Security Definition

SEMANTIC SECURITY

How to evaluate cipher’s security?

Choose an attacker model (attacker’s abilities)

- He can obtain the ciphertext
- Ciphertext-only attacks (COA)

The cipher is *“secure”* if

- Attacker cannot recover *secret key*
  - *Ciphertext does not reveal information about the key*
- Attacker cannot recover the *plaintext*
  - Ciphertext does not reveal information about the plaintext

How to evaluate cipher’s security?

Choose an attacker model (attacker’s abilities)

- He can obtain the ciphertext
- Ciphertext-only attacks (COA)

The cipher is *“secure”* if

- Attacker cannot recover ***secret key***
  - *Ciphertext does not reveal information about the key*

$$E(k, m) = m$$

- Attacker cannot recover the ***plaintext***
  - Ciphertext does not reveal information about the plaintext

$$E(k, m_0 || m_1) = m_0 || m_1 \oplus k$$

Which is our security goal?

m

Adversary knows that

m := { *"I love you"* with prob. **0.5** ; *"I don't love you"* with prob. **0.5** }

---

m

$c \leftarrow E(k, m)$

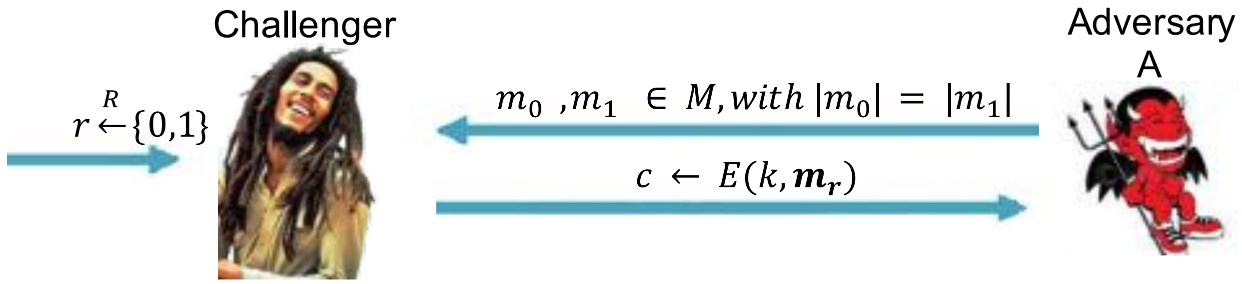
Adversary **still** knows that

m := { *"I love you"* with prob. **0.5** ; *"I don't love you"* with prob. **0.5** }

Adversary Advantage

Define encryption of messages as experiments

- $Exp(0) \rightarrow encrypt\ m_0$
- $Exp(1) \rightarrow encrypt\ m_1$



- Define event: $W_r\ = \{Exp(r)\ =\ 1\}$
- Define advantage: $Adv[A, E]\ = |P\{W_0\} - \ P\{W_1\}|$
  - $Adv = 1 \rightarrow$ *Adversary <u>distinguish</u>* $r = 0$ *and* $r = 1$
  - $Adv = 0 \rightarrow$ *Adversary <u>cannot distinguish</u>* $r = 0$ *and* $r = 1$

Semantic Security

Encryption algorithm E is semantically secure if

$$Adv[A, E] < \varepsilon \rightarrow \text{is negligible}$$

- For all *efficient* algorithm A
- For all explicit $m_0, m_1 \in M$ s.t.

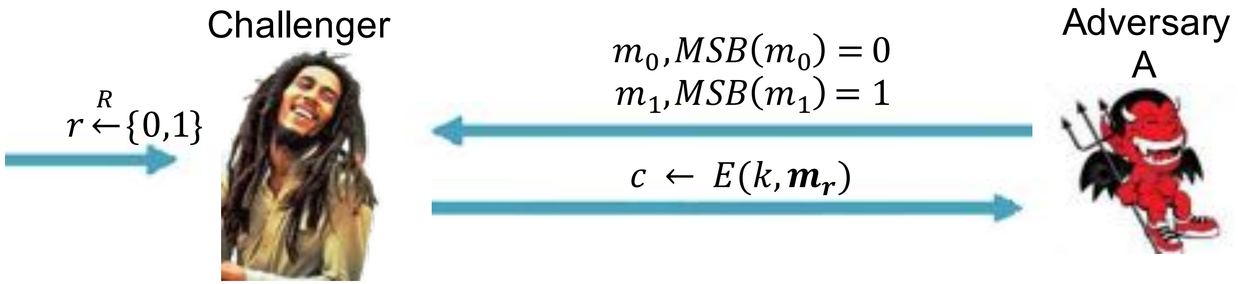
$$P\{E(k, m_0) = c\} = P\{E(k, m_1) = c\}$$

Cannot distinguish encryption of different messages

Example

Suppose the adversary has algorithm A

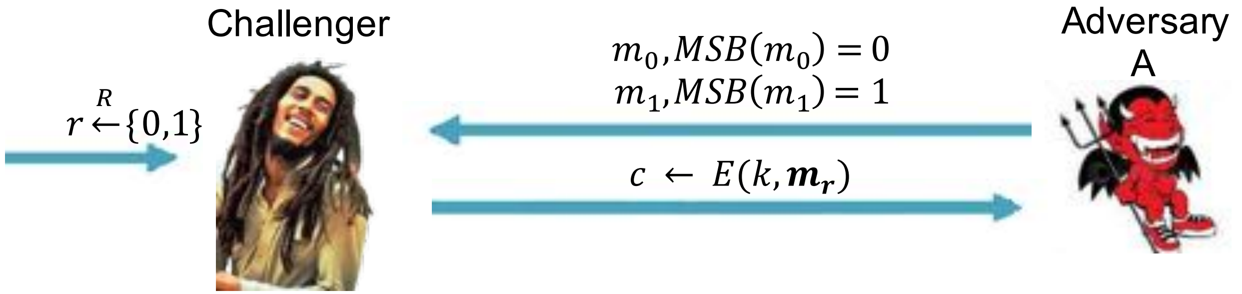
- $Exp(0) \rightarrow 0$
- $Exp(1) \rightarrow 1$



Example

Suppose the adversary has algorithm A

- Can deduce MSB of PT…having CT

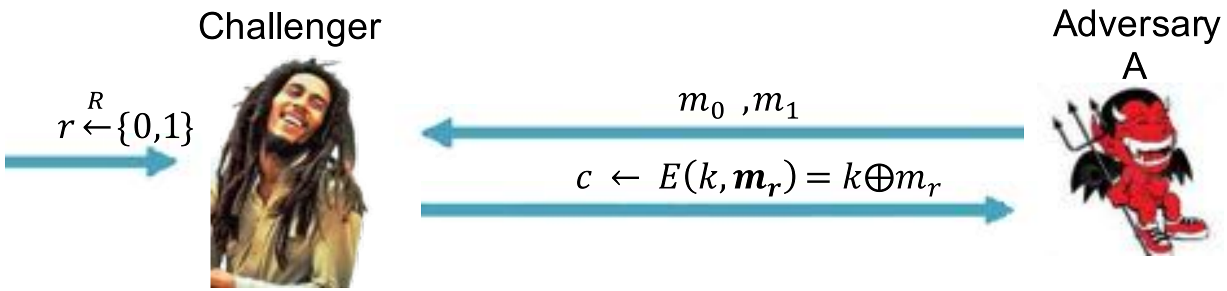


- $W_r \ = \{Exp(r) = \ 1\}$
- $Adv[A, E] = |0 - 1| = 1$
  - $Adv = 1 \rightarrow$ *Adversary* *__distinguish__* $r = 0$ *and* $r = 1$
  - $Adv = 0 \rightarrow$ *Adversary* *__cannot distinguish__* $r = 0$ *and* $r = 1$

Example: One-Time Pad

Take the of course secure OTP

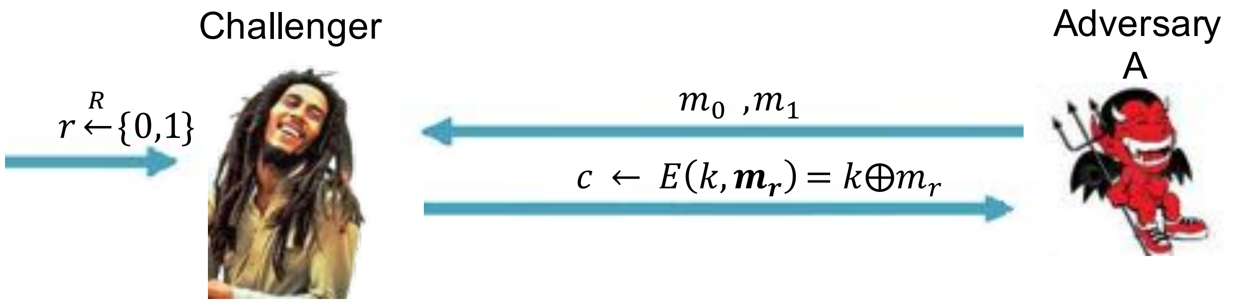
- Another way to prove his security…semantic!



Example: One-Time Pad

Take the of course secure OTP

- Another way to prove his security…semantic!

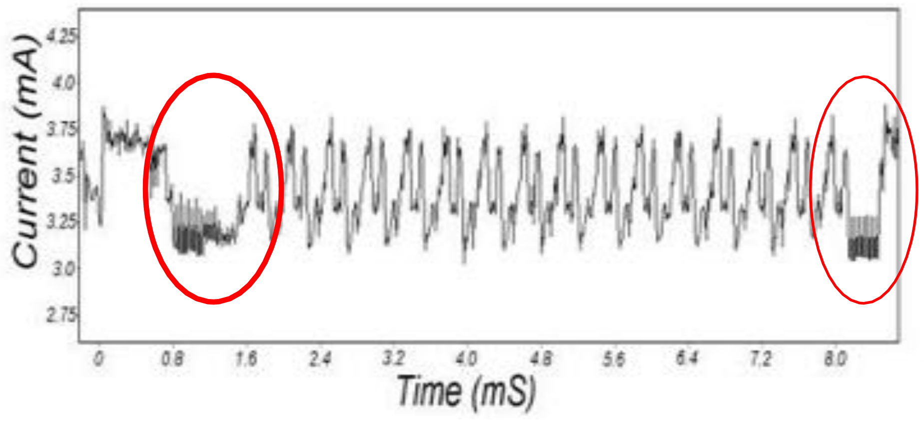


- $Wr = \{Exp(r) = 1\} = \{A[k \oplus m_r] = 1\}$
- $Adv[A, E] = |P\{A[k \oplus m_0] = 1\} - P\{A[k \oplus m_1] = 1\}| = 0$
  - $Adv = 1 \rightarrow$ *Adversary <u>distinguish</u>* $r = 0$ *and* $r = 1$
  - $Adv = 0 \rightarrow$ *Adversary <u>cannot distinguish</u>* $r = 0$ *and* $r = 1$

Attacks on the implementations

Attacker wants to distinguish operations

- Side channel attacks
  - Timing attack
  - Power monitoring
  - Electromagnetic monitoring
  - Acoustic attack
- Fault attacks
  - Induce errors in computation or memory



Implementation accuracy foundamental!

