

CSCI 1515 Applied Cryptography

This Lecture:

- Practical Constructions of Block Cipher
- Practical Constructions of Hash Functions
- Block Cipher Modes of Operation

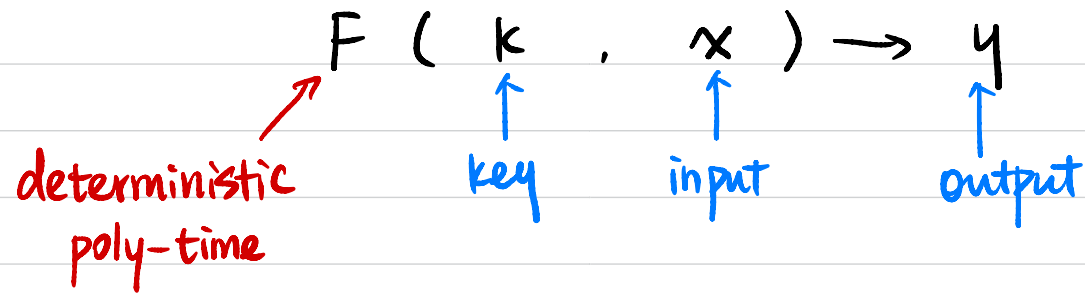
Summary

	Symmetric-Key	Public-Key
Message Secrecy	Primitive: SKE Construction: block cipher	Primitive: PKE Constructions: RSA / ElGamal
Message Integrity	Primitive: MAC Constructions: CBC-MAC / HMAC	Primitive: Signature Constructions: RSA / DSA
Secrecy & Integrity	Primitive: AE Construction: Encrypt-then-MAC	
Key Exchange		Construction: Diffie-Hellman
Important Tool	Primitive: Hash function Construction: SHA	

```
graph TD; BC[block cipher] -- red arrow --> ETM[Encrypt-then-MAC]; SHA[SHA] -- blue arrow --> HMAC[HMAC];
```

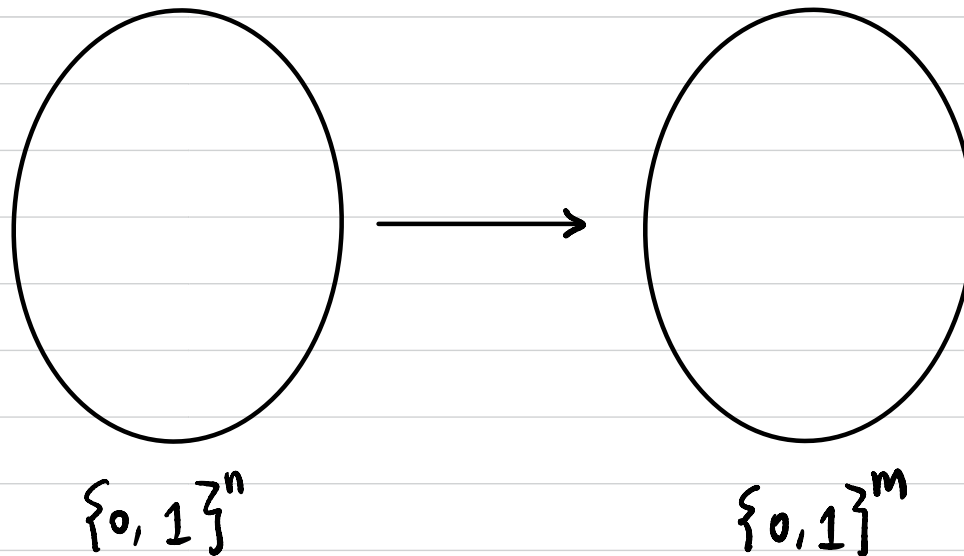
Pseudorandom Function (PRF)

Keyed Function $F: \{0,1\}^k \times \{0,1\}^n \rightarrow \{0,1\}^m$



$k \leftarrow \{0,1\}^k$

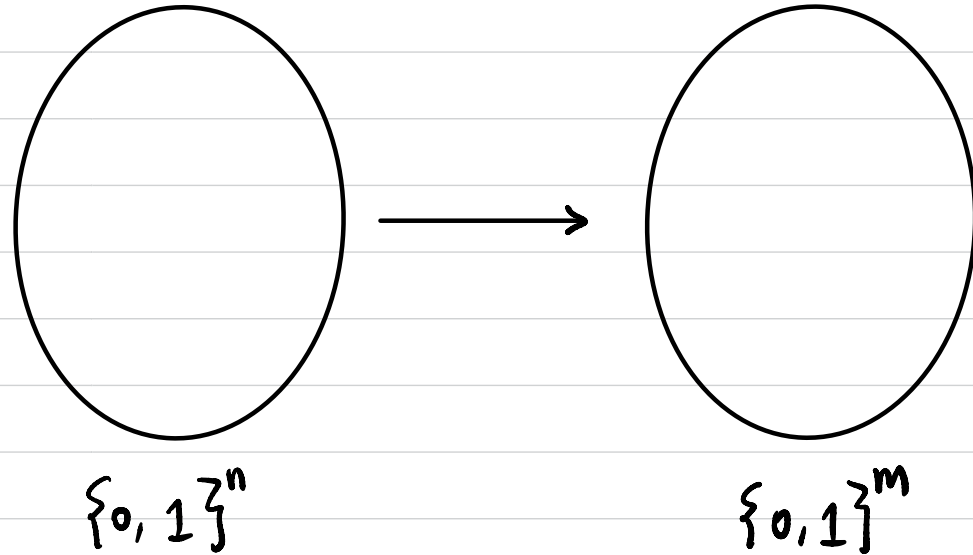
$F_k:$



"looks like a random function"

Pseudorandom Function (PRF)

