Cryptography and Security

Jiaxuan Zhang

January 21, 2022

Forword

Jiaxuan Zhang

January 21, 2022

Contents

Part I

Coursera Stanford Cryptography I

Chapter 1

Introduction

1.1 Goals of Crypto

Confidentality

Integrity

we make sure that the received message is equal to the sent one

Authenticated

we make sure about who sent the message

1.2 Some Historic Examples

1.2.1 Substitution Cipher: Caesar Cipher

How to break a substitution cipher

1. use frequency of English letters
2. use frequency of pairs of letters

1.2.2 Vigener Cipher

The format of Vigener Cipher is shown in Figure 1.1.

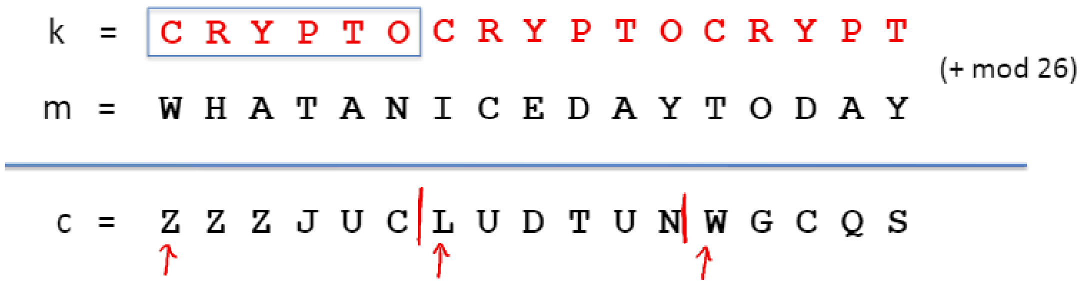

k = C R Y P T O C R Y P T O C R Y P T
(+ mod 26)
m = W H A T A N I C E D A Y T O D A Y
c = Z Z Z J U C L U D T U N W G C Q S

Figure 1.1: Vigener Cipher

1.3 Introduction of Discrete Probability

1.3.1 Randomized Algorithms

Deterministic Algorithm

$$y \leftarrow A(m)$$

.

Randomized Algorithm

$$y \longleftarrow A(m; r) \quad \text{where} \quad r \longleftarrow \{0,1\}^n$$

So the output of $A(m)$ is a random variable

$$\mathrm{y} \leftrightarrow \mathrm{A(m)}$$

1.3.2 An Important Property of XOR

Theorem 1.3.1 (An Important Property of XOR)**.** *An Important Property of XOR: Y a* ***rand. var.*** *over $\{0,1\}^n$, X an* ***indep. uniform var.*** *on $\{0,1\}^n$ Then $Z := Y \oplus X$ is uniform var. on $\{0,1\}^n$*

1.3.3 The Birthday Paradox

Theorem 1.3.2 (The Birthday Paradox)**.** *The Birthday Paradox:*

Let $r_1, \ldots, r_n \in U$ be indep. identically distributed random vars. when $\mathrm{n} = 1.2 \times |\mathrm{U}|^{1/2}$ *then* $\Pr[\exists \mathrm{i} \neq \mathrm{j} : r_{\mathrm{i}} = r_{\mathrm{j}}] \geq 1/2$

1.4 Semantic Security

Here, we summarized different Semantic Security Models and Definitions.

Chapter 2

Stream Cipher

2.1 Information Theoretic Security and The One Time Pad

2.1.1 Symmetric Ciphers

Definition 2.1.1 (Symmetric Cipher)**.** *Symmetric Cipher*

Aa cipher defined over $(\mathcal{K}, M, 6) \quad (k, M, c)$ *is a pair of "efficient" algs* $(\boldsymbol{E}, \boldsymbol{D})$ *where* $E : \mathcal{K} \times M \to 6$, $D : \mathcal{K} \times C \to \mu$ *s.t.* $\forall m \in M, k \in \mathscr{L} : \quad D(k, E(k, m)) = m$

Notes:

- efficient means: can be calculated in a certain polynomial time limit
- E is often Randomized
- D is always deterministic

2.1.2 Perfect Security

Definition 2.1.2 (Shannon Information Theoretic Security)**.** *Shannon Information Theoretic Security:*

A cipher (E, D) *over* (K, M, C) *has perfect secrecy if* $\forall m_0, m_1 \in M \quad (|m_0| = |m_1|)$ *and* $\forall c \in C \Pr\left[E\left(k, m_0\right) = c\right] =$ $\Pr\left[E\left(k, m_1\right) = c\right]$ *where* $k \leftarrow^{R} k$

Notes:

- Given CT, adv. cannot tell if message is m_0 or m_1
- most powerful adv. learns nothing about PT from CT

Theorem 2.1.3. *perfect security* $\Rightarrow \mathcal{K} \geq \mathcal{M}$.

2.1.3 The One-Time Pad: Vernam Cipher

Definition 2.1.4 (One-Time Pad)**.** *A one-time pad is a symmetric cipher* $\mathcal{E} = (E, D)$*, where the keys, messages, and ciphertexts are bit strings of the same length; that is,* $\mathcal{E}$ *is defined over* $(\mathcal{K}, \mathcal{M}, \mathcal{C})$*, where*

$$\mathcal{K} := \mathcal{M} := \mathcal{C} := \{0, 1\}^L$$

for some fixed parameter L*. For a key* $k \in \{0, 1\}^L$ *and a message* $m \in \{0, 1\}^L$ *the encryption function is defined as follows:*

$$E(k, m) := k \oplus m,$$

and for a key $k \in \{0,1\}^L$ and ciphertext $c \in \{0,1\}^L$, the decryption function is defined as follows:

$$D(k,c) := k \oplus c.$$

Theorem 2.1.5 (Inf Secure of OTP). *Inf Secure of OTP:*
OTP has perfect security.

2.2 Pseudo Random Generators

Definition 2.2.1 (Pseudo Random Generators). *Pseudo Random Generators: PRG is a function* $G : \underbrace{\{0,1\}^s}_{\substack{seed\\space}} \to \{0,1\}^n \quad n \gg s$ *(eff. computable by deterministic algorithn)*

2.2.1 Security of RPG

Definition 2.2.2 (negligible and non-negligible). *negligible and non-negligible*

$$\varepsilon \text{ is a function } \varepsilon : \mathbf{Z}^{20} \to \mathbf{R}^{20} \text{ and}$$

$$\begin{aligned} &\varepsilon \text{ non-neg:} \quad \exists d : \varepsilon(\lambda) \geq 1/\lambda^d \text{ inf. often} && (\varepsilon \geq 1/\text{ poly, for many } \lambda) \\ &\varepsilon \text{ negligible: } \forall d, \lambda \geq \lambda_d : \quad \varepsilon(\lambda) \leq 1/\lambda^d && (\varepsilon \leq 1/\text{ poly, for large } \lambda) \end{aligned}$$

Method 2.2.3 (Attack Game of Unpred. PRG). *Attack Game of Unpred. PRG*

*Attack Game 3.2 (Unpredictable **PRG**). For a given PRG G, defined over $(\mathcal{S}, \{0,1\}^L)$, and a given adversary $\mathcal{A}$, the attack game proceeds as follows:*

1. *The adversary sends an index i, with $0 \leq i \leq L-1$, to the challenger.*
2. *The challenger computes*
$$s \overset{\text{R}}{\mathcal{S}}, r \leftarrow G(s)$$
3. *he adversary outputs $g \in \{0,1\}$.*

We say that $\mathcal{A}$ wins if $r[i] = g$, and we define $\mathcal{A}$'s advantage $\text{Predadv}[\mathcal{A}, G]$ *to be* $\mid \Pr[\mathcal{A} \textit{ wins }] - 1/2 \mid$.

Definition 2.2.4 (Unpredictable PRG). *Unpredictable PRG A PRG G is unpredictable if the value Predadv[A, G] is negligible for all efficient adversaries A.*

Notes: When implementation, never use `random()` for crypto!

Method 2.2.5 (Attack Game of PRG). *Attack Game of PRG Attack Game* 3.1(*PRG*). *For a given PRG G, defined over $(\mathcal{S}, \mathcal{R})$, and for a given adversary $\mathcal{A}$, we define two experiments, Experiment 0 and Experiment 1 . For $b = 0,1$, we define: Experiment b : - The challenger computes $r \in \mathcal{R}$ as follows: if $b = 0 : s \longleftarrow \mathcal{R}, r \leftarrow G(s)$; if $b = 1 : r \longleftarrow R\mathcal{R}$.*

Definition 2.2.6 (Secure PRG). *Secure PRG:*

For $b = 0,1$, let W_b be the event that $\mathcal{A}$ outputs 1 in Fixperiment b. We define $\mathcal{A}$'s advantage with respect to G as

$$\text{PRGadv}[\mathcal{A}, G] := |\Pr[W_0] - \Pr[W_1]|$$

A PRGG is secure if the value $\text{PRGadv}[\mathcal{A}, G]$ *is negligible for all efficient adversaries $\mathcal{A}$.*

Theorem 2.2.7 (Unpredictable and Secure PRG)**.** *if* $\forall i \in \{0, \cdots, n-1\}$ *PRG G is unpredictable at pos. i, then G is a secure PRG.*

Notes: A secure PRG is unpredictable. if next-bit predictors cannot distinguish G from random then no statistical test can

2.3 Stream Ciphers: Encrpytion with a PRG

Idea: replace "random" key by "pseudorandom" key.

Method 2.3.1 (Stream Cipher)**.** *Stream Cipher*

Making OTP practical using a PRG: G: $\mathrm{K} \longrightarrow \{0,1\}^{\mathrm{n}}$ *Stream cipher:* $\mathrm{E(k,m) = m \bigoplus G(k)}$, $\mathrm{D(k,c) = c \bigoplus G(k)}$

2.4 Attacks on Stream Ciphers and The One Time Pad

Two-Time Pad is Insecure

$$c_1 \leftarrow m_1 \oplus PRG(k)$$
$$c_2 \leftarrow m_2 \oplus PRG(k)$$

Eavesdropper does:

$$\mathrm{C}_1 \oplus \mathrm{C}_2 \rightarrow m_1 \oplus \mathrm{m}_2$$

Notes: Never use stream cipher key more than once!

- Network traffic: negotiate new key for every session
- Disk encryption: typically do not use a stream cipher

One-Time Pad is malleable: not integrity

Modifications to ciphertext are undetected and have predictable impact on plaintext

2.5 Real-World Stream Ciphers

2.5.1 Old Example: CSS

The process of CSS is shown in Figure 2.1.

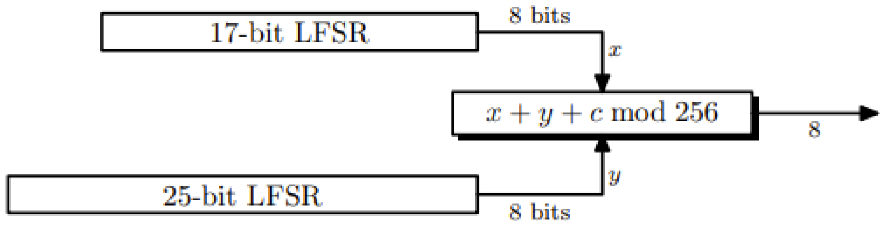


Figure 2.1: CSS

Linear feedback shift registers (LFSR)

The LFSR is a key structure in CSS. The structure of LFSR is shown in Figure 2.2.

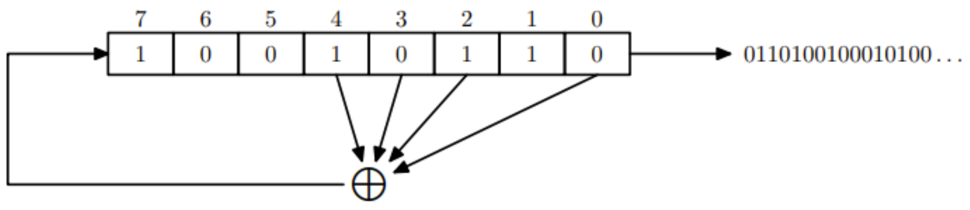


Figure 2.2: LFSR

2.5.2 Modern Example: eStream: Salsa20

eStream Framework

PRG: $\underbrace{\{0,1\}^s}_{\text{seed}} \times R \rightarrow \{0,1\}^n \quad n >> s$

Nonce: a non-repeating value for a given key. R is Nonce

$\mathrm{E(k, m; r) = m \bigoplus PRG(k; r)}$

The pair (k, r) is never used more than once.