Attacks on the implementations

Example: Acoustic cryptanalysis, Crypto 2014

Computers emit noise due to vibration of their components

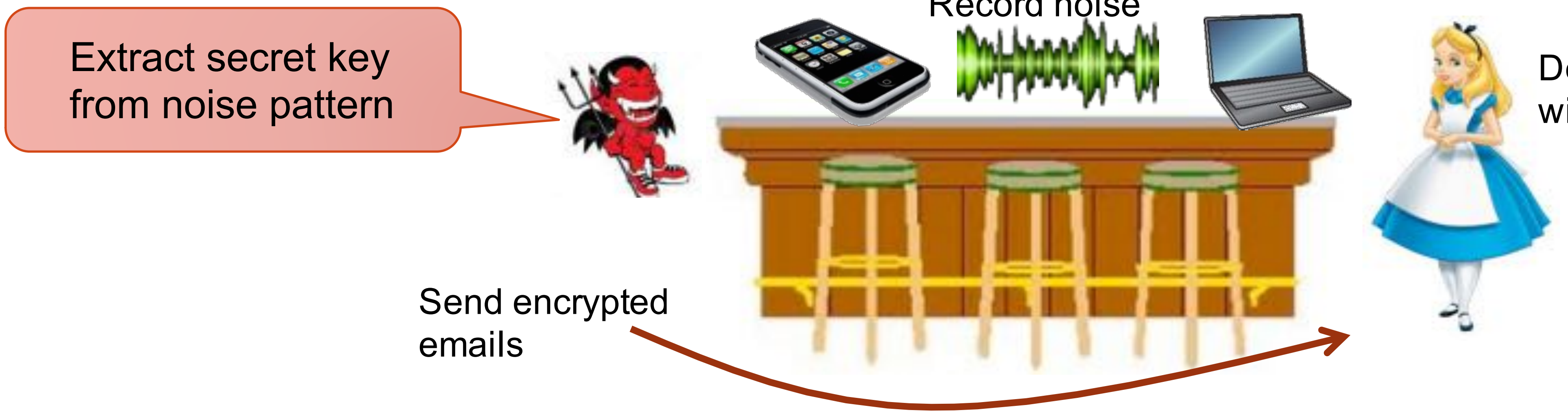
If computer computes with secret key, then noise pattern depends on key ➔ extract key

Attacks on the implementations

Example: Acoustic cryptanalysis, Crypto 2014

Computers emit noise due to vibration of their components

If computer computes with secret key, then noise pattern depends on key ➔ extract key



Model does not cover all real world attacks!


Reality
Model does not cover all real world attacks!

Symmetric Ciphers

BLOCK CIPHERS

Symmetric Cipher

A symmetric cipher is defined as

- $E(\cdot,\cdot)$ → *Encryption Algorithm*
- $D(\cdot,\cdot)$ → *Encryption Algorithm*
- K → *Secret Key*

We have two types of messages

- M → *Plaintext (original message)*
- CT → *Ciphertext (encrypted message)*

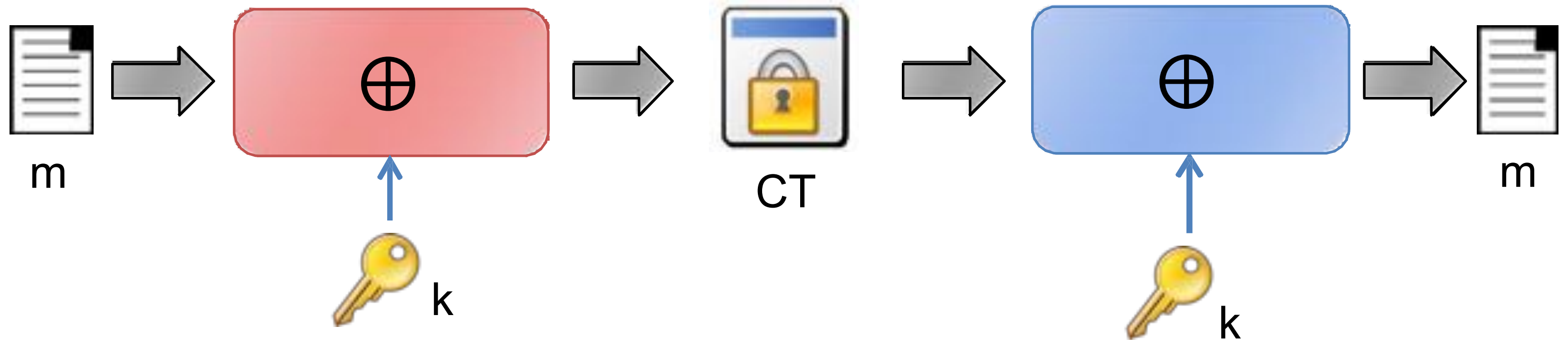
Common key and common cipher!

One Time Pad

Perfect secrecy but not easy to apply

- Truly random key
- Same key and plaintext size
- Different keys for different encryptions

- $CT = E(K, M) = K \oplus M$
- $D(K, CT) = K \oplus CT$

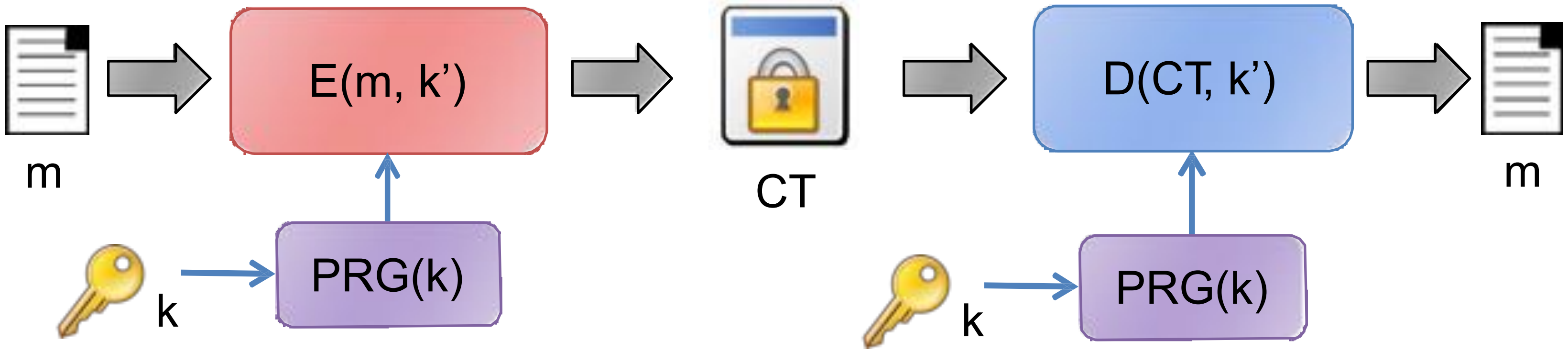


Stream Cipher

Approximating OTP

- Replace random key with pseudo-random
- Exploits PRG to replace the key
- One truly random key used as seed

- $CT = E(K, M) = PRG(K) \oplus M$
- $D(K, CT) = PRG(K) \oplus CT$



Properties of Good Ciphers

***Confusion** and **diffusion** are two properties of the operation of a secure cipher which were identified by **Shannon** in 1949.*

Confusion refers to making the relationship between the key and the ciphertext as complex as possible

- Substitution is one of the mechanism for primarily confusion

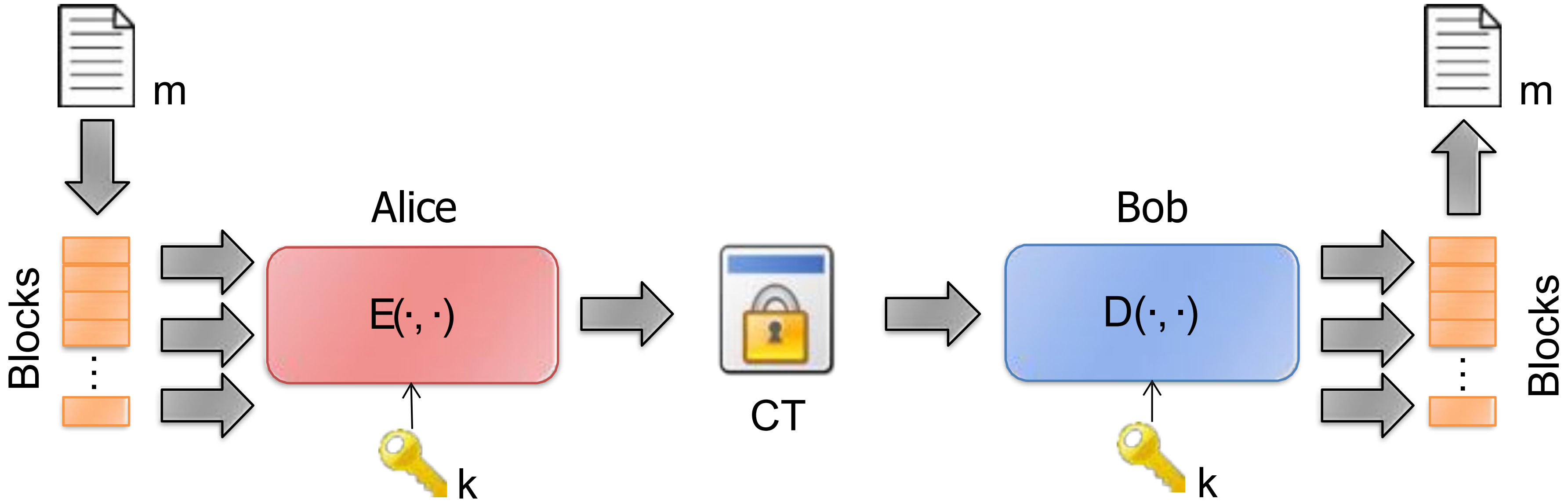
Diffusion refers to the property that redundancy in the statistics of the plaintext is "dissipated" in the statistics of the ciphertext

- Transposition (Permutation) is a technique for diffusion

Block Ciphers

Mostly based on a Feistel Cipher Structure

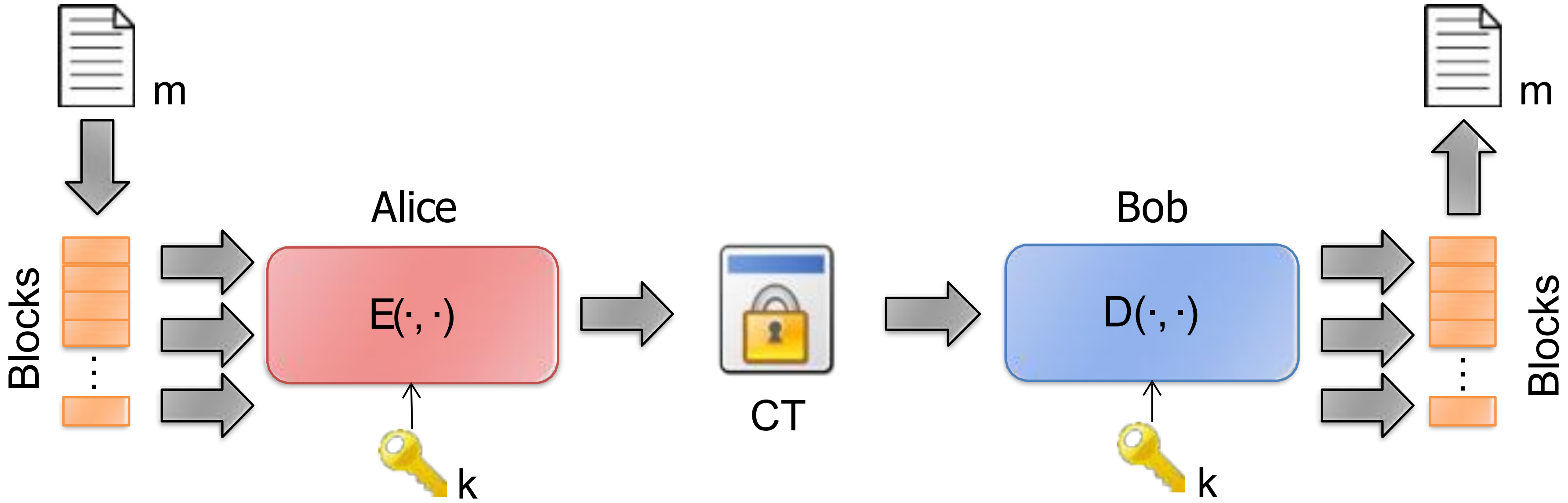
- Takes one block (plaintext) and transform it into a block of the same length using a the provided secret key
- Decrypt by applying the reverse transformation to the ciphertext block using the same secret key
- Encrypt/Decrypt blocks of data of fixed length (e.g. 64bits, 128bits, etc…)