$$k \leftarrow \{0, 1\}^\lambda \quad F_k:$$

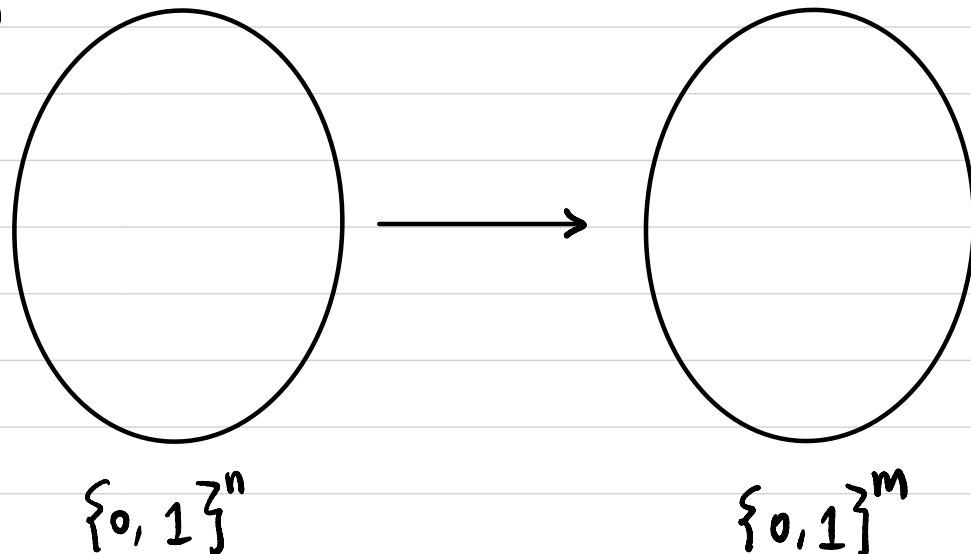


How many possible F_k 's?
 2^λ

$$f \leftarrow \{F \mid F: \{0, 1\}^n \rightarrow \{0, 1\}^m\}$$

$$(2^m)^{2^n} = 2^{m \cdot 2^n} \quad f:$$

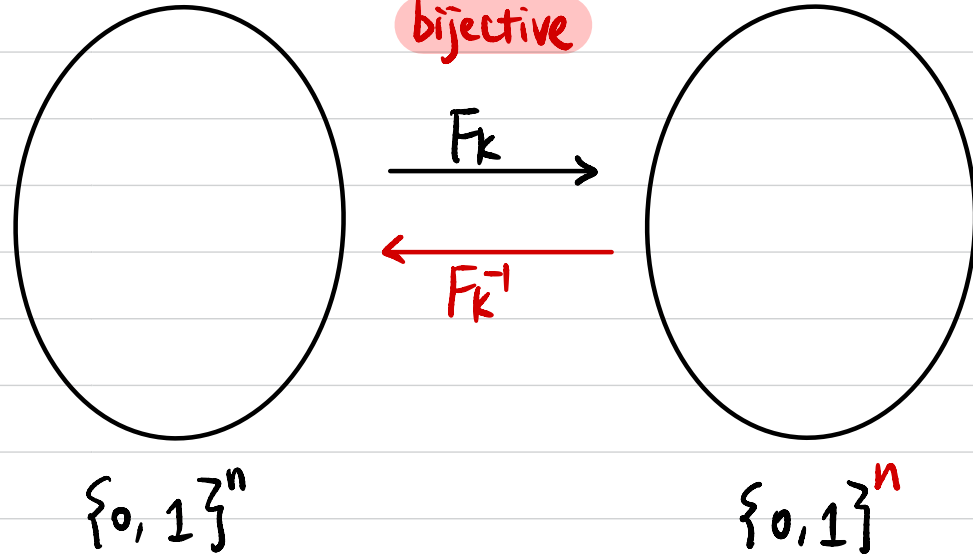
$\sum \sum c$ (not knowing k)



How many possible f 's?

Pseudorandom Permutation (PRP)

$$k \leftarrow \{0, 1\}^n \quad F_k:$$

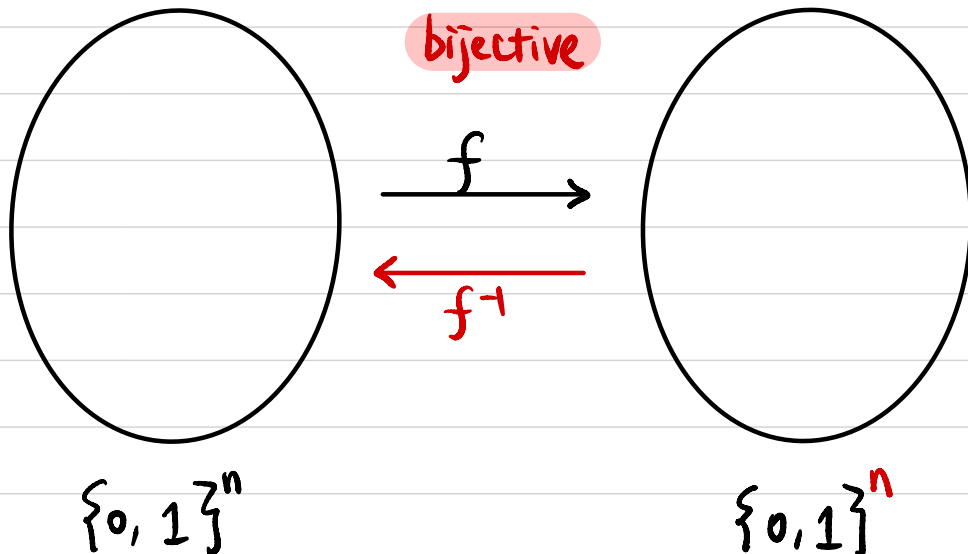


How many possible F_k 's ?
 2^n

$$f \leftarrow \{ F \mid F: \{0, 1\}^n \rightarrow \{0, 1\}^n, \\ F \text{ is bijective} \}$$

$$2^n! \quad f:$$

$\sum \sum^c$ (not knowing k)



How many possible f 's ?

Block Cipher

$$F: \{0,1\}^\lambda \times \{0,1\}^n \rightarrow \{0,1\}^n$$

λ : key length

n : block length

$F_k(\cdot)$: permutation / bijective $\{0,1\}^n \rightarrow \{0,1\}^n$

$F_k^{-1}(\cdot)$: efficiently computable given k .

It is assumed to be a pseudorandom permutation (PRP).

Construction: Advanced Encryption Standard (AES)

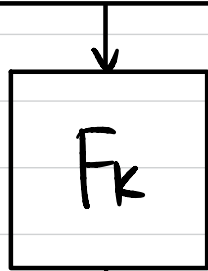
- $\lambda = 128/192/256$, $n = 128$

Before AES: Data Encryption Standard (DES)

- $\lambda = 56$, $n = 64$

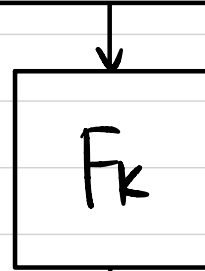
Substitution-Permutation Network (SPN)

$X_1 =$ 1001101011



0110100110

$X_2 =$ 0001101011

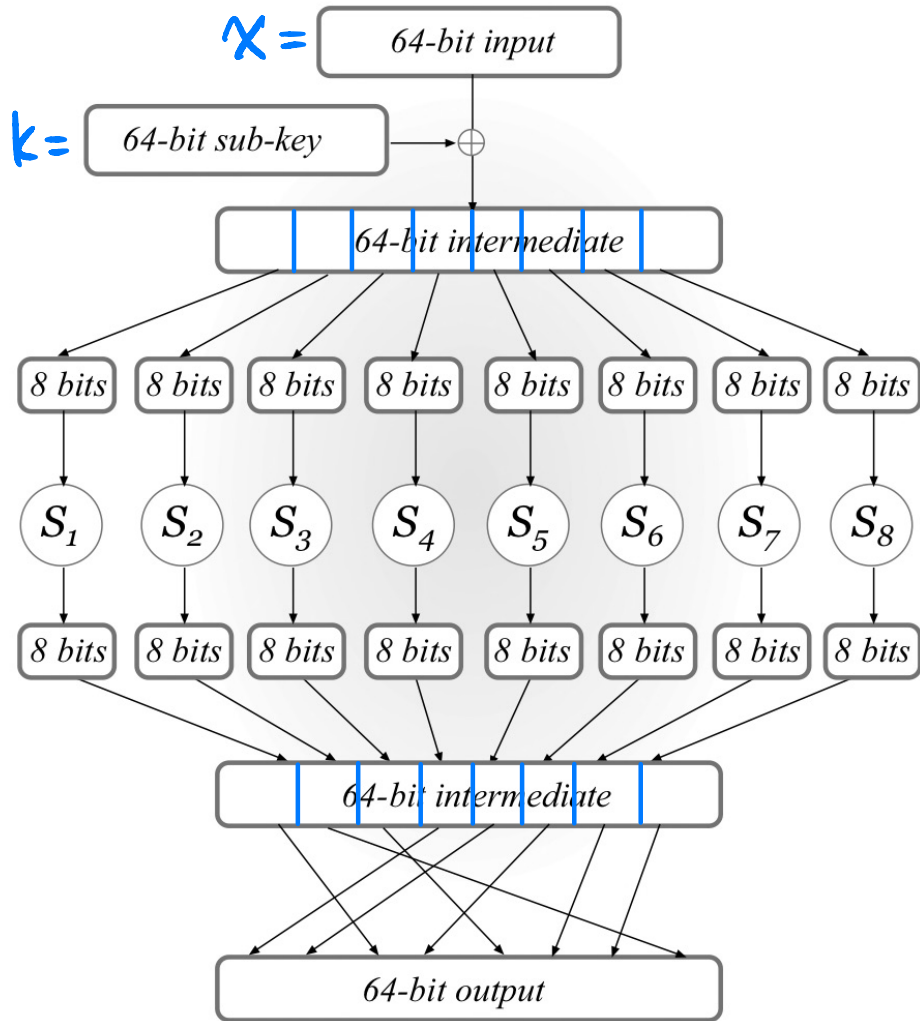


1100101101

Design Principle: "Avalanche Effect"

A one-bit change in the input should "affect" every bit of the output.

