CSA E0 235: Cryptography March 16, 2015

(Extra) Lecture 3

Instructor: Arpita Patra *Submitted by: Ajith S*

1 Chosen Plaintext Attack

A chosen-plaintext attack (CPA) is an attack model for cryptanalysis which presumes that the attacker can obtain the ciphertexts for plaintexts of its choice. This is formalized by allowing the adversary to interact with an *encryption oracle*, viewed as a black box. The goal of the attack is to gain information to break the security of the encryption scheme.

This appears, at first glance, to be an unrealistic model; as it is unlikely that an attacker could persuade a honest sender/receiver to encrypt large amounts of plaintexts of the attacker's choosing. However, modern cryptography is implemented in software or hardware and is used for a diverse range of applications; for many cases, a chosen-plaintext attack is often very feasible.

1.1 Security against Chosen Plaintext Attack

In realistic scenarios, an adversary has knowledge of plaintext-ciphertext pairs. A broadly (but not fully) general way to capture this knowledge is to look at a model in which the adversary is able to see *encryptions of arbitrary messages of his choice.*

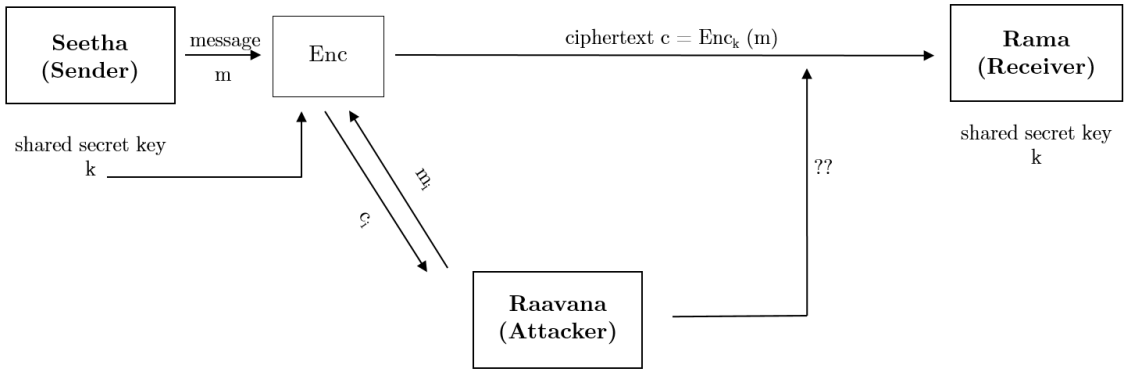

Fig 1 Chosen-Plaintext Attacks (CPA)
(Single message security)

Here *Seetha*(sender) and *Rama*(receiver) had agreed on a pre-shared key, k before the start of the protocol. *Seetha* will encrypt the message m using the key k to get the corresponding ciphertext c. This is send to *Rama* through an unsecure channel. *Raavana*(attacker) can see the ciphertexts going through the channel. The difference of this CPA from that

of Ciphertext Only Attack (COA) is that here *Raavana* is given more power by giving him access to an *Encryption Oracle*. *Raavana* is allowed to send messages, say m_i's to the *Encryption Oracle* at any point in time and the *oracle* will send back the ciphertexts corresponding to messages m_i's, say c_i's. The goal of *Raavana* is to decrypt a new ciphertext that he hasn't queried before and thus to obtain the underlying message.

1.2 Chosen-plaintext attacks in practice

In World War II US Navy cryptoanalysts discovered that Japan was planning to attack a location referred to as "AF". They believed that "AF" might be Midway Island, because other locations in the Hawaiian Islands had codewords that began with "A". To prove their hypothesis that "AF" corresponded to "Midway Island" they asked the US forces at Midway to send a plaintext message about low supplies. The Japanese intercepted the message and immediately reported to their superiors that "AF" was low on water, confirming the Navy's hypothesis and allowing them to position their force to win the battle.

Also during World War II, Allied codebreakers at Bletchley Park would sometimes ask the Royal Air Force to lay mines at a position that didn't have any abbreviations or alternatives in the German naval system's grid reference. The hope was that the Germans, seeing the mines, would use an Enigma machine to encrypt a warning message about the mines and an "all clear" message after they were removed, giving the allies enough information about the message to break the German naval Enigma.

1.3 CPA Indistinguishability Experiment

The following experiment is based on *computational security* assumption and hence the *attacker* is modeled by a probabilistic polynomial time Turing machine, meaning that it must complete the game and output a "guess" within a polynomial number of time steps.

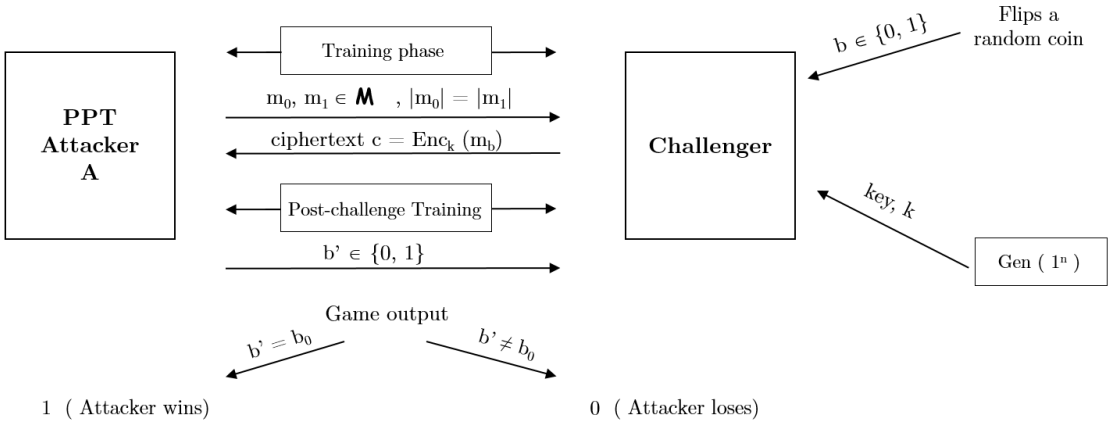

Fig 2 CPA Indistinguishability Experiment $\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\Pi}(n)$