Salsa20

$$\begin{aligned}&\text{Salsa20: } \{0,1\}^{128 \text{ or } 256} \times \{0,1\}^{64} \longrightarrow \{0,1\}^{\mathrm{n}} \quad (\max \mathrm{n} = 2^{73} \ \text{ bits})\\ &\mathrm{Salsa20\ (k; r) := H(k, (r, 0)) \| H(k, (r, 1)) \| \ldots}\end{aligned} \tag{2.1}$$

The structure of Salsa20 is shown in Figure 2.3.

2.6 Security Analysis of Stream Ciphers

2.6.1 Semantic Security for One-Time Pad

Method 2.6.1 (Attack Game for Semantic Security). *Attack Game for Semantic Security*

For $b = 0, 1$ define experiments EXP(0) and EXP(1) as

1. *The adversary computes $m0, m1 \in M$, of the same length, and sends them to the challenger.*
2. *The challenger computes $k \xleftarrow{R} K$, $c \xleftarrow{R} E(k, mb)$, and sends c to the adversary.*
3. *The adversary outputs a bit $\hat{b} \in \{0, 1\}$.*

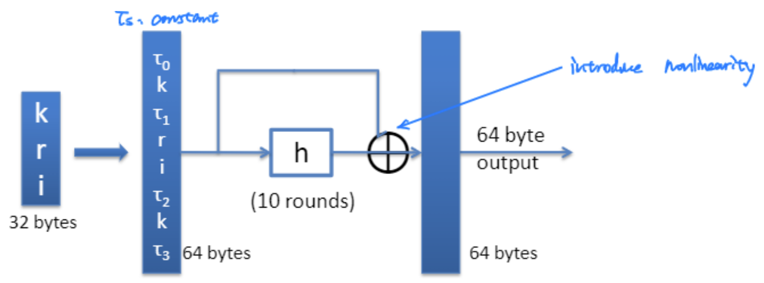


Figure 2.3: Salsa20

Definition 2.6.2 (Semantic Security). *Semantic Security*

For $b = 0, 1$, let W_b be the event that $\mathcal{A}$ outputs 1 in Experiment b. We define $\mathcal{A}$'s semantic security advantage with respect to $\mathcal{E}$ as

$$\mathrm{SSadv}[\mathcal{A}, E] := |\Pr[W_0] - \Pr[W_1]|$$

A cipher E is semantically secure if for all efficient adversaries A, the value $SSadv[A, E]$ is negligible

2.6.2 Stream Ciphers are Semantically Secure

Theorem 2.6.3 (OPT Sem. Sec.). *OPT Sem. Sec.*

OTP is semantically secure

2.7 Summary

Starting from OTP and the theorem of **Perfect Security**, we point out the the OTP met the Confidentiality as long as the key length is equal to message length. But it is impossible in practice.

Stream cipher tries to solve this problem by using a key to generate a series of pseudorandom keys. In order to do that, we need to guarantee the process of generate a pseudorandom keys do not harm the security.

So we define what is a secure PRG and relate it with unpredictable. We can also prove that the OTP is Sem. secure with a secure RPG.

Chapter 3

Block Cipher

3.1 PRPs and PRFs

Definition 3.1.1 (PRP). *PRP:*

Pseudo Random Permutation (PRP) defined over (K,X): $\mathrm{E} : \mathrm{K} \times \mathrm{X} \to \mathrm{X}$ *such that:*

1. *Exists "efficient" deterministic algorithm to evaluate* $E(k, x)$
2. *The function* $E(k, \cdot)$ *is one-to-one*
3. *Exists "efficient" inversion algorithm* $D(k, y)$

Definition 3.1.2 (PRF). *PRF:*

Pseudo Random Function (PRF) defined over (K,X,Y): $F : K \times X \to Y \to$ *such that exists "efficient" algorithm to evaluate* $F(k, x)$

3.1.1 Secure PRFs and Secure PRPs

From intuition, a PRF is secure if a random function in $Funs[X, Y]$ is indistinguishable from a random function in S_F. The experiment for secure PRF is shown in Figure 3.1.

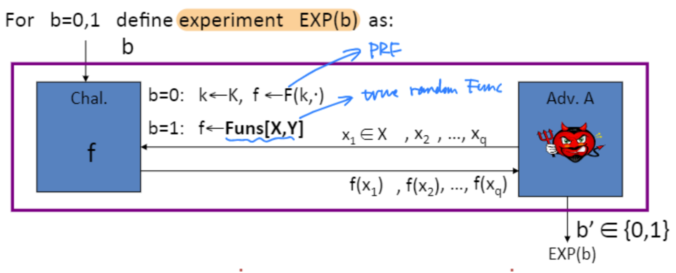


Figure 3.1: Secure PRF Experiment

Definition 3.1.3 (Secure PRF). *Secure PRF*

F is a secure PRF if for all "efficient" A:

$$\mathrm{Adv}_{\mathrm{PRF}}[A,\ \mathrm{F}] := |\,\Pr[\mathrm{EXP}(0) = 1] - \Pr[\mathrm{EXP}(1) = 1]|$$

is "negligible."

The experiment for secure PRF is shown in Figure 3.2.

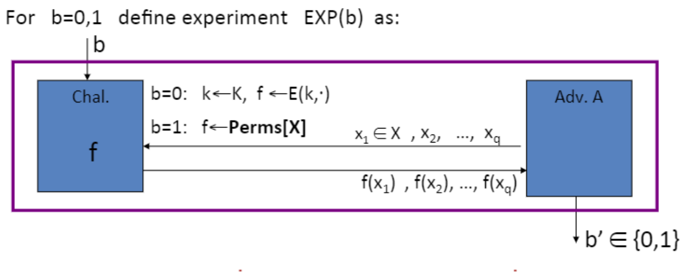


Figure 3.2: Secure PRP Experiment

Definition 3.1.4 (Secure PRP). *Secure PRP:*

E is a secure PRP if for all "efficient" A:

$$\mathrm{Adv}_{\mathrm{PRP}}[A, E] = |\Pr[\mathrm{EXP}(0) = 1] - \Pr[\mathrm{EXP}(1) = 1]|$$

is "negligible."

AES, 3DES are PRPs that is believed to be secure.

There is a Lemma about the relation between secure PRP and secure PRF. The lemma tell us that "any secure PRP is also a secure PRF, if $|X|$ is sufficiently large"

Lemma 3.1.5 (PRF Switching Lemma). *PRF Switching Lemma Let* E *be a PRP over* (K, X) *Then for any q-query adversary A:*

$$|\mathrm{Adv}_{PRF}\,[A, E] - \mathrm{Adv}_{PRP}\,[A, E]| < q^2/2|X|$$

3.1.2 Constructing PRGs from PRFs

Method 3.1.6 (From PRFs to PRGs: 1). *From PRFs to PRGs:*

Let $\mathrm{F} : \mathrm{K} \times \{0,1\}^{\mathrm{n}} \to \{0,1\}^{\mathrm{n}}$ *be a secure PRF. Then the following* $G : K \to \{0,1\}^{nt}$ *is a secure PRG:*

$$\mathrm{G(k)} = \mathrm{F(k, 0)} \| \mathrm{F(k, 1)} \| \cdots \| \mathrm{F(k, t-1)}$$

One advantage of this method is it is parallelizable.

3.1.3 Constructing PRFs from PRGs

The first method is Tree Construction that helps us generate PRFs from PRGs, the structure of the Tree Construction is shown in Figure 3.3.

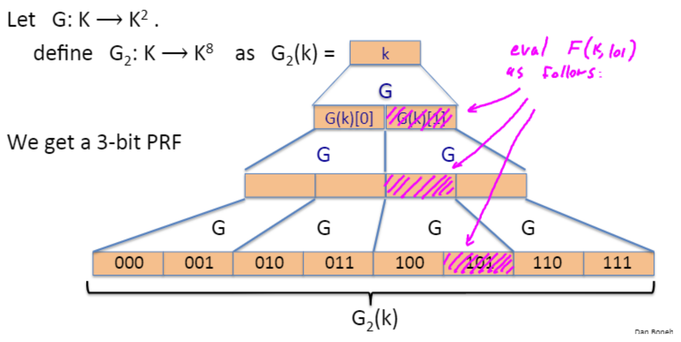
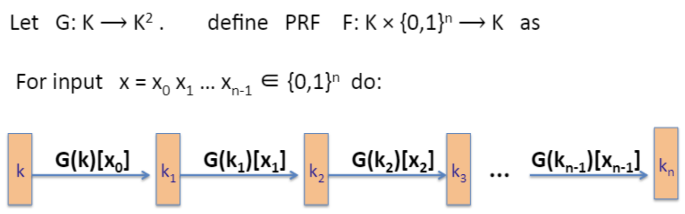


Figure 3.3: Tree Construction

Another method to generate PRFs to PRGs is the GGM PRF method. Which is shown in Figure 3.4.



Let $G: K \longrightarrow K^2$. define PRF $F: K \times \{0,1\}^n \longrightarrow K$ as

For input $x = x_0 x_1 \dots x_{n-1} \in \{0,1\}^n$ do:

$$\boxed{k} \xrightarrow{G(k)[x_0]} \boxed{k_1} \xrightarrow{G(k_1)[x_1]} \boxed{k_2} \xrightarrow{G(k_2)[x_2]} \boxed{k_3} \dots \xrightarrow{G(k_{n-1})[x_{n-1}]} \boxed{k_n}$$

Figure 3.4: GGM PRF

However, this method is quite slow, so it is not used in practice.

A property exists for these two methods: **If G is a secure PRG then the F is a secure PRF**

3.2 Block Ciphers

The structure of the block cipher is shown in Figure 3.5.

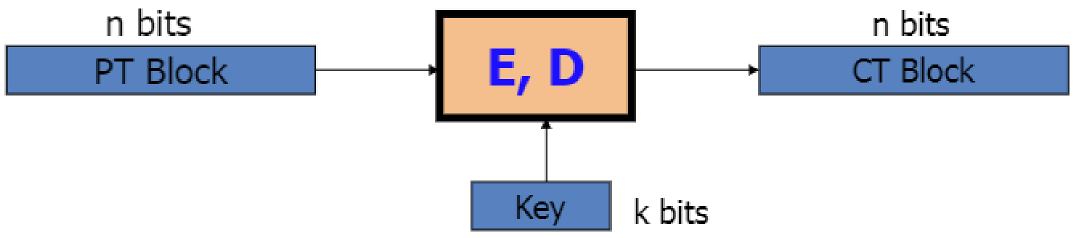
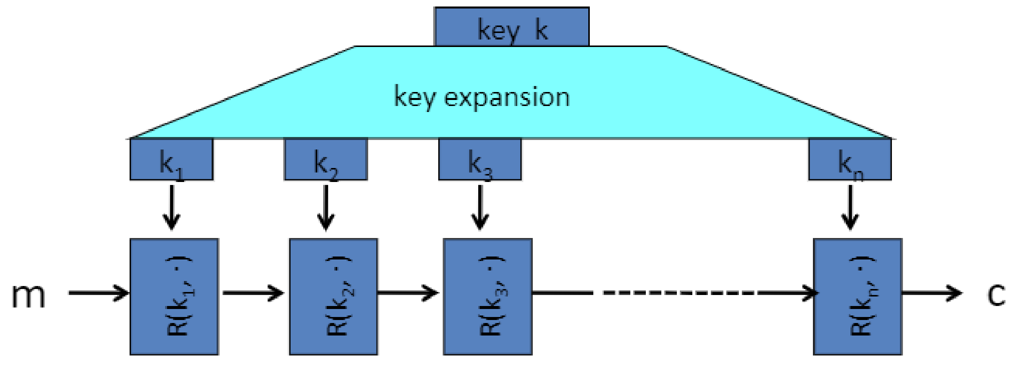


Figure 3.5: Block Cipher

The general framework of the block cipher is shown in Figure 3.6



R(k,m) is called a round function

Figure 3.6: Block Cipher Structure

3.2.1 The Data Encryption Standard (DES)

The DES encryption algorithm is a **16-round Feistel network** where each round uses a different function $f : \mathcal{X} \to \mathcal{X}$(round function).

Feistel Network

The encryption process of Feistel Network is shown in Figure 3.7.