Substitution-Permutation Network (SPN)



A single round of SPN

"Confusion-Diffusion Paradigm"

Step 1: Key Mixing

$$X := X \oplus k$$

Step 2: Substitution (Confusion Step)

$$S_i: \{0,1\}^8 \rightarrow \{0,1\}^8 \quad (\text{S-box})$$

Public permutation/one-to-one map

1-bit change of input

→ at least 2-bit change of output

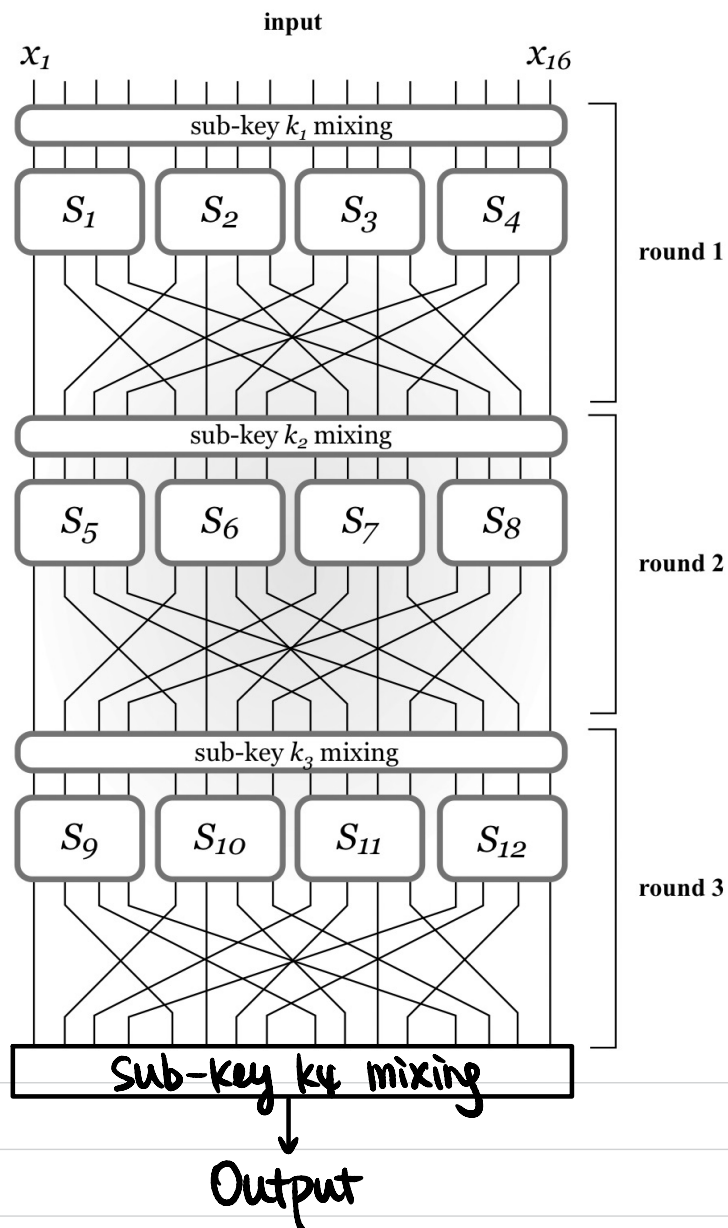
Step 3: Permutation (Diffusion Step)

$$P: [64] \rightarrow [64]$$

Public mixing permutation

↓
affect input to multiple S-boxes next round

Substitution-Permutation Network (SPN)



3-round SPN:

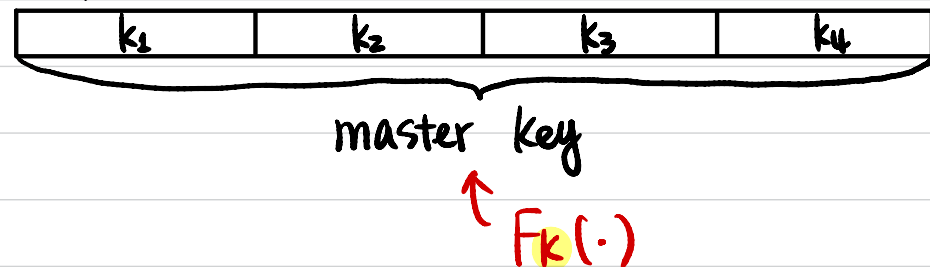
3-round { key mixing
substitution
permutation

1 final-round key mixing

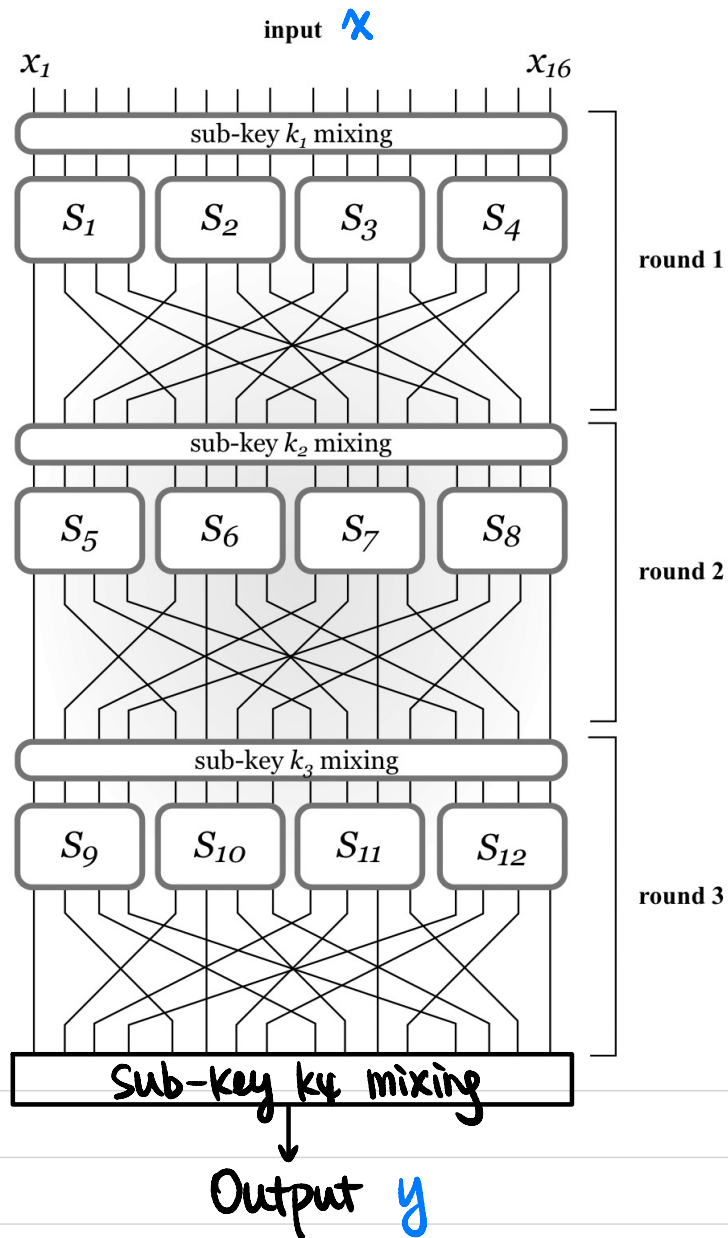
Key Schedule:

How we derive sub-keys from master key.