The experiment $\mathsf{PrivK}^{\mathsf{cpa}}_{A,\Pi}(n)$ consists of the following four phases:

- **Training Phase :** *Attacker* A is given *Oracle* access in this phase. A adaptively submits its query messages and receives their encryptions.
- **Challenge Phase :** A submits two equal length challenge plaintexts, m_0 and m_1 to the *Challenger*. Here A is free to submit any message of its choice (including the ones already queried during the training phase). *Challenger* obtains the key, k, for encryption from the *Gen* algorithm. He flips a random coin to obtain the value of bit $b \in \{0, 1\}$, and encrypts the corresponding challenge plaintext, say m_b, and sends the ciphertext to A.
- **Post-Challenge Training Phase :** A can use the *Oracle* access even after the challenge. Here also A adaptively submits its query messages and receives their encryptions.
- **Response Phase :** A finally submits its guess regarding encrypted challenge plaintext, in the form of a bit, b' . A wins the experiment if its guess is correct.

Definition 1 A *private-key encryption scheme* $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ has indistinguishable encryptions under a *chosen-plaintext attack*, or is *CPA-secure*, if for all probabilistic polynomial-time adversaries A there is a negligible function $\mathsf{negl}(n)$ such that

$$\mathsf{Pr}[\mathsf{PrivK}^{\mathsf{cpa}}_{A,\Pi}(n) = 1] \leq \frac{1}{2} + \mathsf{negl}(n)$$

$\diamondsuit$

1.4 Search for Ingredients of CPA-Secure Scheme

The CPA-Secure scheme MUST be *randomized*. If not, one simple attack for *Attacker* is to query the challenge plaintexts in the Post-Challenge Training Phase and can compare the results with the ciphertext returned by the *Challenger* for the challenge.

Thus we are in need of "fresh randomness" for each run of Enc and at the same time we want to use a "single key". One solution is to pre-share a look up table between *Seetha* and *Rama*. The table consists of 2^n entries of the form (x_i, y_i) where $|x_i| = |y_i| = n$. each of the x_i is mapped to a random y_i. For encryption, *Seetha* chooses a random x_i and uses the corresponding y_i as key for encryption.

The encryption is defined as $\mathsf{Enc}(m) : c \leftarrow (x_i, m \oplus y_i)$. Here since the pad y_i is truly random, this scheme is an instance of One Time Pad (OTP) and thus is CPA-Secure. But the problem with the scheme is size of the look-up table, which is $n.2^n$ bits. Thus we need a smarter tool which requires only short key.

2 Encryption Using Pseudo Random Functions

A function is said to be *truly random* (TRF) if its output behavior is completely unpredictable. Given an input, it randomly assigns one element from the co-domain as the output and every element from the co-domain is a possible image with equal probability. Intuitively, a *pseudo random function* (PRF) is a function whose output behavior looks like that of a TRF for an observer who is computationally bounded.

2.1 Pseudo Random Functions (PRF)

Consider the set of all functions from $\{0,1\}^n$ to $\{0,1\}^n$. Let $Func_n = \{f_1, f_2, ..., f_{2^{n.2^n}}\}$ which represents the family of all such functions. Let $\overline{Func_n} = \{F_{k_1}, F_{k_2}, ..., F_{k_{2^n}}\}$ be the family of all keyed functions with key length n. It is easy to see that a function chosen uniformly at random from $Func_n$ is a TRF while that from $\overline{Func_n}$ is a PRF.

One possible definition of PRF can be as follows : Give the PPT distinguisher D, a function F uniformly sampled from either $Func_n$ or $\overline{Func_n}$ and ask him to distinguish. This approach doesn't work since the description of the function is of exponential size. Thus we give D oracle access to either a TRF or a PRF and asks him to tell with whom he is interacting with. If D fails to distinguish, then we say that the function is a PRF.

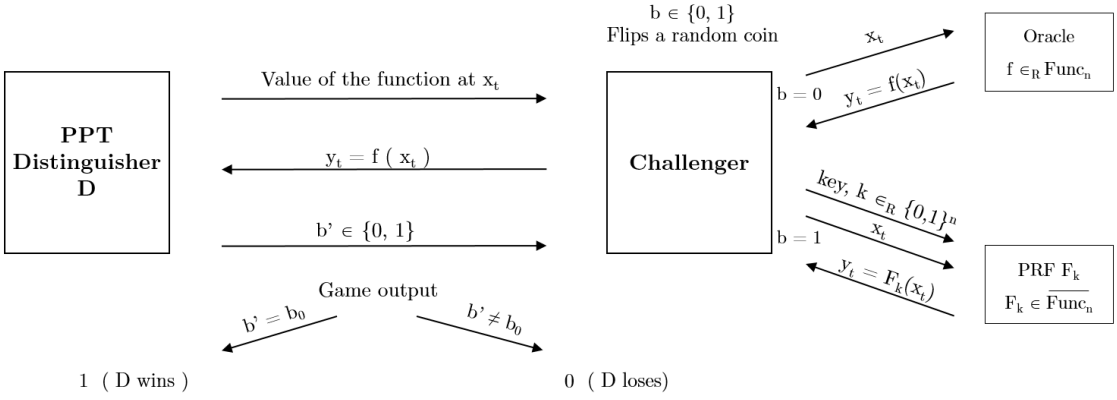

Fig 3 PRF Indistinguishability Experiment

Challenger will flip a random coin and based on its output, he selects either a TRF or a PRF. Now D can adaptively ask its queries in form of x_t and the challenger will return the value of the chosen function at that point, say y_t. The number of queries that D can make is bounded polynomially in n. After satisfied with the queries, D has to predict the family to which the function he interacted belongs to.