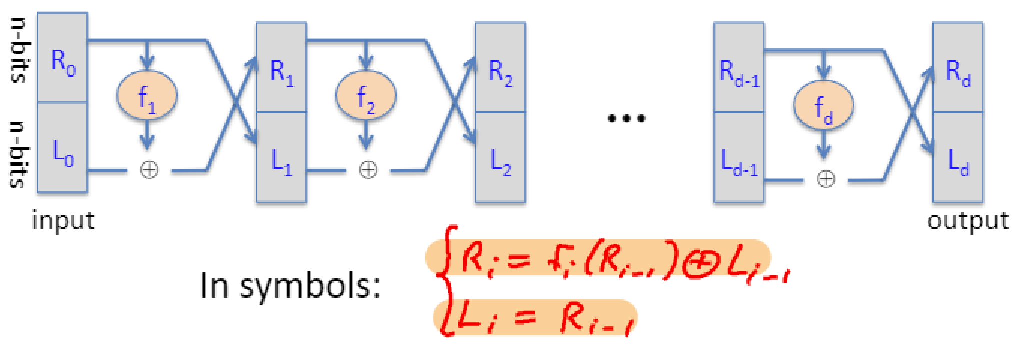


Figure 3.7: Feistel Network Encryption

The decryption process of Feistel Network is shown in Figure 3.8.

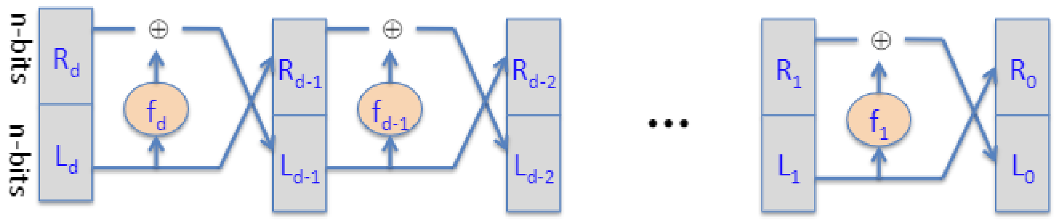


Figure 3.8: Feistel Network Decryption

The inversion part is basically the same circuit with $f_1, \cdots f_d$ applied in reverse order.

Theorem 3.2.1 (Theorem about Security of Feistel Network)**.** *Theorem about Security of Feistel Network:* *f:* $\mathrm{K} \times \{0,1\}^{\mathrm{n}} \rightarrow \{0,1\}^{\mathrm{n}}$ *a secure PRF* $\Rightarrow$ *3-round Feistel* $\mathrm{F} : \mathrm{K}^3 \times \{0,1\}^{2\mathrm{n}} \rightarrow \{0,1\}^{2\mathrm{n}}$ *a secure PRP*

3.2.2 DES Round Function

In round number i the function f is defined as

$$f(x) := F(k_i, x)$$

where k_i is a 48 -bit key for round number i and F is a fixed function called the DES round function. The function F is the centerpiece of the DES algorithm and is shown in Figure 3.9.

The auxiliary functions in the round function is shown as follows

1. The function E expands a 32-bit input to a 48-bit output by rearranging and replicating the input bits. For example, E maps input bit number 1 to output bits 2 and 48 ; it maps input bit 2 to output bit number 3 , and so on.
2. The function P, called the mixing permutation, maps a 32 -bit input to a 32 -bit output by rearranging the bits of the input. For example, P maps input bit number 1 to output bit number 9 ; input bit number 2 to output number 15 , and so on.
3. At the heart of the DES algorithm are the functions $S_1, \ldots, S_8$ called S-boxes. Each S-box S_i maps a 6-bit input to a 4-bit output by a lookup table. The DES standard lists these 8 look-up tables, where each table contains 64 entries.

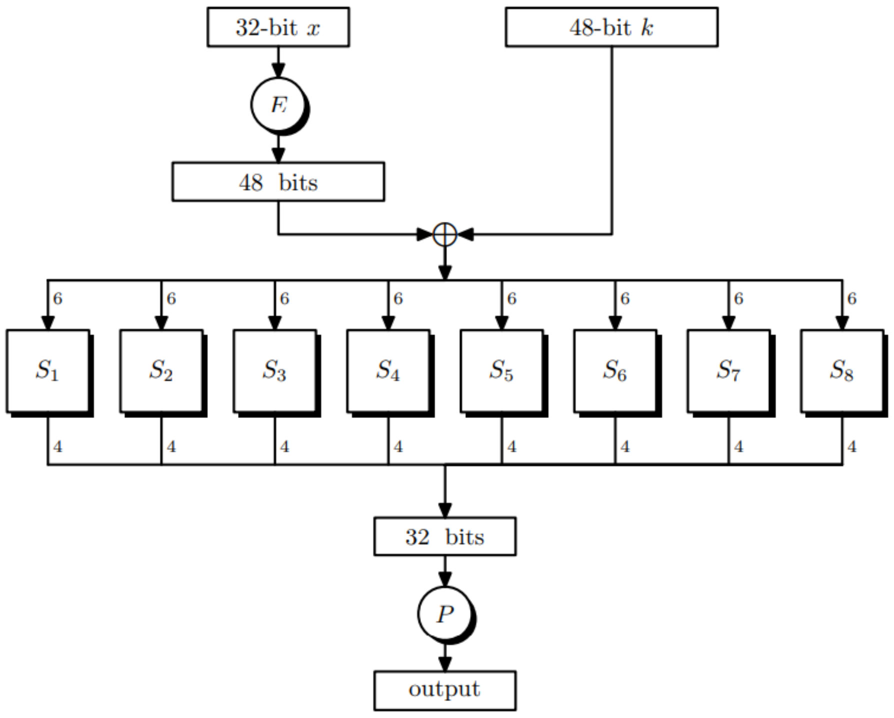


Figure 3.9: DES Round Function

It is important to choose appropriate S-BOXES and P-BOXES.

Triple DES

For Normal DES with a 56-bit cipher, it can be broken in 7 days. So we need more method to make DES more safety under exhausitve search attack.

Method 3.2.2 (Triple DES)**.** *The Triple-DES standard. NIST approved Triple-DES for government use through the ycar 2030. Strictly speaking, the NIST version of Triplc-DES is defined as*

$$E_3\left(\left(k_1, k_2, k_3\right), x\right) := E\left(k_3, D\left(k_2, E\left(k_1, x\right)\right)\right).$$

The reason for this is that setting $k_1 = k_2 = k_3$ reduces the NIST Triple-DES to ordinary DES and hence Triple-DES hardware can be used to implement single DES.

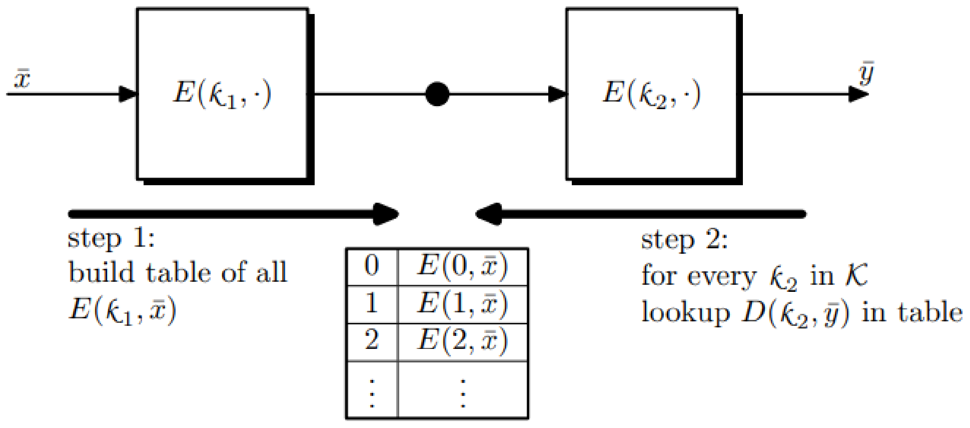
Meet in the Middle Attack

The problem is why we do not use a Double-DES. The problem is for Double-DES there is an efficient attack: Meet in the Middle Attack. The structure of meet in the middle attack is shown in Figure 3.10.

DESX

Another method is DESX.

Method 3.2.3 (DESX)**.** *DESX:*



0	$E(0, \bar{x})$
1	$E(1, \bar{x})$
2	$E(2, \bar{x})$
⋮	⋮

Figure 4.10: Meet in the middle attack on $2\mathcal{E}$

Figure 3.10: DES Meet in the Middle Attack

$$
\begin{aligned}
&\mathrm{E} : \mathrm{K} \times \{0,1\}^{\mathrm{n}} \longrightarrow \{0,1\}^{\mathrm{n}} \ \textit{a block cipher} \\
&\textit{Define EX as } \mathrm{EX}\left((\mathrm{k}_1, \mathrm{k}_2, \mathrm{k}_3), \mathrm{m}\right) = \mathrm{k}_1 \oplus \mathrm{E}\left(\mathrm{k}_2,\ \mathrm{m} \oplus \mathrm{k}_3\right) \\
&\textit{For DESX: key-len } = 64 + 56 + 64 = 184 \textit{ bits}
\end{aligned} \tag{3.1}
$$

DESX has an easy attack in time $2^{64+56} = 2^{120}$ compared to its key length

3.2.3 AES

The high level overview of AES is shown in Figure 3.11. It is a substitution-permutation network.

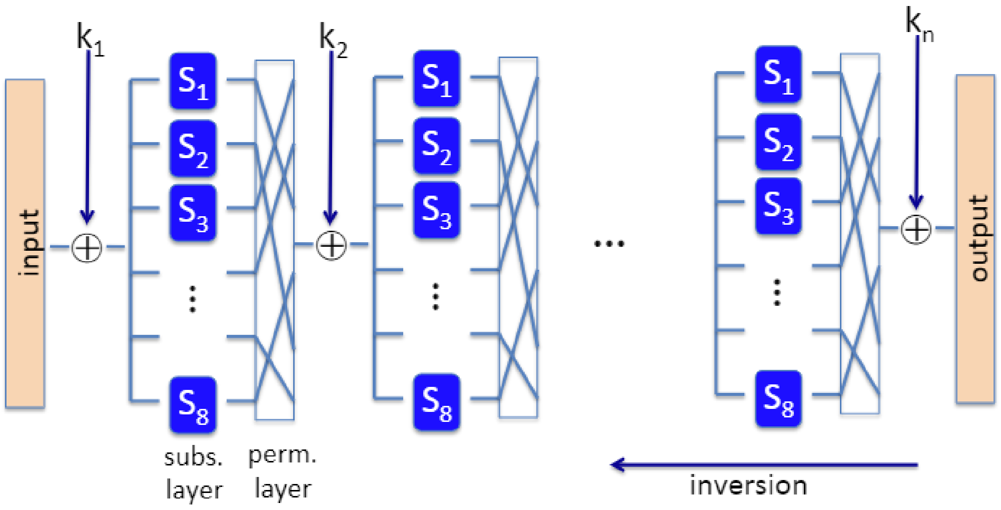


Figure 3.11: AES High-level

Schematic of AES-128

The schematic of AES-128 is shown in Figure 3.12.

