$.

$B$ faithfully answers all decryption queries using $s_k = (x, y, u_1, u_2, v_1, v_2)$. In the challenge phase, $A$ chooses $M_0, M_1 \in G$. Then $B$ picks a bit $\beta \leftarrow \{0, 1\}$ and computes

$$C^* = (c_0^*, c_1^*, c_2^*, c_3^*) = (M_\beta (g^a)^x T^y, g^a, T, (g^a)^{u_1 + u_2 T^{v_1 + v_2}})$$

where $a^* = H(c_0^*, c_1^*, c_3^*)$. If $A$ queries the decryption of $c = (c_0, c_1, c_2, c_3)$ such that $a = H(c_0, c_1, c_2) = H(c_0^*, c_1^*, c_3^*)$, but $(c_0, c_1, c_2) = (c_0^*, c_1^*, c_3^*)$. $B$ halts since it found a collision for $H$.

When $A$ stops and outputs $\beta' \in \{0, 1\}$, $B$ outputs 1 (meaning $T = g^{ab}$) if $\beta' = \beta$, $B$ outputs 0 (meaning $T \in_R G$) if $\beta' \neq \beta$.

If $T = g^{ab}$, then $C^a = (M_\beta X^a, g^a h^a, (Y_1^a Y_2)^a) = (g^a, (g^a)^a, (g^a)^2)$ is a correct encryption of $M_\beta \Rightarrow P(\beta' = \beta' \mid T = g^{ab}) = P(\beta = 1 \mid T = g^{ab}) = \frac{1}{2} + \varepsilon$ where $\varepsilon = Adv(A)$.

What if $T \in_R G$? $C^* = (M_\beta, g^a, h^a, (Y_1^a Y_2)^a g^{c(v_1 a + v_2)})$.

An invalid ciphertext $c = (c_0, c_1, c_2, c_3)$ is not rejected if $c_3 = c_1^{u_1 + u_2} c_2^{v_1 + v_2}$ i.e. if $c_3 = (Y_1^a Y_2)^x g^{c(v_1 a + v_2)}$. However $v_1 a + v_2$ is unpredictable if $a \neq a^*$. \qed
Chapter 10

Commitment schemes

10.1 Definition

Digital equivalent of a safe.

Definition 23.

A commitment scheme is a tuple of algorithms $(\text{Setup}, \text{Com}, \text{Open}, \text{Verify})$

$\text{Setup}(1^n)$: given a security parameter $1^n$ generates a commitment key $c_k$ (and sometimes a trapdoor $t_k$).

$\text{Com}_{c_k}(m)$: randomized algorithm that outputs a commitment $\text{com}$ of $m$ and a decommitment $\text{dec}$.

$\text{Open}_{c_k}(\text{com}, m, \text{dec})$: reveals evidence $\text{dev}$ that $\text{com}$ was a commitment of $m$

$\text{Verify}_{c_k}(m, \text{Com}, \text{Dec})$: outputs 0 or 1

Properties:

Hiding

$\forall m_0, m_1$ with $m_0 \neq m_1$ $D_0 := \{\text{Com}_{c_k}(m_0, r) \mid r \leftarrow \mathcal{R}\}$ and $D_1 := \{\text{Com}_{c_k}(m_1, r) \mid r \leftarrow \mathcal{R}\}$

$D_0$ is computationally indistinguishable from $D_1$.

Biding: given $c_k$, no PPT adversary can output $\text{com}$ and pairs $(m, \text{dec}), (m', \text{dec}')$ with $m \neq m'$ such that

$\text{Verify}_{c_k}(m, \text{Com}, \text{Dec}) = 1 \land \text{Verify}_{c_k}(m', \text{Com}, \text{Dec}') = 1$

10.2 Application

Coin-tossing over the internet. Let $A$ and $B$ be distrustful players who want to jointly generate a random bit $b \in_R \{0, 1\}$.

1. $B$ picks $b_N \leftarrow \{0, 1\}$ and keeps it secret.
2. $A$ picks $b_A \leftarrow \{0, 1\}$ and computes $(\text{Com}, \text{Dec}) \leftarrow \text{Com}_{c_k}(b_A)$. Sends $\text{Com}$ to $B$.
3. Breaks $b_B \in \{0, 1\}$. $A$ reveals $b_A$ and $\text{Dec}$.
4. $B$ checks that $\text{verify}_{c_k}(b_A, \text{Com}, \text{Dec}) = 1$. $A$ and $B$ compute $b = b_A \oplus b_B$. 

53
Bibliography