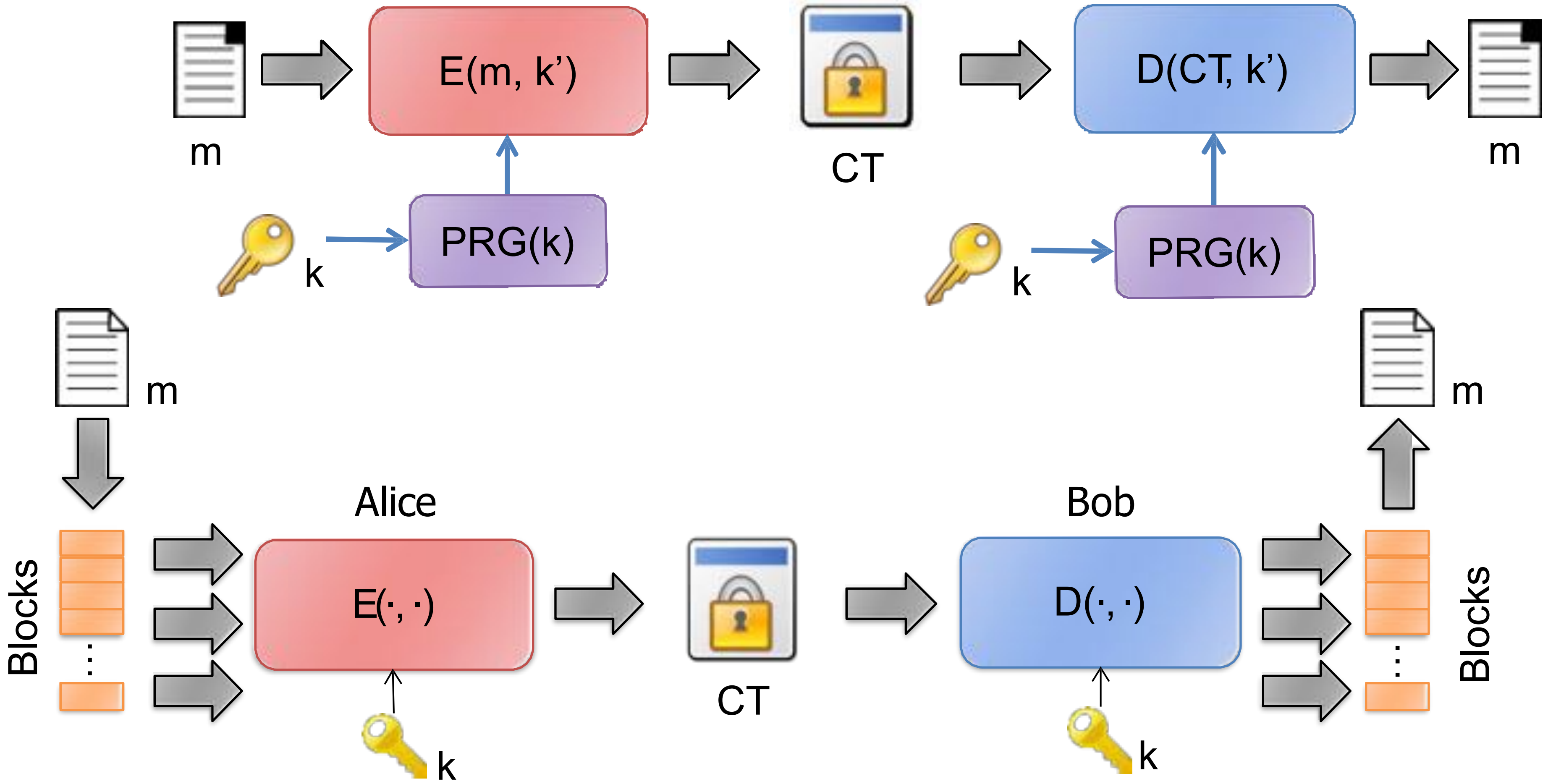
Block Ciphers

Fixed key and block length

- **DES**: $n = 64\ bits,\ k = 56\ bits$
- **3-DES**: $n = 64\ bits,\ k = 168\ bits$
- **RC6**: $n = 128\ bits,\ k = 128/192/256\ bits$
- **AES**: $n = 128\ bits,\ k = 128/192/256\ bits$



Stream vs Block Ciphers



Block Ciphers

COMPONENTS

Substitution and Permutation

In 1949, Shannon introduced the idea of substitution-permutation (S-P) networks which form the basis of modern block ciphers

S-P networks are based on the two primitives:

- Substitution (S-box) → Confusion
- Permutation (P-box) → Diffusion

A good block cipher uses also:

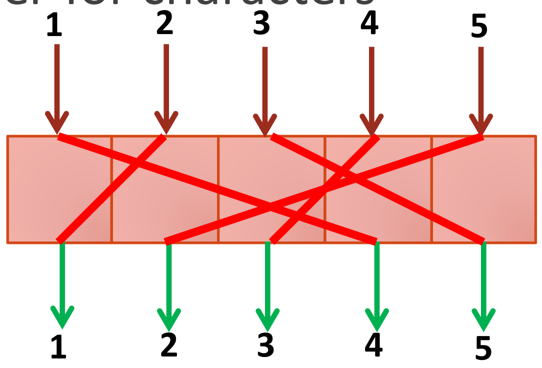
- XOR
- Circular Shift
- Swap
- Split and Combine

Permutation Boxes

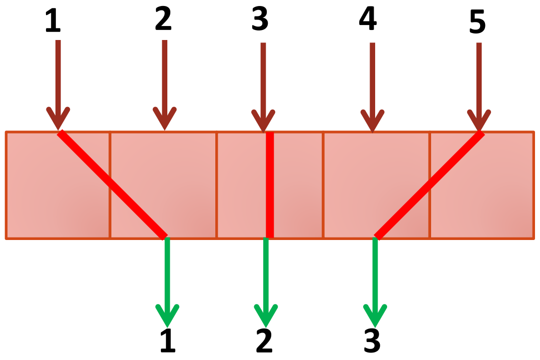
A P-Box (permutation box) is like

- The traditional transposition cipher for characters
- But it transposes bits

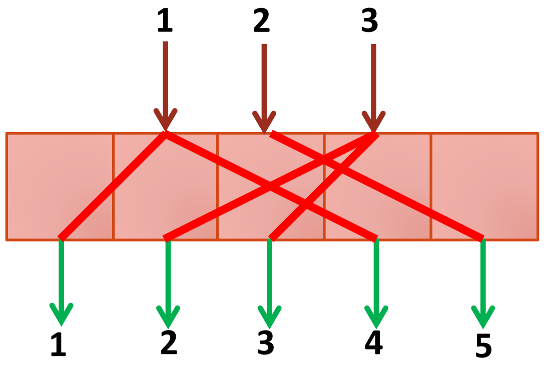
Straight P-Box



Compression P-Box



Expansion P-Box



Permutation Boxes: Example

Straight P-Box: n (inputs) x n (outputs)

- Example 64x64 permutation table

58	50	42	34	26	18	10	02	60	52	44	36	28	20	12	04
62	54	46	38	30	22	14	06	64	56	48	40	32	24	16	08
57	49	41	33	25	17	09	01	59	51	43	35	27	19	11	03
61	53	45	37	29	21	13	05	63	55	47	39	31	23	15	07

Compression P-Box: n (inputs) x m (outputs) → $m < n$

- Example 32x24 permutation table

01	02	03	21	22	26	27	28	29	13	14	17
18	19	20	04	05	06	10	11	12	30	31	32

Expansion P-Box: n (inputs) x m (outputs) → $m > n$

- Example 12x16 permutation table

01	09	10	11	12	01	02	03	03	04	05	06	07	08	09	12

Permutation Boxes: Example

Straight P-Box: n (inputs) x n (outputs)

Compression P-Box: n (inputs) x m (outputs) → $m < n$

Expansion P-Box: n (inputs) x m (outputs) → $m > n$

Which one is invertible?

Permutation Boxes: Example

Straight P-Box: n (inputs) x n (outputs)

Compression P-Box: n (inputs) x m (outputs) → $m < n$

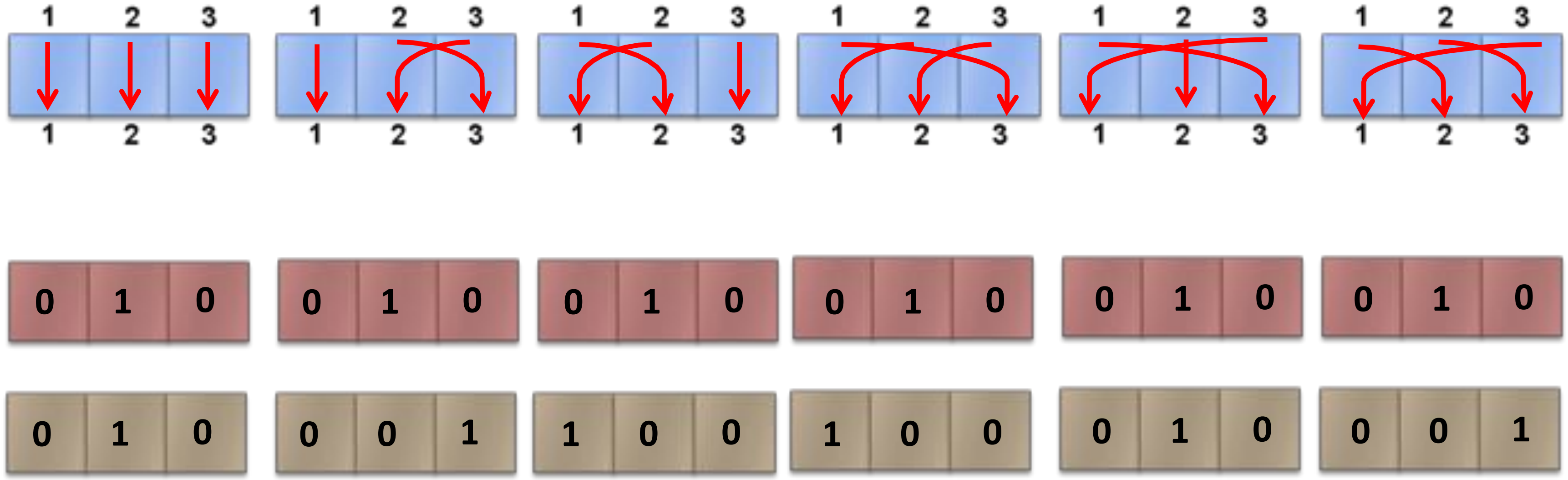
Expansion P-Box: n (inputs) x m (outputs) → $m > n$

Which one is invertible?

Permutation Boxes: Example

Straight 3x3 P-Box (permutation box)

- 6 possible mappings
- Same number of inputs and outputs



Substitution Boxes

An S-Box (substitution box) is

- A box that realizes a miniature substitution cipher
- Is an $m\ x\ n$ substitution cipher
- Invertible if $m = n$!