Example:



Substitution-Permutation Network (SPN)

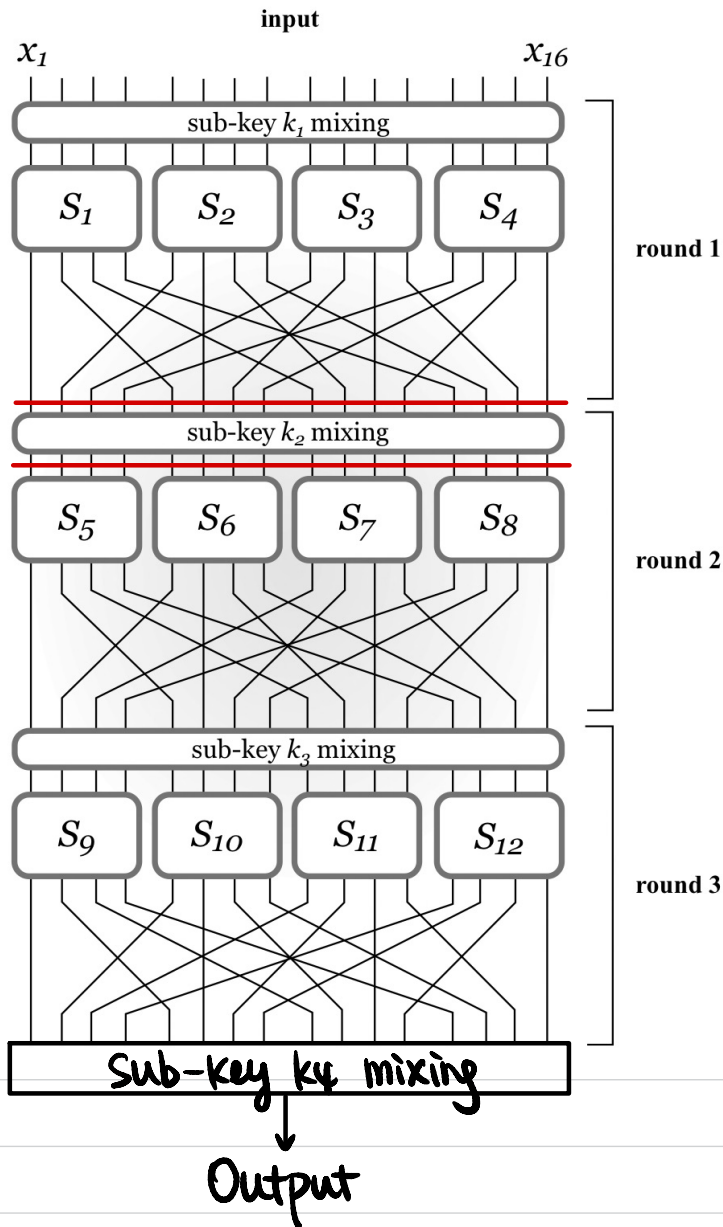


An SPN is invertible given the master key.

↓
permutation

How to compute $F_k^{-1}(y)$?

Attacks on Reduced-Round SPN



1-round SPN without final key mixing?

Given $(x, y) \Rightarrow$ recover key?

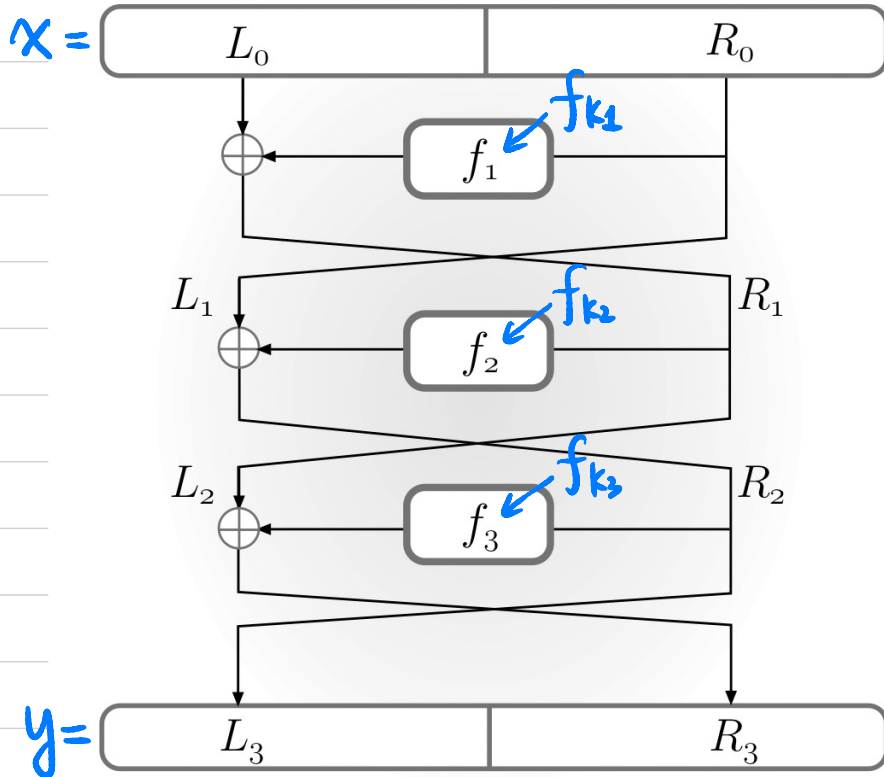
1-round SPN with final key mixing?

Given $(x, y), (x', y') \Rightarrow$ recover key?

Why do we need a final key mixing step?

Can we do r -round key mixing, then r -round substitution, then r -round permutation?

Feistel Network



3-round Feistel Network

$f_{ki}: \{0,1\}^{n/2} \rightarrow \{0,1\}^{n/2}$
↑
round function

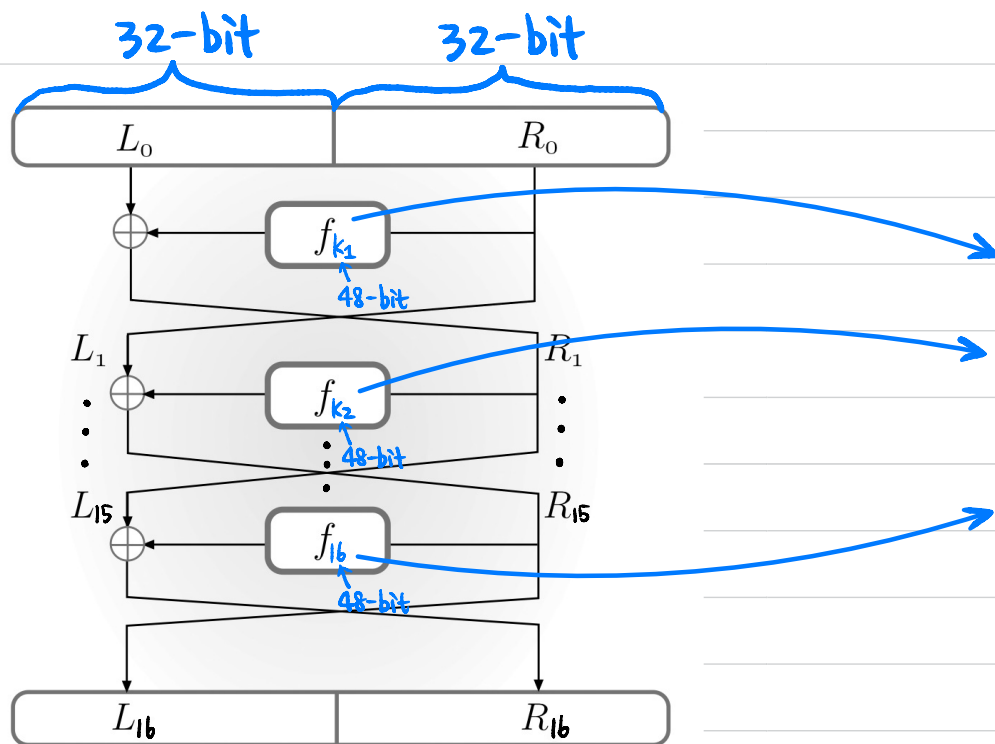
How to compute $F_k^{-1}(y)$?