The auxiliary functions of AES is shown as follows:

1. ByteSub: $A[i,7]) \in S[A(i,J)]$

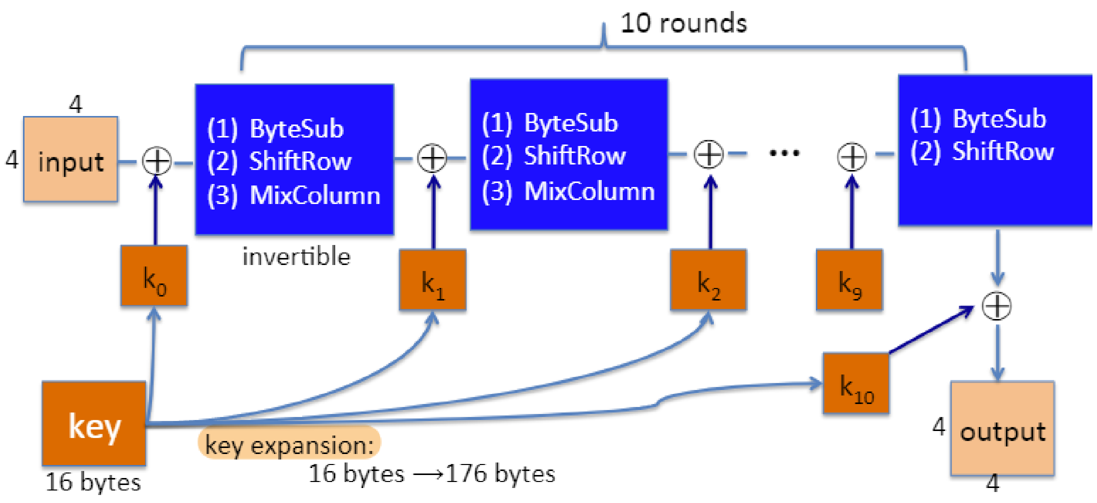
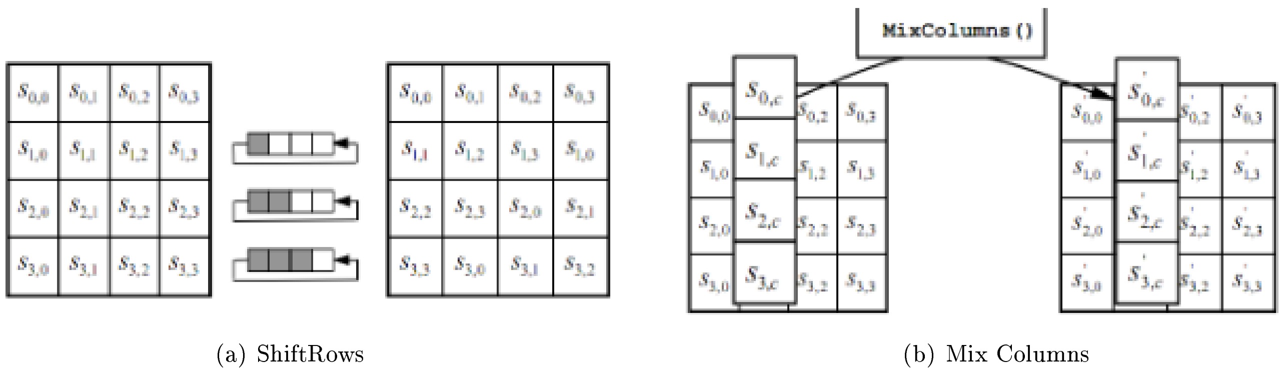


Figure 3.12: AES128 Schematic

2. ShiftRows: shown in Figure 3.13.

3. MixColumns: shown in Fiugre 3.13



(a) ShiftRows (b) Mix Columns

Figure 3.13: AES Auxialiary Functions

3.3 Security of Block Ciphers

3.3.1 Exhaustive Search Attacks

The goal of exhaustive search attack is: given a few input output pairs $(m_i, c_i = E(k, m_i))$, find key.

3.3.2 Semantic Security for Many-Time Key

For the many-time key cases, the adversary is able to see many CTs with same key.

Definition 3.3.1 (Chosen-Plaintext Attack(CPA)). *Chosen-Plaintext Attack(CPA)*
The adversary is able to obtain the encryption of arbitrary messages of his choice.

The attack game of many-time key is shown in Figure 3.14.

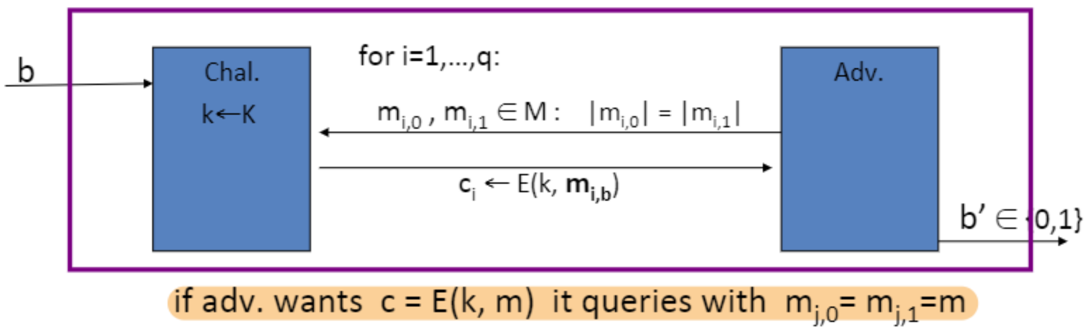

b
Chal.
k←K
for i=1,...,q:
m_{i,0} , m_{i,1} ∈ M : |m_{i,0}| = |m_{i,1}|
c_i ← E(k, m_{i,b})
Adv.
b' ∈ {0,1}
if adv. wants c = E(k, m) it queries with m_{j,0}= m_{j,1}=m

Figure 3.14: Attack of Sem Secure for Many-Time Key

Definition 3.3.2 (Sem. Sec with CPA). *E is sem. sec. under CPA if for all "efficient" A:*

$$\text{Adv}_{CPA}\,[A, E] = |\Pr[\text{EXP}(0) = 1] - \Pr[\text{EXP}(1) = 1]|$$

is "negligible."

If a system is not Sem. Sec. with CPA, this means an attacker can learn that two encrypted files are the same or two encrypted packets are the same. This is malicious, because like in a voice stream, the attacker will know the break time.

Property

Suppose $E(k, m)$ always outputs the same ciphertext for msg m, then it will not be secure with CPA (shown in Figure 3.15).

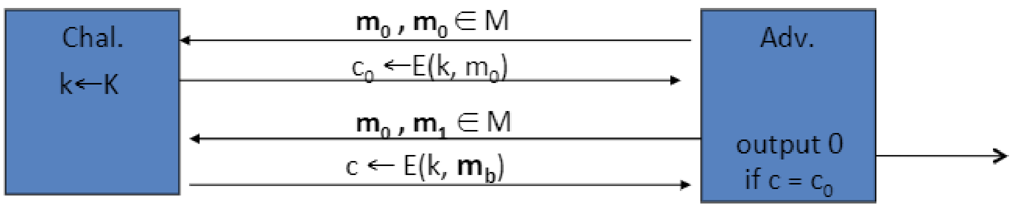


Figure 3.15: Ciphers Insecure under CPA

This means, if a secret key is to be used multiple times, we should have: given the same plaintext message twice, the encryption must produce different outputs.

Solution 1: Randomized Encryption

The structure of the randomized encryption is shown in Figure 3.16.

Which means we should use a $E(k, m)$ that is a randomized algorithm. However, it also means the ciphertext must be longer than plaintext. (because it is a one to n mapping)

Solution 2: Nonce-Based Encryption

Definition 3.3.3 (nonce). *nonce*

a value that changes from msg to msg.

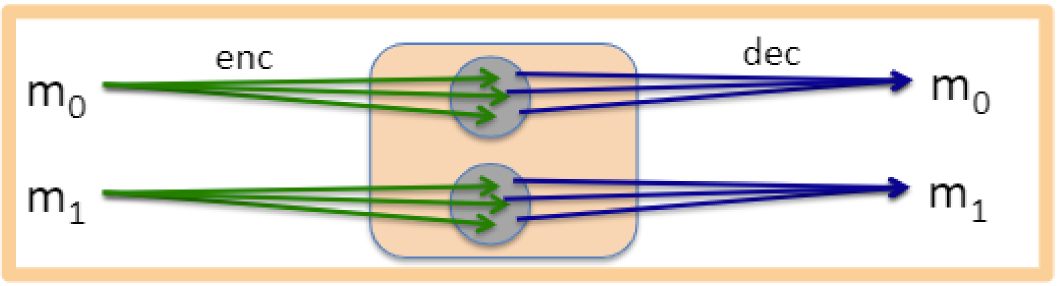


Figure 3.16: Randomized Encryption

(k,n) pair never used more than once

The structure of the nonce-based encryption is shown in Figure 3.17.

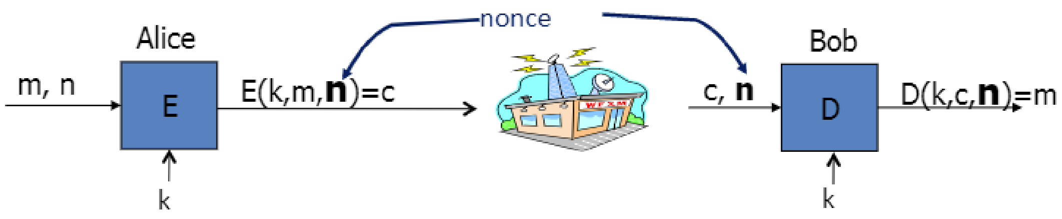


Figure 3.17: Nonce-Based Encryption

There are mainly two ways to define a nonce:

- nonce is a counter, this is used when encryptor keeps state from msg to msg
- encryptor chooses a random nonce

The attack of nonce-based encryption is shown in Figure 3.18.

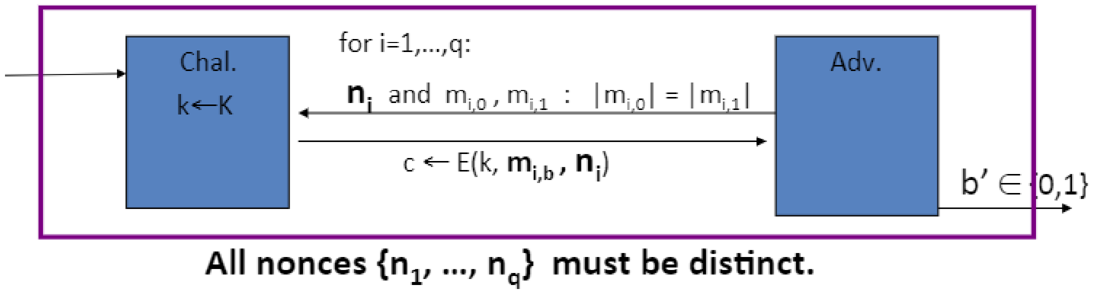

Chal.
k←K
for i=1,...,q:
n_i and m_i,0 , m_i,1 : |m_i,0| = |m_i,1|
c ← E(k, m_i,b , n_i)
Adv.
b' ∈ {0,1}
All nonces {n_1, ..., n_q} must be distinct.

Figure 3.18: Attack for nonce-based encryption

Definition 3.3.4 (Sem.Secure for Nonce-Based E under CPA). *Sem.Secure for Nonce-Based E under CPA: nonce-based E is sem. sec. under CPA if for all "efficient" A:* $\text{Adv}_{nCPA}[A, E] = |\Pr[\text{EXP}(0) = 1] - \Pr[\text{EXP}(1) = 1]|$ *is "negligible."*

3.3.3 More Attacks on Block Ciphers

Attacks on the implementation

There are a lot of method to attack block cipher based on the implementation. So the main lesson is: **Don't Design Ciphers Yourself**

3.3.4 Modes of Operations

3.3.5 ECB: An Incorrect Use of a PRP

The electronic Code Block is a classical incorrect way of using a PRP. The structure of ECB is shown in Figure 3.19.

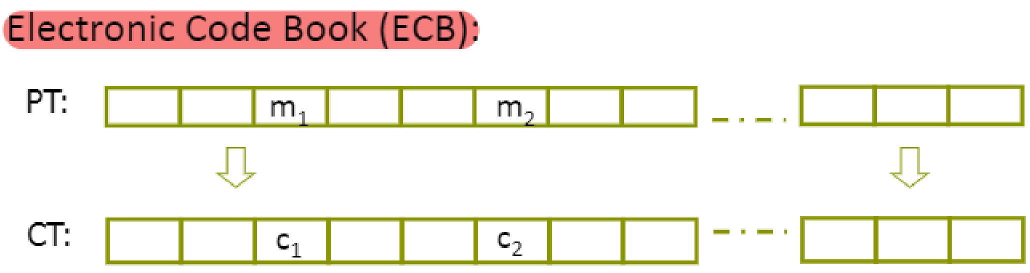


Figure 3.19: ECB Mode

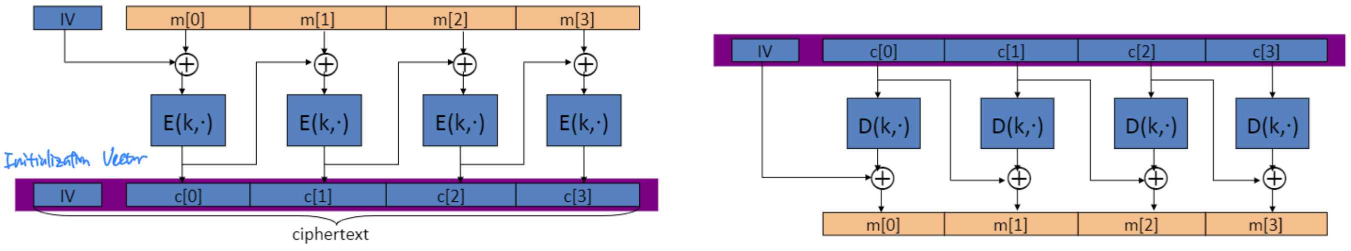
The problem of ECB is for the same message $m_1 = m_2$ it will produce the same ciphertext $c_1 = c_2$. So it will lost the CPA Attack.

3.3.6 CBC with random IV

CBC: Cipher Block Chain

IV: Initial Vector

The encrpytion and decryption process of CBC with random IV is shown in Figure 3.20.