Definition 2 A function F is said to be *Pseudo Random Function* (PRF) if for every PPT Distinguisher D, there exists a negligible function negl(n) such that

$$|\mathsf{Pr}[D^{F_k(.)}(1^n) = 1] - \mathsf{Pr}[D^{f(.)}(1^n) = 1]\ | \leq \mathsf{negl}(n)$$

where the first probability is taken over uniform choice of $k \in \{0,1\}^n$ and the randomness of D, and the second probability is taken over uniform choice of $f \in Func_n$ and the randomness of D. ♢

2.2 PRF-based CPA-Secure Scheme

Let F be a pseudorandom function. Define a private-key encryption scheme for messages of length n, $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ as follows:

- Gen: on input 1^n, choose uniform $k \in \{0,1\}^n$ and output it
- Enc: on input a key $k \in \{0,1\}^n$ and a message $m \in \{0,1\}^n$, choose uniform $r \in \{0,1\}^n$ and output the ciphertext

$$c \leftarrow < r, F_k(r) \oplus m >$$

- Dec : on input a key $k \in \{0,1\}^n$ and a ciphertext $c = < c_0, c_1 >$, output the plaintext message

$$m \leftarrow \ F_k(c_0) \oplus c_1$$

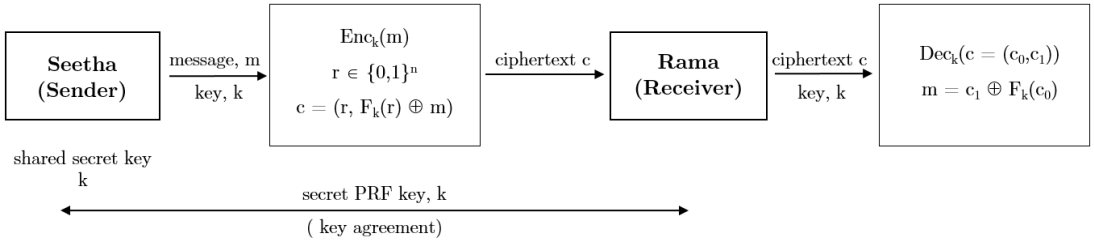

Fig 4 PRF Based CPA Secure Scheme

Theorem 1 *If F_k is a PRF, then Π is a CPA-Secure scheme.*

Proof We prove the theorem using proof by reduction. Assume Π is not a CPA-Secure Scheme. Then there exists an adversary A_{CPA}, who can break Π with some non-negligible probability. That is, there exists some polynomial $p(n)$ such that:

$$\mathsf{Pr}[\mathsf{PrivK}^{\mathsf{cpa}}_{A_{CPA},\Pi}(n) = 1\ ] \ \geq \frac{1}{2} + \frac{1}{p(n)} \tag{1}$$

Using A_{CPA}, we will create another attacker A_{PRF} who can distinguish the F_k from a TRF which leads to a contradiction showing that our assumption was wrong. Ch_{PRF} represents the challenger for the PRF Indistinguishability experiment. Ch_{PRF} flips a random coin and based on the output, it selects a function f which is either a TRF or a PRF. The scheme of A_{PRF} works as follows:

- **Training Phase :** A_{CPA} adaptively submits its query message m to A_{PRF}. A_{PRF} picks a random $r \in \{0,1\}^n$ and forwards it to Ch_{PRF}. Ch_{PRF} returns value of f, say y to A_{PRF}. A_{PRF} calculates ciphertext as $(r, m \oplus y)$ and returns it to A_{CPA}.
- **Challenge Phase :** A_{CPA} submits two equal length challenge plaintexts, m_0 and m_1 to the A_{PRF}. Here A_{CPA} is free to submit any message of its choice (including the ones already queried during the training phase). A_{PRF} picks a random $r^* \in \{0,1\}^n$ and forwards it to Ch_{PRF} and obtains y^* in return. Now A_{PRF} picks a random $b \in \{0,1\}$ and based on value of b, he calculates ciphertext as $(r^*, m_b \oplus y^*)$ and returns it to A_{CPA}.
- **Post-Challenge Training Phase :** A_{CPA} adaptively submits its query messages and receives their encryptions in a way similar to that in the Training phase.
- **Response Phase :** A_{CPA} finally submits its guess regarding encrypted challenge plain-text, in the form of a bit, b" . If b" $= b$, then A_{PRF} outputs its guess $b' = 1$ to Ch_{PRF}, else outputs $b' = 0$.

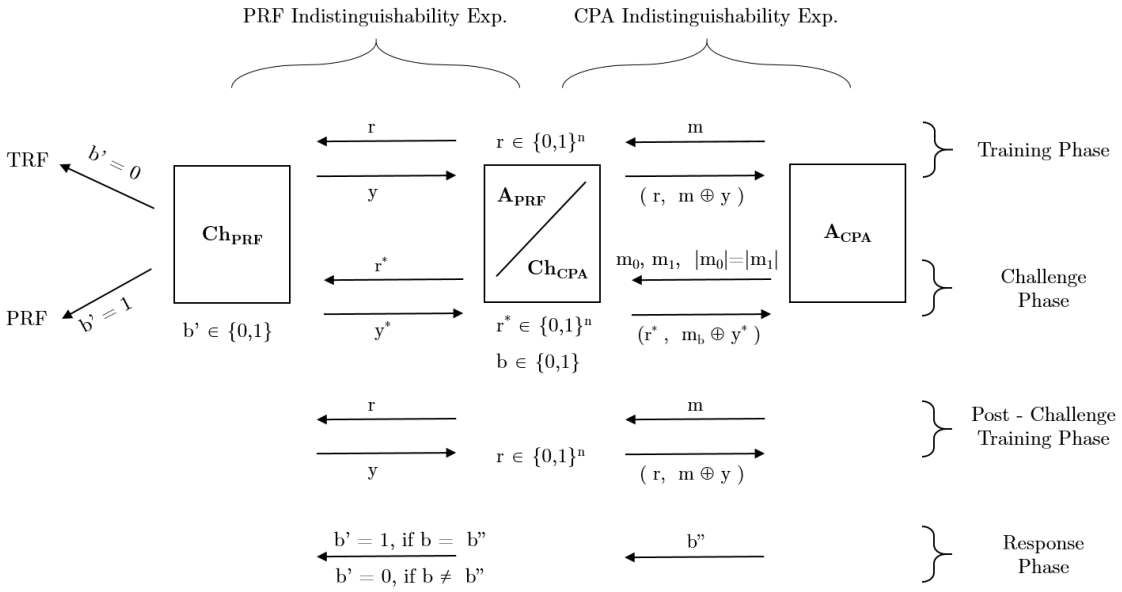

Fig 5 Attacking scheme of A_{PRF} using A_{CPA}

Let $\widetilde{\Pi}$ denotes the game played, when the function chosen by Ch_{PRF} is a TRF. Let `Repeat` denotes the event where the random number chosen for challenge phase, r^*, is used somewhere in either of the Training phase or in the Post - Challenge Training Phase. Let $q(n)$ denote the total number of queries made by A_{CPA}. There are two possibilities:

- **Case 1:** $r^* = r$ for some query in the training phase. In this case A_{CPA} obtains $< r, m \oplus f(r) >$ and $< r^*, m_b \oplus f(r^*) >$ as ciphertexts to m and m_b respectively. Now

$m \oplus f(r) \oplus m_b \oplus f(r^*)$ will give A_{CPA} the value $m \oplus m_b$. Since A_{CPA} has got both m and m_b with him, he can distinguish which among m_0 and m_1 is m_b and can win the game with a probability of 1.

Since A_{CPA} makes at most $q(n)$ queries and since r^* is chosen uniformly from $\{0,1\}^n$, the probability of event `Repeat` is at most $q(n)/2^n$. That is

$$\mathsf{Pr}[\ \texttt{Repeat}\ ] \ \leq \frac{q(n)}{2^n} \tag{2}$$

- **Case 2:** $r^* \neq r$ for any query in the training phase. Here A_{CPA} can win only by guessing b, which happens with a probability of $\frac{1}{2}$.

Now, consider the probability that A_{CPA} wins in the game $\widetilde{\Pi}$

$$\begin{aligned}
\mathsf{Pr}[\ \mathsf{PrivK}^{\mathsf{cpa}}_{A_{CPA},\widetilde{\Pi}}(n) = 1\ ] &= \mathsf{Pr}[\ \mathsf{PrivK}^{\mathsf{cpa}}_{A_{CPA},\widetilde{\Pi}}(n) = 1 \ \wedge \texttt{Repeat}\ ] + \\
&\quad \mathsf{Pr}[\ \mathsf{PrivK}^{\mathsf{cpa}}_{A_{CPA},\widetilde{\Pi}}(n) = 1 \ \wedge \overline{\texttt{Repeat}}\ ] \\
&= \mathsf{Pr}[\ \mathsf{PrivK}^{\mathsf{cpa}}_{A_{CPA},\widetilde{\Pi}}(n) = 1 \ /\ \texttt{Repeat}\ ].\mathsf{Pr}[\ \mathit{Repeat}\ ] + \\
&\quad \mathsf{Pr}[\ \mathsf{PrivK}^{\mathsf{cpa}}_{A_{CPA},\widetilde{\Pi}}(n) = 1 \ /\ \overline{\texttt{Repeat}}\ ].\mathsf{Pr}[\ \overline{\texttt{Repeat}}\ ] \\
&\leq \mathsf{Pr}[\ \texttt{Repeat}\ ] + \mathsf{Pr}[\ \mathsf{PrivK}^{\mathsf{cpa}}_{A_{CPA},\widetilde{\Pi}}(n) = 1\ /\overline{\texttt{Repeat}}\ ] \\
&\leq \tfrac{q(n)}{2^n} + \tfrac{1}{2}
\end{aligned}$$

Using the above, consider the probability that A_{PRF} can win in the PRF Indistinguishability Experiment.

$$\begin{aligned}
|\mathsf{Pr}[\ A^{F_k(\cdot)}_{PRF}(1^n) = 1\ ] &- \mathsf{Pr}[\ A^{f(\cdot)}_{PRF}(1^n) = 1\ ]| \\
&= |\mathsf{Pr}[\ \mathsf{PrivK}^{\mathsf{cpa}}_{A_{CPA},\Pi}(n) = 1\ ] - \mathsf{Pr}[\ \mathsf{PrivK}^{\mathsf{cpa}}_{A_{CPA},\widetilde{\Pi}}(n) = 1\ ]| \\
&= (\tfrac{1}{2} + \tfrac{1}{p(n)}) - (\tfrac{q(n)}{2^n} + \tfrac{1}{2}) \\
&> \tfrac{1}{p(n)} - \tfrac{q(n)}{2^n} \\
&> \tfrac{1}{p'(n)}
\end{aligned}$$

for some polynomial $p'()$ in n.

Thus we proved that with the help of A_{CPA}, A_{PRF} can win in the PRF Indistinguishabilty Experiment with a non-negligible probability. This contradicts the assumption that F_k is a PRF. Thus proved. ■

2.3 Pseudo Random Permutation (PRP)

The definition of a *pseudo random permutation* (PRP) is exactly analogous to that of a PRF, with the only difference being that now we require F_k to be indistinguishable from a uniform *permutation* rather than a uniform function. Similar to $Func_n$ and $\overline{Func_n}$ we defined for PRF, let $Perm_n = \{f_1, f_2, ..., f_{(2^n)!}\}$ represents the family of all uniform permutations and $\overline{Perm_n} = \{F_{k_1}, F_{k_2}, ..., F_{k_{2^n}}\}$ represents the family of all keyed functions with key length n. The Indistinguishability experiment and the security definitions remains the same, except with the change that $Func_n$ and $\overline{Func_n}$ are now replaced with $Perm_n$ and $\overline{Perm_n}$.

If F_k satisfies the stronger requirement that for any PPT Distinguisher, F_k is indistinguishable from a uniform permutation *even if the distinguisher is additionally given oracle access to the inverse of the permutation*, say F_k^{-1}, then it is called a *strong pseudo random permutation*.