Leftmost

Rightmost

	00	**01**	**10**	**11**
0	011	101	111	100
1	000	010	001	110

	00	**01**	**10**	**11**
0	00	10	01	11
1	10	00	11	01

Substitution Boxes: Examples

Invertible if same input and output size

- If the input to the left box is **001**, the output is **101**
- The input **101** in the right table creates the output **001**
- The two tables are inverses of each other

Encryption S-Box

	00	01	10	11
0	011	101	111	100
1	000	010	001	110

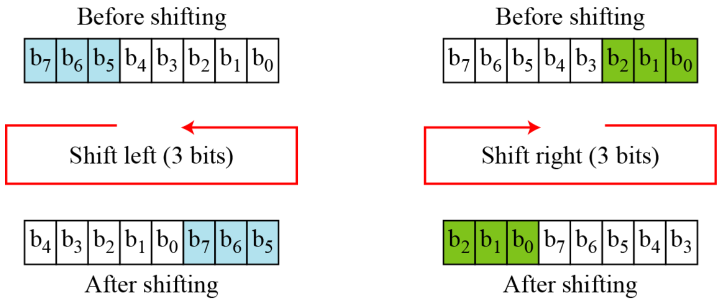
Decryption S-Box

	00	01	10	11
0	100	110	101	000
1	011	001	111	010

Other components

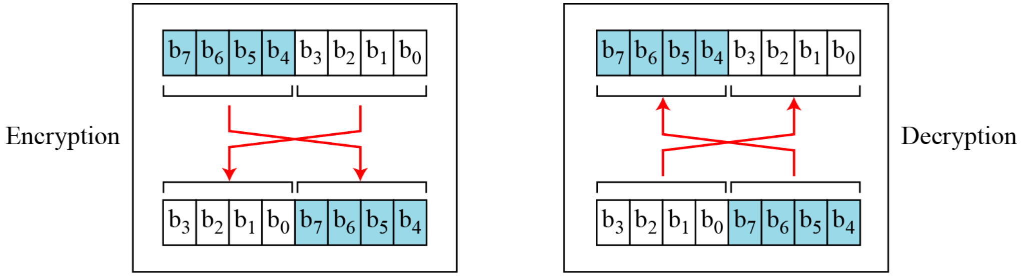
Circular Shift

- Shift bits to the left or to the right



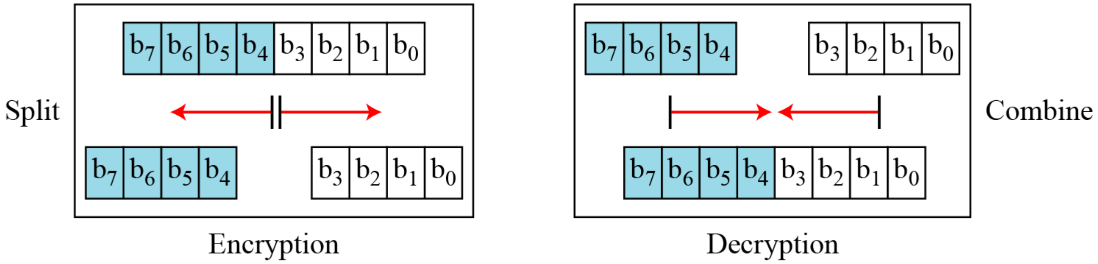
Swap

- Particular case of the shift
- Size of shift = $n/2$



Split & Combine

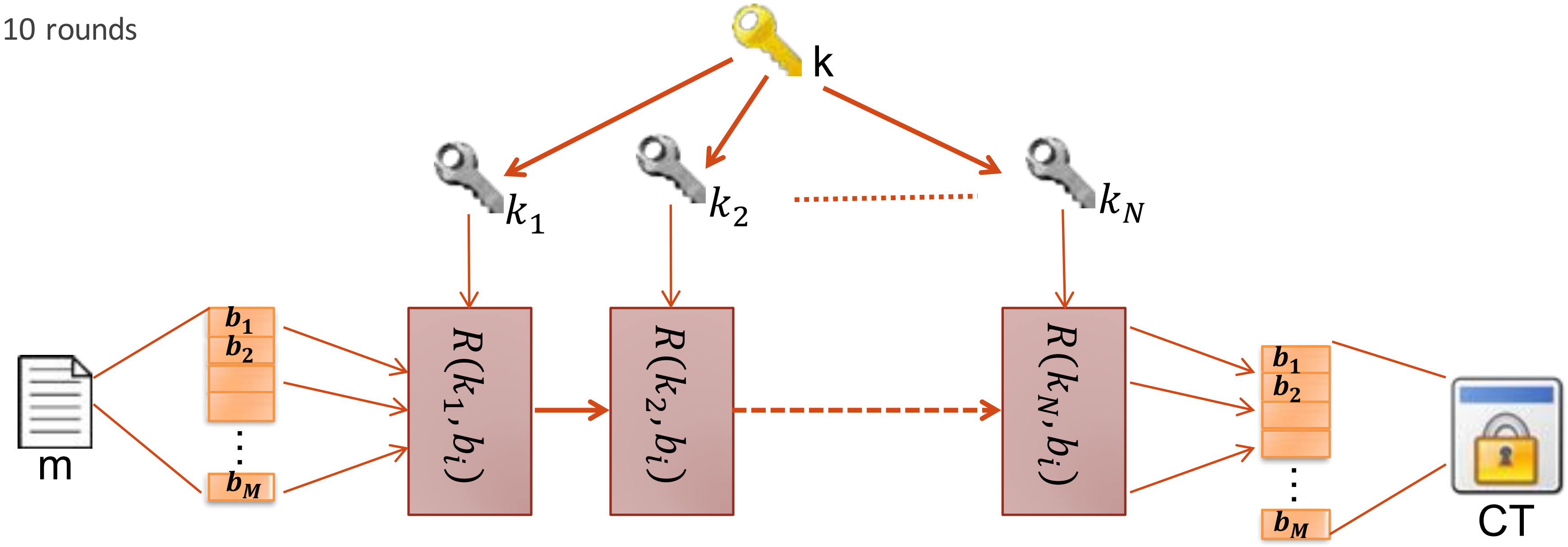
- In encryption we need to split words
- In decryption we need to re-combine words



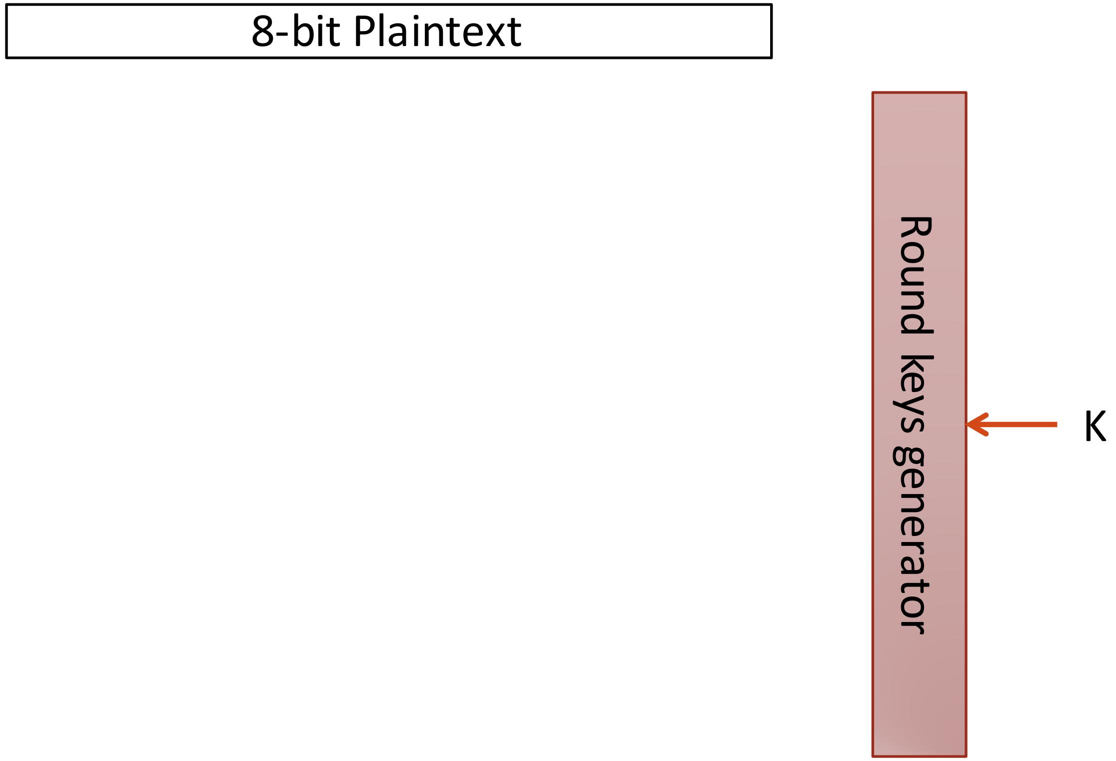
Encrypt by iterations

High level structure

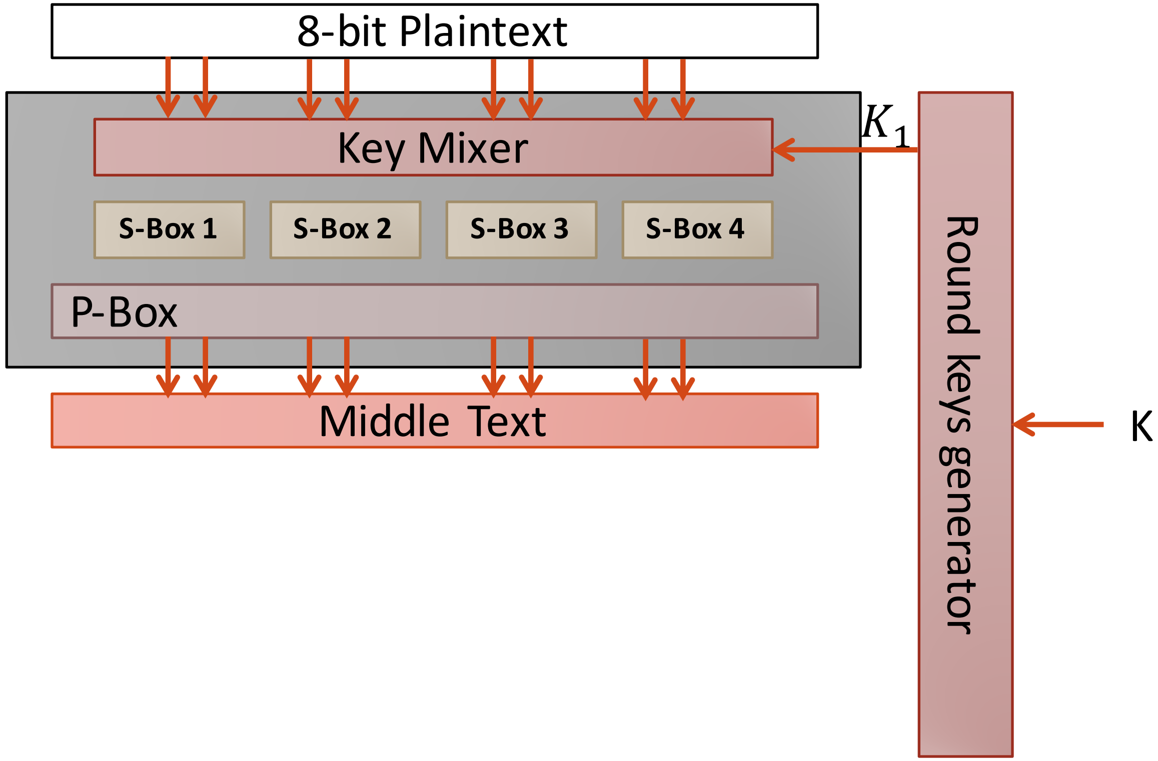
- Define N rounds
- Derive $k_{1..N}$ keys
- Iteratively apply round functions $R(k_i,\ b_i)$ to each block
  - DES: 16 rounds → 3-DES: 48 rounds (16x**3**)
  - AES: 10 rounds