Attacks on reduced-round Feistel Network

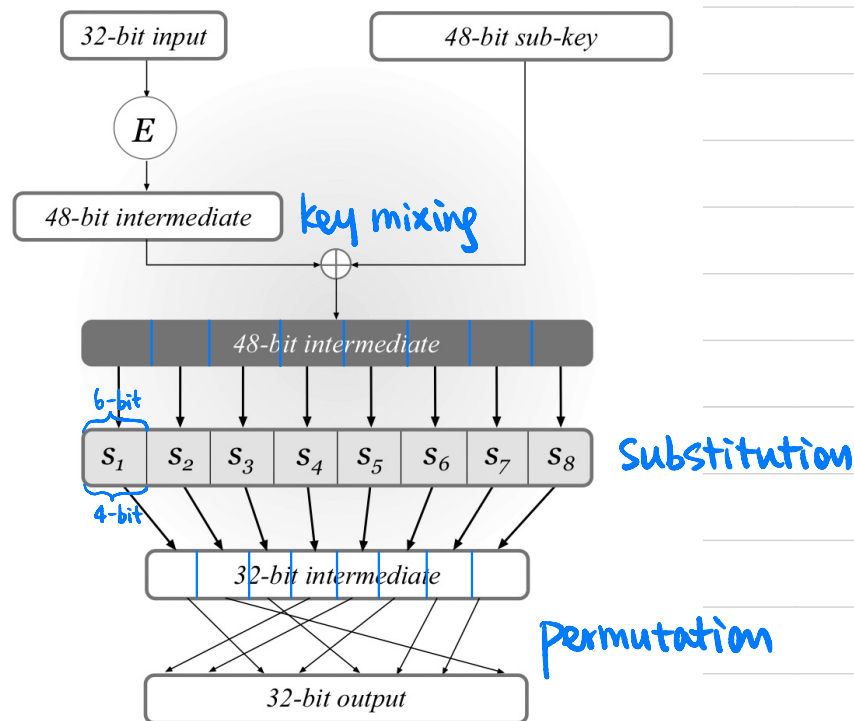
Data Encryption Standard (DES)

$F: \{0, 1\}^{\lambda} \times \{0, 1\}^n \rightarrow \{0, 1\}^n$
 block length $n=64$
 master key length $\lambda=56$

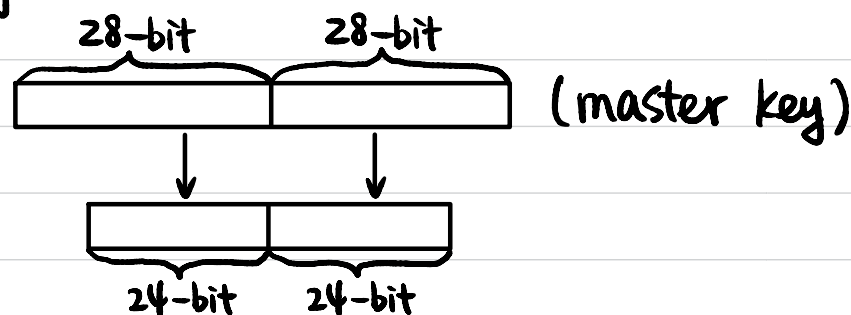
16-round Feistel Network



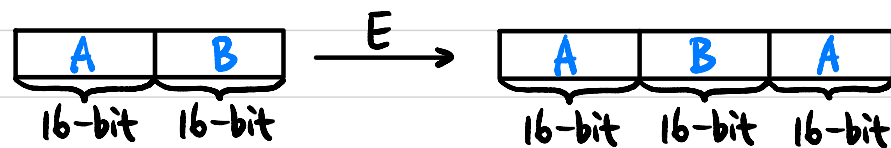
DES mangler function



Key Schedule:

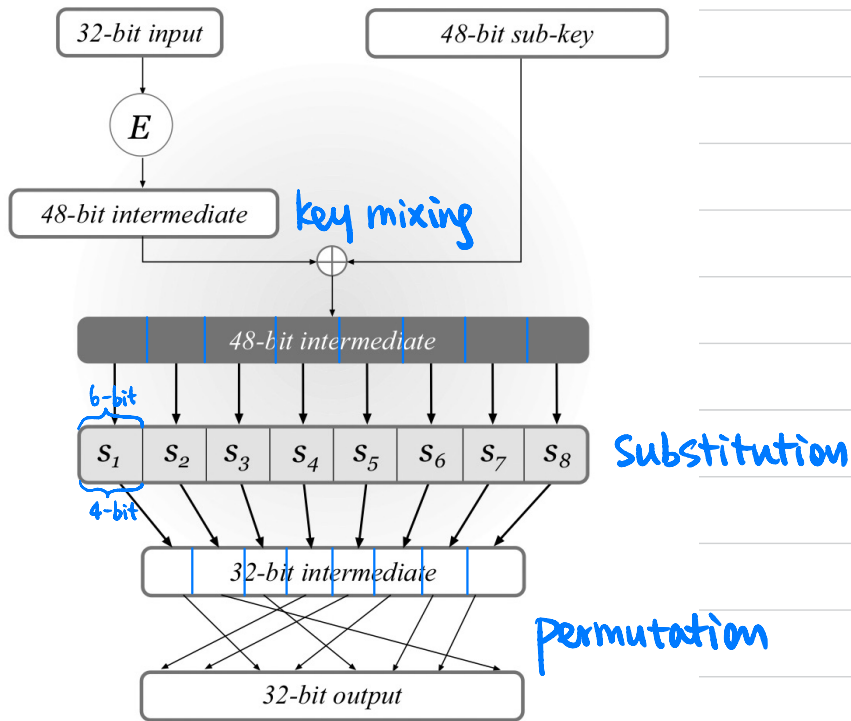


E: expansion function



Data Encryption Standard (DES)

DES mangler function



S-box: $\{0,1\}^6 \rightarrow \{0,1\}^4$

① "4-to-1":