Definition 3 A function F is said to be *Strong Pseudo Random Permutation* (SPRP) if for every PPT Distinguisher D, there exists a negligible function negl(n) such that

$$|\mathsf{Pr}[D^{F_k(.),F_k^{-1}(.)}(1^n) = 1] - \mathsf{Pr}[D^{f(.),f^{-1}(.)}(1^n) = 1]\ | \leq \mathsf{negl}(n)$$

where the first probability is taken over uniform choice of $k \in \{0,1\}^n$ and the randomness of D, and the second probability is taken over uniform choice of $f \in Perm_n$ and the randomness of D. ♢

Theoretically, a CPA-Secure SKE scheme can be instantiated using any of the PRF or PRP or SPRP primitives. But for practical purposes, we use only PRP and SPRP for constructing CPA-Secure SKE. Some of the examples include AES and DES. These schemes operates on block of a message at a time and hence these are known as Block ciphers.

3 Multiple Encryptions and Theoretical Constructions

In this section, we will see CPA secure scheme for multiple encryptions and some theoretical constructions for messages of arbitrary lengths.

3.1 CPA-Security for multiple encryptions

This scheme is similar to CPA-Secure scheme for single encryptions. Here the difference is that instead of a single message, *AttackerA* can submit message vector of form $M_i = (m_1, ..., m_t)$ and the *Challenger* returns corresponding ciphertext vector $C_i = (c_1, ..., c_t)$ where $c_i = \mathsf{Enc}_k(m_i)$. After the training phase, A submits two message vectors of equal length. *Challenger* encrypts one of them and returns the corresponding ciphertext vector. A wins if he can distinguish the message vector corresponding to the ciphertext vector.

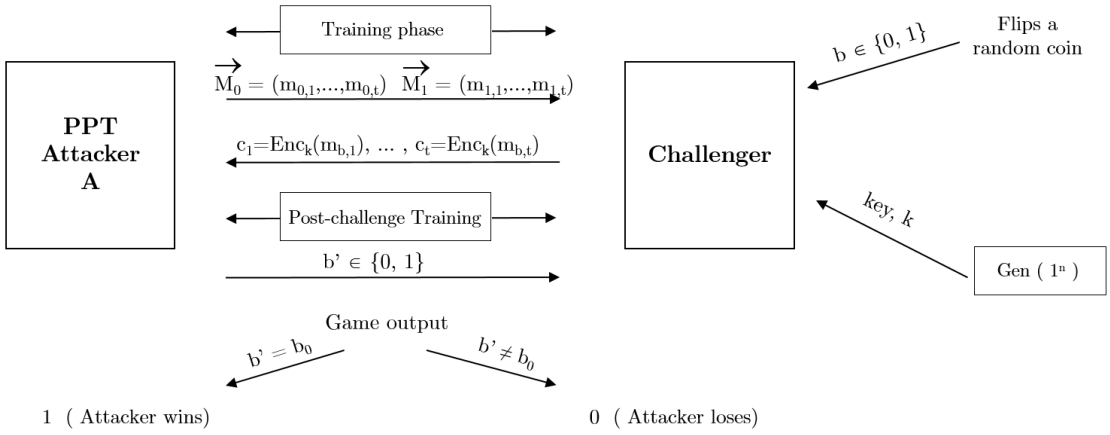

Fig 6 CPA Indistinguishability Experiment for multiple encryptions $\mathsf{PrivK}^{\mathsf{cpa\text{-}mult}}_{A,\Pi}(n)$

Definition 4 A *private-key encryption scheme* $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ has indistinguishable encryptions under a *chosen-plaintext attack* for multiple encryptions, or is *CPA-secure* for multiple encryptions, if for all probabilistic polynomial-time adversaries A there is a negligible function $\mathsf{negl}(n)$ such that

$$\mathsf{Pr}[\mathsf{PrivK}^{\mathsf{cpa\text{-}mult}}_{A,\Pi}(n) = 1] \leq \frac{1}{2} + \mathsf{negl}(n)$$

♢

3.2 CPA Multiple-message vs Single-message Security

It is easy to see that the Experiment $\mathsf{PrivK}^{\mathsf{cpa}}_{A,\Pi}(n)$ is a special case of $\mathsf{PrivK}^{\mathsf{cpa\text{-}mult}}_{A,\Pi}(n)$ and is obtained by setting $\mid \overrightarrow{M_0} \mid = \mid \overrightarrow{M_1} \mid$.

Theorem 2 *Any cipher that is CPA-secure is also CPA-secure for multiple encryptions.*

3.3 CPA Security for Arbitrary-length Messages

Let $\Pi = (\text{Gen, Enc, Dec})$ be a fixed-length CPA-secure encryption based on PRF/PRP/SPRP, that encrypts messages of size, say n . Now consider a message m of length $t.n$. In the theoretical construction below, we will first divide m into t n-length blocks and then encrypts each block using Π and key k generated by Gen algorithm. The ciphertext is obtained by appending the ciphertexts obtained from each block in order.

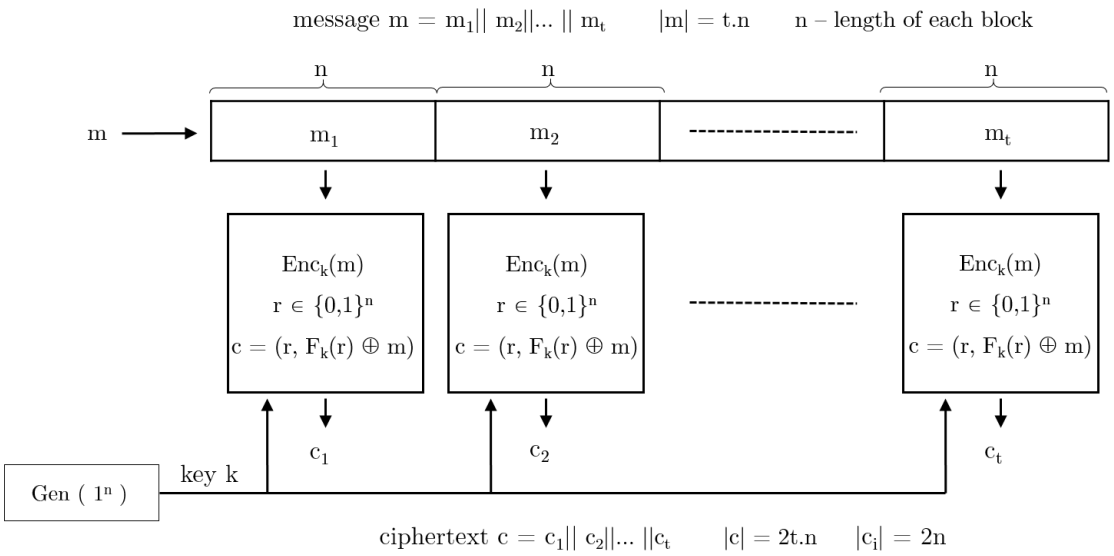

Fig 7 CPA Security for Arbitrary length messages (Theoretical Construction)