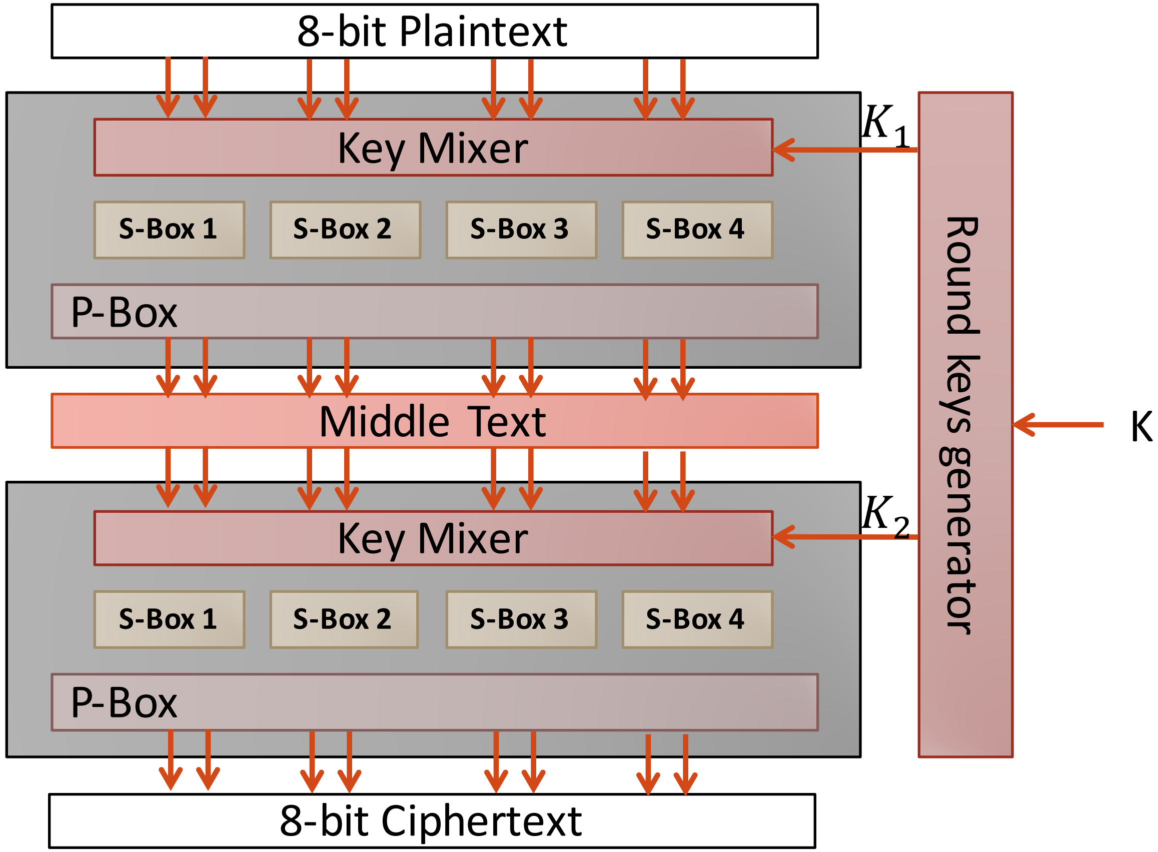
Encrypt by iterations



Encrypt by iterations



Encrypt by iterations



Performance

Iterations (rounds) drawback

- Stream Ciphers notably faster than Block Ciphers
- Crypto++ benchmarks (**http://www.cryptopp.com/benchmarks.html**)

Cipher	Block Size	Key Size	Throughput [MB/s]
RC4	-	-	126
Salsa20/12	-	-	674
Sosemanuk	-	-	767
DES	64	56	46
3-DES	64	168	17
AES	128	128	148

Block Ciphers

FEISTEL NETWORK

Motivation for Feistel Network

Product cipher

- Sequence of two or more simple ciphers
- Final result or product is cryptographically stronger than any of the component ciphers

S-P network

- A special form of substitution-permutation product cipher
- Feistel Network
- Non-Feistel Network

Motivation for Feistel Network

Feistel ciphers

- In 1970's, Horst Feistel (IBM) proposed a suitable (and practical) structure for Shannon's S-P network
- Encryption and decryption use the same structure
- Three types of components:
  - Self-invertible
  - Invertible
  - Non-Invertible

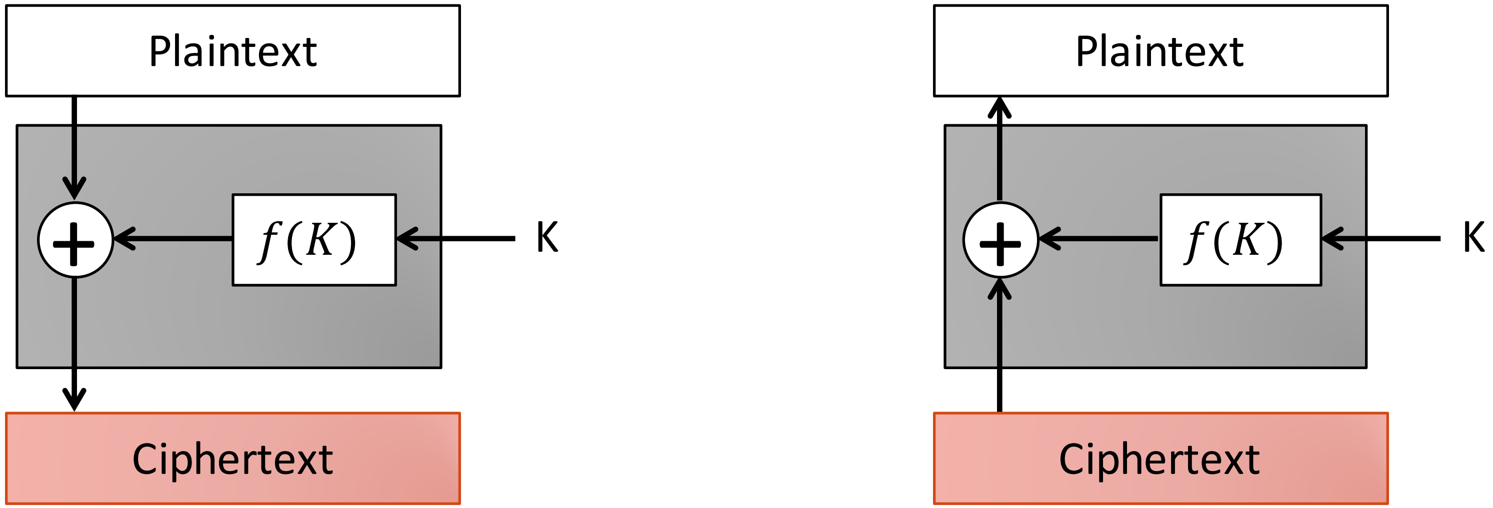
Non-Feistel ciphers

- Only invertible components
- A component in the encryption cipher has the corresponding component in the decryption cipher

Feistel Network

First sketch of the Feistel design

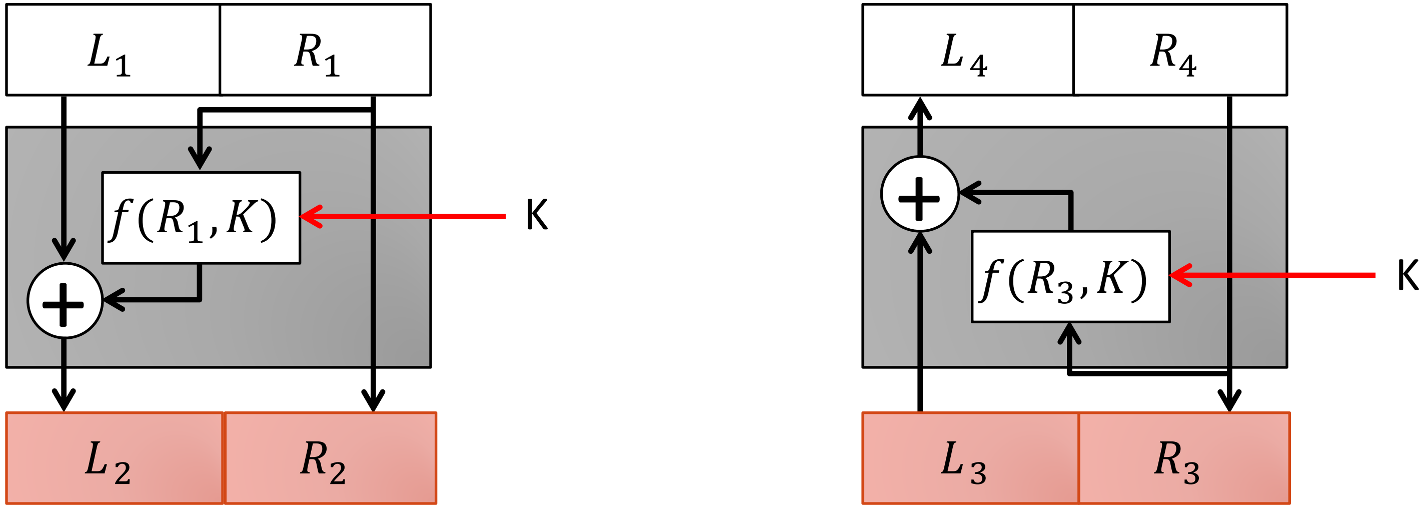
- Any function $f(K)$



Feistel Network

Improvement of the Feistel design

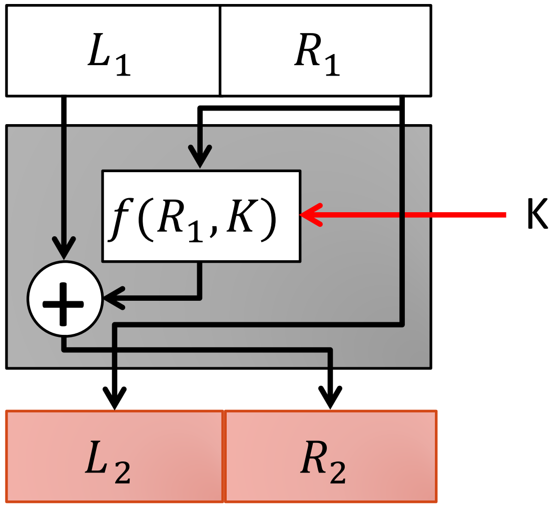
- Any function $f(K, R_i)$

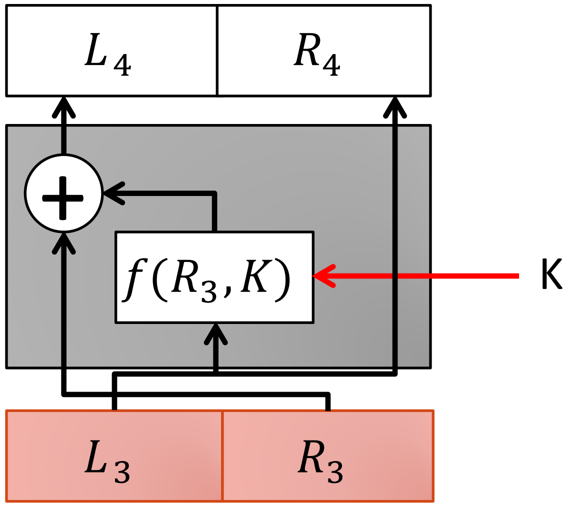


Feistel Network

Improvement of the Feistel design

- Any function $f(K, R_i)$
- Swap output of each round





Feistel Network

Block size
- Increasing size improves security

Key size
- Increasing size improves security
- Makes exhaustive key searching harder