(a) Encryption (b) Decyprtion

Figure 3.20: CBC with random IV

Nonce-Based CBC

The nonce-based CBC use a key pair $key = (k, k_1)$. It is a CBC with unique nonce, i.e. (key,n) pair is used for only one message. The structure of the nonce-based CBC is shown in Figure 3.21.

Padding in CBC

The CBC mode need a padding part to extend the message in order to let the length becomes a multiple of block size. The pad is removed during decryption.

One example is the padding used in the TLS: for $n > 0, n$ byte pad is $\boxed{n}\boxed{n}\boxed{n}\boxed{\cdots}\boxed{n}$ if no pad needed, add a dummy block

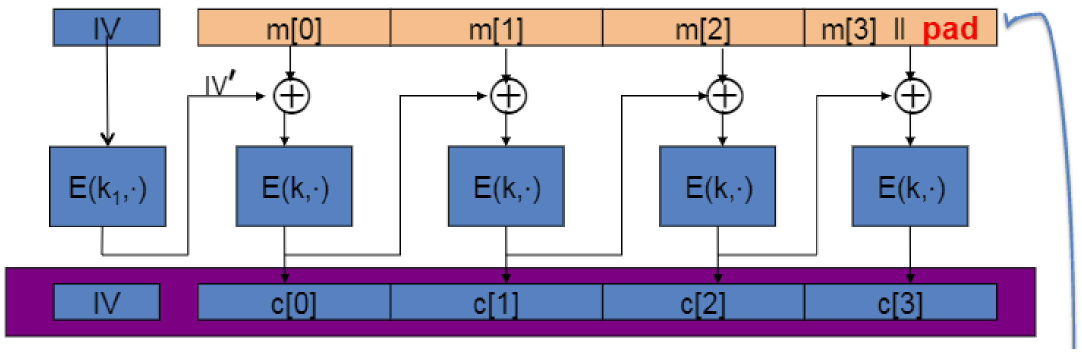


Figure 3.21: Nonce-based CBC

CPA Security

Theorem 3.3.5 (CBC CPA Security). *For any* $L > 0$*, If* E *is a secure PRP over* (K, X) *then* E_{CBC} *is a sem. sec. under CPA over* (K, X^L, X^{L+1})*. In particular, for a q-query adversary* A *attacking* E_{CBC} *there exists a PRP adversary* B *s.t.:* $\mathrm{Adv}_{\mathrm{CPA}}[A, E_{\mathrm{CBC}}] \leq 2 \cdot \mathrm{Adv}_{\mathrm{PRP}}[B, E] + 2q^2L^2/|X|$

Based on the theorem, we know that after w^{48} AES blocks, we must change key.

Notes: CBC where attacker can predict the IV is not CPA-secure.

3.3.7 Random Counter(ctr) Mode

Let $\mathrm{F} : \mathrm{K} \times \{0, 1\}^{\mathrm{n}} \to \{0, 1\}^{\mathrm{n}}$ be a secure PRF. $\mathrm{E(k, m)}$: choose a random $\mathrm{IV} \in \{0, 1\}^{\mathrm{n}}$.

Then the structure of the rand ctr-mode is shown in the Figure 3.22.

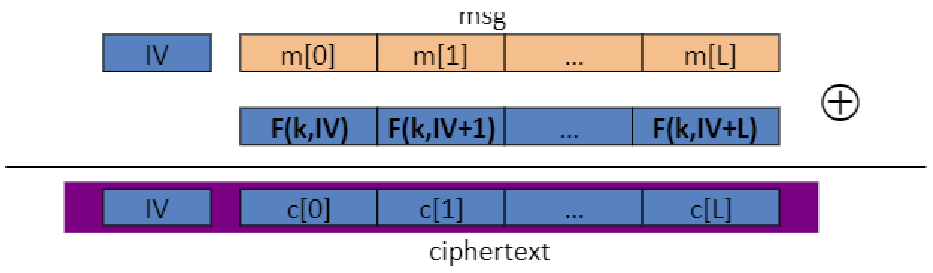


Figure 3.22: rand ctr-mode

This is a **parallelizable structure**

Nonce-Based ctr mode

CPA Security

Theorem 3.3.6 (CPA Security for CTR Mode). *CPA Security for CTR Mode:*

Counter-mode Theorem: For any $L > 0$*, If* F *is a secure PRF over* (K, X, X) *then* E_{CTR} *is a sem. sec. under CPA over* (K, X^L, X^{L+1})*. In particular, for a q-query adversary* A *attacking* E_{CTR} *there exists a PRF adversary* B *s.t.:* $\mathrm{Adv}_{\mathrm{CPA}}[A, E_{\mathrm{CTR}}] \leq 2 \cdot \mathrm{Adv}_{\mathrm{PRF}}[B, F] + 2q^2L/|X|$

ctr-mode only secure as long as $q^2L < |X|$. Better than *CBC*. Based on the theorem, if we use AES in CTR mode, after 2^32 CTs each of len 2^32, we must change key.

3.4 Summary

This chapter, we start from PRP and PRF. We defined what is secure PRF and secure PRP. And we propose method to construct PRF or PRG from each other.

Then we introduce two useful method of block cipher: AES and DES. AES is much more safer while DES is less safer and we always use 3-DES.

Then we define the model of CPA Sem. Sec. Based on the CPA Sem. Sec., we knows that if we want to use the same key many times and keep safe, we need to let the same message generate different ciphertext. Based on that, we need the randomized algorithm or nonce-based algorithm. Which means we need to use PRG or PRF.

Besides, although we use some randomized component, how to use it is also a big problem. We knows that ECB is not secure. Classical secure modes of operation contains: CBC mode and CTR mode.

Chapter 4

Integrity

4.1 Message Integrity

The goal of message integrity is integrity, that is, the information must be protected against any malicious modification.

4.1.1 MACs

MAC Framework

Definition 4.1.1 (MAC). *MAC:*

MAC I=(S,V) defined over (K,M,T) is a pair of algorithms:

- *Signing Function: S(k,m) outputs a tag t in time T*
- *Verification Function: V(k,m,t) outputs 'yes' or 'no'*

The basic framework of MAC is shown in Figure 4.1.

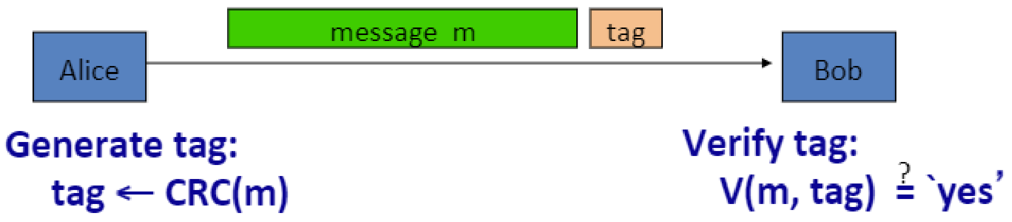


Figure 4.1: MAC FrameWork

4.1.2 Secure MAC

Attackers in MACs

Attacker's Power: chosen message attack, that is attacker can given $m_1, m_2, \cdots, m_q$, and an oracle will give back $\mathrm{t_i} \leftarrow \mathrm{s}(\mathrm{k}, \mathrm{m_i})$.

Attacker's Goal: existential forgery, that is to produce some new valid message/tag pair (m,t), $(\mathrm{m}, \mathrm{t}) \notin \{(\mathrm{m_1}, \mathrm{t_1}), \ldots, (\mathrm{m_q}, \mathrm{t_q})\}$.

Motivation of Secure MAC

intuitively, we have two expectations, A secure MAC should be able to:

1. an attacker cannot produce a valid tag for a new message, even for a gibberish message
2. given (m,t), attacker cannot even produce (m,t') for $t' \neq t$

Definition of Secure MAC

A game in MAC is defined as in Figure 4.2.

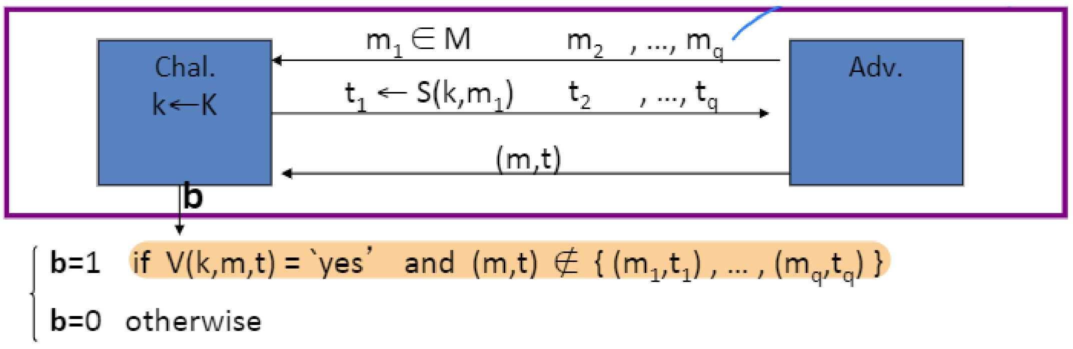

Chal.
k←K
$m_1 \in M$ m_2 , ..., m_q
$t_1 \leftarrow S(k,m_1)$ t_2 , ..., t_q
(m,t)
Adv.
b
b=1 if V(k,m,t) = 'yes' and $(m,t) \notin \{(m_1,t_1), \ldots, (m_q,t_q)\}$
b=0 otherwise

Figure 4.2: Game for MAC

One thing should be notice that, here, the adversary actually do not know the Ciphertext.

So we can generate the following definition of **secure MAC**

Definition 4.1.2 (Secure MAC). *Secure MAC*

I=(S,V) is a ***secure MAC*** *if for all "efficient" A,*

$$\mathrm{Adv}_{\mathrm{MAC}}[A, I] = \Pr[\ \textit{Chal. outputs } 1] \quad \textit{is "negligible."}$$

4.2 Collision Resistance

4.2.1 Introduction

Here are the definitions of **collision** and **collision resistant**.

Definition 4.2.1 (collision and collision resistant). *collision and collision resistant:*

Let $H : M \to T$ be a hash function $(|M| >> |T|)$ *A collision for* H *is a pair* $\mathrm{m}_0,\ \mathrm{m}_1 \in \mathrm{M}$ *such that:*

$$\mathrm{H}(\mathrm{m}_0) = \mathrm{H}(\mathrm{m}_1) \ \textit{ and } \mathrm{m}_0 \neq \mathrm{m}_1$$

A function H *is collision resistant if for all (explicit) "eff" algs. A:* $\mathrm{Adv}_{\mathrm{CR}}[\mathrm{A}, \mathrm{H}] = \Pr[\mathrm{A}$ *outputs collision for* H] *is "neg".*

4.2.2 Generic Birthday Attack

Generic Attack on C.R. functions

Method 4.2.2 (Generic Attack). *Generic Attack:*

1. *Choose* $2^{n/2}$ *random messages in* $M : m_1, \ldots, m_2^{n/2}$ *(distinct w.h.p)*

2. *For* $\mathrm{i} = 1, \ldots, 2^{\mathrm{n}/2}$ *compute* $\mathrm{t_i} = \mathrm{H}(\mathrm{m_i}) \quad \in \{0,1\}^{\mathrm{n}}$
3. *Look for a collision* $(t_i = t_j)$. *If not found, got back to step 1 .*

The time complexity of this method is $\mathcal{O}(2^{n/2})$

The Birthday Paradox

Theorem 4.2.3 (The Birthday Paradox)**.** *The Birthday Paradox:*
when $\mathrm{n} = 1.2 \times \mathrm{B}^{1/2}$ *then* $\Pr[\exists \mathrm{i} \neq \mathrm{j} : \mathrm{r_i} = \mathrm{r_j}] \geq 1/2$

Notes: If the distribution is uniform distribution, than it is the best case. Other distribution will lead to worse case.

The birthday paradox is illustrated in Figure 4.3.