3.4 Block-cipher Modes of Operations

We are given a length-preserving block cipher F (may be a PRF/PRP/SPRP) with block length equals n. It takes as input a key $k \in \{0,1\}^n$ and a value $x \in \{0,1\}^n$ and outputs $F_k(x) = F(k,x) \in \{0,1\}^n$. Our goal is to encrypt messages of the form $m = m_1||m_2||...||m_t$, where $|m_i| = n$ for $i = 1,..,t$ using F with ciphertext length as small as possible and with minimum randomness. The following modes of operations are explained in this section:

- Electronic Code Book (ECB) Mode
- Cipher Block Chaining (CBC) Mode
- Output Feedback (OFB) Mode
- Counter (CTR) Mode

3.4.1 Electronic Code Book (ECB) Mode

Given message m of the form $m_1||m_2||...||m_t$ and a SPRP F_k, ECB encodes m as follows:

$$c_i = F_k(m_i)$$

where c_i is the i^{th} block of the ciphertext.

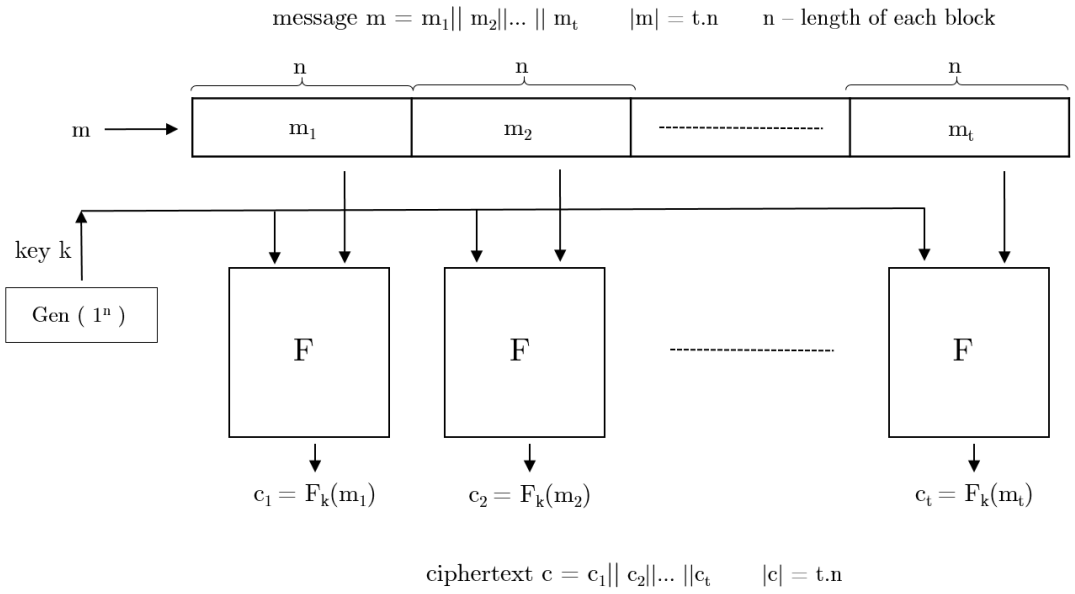

Fig 8 Electronic Code Book (ECB) Mode

Since the scheme is *deterministic*, this is not CPA-secure. One simple attack makes use of the fact that encrypting the same plaintext block always results in the same ciphertext.Consider the following attack. Let $m_0 = aa$ and $m_1 = ab$, where $a, b \in \{0,1\}^n$. Let $c_a = F_k(a)$ and $c_b = F_k(b)$. Now encryptions of m_0 and m_1 will be $c_a c_a$ and $c_a c_b$ respectively. Thus by looking to the two blocks in the ciphertext returned by the *Challenger*, the *Attacker* can guess the bit b with probability equals 1.To perform the decryption, we need F_k^{-1} to be efficiently computable. Thus we need F_k to be a SPRP.

3.4.2 Cipher Block Chaining (CBC) Mode

Now we will consider a scheme that is non-deterministic. First, we choose an *Initialization Vector*, IV, of block length, n, uniformly at random from $\{0,1\}^n$. Then we encrypt the message $m = m_1||m_2||...||m_t$ as follows:

$$\begin{aligned} c_0 &= IV \\ c_i &= F_k(m_i \oplus c_{i-1}) \; for \; 1 \leq i \leq t \end{aligned}$$

and the ciphertext will be $c = c_0||c_1||...||c_t$ which is of length $(t+1)n$.