Number of rounds
- Increasing number improves security

Sub-key generation
- Greater complexity can make analysis harder

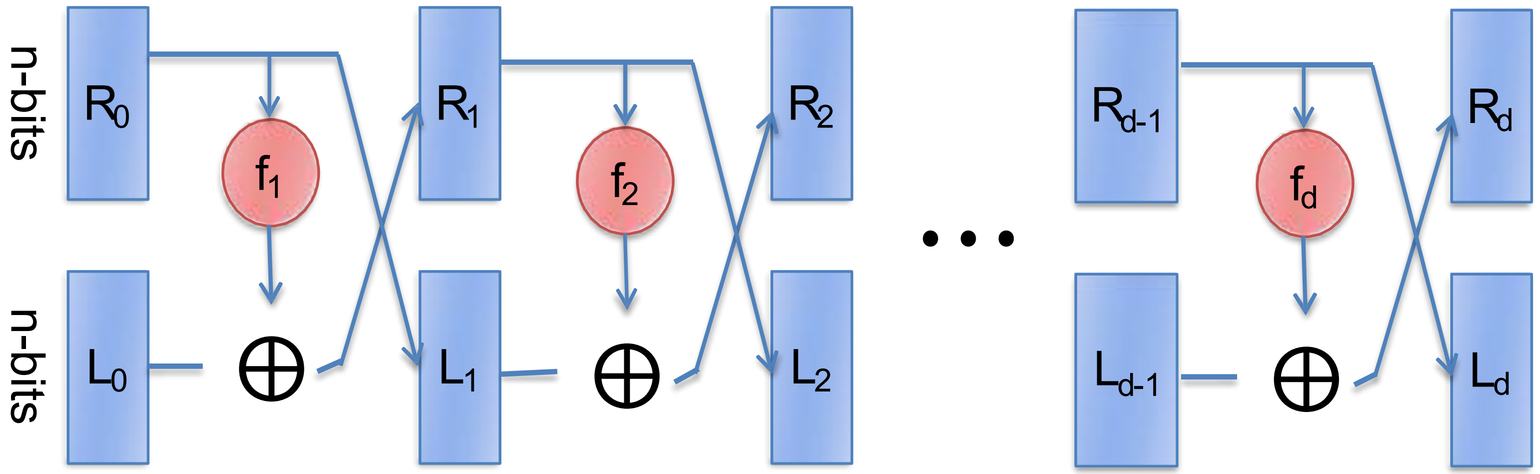
Round function
- Greater complexity can make analysis harder

Slows encryption/decryption

Feistel Network

Make the network

- Use generic round functions
  - $f_1, \ldots, f_d : \{0,1\}^n \to \{0,1\}^n$
- To make invertible function
  - $F(f_1, \ldots, f_d) : \{0,1\}^{2n} \to \{0,1\}^{2n}$
  - $\begin{cases} R_{i+1} = L_i \oplus f_{i+1}(R_i) \\ L_{i+1} = R_i \end{cases}$



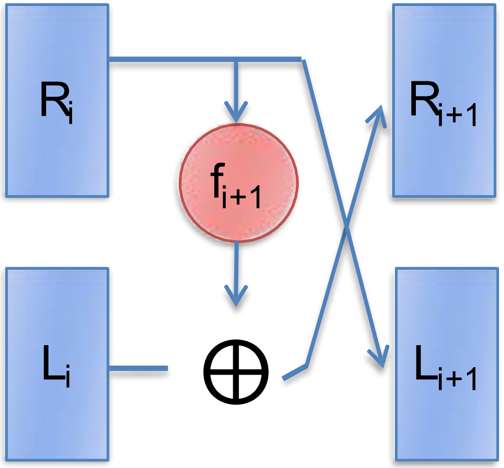
Feistel Network

Always invertible

- Even if f_i is not invertible

$$\begin{cases} R_{i+1} = L_i \oplus f_{i+1}(R_i) \\ L_{i+1} = R_i \end{cases}$$

inverse



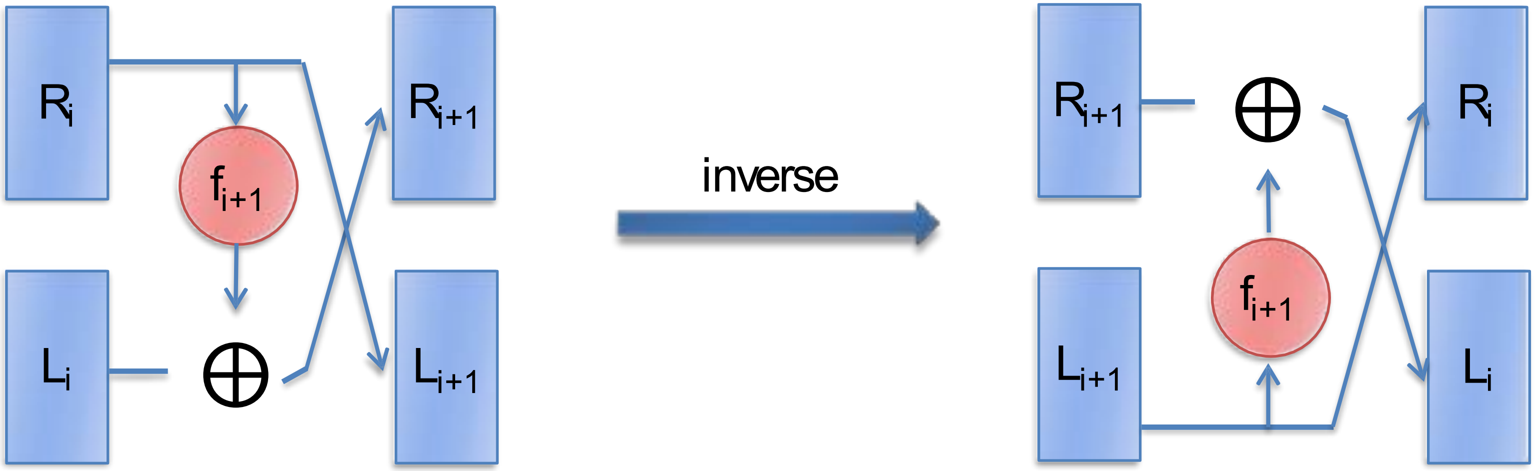
inverse

Feistel Network

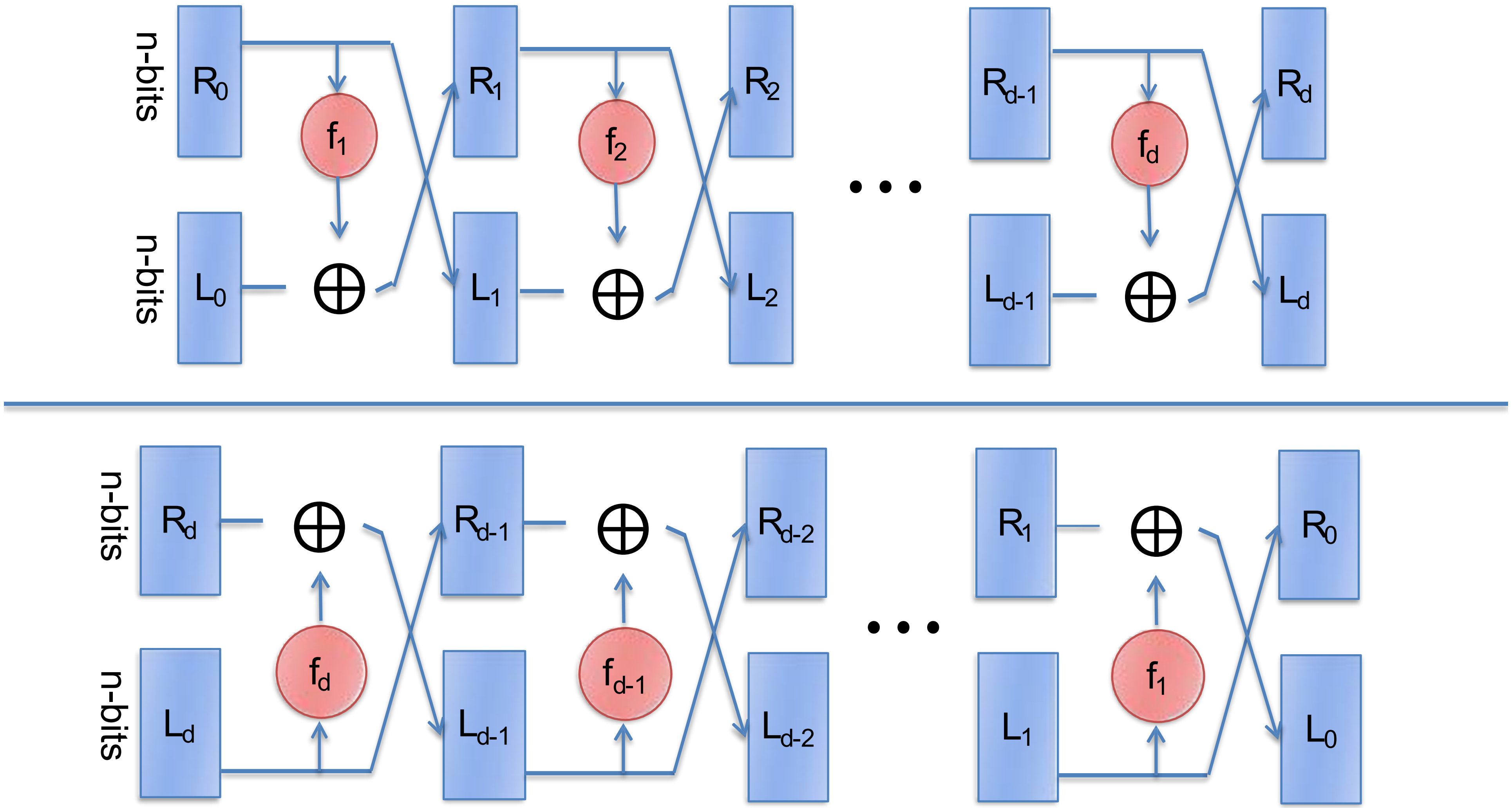
Always invertible

- Even if f_i is <u>not invertible</u>

$$\begin{cases} R_{i+1} = L_i \oplus f_{i+1}(R_i) \\ L_{i+1} = R_i \end{cases} \xrightarrow{\text{inverse}} \begin{cases} R_i = L_{i+1} \\ L_i = R_{i+1} \oplus f_{i+1}(L_{i+1}) \end{cases}$$