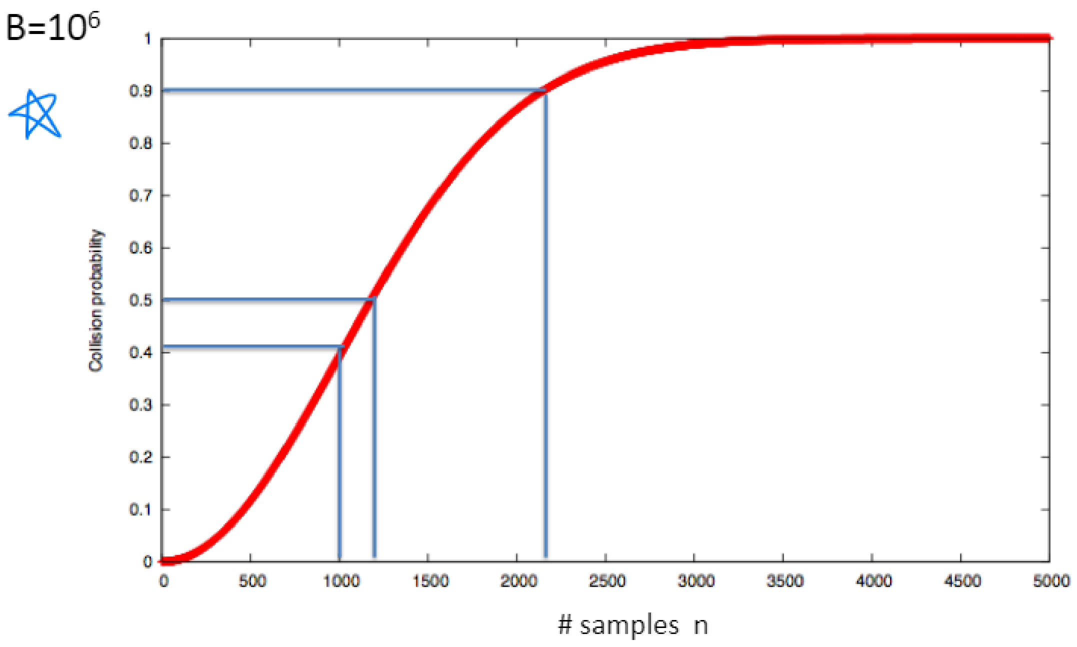


Figure 4.3: The Birthday Paradox

Based on the birthday paradox, if we use the generic attack, the expected number of iteration is roughly equal to 2.

4.3 Construction of Collision Resistance

The construction will be divided into two steps:

1. given C.R. function for short messages, we can construct C.R. function for long messages. (The Merkle-Damagard Paradigm). That is, if we can find a C.R. compression function, then we can build a C.R. Hash function.
2. Then we will propose some method to build C.R. compression function

4.3.1 The Merkle-Damagard Paradigm

The structure of the Merkle-Damagard Paradigm is shown in Figure 4.4.

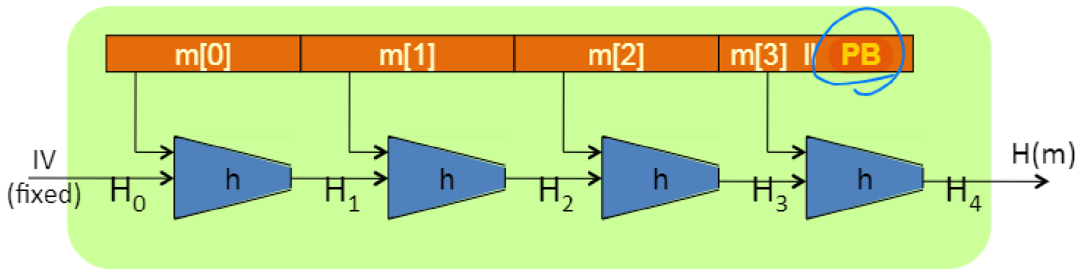


Figure 4.4: MD Framework

That is given $\mathrm{h} : \mathrm{T} \times \mathrm{X} \rightarrow \mathrm{T}$ (compression function), we obtain $\mathrm{H} : \mathrm{X}^{\leq \mathrm{L}} \longrightarrow \mathrm{T}$. ($\mathrm{H_i}-$chaining variables).

The **PB** part is the padding block. [10000$\cdots$000 || msg_len(64-bits)]. If no space for PB, then add another block.

Based on Merkle-Damagard Paradigm, we have a theorem

Theorem 4.3.1 (MD Collision Resistance). *MD Collision Resistance:*

If h is C.R., then so is H

The proof is quite simple from final part and rollback.

Notes: The MD C.R. theorem told us, if we want to construct C.R. function, suffices to construct compression function.

4.3.2 Constructing Compression Functions

Construct Compression Function from Block Cipher

One method called **Davies-Meyer** Method: $\mathrm{h(H, m) = E(m, H) \oplus H}$ and is shown in Figure 4.5.

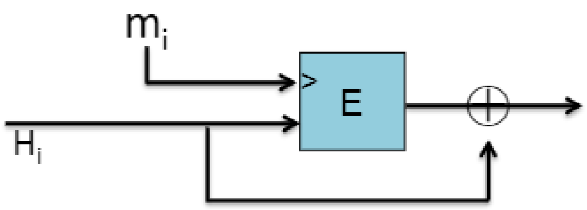


Figure 4.5: Davies_Meyer Method

Theorem 4.3.2. *Suppose E is an ideal cipher (collection of |K| random perms.). Finding a collision* $\mathrm{h(H, m) = h(H', m')}$ *takes* $\mathrm{O}\left(2^{n/2}\right)$ *evaluations of* $\mathrm{(E, D)}$.

One application of the Davies-Meyer method is the SHA-256, which is shown in Figure 4.6.

Provable Compression functions

Method 4.3.3 (Provable Compression functions). *Provable Compression functions:*

Choose a random 2000-bit prime p and random $1 \leq u, v \leq p$. *For* $m, h \in \{0, \ldots, p-1\}$ *define* $h(H, m) = u^H \cdot v^m \pmod{p}$

Fact: finding collision for h is as hard as solving "discrete-log" module p. The problem of this method is that it is too slow.

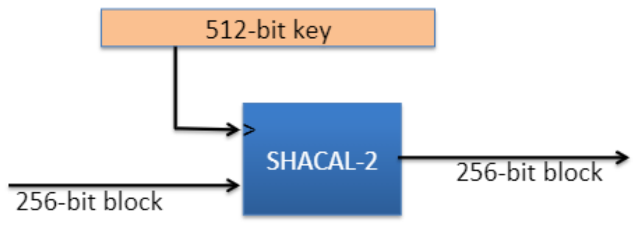


Figure 4.6: SHA256 DM Method

4.4 MACs based on PRFs

Property

Theorem 4.4.1 (From Secure PRF to Secure MAC). *From Secure PRF to Secure MAC:*

If $\mathrm{F} : \mathrm{K} \times \mathrm{X} \to \mathrm{Y}$ *is a secure PRF and* $1/|\mathrm{Y}|$ *is negligible (i.e.* $|Y|$ *is large) then* I_F *is a secure MAC.*

In particular, for every eff. MAC adversary A attacking I_F *there exists an eff. PRF adversary B attacking F s.t.:*

$$\mathrm{Adv}_{\mathrm{MAC}}\left[\mathrm{A}, \mathrm{I}_F\right] \leq \mathrm{Adv}_{\mathrm{PRF}}[\mathrm{B}, \mathrm{F}] + 1/|\mathrm{Y}|$$

Notes: I_F **is secure as long as** $|Y|$ **is large**

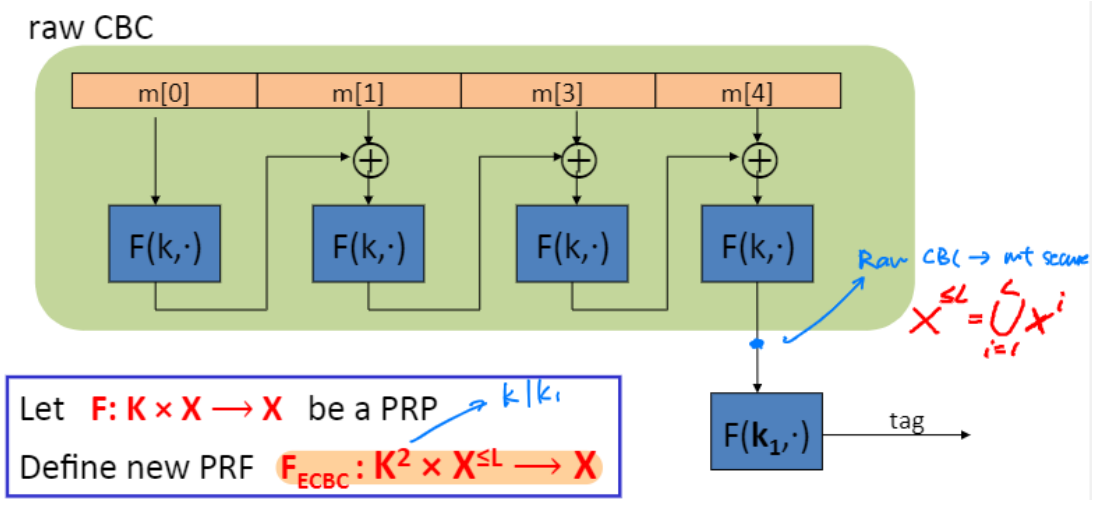
And we can regard **AES** as a MAC for 16-byte messages.

Motivation

Given a PRF for short messages (AES), construct a PRF for long messages.

4.4.1 ECBC-MAC

The structure of ECBC-MAC is shown in Figure 4.7.



Let $\mathbf{F: K \times X \longrightarrow X}$ be a PRP $k | k_1$

Define new PRF $\mathbf{F_{ECBC} : K^2 \times X^{\leq L} \longrightarrow X}$

Figure 4.7: ECBC-MAC Structure

One problem is Why the last Encryption step in ECBC-MAC?

Suppose we define a MAC $I_{\mathrm{RAW}} = (S, V)$ where

$$S(k, m) = \mathrm{rawCBC}(k, m)$$

Then I_{RAW} is easily broken using a 1-chosen msg attack.

Adversary works as follows:

1. choose an arbitrary one-block message $m \in X$
2. request tag for m, Get $t = F(k, m)$
3. Output t as MAC forgery for the 2-block message (m, t $\oplus$ m)

Then the adversary win the challenge: rawCBC$(k, (m, t \oplus m)) = F(k, F(k, m) \oplus (t \oplus m)) = F(k, t \oplus (t \oplus m)) = t$

CBC-MAC Padding

For security, padding must be invertible, i.e. $m_0 \neq m_1 \Rightarrow \quad \text{pad}\,(m_0) \neq \text{pad}\,(m_1)$.

The first method is the **ISO** standard: pad with "1000...00". Add new dummy block if needed. The " 1 " indicates beginning of pad. The structure of the ISO standard padding is shown in Figure 4.8.

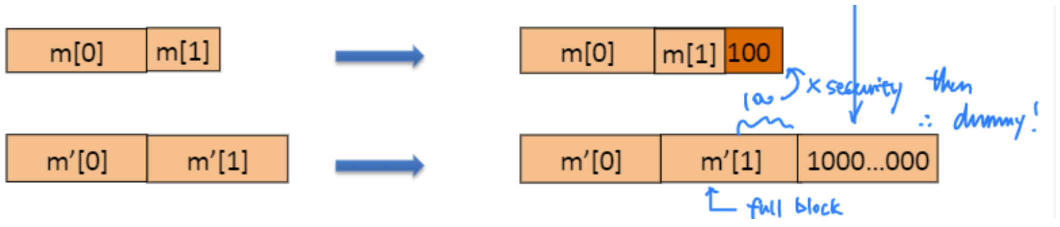


Figure 4.8: ISO CBC Padding

The second method is the **NIST** standard: it use a key tuple: $key = (k, k_1, k_2)$, in which, (k_1, k_2) are derived from k. From this standard, we do not need the final encryption step of ECBC, and we do not need dummy block. The structure of the NIST standard padding is shown in Figure 4.9.

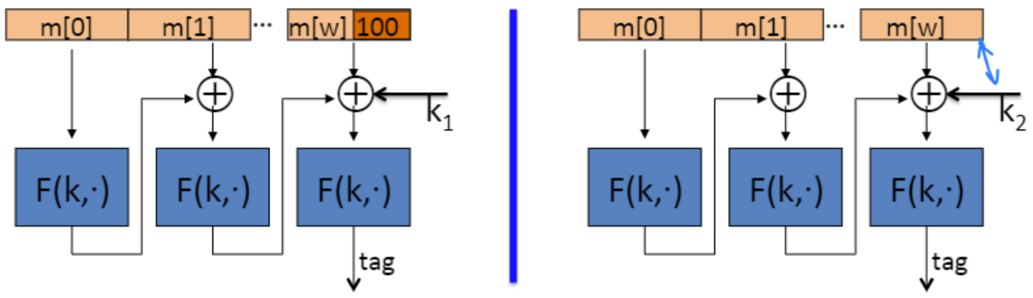


Figure 4.9: NIST CBC Padding

Randomize MAC

Another form of MAC with better security is shown in Figure 4.10

The security of this method is: $Adv_{\text{MAC}}\,[A, I_{\text{RCBC}}] \leq Adv_{\text{PRP}}[B, F] \cdot \left(1 + 2q^2/|X|\right)$

4.4.2 NMAC

The structure of NMAC is shown in Figure 4.11.