Exactly 4 inputs map to same output

② 1-bit change of input

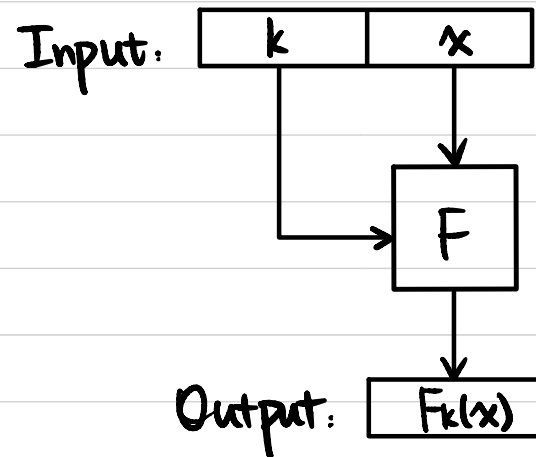
→ at least 2-bit change of output

Mixing Permutation: $[32] \rightarrow [32]$

4 bits from each S-box will affect the input to 6 S-boxes in the next round

Hash Function from Block Cipher

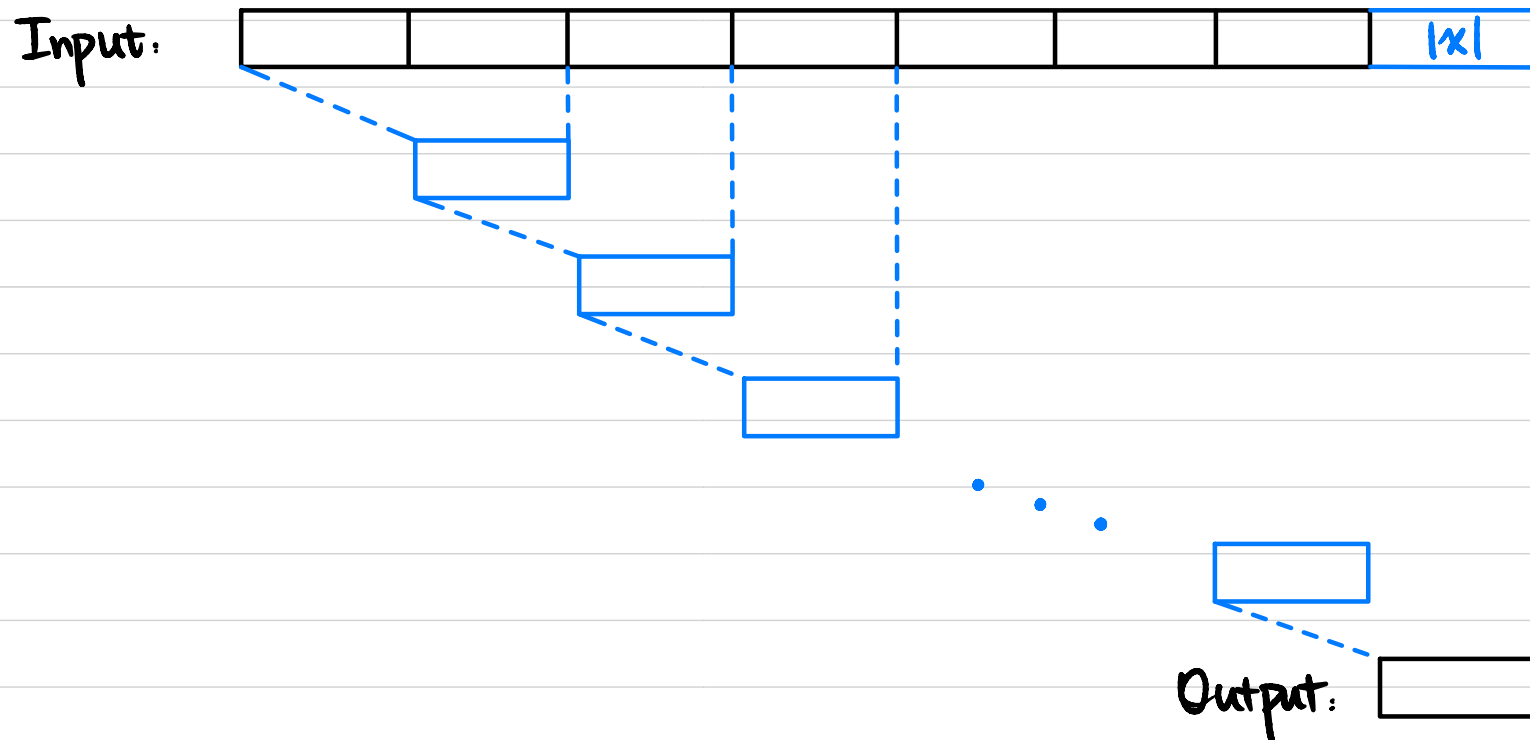
Block Cipher $\xRightarrow{\text{Davies-Meyer}}$ Compression Function



If F is model as an "ideal cipher", then Davies-Meyer Construction is Collision-resistant.

Hash Function from Block Cipher

Compression Function $\xrightarrow{\text{Merkle-Damgård}}$ Arbitrary-length Hash Function
(fixed-length hash function)



If the compression function is a collision-resistant hash function for fixed-length inputs, then Merkle-Damgård is a collision-resistant hash function for arbitrary-length inputs.

Block Cipher Modes of Operation

Given: $F: \{0,1\}^\lambda \times \{0,1\}^\lambda \rightarrow \{0,1\}^\lambda$ ¹²⁸

Assumed to be a pseudorandom permutation (PRP).

Goal: Construct a symmetric-key encryption (SKE) scheme for arbitrary-length messages with chosen-plaintext Attack (CPA) security.

- $k \xleftarrow{\$} \{0,1\}^\lambda$
- $\text{Enc}_k(m)$ $|m| = \alpha \cdot \lambda$ (If not, pad it to a multiple of λ)
- $\text{Dec}_k(c)$

Chosen-Plaintext Attack (CPA) Security

$k \leftarrow \text{Gen}(1^\lambda)$

Alice



$c_0 \leftarrow \text{Enc}_k(m_0)$

$c_1 \leftarrow \text{Enc}_k(m_1)$

$c_2 \leftarrow \text{Enc}_k(m_2)$

$\vdots$

$m_0, m_1 \in \{0, 1\}^n$

$b \in \{0, 1\}$

$c \leftarrow \text{Enc}_k(m_b)$

③ choose



(PPT)

② see

④ see

⑤ Output b'