Feistel Network: Encryption/Decryption



Feistel Network

Decryption is basically the same circuit

- Applied in the inverse order
- i.e. $f_1, \dots, f_d$ → $f_d, \dots, f_1$

General methodology to build ciphers

- Arbitrary atomic function (also not invertible)
- Always invertible

Design for many block ciphers

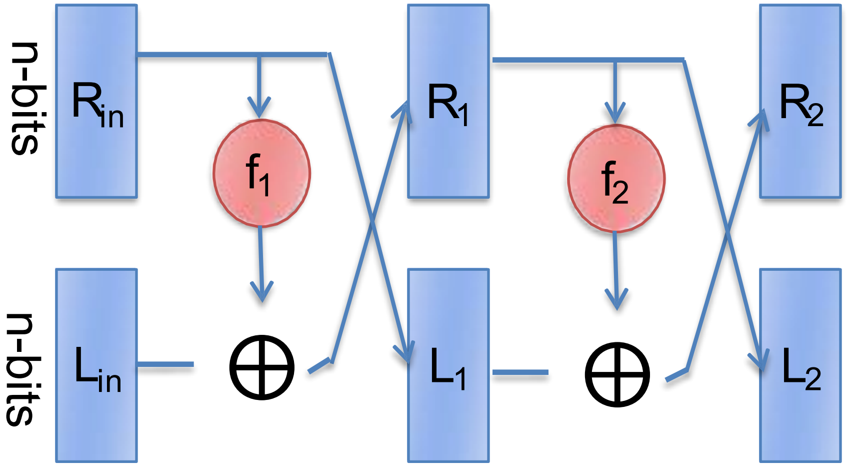
- DES
- 3-DES
- RC5
- …Not AES…

Feistel Network: two-rounds

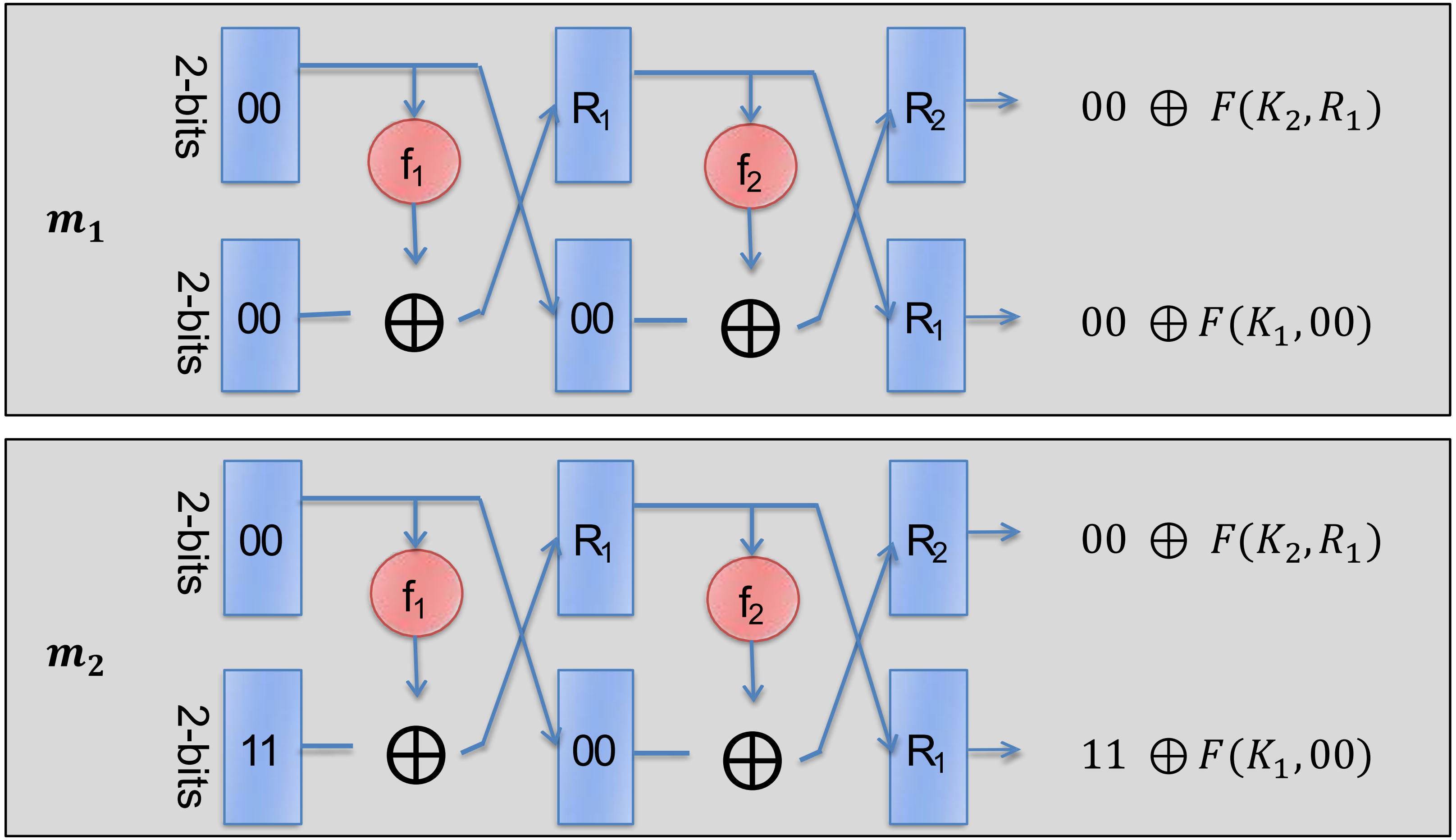
Never use two-rounds Feistel network

- It is not secure

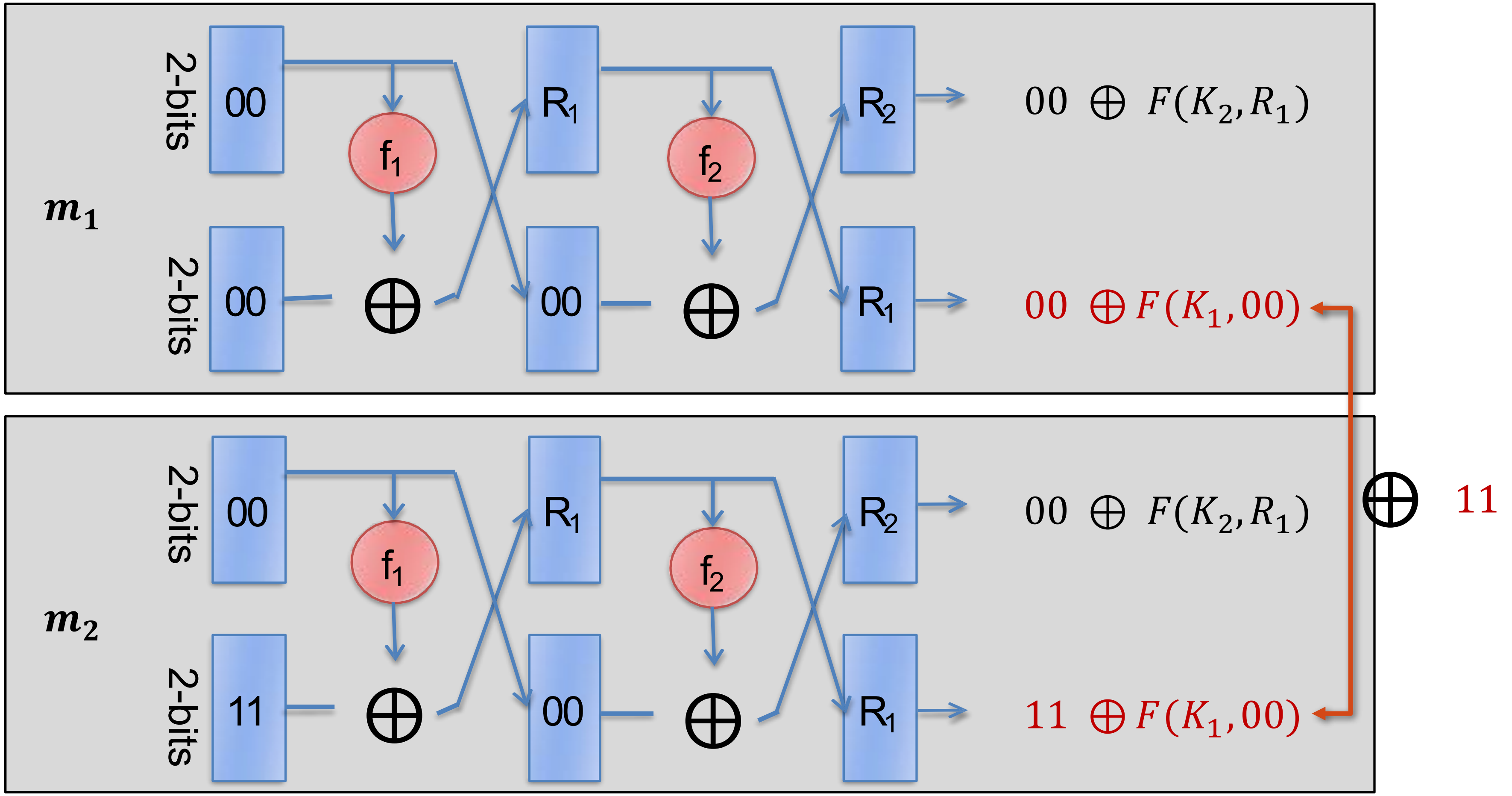
- Suppose $f_i = F(k_i, \cdot)$ is a secure function
- Compare and exploit output from:
  - $m_1 = 0000$
  - $m_2 = 0011$



Feistel Network: two-rounds



Feistel Network: two-rounds



Block Ciphers

DATA ENCRYPTION STANDARD

Encryption Standardization

1960

- The first commercial Feistel Cipher developed by IBM
- **Lucifer** by Feistel and Coppersmith

1972

- US National Bureau of Standards (NBS) call for proposals

1974-1977

- Lucifer refined, renamed the Data Encryption Algorithm (DEA)
- Adopted as standard by NBS
- First official U.S. government cipher for commercial use
- Most widely used block cipher

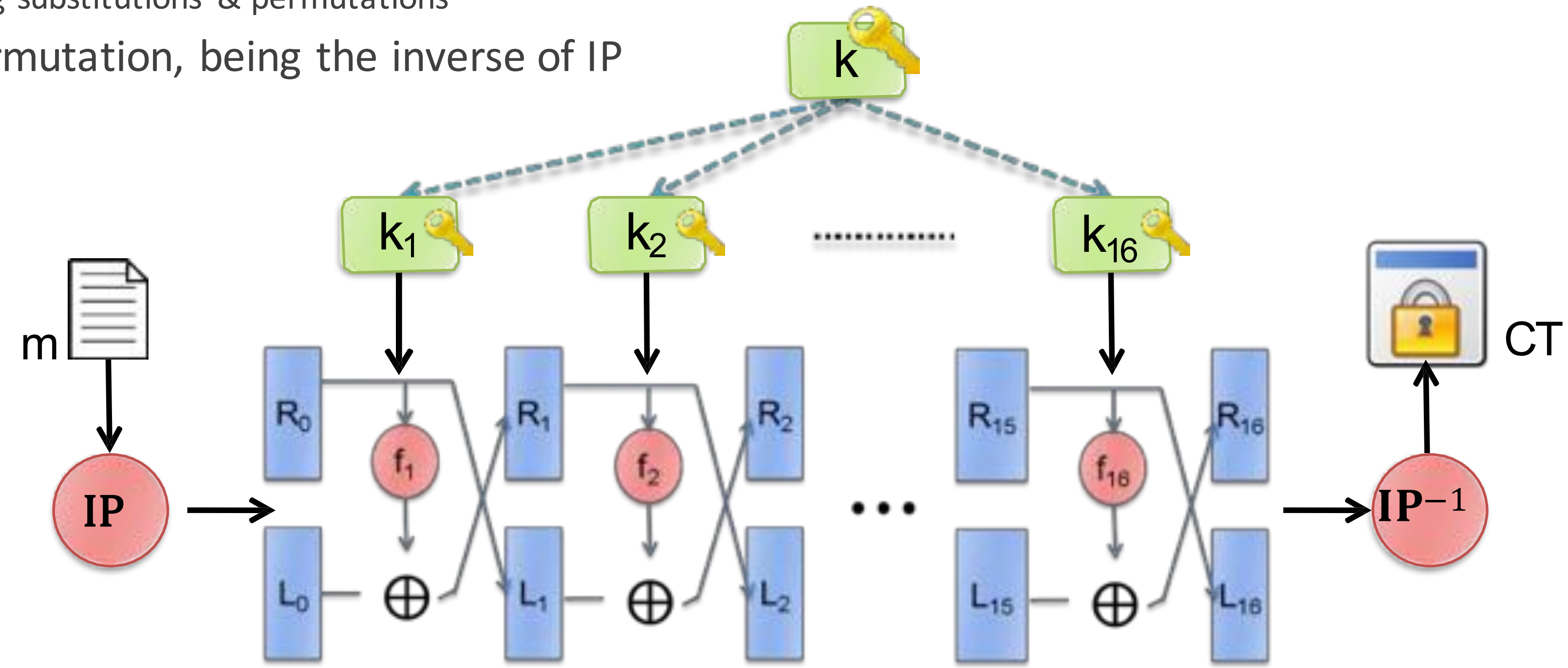
1997

- DES theoretically broken
    - Exhaustive search
    - Differential and linear cryptanalysis