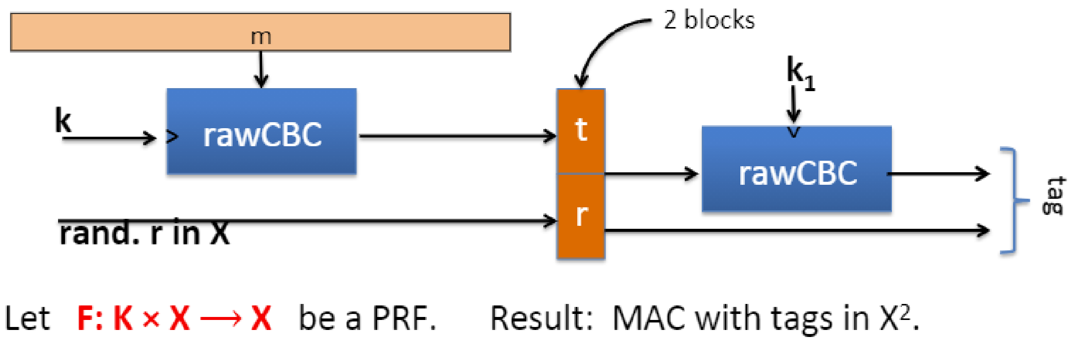


Figure 4.10: Randomize MAC

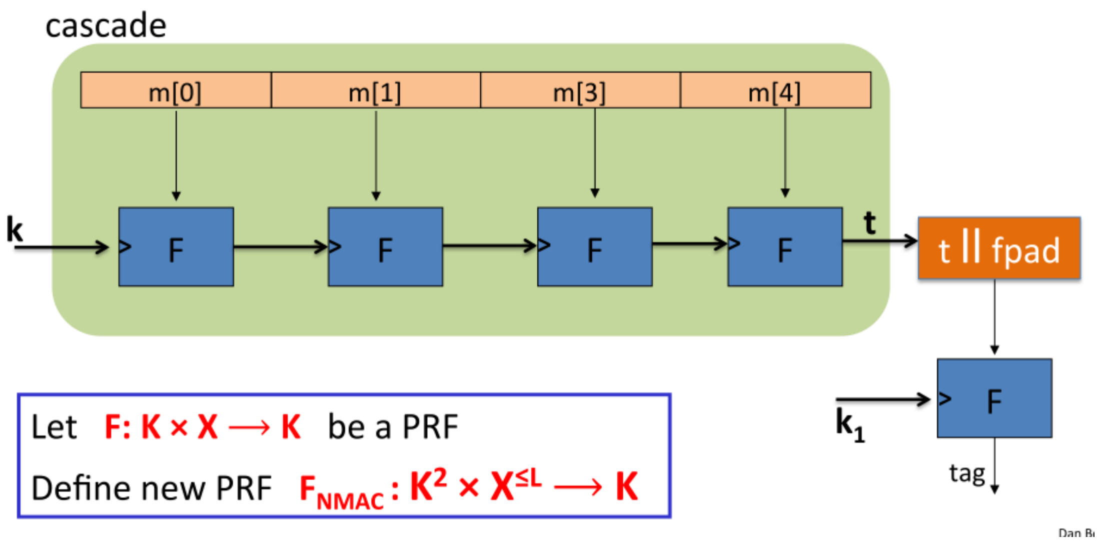

cascade
m[0]
m[1]
m[3]
m[4]
k
F
F
F
F
t
t || fpad
k₁
F
tag
Let F: K × X ⟶ K be a PRF
Define new PRF F_NMAC : K² × X^≤L ⟶ K
Dan B

Figure 4.11: NMAC Structure

4.4.3 Security Analysis of ECBC-MAC and NMAC

Theorem 4.4.2 (Security of ECBC-MAC and NMAC). *Security of ECBC-MAC and NMAC:*

For any $L > 0$, For every eff. q-query PRF adv. A attacking F_{ECBC} or F_{NMAC} there exists an eff. adversary B s.t.:

$$\mathrm{Adv}_{\mathrm{PRF}}\,[A, F_{\mathrm{ECBC}}] \le \mathrm{Adv}_{\mathrm{PRP}}[B, F] + 2q^2/|X|$$
$$\mathrm{Adv}_{\mathrm{PRF}}\,[A, F_{\mathrm{NMAC}}] \le q \cdot L \cdot \mathrm{Adv}_{\mathrm{PRF}}[B, F] + q^2/2|K|$$

Notes: CBC-MAC is secure as long as $q \lll |X|^{1/2}$ NMAC is secure as long as $q < |K|^{1/2}$

An example using this theorem:

Suppose we want $\mathrm{Adv_{PRF}}\,[\mathrm{A}, \mathrm{F_{ECBC}}] \le 1/2^{32} \Leftrightarrow \mathrm{q}^2/|\mathrm{X}| < 1/2^{32}$ Then for AES: $|X| = 2^{128} \Rightarrow q < 2^{48}$ So, after 2^{48} messages must, must change key

Comparison of ECBC-MAC and NMAC

ECBC-MAC is commonly used as an AES-based MAC.

NMAC not usually used with AES or 3DES. Main reason is: it need to change AES key on every block, requires re-computing AES key expansion.

4.4.4 PMAC

The structure of paralle MAC (PMAC) is shown in Figure 4.12

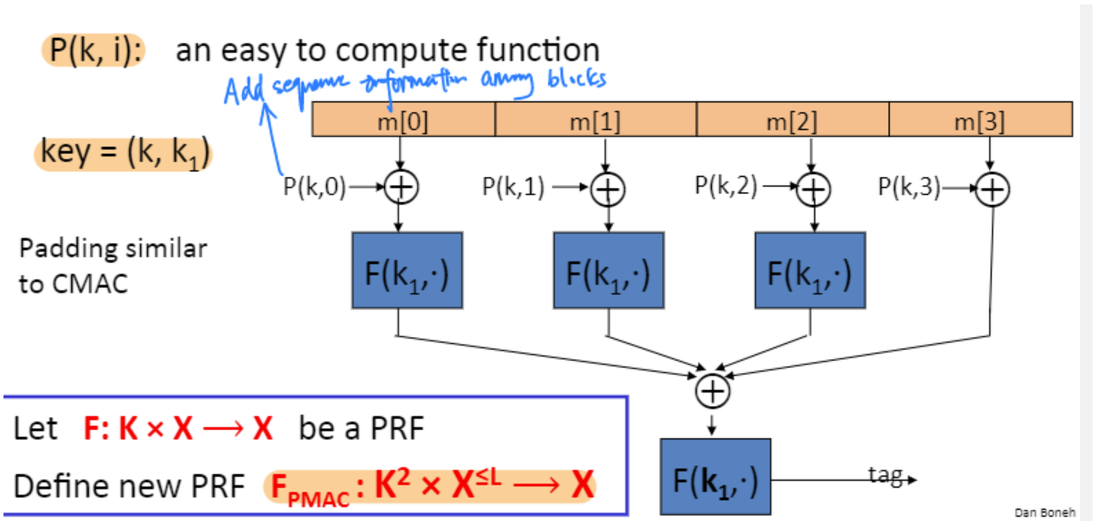

P(k, i): an easy to compute function
Add sequence information among blocks
key = (k, k1)
m[0]
m[1]
m[2]
m[3]
P(k,0)
P(k,1)
P(k,2)
P(k,3)
Padding similar to CMAC
F(k1,·)
F(k1,·)
F(k1,·)
F(k1,·)
tag
Let F: K × X ⟶ X be a PRF
Define new PRF F_PMAC : K² × X^≤L ⟶ X
Dan Boneh

Figure 4.12: PMAC Structure

Property of PMAC

PMAC is **incremental**. Which means we can easily generate a new tag when only one part of the PMAC is changed. For example when $m[1] \leftarrow m'[1]$, we can just do $\mathrm{F}^{-1}(\mathrm{k}_1, \mathrm{tag}) \oplus \mathrm{F}(\mathrm{k}_1,\ \mathrm{m}[1] \oplus \mathrm{P}(\mathrm{k}, 1)) \oplus \mathrm{F}(\mathrm{k}_1,\ \mathrm{m}'[1] \oplus \mathrm{P}(\mathrm{k}, 1))$

Security Analysis

Theorem 4.4.3 (Security of PMAC). *Security of PMAC:*

For any $L > 0$, If F is a secure PRF over (K, X, X) then F_{PMAC} is a secure PRF over $(K, X \leq L, X)$. For every eff. q-query PRF adv. A attacking F_{PMAC} there exists an eff. PRF adversary B s.t.: $\mathrm{Adv}_{\mathrm{PRF}}[\mathrm{A}, \mathrm{F}_{\mathrm{PMAC}}] \leq \mathrm{Adv}_{\mathrm{PRF}}[\mathrm{B}, \mathrm{F}] + 2\mathrm{q}^2\ \mathrm{L}^2/|\mathrm{X}|$

From the theorem, we knows that: PMAC is secure as long as $qL \ll |X|^{1/2}$.

4.4.5 Carter-Wegman MAC

One-time MAC

4.5 MACs from Collision Resistance

4.5.1 HMAC

The HMAC method need two dependent keys and an IV. The structure of HMAC is shown in Figure 4.13. It can be presented by:

$$S(k, m) = H(k \oplus opad \| H(k \oplus ipad \| m))$$

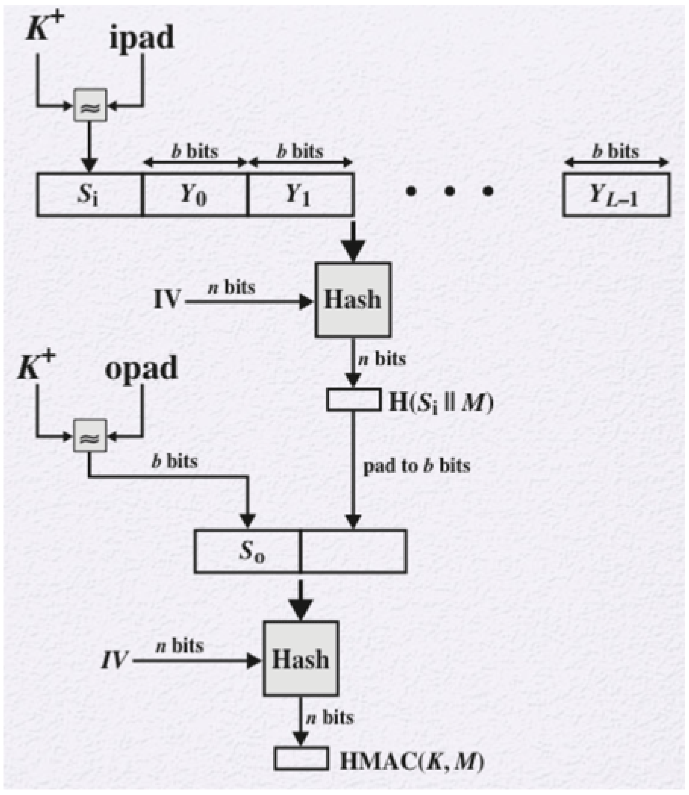


Figure 4.13: HMAC Structure

Properties:

HMAC is assumed to be a secure PRF

- Can be proven under certain PRF assumptions about h(.,.)
- Security bounds similar to NMAC: Need $q^2/|T|$ to be negligible $\left(q < |T|^{1/2}\right)$

4.6 Timing Attacks on MAC Verification

Timing attacks try to use the step of Verification of the HMAC, because originally the earlier byte will be test first. So based on the feedback timing, some attacker try to find the real MAC.

So the lesson is **Don't implement crypto yourself**.

4.7 Summary

In this chapter, we first introduce what is an MAC, and we defined what is a secure MAC. The basic intuition is "the adversary cannot build a new (m,t) pair that is available".

Then we defined what is collision resistant and how to build collision resistant function from compression function and how to build compression function from Block Cipher. We also declared that we can regard AES as a PRF.

Then we illustrate three method for building MAC based on PRFs: ECBC-MAC and NMAC and PMAC. And based on previous discussion of collision resistant function, we discussed the MAC based on collision resistant Hash Function: HMAC.

Chapter 5

Authenticated Encryption

5.1 Passive Attack and Active Attack

The CPA Security can defend passive attack, however, it cannot defend active attack. In active attack, the attacker can not only get the plaintext, it can also get the ciphertext.