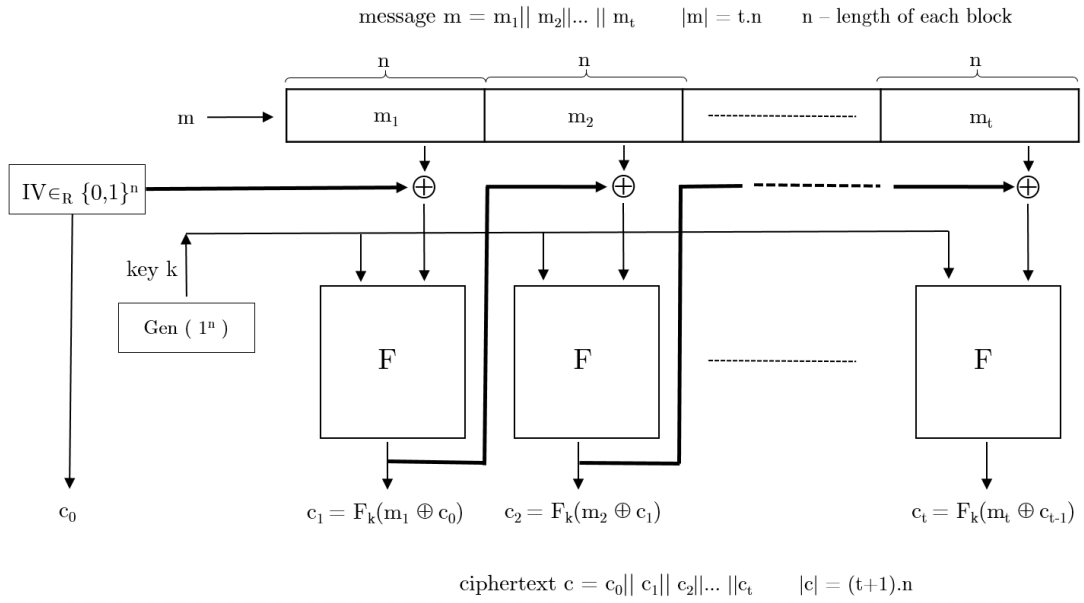

Fig 9 Cipher Block Chaining (CBC) Mode

Unlike ECB, it turns out that this scheme is IND-CPA secure. This scheme cannot do parallel encryption because each block must wait for the encryption of the previous block. However, we can do random access decryption of the i^{th} block using C_i and C_{i-1}, as follows,

$$m_i \quad = \quad c_{i-1} \oplus F_k(c_i)$$

Chained CBC Mode

In this variant of CBC-mode encryption, the last block of the previous ciphertext is used as the IV when encrypting the next message. This reduces the bandwidth, as the IV need not be sent each time. In Fig 10, an initial message $m = m_1||m_2||...||m_t$ is encrypted using a random IV, and then subsequently a second message $\overline{m} = \overline{m}_1||\overline{m}_2||...||\overline{m}_t$ is encrypted using c_t as the IV.Chained CBC-mode is vulnerable to chosen-plaintext attack. The basis of the attack is that the adversary knows in advance the initialization vector that will be used for encrypting the next message.

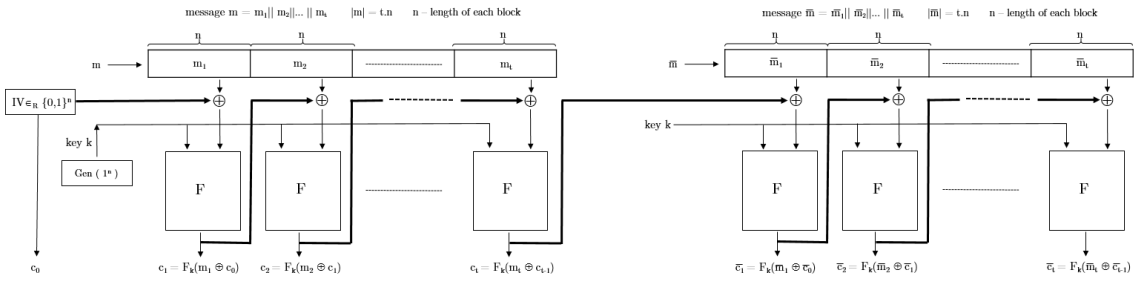

Fig 10 Chained CBC Mode

3.4.3 Output Feedback (OFB) Mode

For OFB, we select an *Initialization Vector*, IV, of block length, n, uniformly at random from $\{0,1\}^n$. Then we encrypt the message $m = m_1||m_2||...||m_t$ as follows:

$$\begin{aligned} y_0 &= IV \\ y_i &= F_k(y_{i-1}) \ for \ 1 \leq i \leq t \\ c_i &= m_i \oplus y_i \ \ for \ 1 \leq i \leq t \end{aligned}$$

and the ciphertext will be $c = c_0||c_1||...||c_t$ which is of length $(t+1)n$.

The y_i's do not depend on the message, so we can pre-compute these and then only compute the $\oplus$ operations for any given message. The stream of these y_i's are called *pseudorandom stream* since these are outputs of some PRF's. However, we cannot do parallel encryption or random access decryption because each block is dependent on the last. However, if we do the pre-computation first, then we can do the $\oplus$ operations in parallel and do random access decryption using the stored y_i values.