Bob



$m_0 := \text{Dec}_k(c_0)$

$m_1 := \text{Dec}_k(c_1)$

$m_2 := \text{Dec}_k(c_2)$

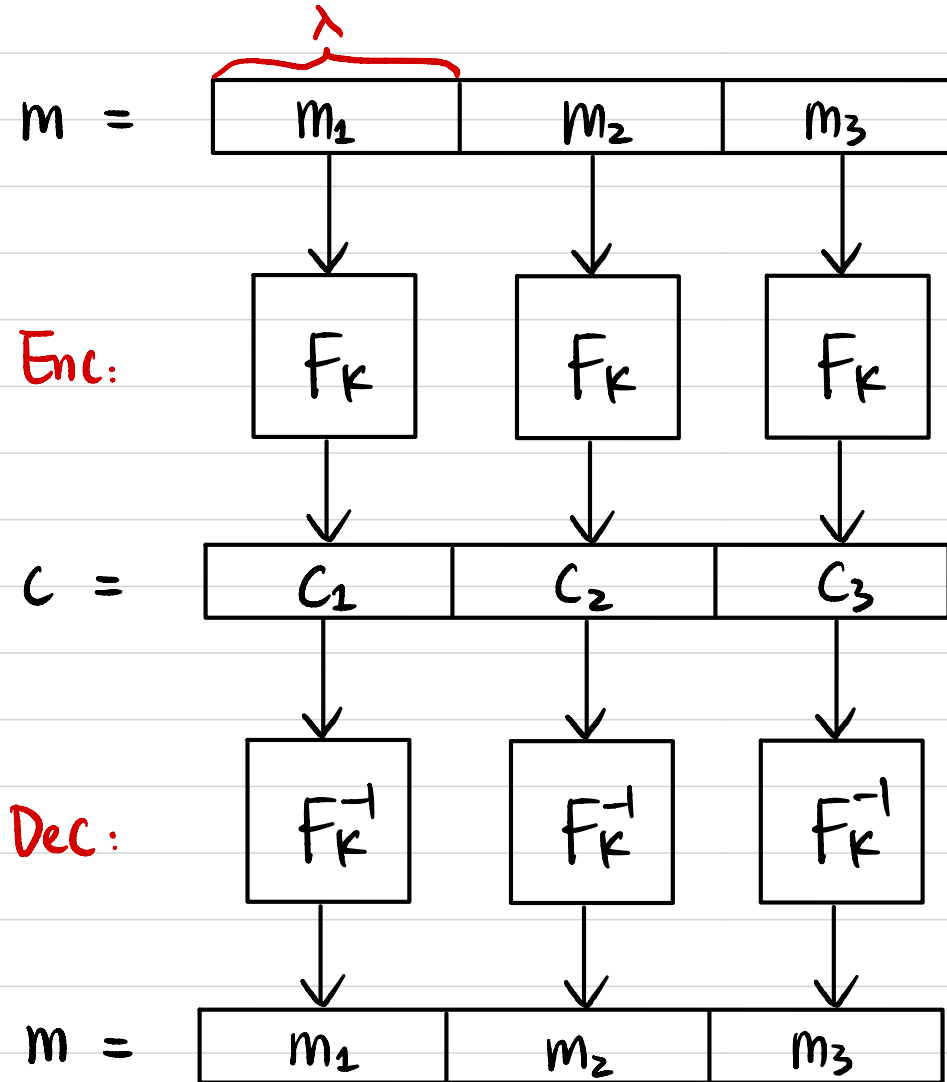
$\vdots$

$c_0, c_1, c_2, \dots$

c

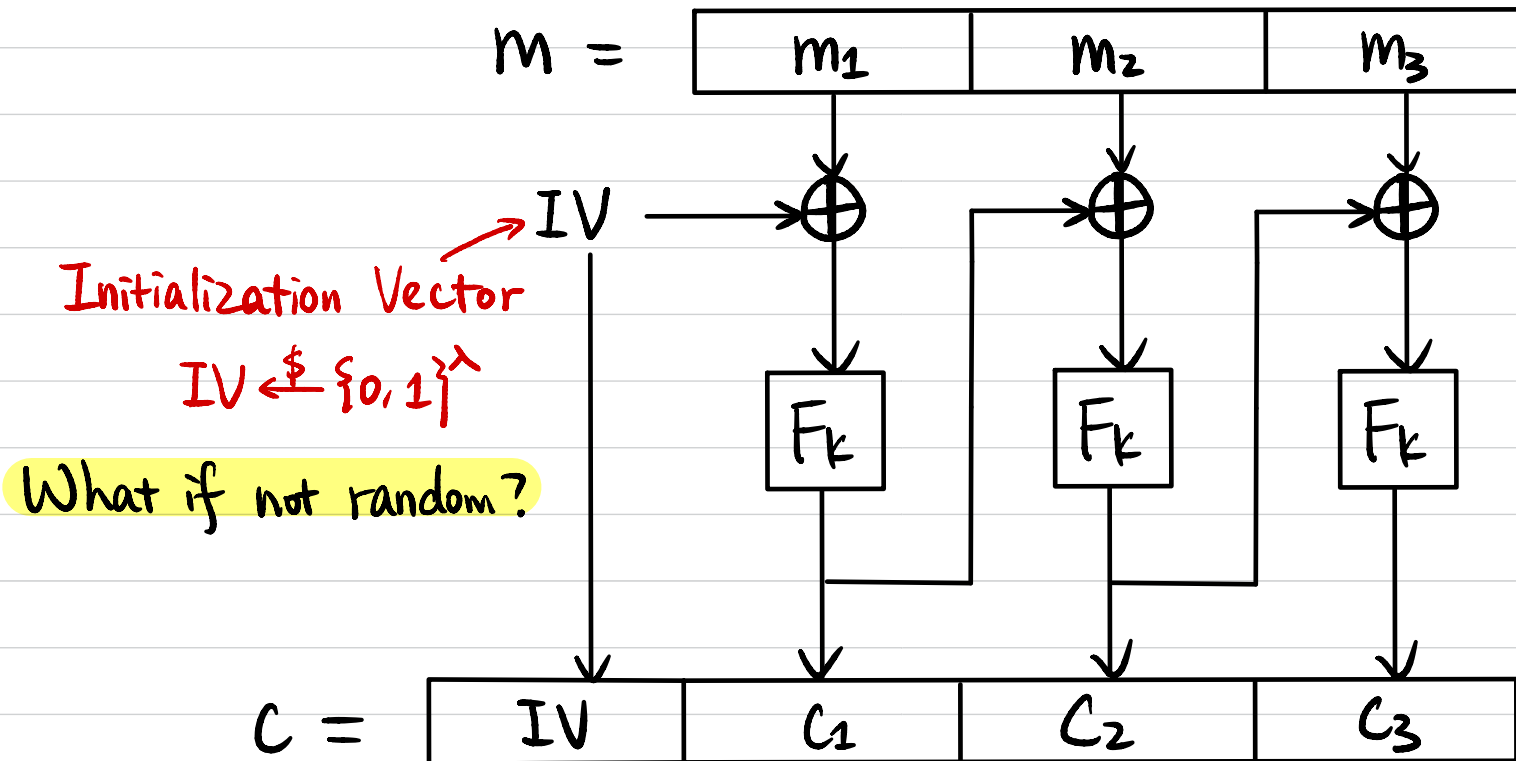
2^{-128}
 $\Pr[b = b'] \leq \frac{1}{2} + \text{negligible}$

Electronic Code Book (ECB) Mode



CPA Secure?