For example in Figure 5.1. If the attacker get to know the ciphertext and message of the two message. It can easily generate a new (m,c,t) pair that is available. It can be done by just use $IV' = IV \oplus (\ldots 80 \ldots) \oplus (\ldots..25 \ldots)$.

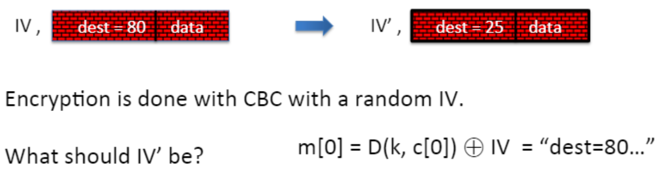

IV ,
dest = 80
data
IV' ,
dest = 25
data
Encryption is done with CBC with a random IV.
What should IV' be?
m[0] = D(k, c[0]) ⊕ IV = "dest=80..."

Figure 5.1: Example of Active Attack to MAC

So we have the conclusion that "CPA security cannot guarantee secret under active attacks". We should choose methods from:

1. If message needs integrity but no confidentiality, use a MAC
2. If message needs both integrity and confidentiality, use authenticated encryption modes

5.2 Chosen Ciphertext Attacks

Adversary's power: both CPA and CCA

- Can obtain the encryption of arbitrary message of his choice
- Can decrypt any ciphertext of his choice, other than challenge

The experiment about CCA is shown in Figure 5.2

So we can define the security under CCA:

Definition 5.2.1 (CCA Security). E *is CCA secure if for all "efficient" A:*

$$\mathrm{Adv}_{CCA}[A, E] = |\Pr[\mathrm{EXP}(0) = 1] - \Pr[\mathrm{EXP}(1) = 1]| \textit{ is "negligible."}$$

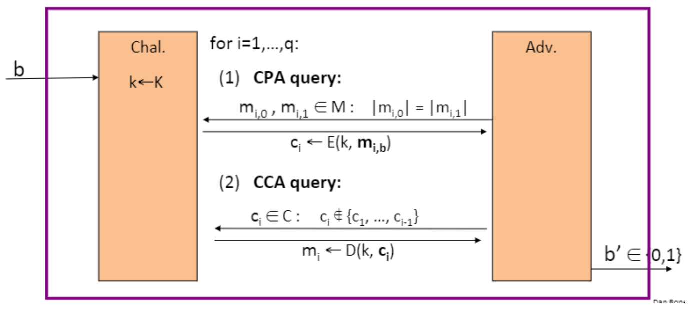


Figure 5.2: CCA Experiment

5.3 Ciphetext Integrity

The experiment for Ciphertext Integrity is shown in Figure 5.3.

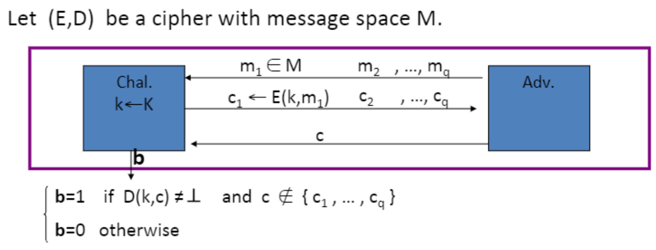

Let (E,D) be a cipher with message space M.
Chal.
k←K
m_1 ∈ M m_2 , ..., m_q
c_1 ← E(k,m_1) c_2 , ..., c_q
c
Adv.
b
b=1 if D(k,c) ≠ ⊥ and c ∉ { c_1 , ... , c_q }
b=0 otherwise

Figure 5.3: Ciphertext Integrity

Then we can define the Ciphertext Security:

Definition 5.3.1 (Ciphertext Security)**.** *Ciphertext Security*

(E,D) has ciphertext integrity if for all "efficient" A : $\mathrm{Adv}_{\mathrm{CI}}[A, E] = \mathrm{Pr}[$ *Chal. outputs 1] is "negligible."*

5.4 Authenticated Encryption

5.4.1 Conceptions

Definition 5.4.1 (Authenticated Encryption)**.** *Authenticated Encryption*

An authenticated encryption system (E, D) *is a cipher where As usual:* $\mathrm{E} : \mathrm{K} \times \mathrm{M} \times \mathrm{N} \to \mathrm{C}$ *but D:* $K \times C \times N \to M \cup \{\perp\}$ ***(which means the ciphertext is rejected)***

And we want the system provide:

- Sem.Sec under a CPA attack
- Ciphertext integrity: attacker cannot create new ciphertexts that decrypt properly.

Definition 5.4.2 (Authenticated Encryption Cipher)**.** *Authenticated Encryption Cipher:*

Cipher (E,D) provides authenticated encryption (AE) if it is:

- *semantically secure under CPA, and*
- *has ciphertext integrity*

A authenticated encryption cipher imply two things:

1. Attacker cannot fool Bob into thinkging a message was sent from Alice: if $D(k, c) \neq \perp$ Bob knows message is from someone who knows k (but message could be a replay)

2. security against chosen ciphertext attack

5.4.2 Security Analysis

Theorem 5.4.3 (Authenticated Encryption and CCA Security). *Authenticated Encryption and CCA Security*

Let (E, D) be a cipher that provides AE. Then (E,D) is CCA secure! In particular, for any q-query eff. A there exist eff. B_1, B_2 s.t. $Adv_{CCA}[A, E] \leq 2q \cdot Adv_{CC}[B_1, E] + Adv_{CPA}[B_2, E]$

5.4.3 Constructions from Ciphers and MACs

MAC-then-Encrypt

One example (SSL) of MAC-then-Encrypt is shown in Figure 5.4.

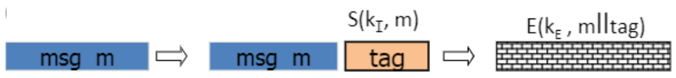


Figure 5.4: MAC-then-Encrypt

Encrypt-then-MAC

One example (IPsec) of Encrypt-then-MAC is shown in Figure 5.5.

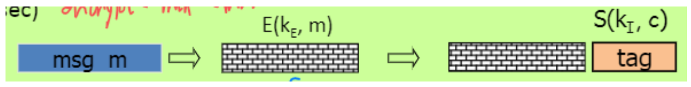


Figure 5.5: Encrypt-then-MAC

Encrypt-and-MAC

One example (SSH) of MAC-and-Encrypt is shown in Figure 5.6.

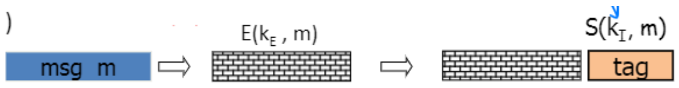


Figure 5.6: Encrypt-and-MAC

Some Standard

GCM, CCM, EAX.

All support AEAD (auth.enc.with associated data). All are nonce-based.

5.4.4 Security Analysis

Let (E,D) be CPA secure cipher and (S,V) secure MAC, then:

1. Encrypt-then-MAC always provides A.E.

2. MAC-then-Encrypt may be insecure against CCA attacks. However, wehn (E,D) is rand-CTR mode or rand-CBC, then it will be A.E.

5.4.5 Direct Construction from PRP: OCB

The OCB frame work is shown in Figure 5.7.

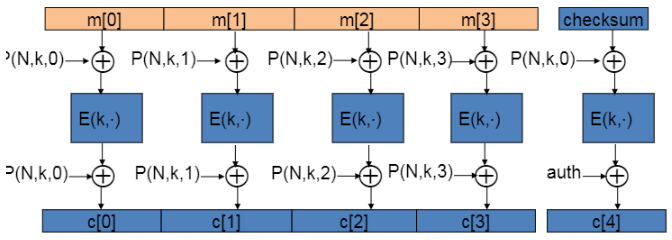


Figure 5.7: OCB

5.4.6 CBC padding attacks

Some lessons about CBC padding attacks:

- Encrypt-then-MAC would completely avoid this problem
- MAC-then-CBC provides A.E., but padding oracle destroys it

5.5 Summary

Starting from the property of active attacker, we explained why the MAC cannot defend active attacker with an example. Then based on the fact that an active attacker can always see the ciphertext, we define the definition of ciphertext integrity and CCA Security.

Then we propose that Authenticated Encryption, that should be able to meet integrity and authenticated and be Sem. Sec under CCA. Then we introduce 3 ways to build an AE from MAC: E then M, M then E, M and E. And the fact is E then M always meet AE. Besides, we also mentioned one method to build an AE from PRP, the OCB method.

Chapter 6

Other Thing of Symmetric Encryption

6.1 Key Derivation

6.1.1 Motivation

Typically, we can generate a single source key by:

1. Hardware Random Number Generator
2. A key exchange protocol

However, based on previous contents, there are a lot of time that we need many keys for security. So the motivation of the Key Derivation is to generate many keys from this one source key.

6.1.2 Case 1: Source Key is Uniform

Suppose Source key SK is uniform in K. We can define key derivation function (KDF) as:

$$\begin{aligned}&\text{KDF(SK, CTX, L)} := \\ &\quad F(SK,(CTX\|0))\|F(SK,(CTX\|1))\|\cdots\|F(SK,(CTX\|L))\end{aligned} \tag{6.1}$$

where CTX is a string that uniquely identifies the application. The purpose of CTX is we want get independent keys even if two apps has sampled the same SK.

6.1.3 Case 2: Source Key is not Uniform

When SK is not uniform, the PRF output may not look random. And the fact is the source key often not uniformly random.

Extract-then-Expand Paradigm

The Extract-then-Expand Paradigm has two steps:

step 1: extract pseudo-random key k from source key SK, it is shown in Figure 6.1. where **salt** is a fixed non-secret string chosen at random.

step 2: expand k by using it as a PRF keys as before.

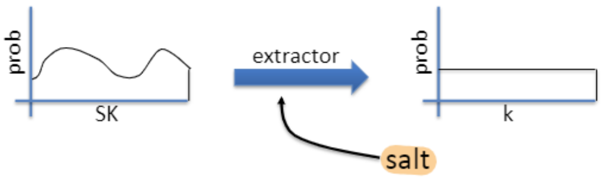


Figure 6.1: Extract-then-Expland Paradigm

HKDF: a KDF from HMAC

Implements the extract-Then-expand paradigm:

Extract by: k $\leftarrow HMAC(salt, SK)$ (salt is key, and SK is data)

Then expand using HMAC as a PRF with key k

6.2 Deterministic Encryption

Deterministic Encryption enables later lookup. However, attacker can tell when two ciphertexts encrypt the same message, which means we leak some information.

6.2.1 Deterministic CPA Security

The experiment of Deterministic CPA Security is shown in Figure 6.2.

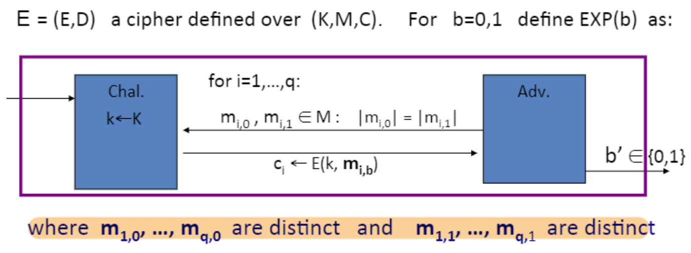

E = (E,D) a cipher defined over (K,M,C). For b=0,1 define EXP(b) as:
Chal.
k←K
for i=1,...,q:
$m_{i,0}$, $m_{i,1}$ ∈ M : $|m_{i,0}| = |m_{i,1}|$
$c_i \leftarrow E(k, m_{i,b})$
Adv.
b' ∈ {0,1}
where $m_{1,0}, \ldots, m_{q,0}$ are distinct and $m_{1,1}, \ldots, m_{q,1}$ are distinct

Figure 6.2: Deterministic CPA Security

Definition 6.2.1 (Sem Sec under det.CPA). *em Sec under det.CPA*

E is sem. sec. under det. CPA if for all efficient A :

$$\text{Adv}_{\text{dCPA}}[A, E] = |\Pr[\text{EXP}(0) = 1] - \Pr[\text{EXP}(1) = 1]| \quad \textit{is negligible.}$$

One should notice that: CBC with fixed IV is not det.CPA secure.

6.3 Tweakable Encryption

6.4 Format Preserving Encryption

Chapter 7

Basic Key Exchange

7.1 Trusted 3rd Parties

7.2

7.2.1 Merkle Puzzles

7.2.2 The Diffie-Hellman Protocol

7.2.3 Public-Key Encryption

7.3 Modular Arithmetic

7.4 Easy and Hard Problems