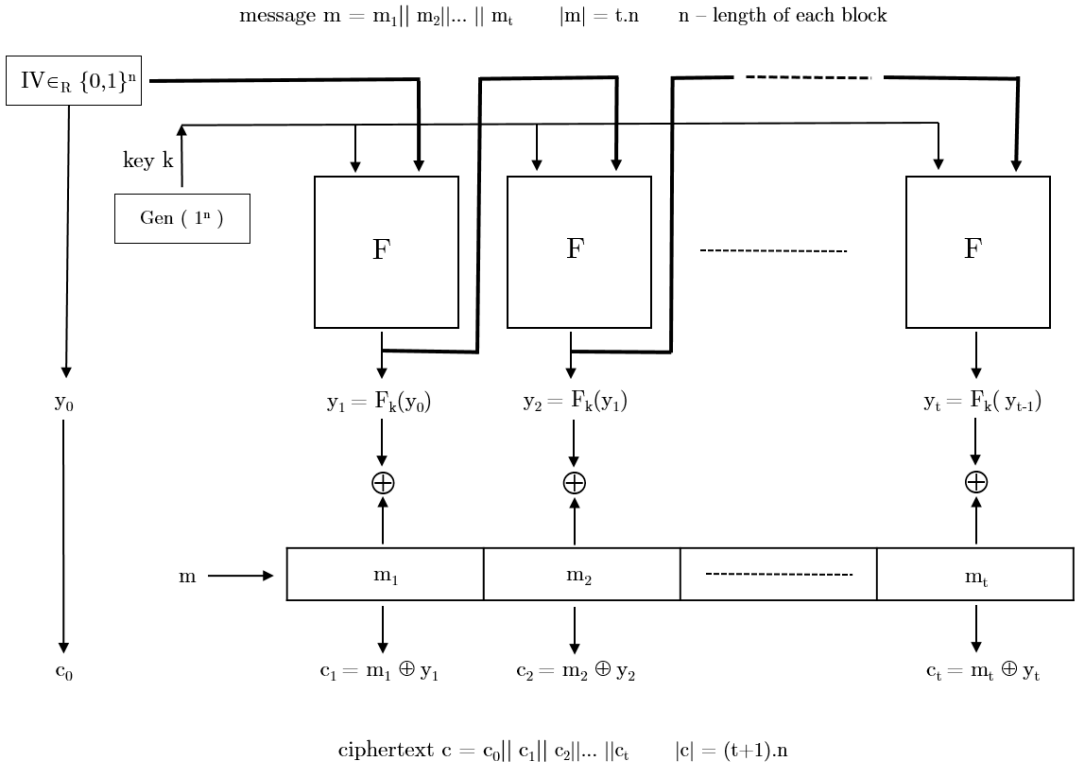

Fig 11 Output Feedback (OFB) Mode

3.4.4 Counter (CTR) Mode

Here we select a counter, CTR, of block length, n, uniformly at random from $\{0,1\}^n$. Then we encrypt the message $m = m_1||m_2||...||m_t$ as follows:

$$\begin{aligned} c_0 &= CTR \\ c_i &= m_i \oplus F_k(CTR+i) \ for \ 1 \leq i \leq t \end{aligned}$$

and the ciphertext will be $c = c_0||c_1||...||c_t$ which is of length $(t+1)n$.

Here the decryption does not require F to be invertible, or even a permutation. Also we can use pre-computation here to generate the pseudorandom stream before the message is known. The CTR mode has the advantage that encryption and decryption can be fully *parallelized*, since all the blocks of the pseudorandom stream can be computed independently of each other. Also it is possible to decrypt the i^{th} block of the ciphertext using only a single evaluation of F.

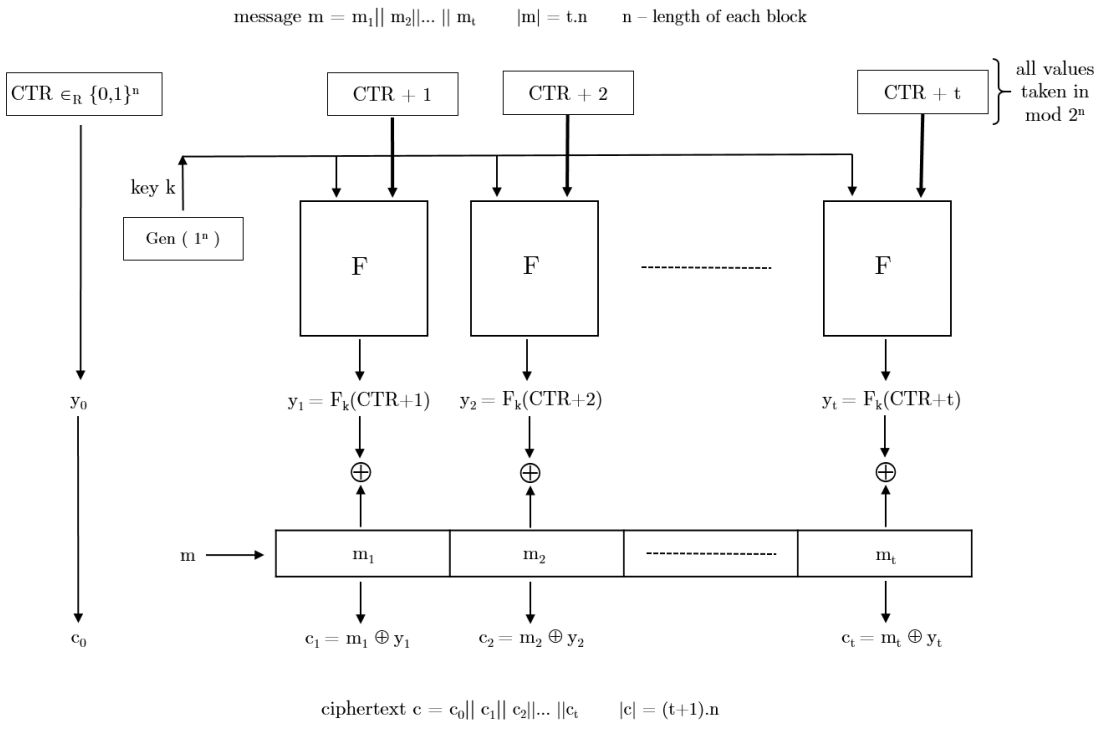

Fig 12 Counter (CTR) Mode

Theorem 3 *If F is a pseudorandom function, then CTR mode is CPA-Secure*

3.4.5 Analysis of Modes of Operations

The table shows the comparison of all the modes we have discussed so far against their features. Here we assumed that the message m is of size $t.n$ and it is divided into t blocks, each of length n.

Mode / Features	Theoretical	ECB	CBC	OFB	CTR
Randomness Usage	n / Block = t.n	No randomness	n	n	n
Ciphertext Expansion	2n / Block = 2t.n	t.n	(t+1).n	(t+1).n	(t+1).n
Ciphertext Computation Parallelizable	YES	YES	NO	NO (But pre-computable)	YES
Randomness Reusability	NO	---	---	YES	YES
Minimal Assumption (PRF/PRP/SPRP)	PRF	SPRP	SPRP	PRF	PRF
CPA Security	YES	NO	YES	YES	YES

Table 1 Analysis of Modes of Operations

4 References

1. Jonathan Katz, Yehuda Lindell : Introduction to Modern Cryptography, Second Edition

2. M. Luby: Pseudorandomness and Cryptographic Applications; Princeton University Press, 1996

3. Mihir Bellare, Anand Desai, E. Jokipii, Phillip Rogaway: A Concrete Security Treatment of Symmetric Encryption. FOCS: 394-403, 1997

4. O. Goldreich, S. Goldwasser and S. Micali. How to Construct Random Functions. JACM, 33(4), 792-807,1986