Cipher Block Chaining (CBC) Mode

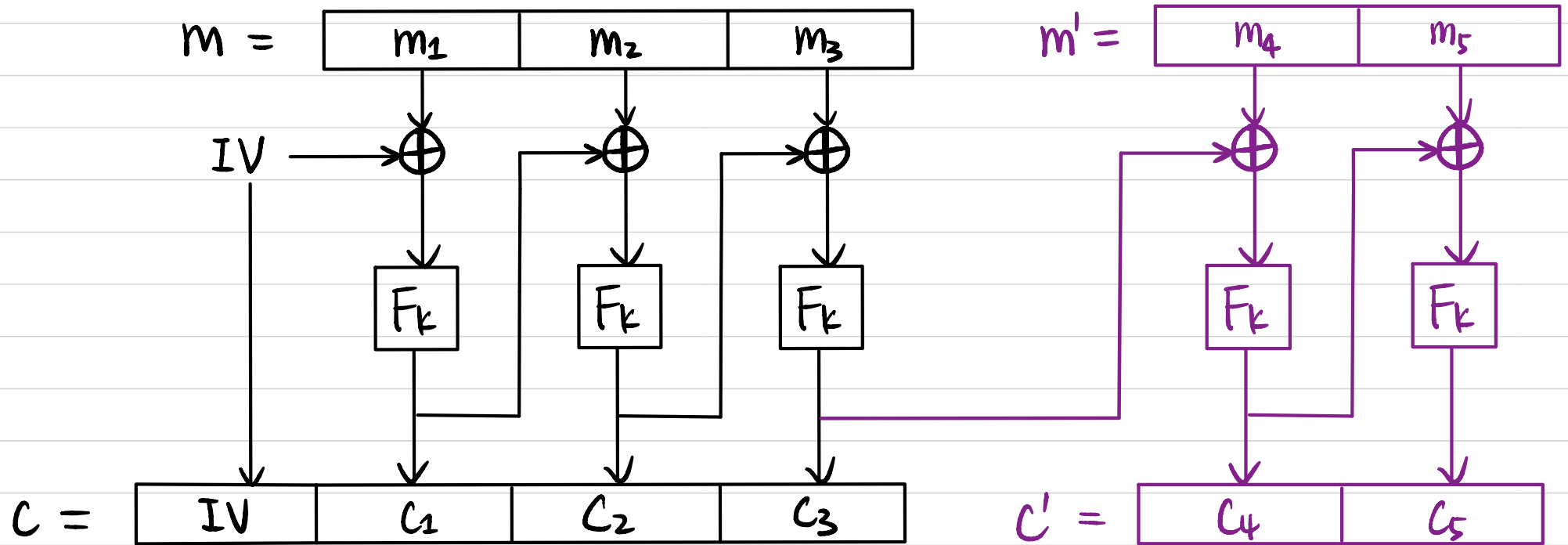


How to decrypt?

CPA Secure?

Can we parallelize the computation?

Chained Cipher Block Chaining (CBC) Mode

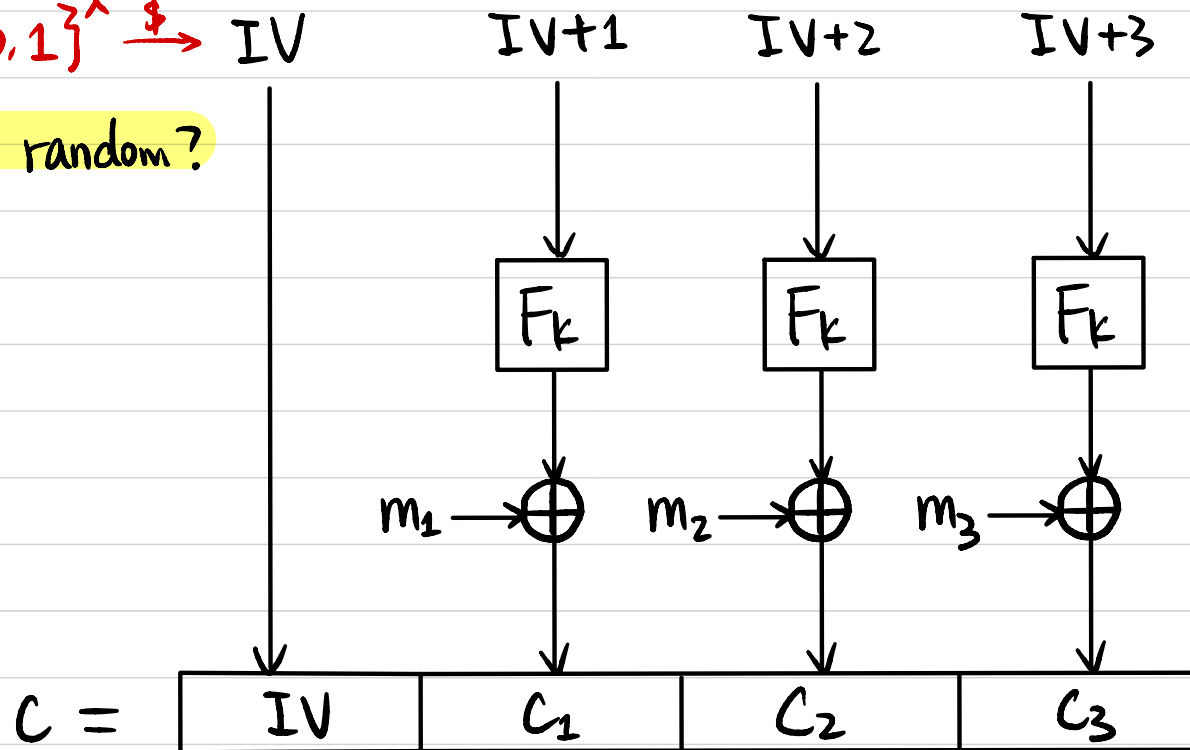


CPA Secure?

Counter (CTR) Mode

$\{0,1\}^n \xrightarrow{\$}$

What if not random?



How to decrypt?

CPA Secure?

"Stateful" CTR Mode?

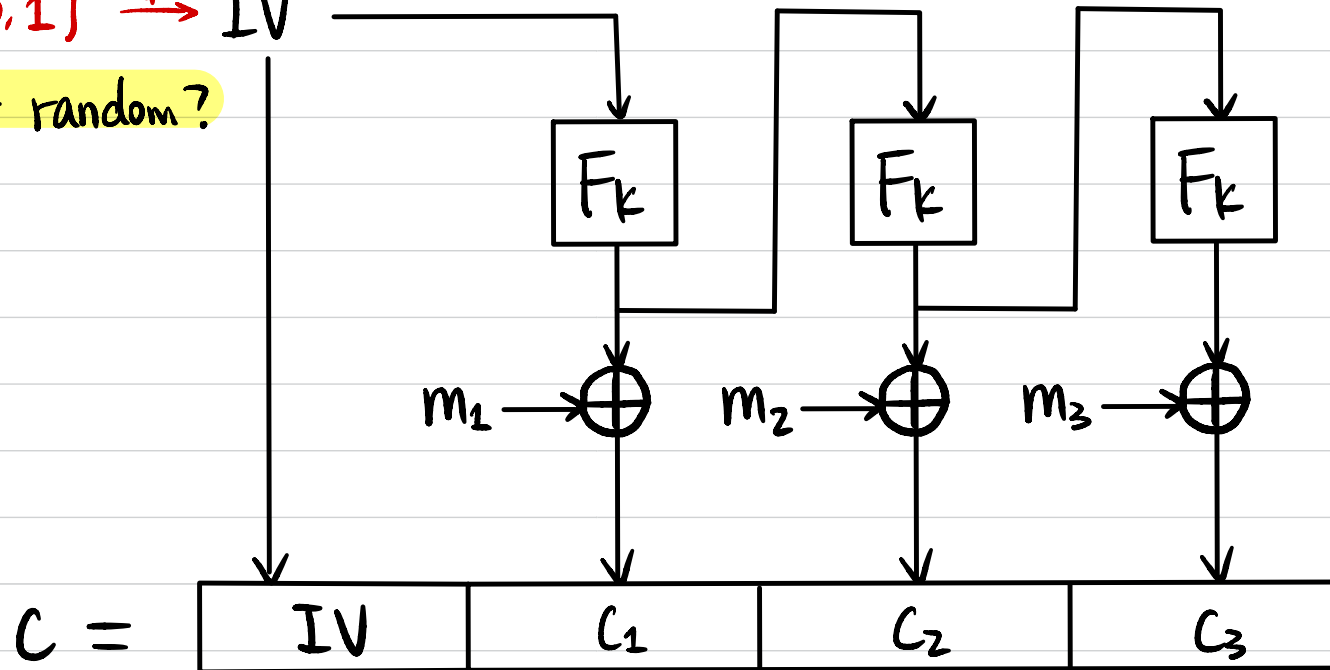
Can we parallelize the computation?

PRG from PRF

Output Feedback (OFB) Mode

$\{0,1\}^n \xrightarrow{\$} \text{IV}$

What if not random?



How to decrypt?

CPA Secure?

"Stateful" OFB Mode?

Can we parallelize the computation?

PRG from PRF