DES Structure

Basic process to encrypt a 64-bit data block

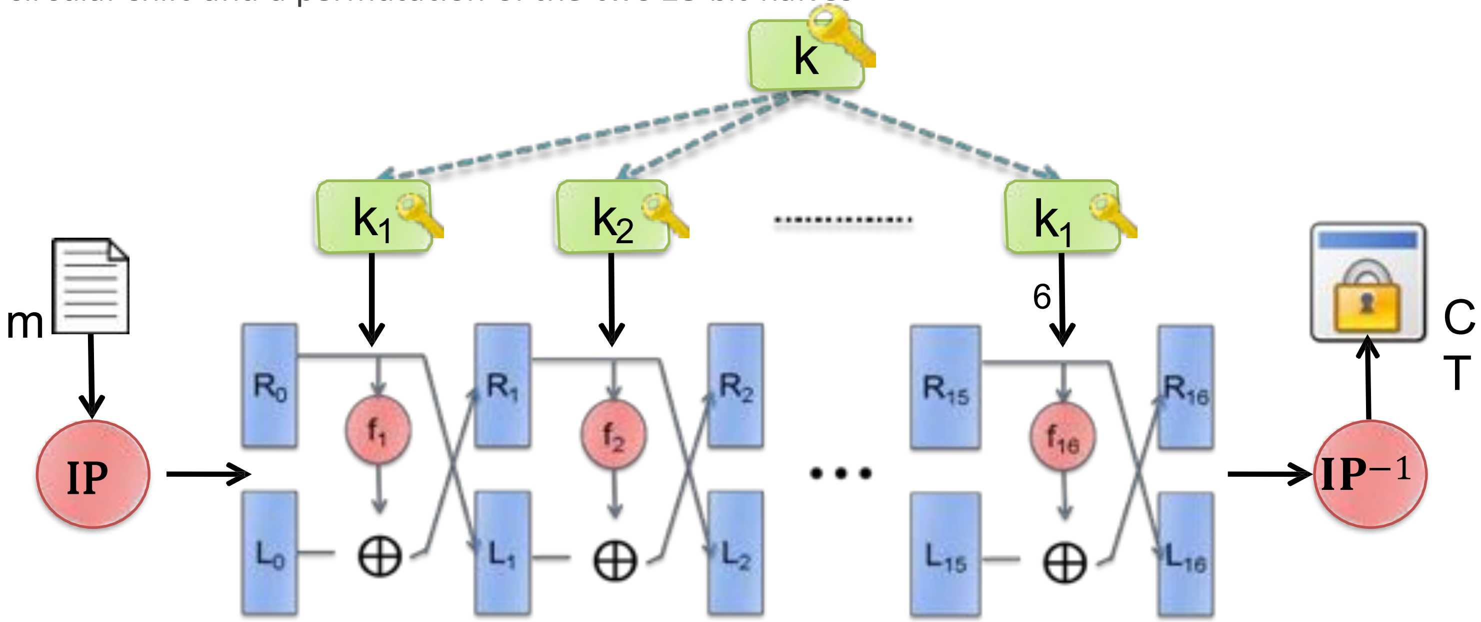
- Initial permutation (IP) which shuffles the 64-bit input block
- 16 rounds of a complex key dependent round function
  - Involving substitutions & permutations
- Final permutation, being the inverse of IP



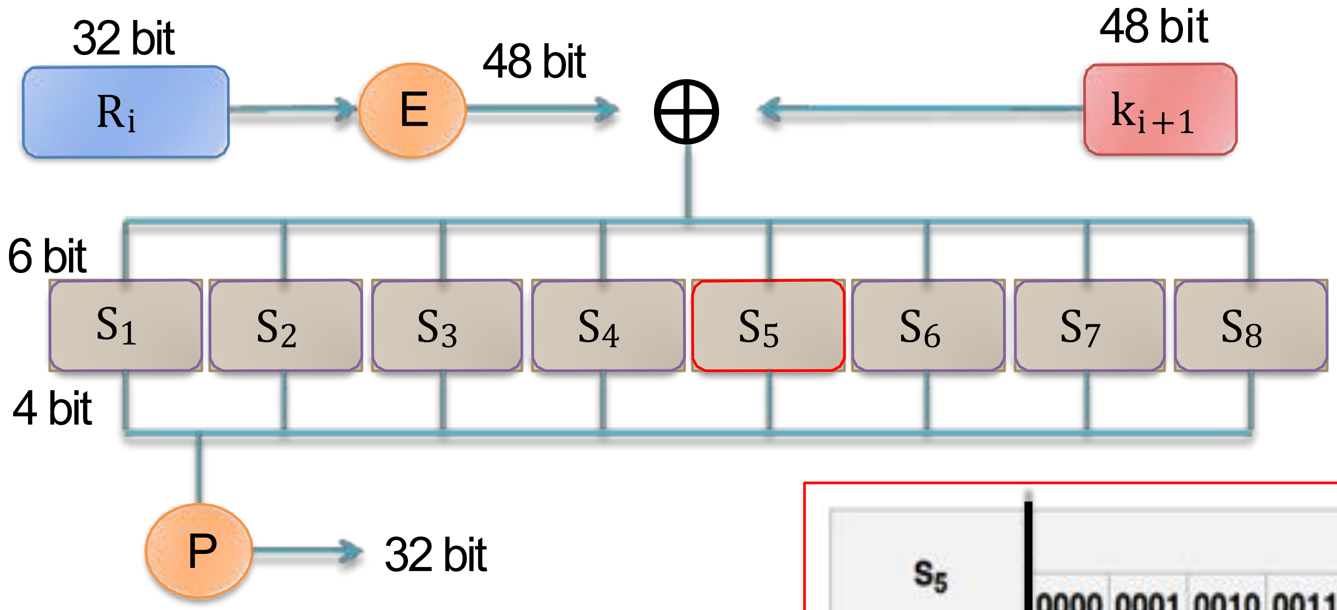
DES Structure

16 sub-keys are derived by the 64-bit key (56+8 parity):

- Initial permutation of the key (K)
  - Selects 56-bits out of the 64-bits input, in two 28-bit halves
- 16 stages to generate the 48-bit sub-keys
  - Using a left circular shift and a permutation of the two 28-bit halves

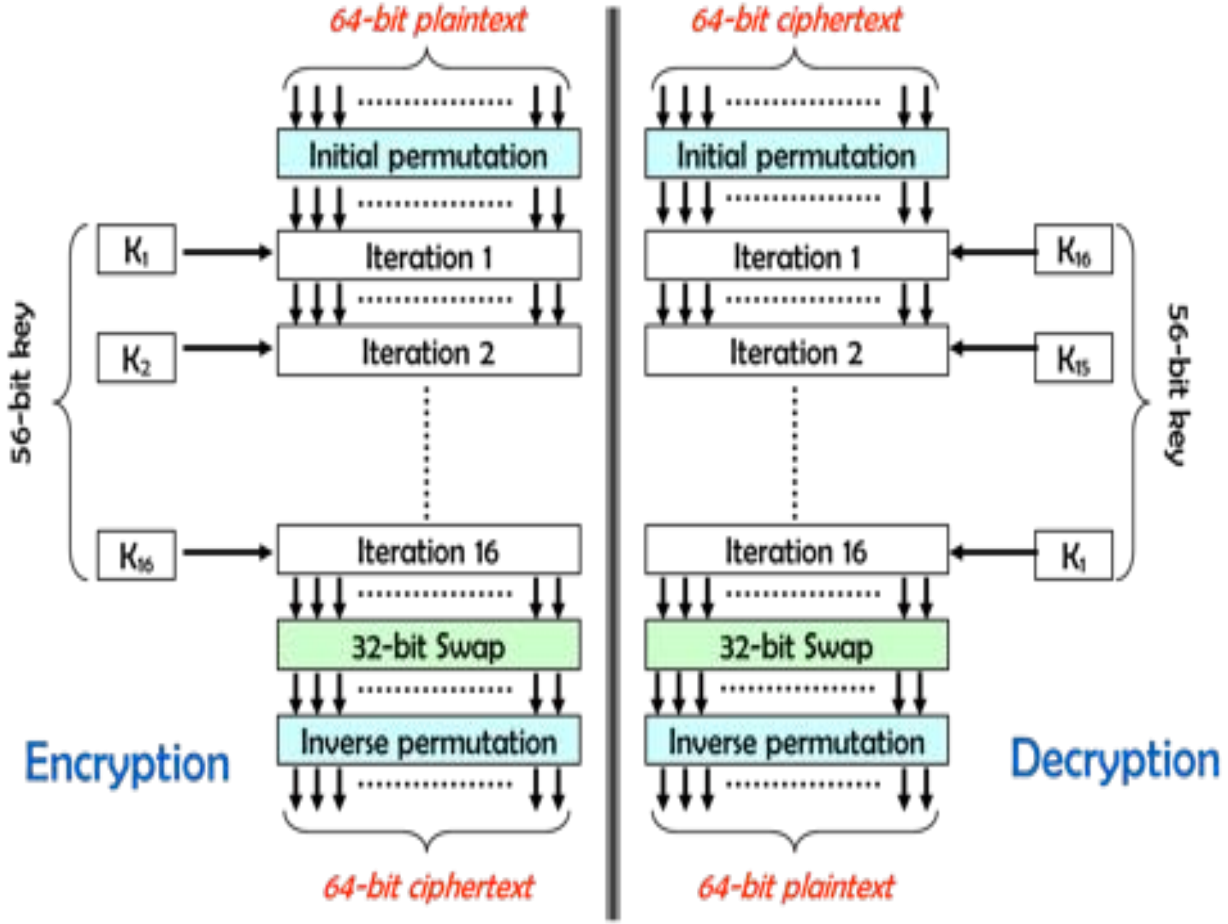


DES: Feistel Round Function



S_5		Middle 4 bits of input															
		0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
Outer bits	00	0010	1100	0100	0001	0111	1010	1011	0110	1000	0101	0011	1111	1101	0000	1110	1001
	01	1110	1011	0010	1100	0100	0111	1101	0001	0101	0000	1111	1010	0011	1001	1000	0110
	10	0100	0010	0001	1011	1010	1101	0111	1000	1111	1001	1100	0101	0110	0011	0000	1110
	11	1011	1000	1100	0111	0001	1110	0010	1101	0110	1111	0000	1001	1010	0100	0101	0011

DES Overview



DES Decryption

Decrypt must “undo” steps of data computation

- Exploit Feistel design, do encryption steps again
- Using sub-keys in reverse order (K_{16} … K_1)

Note that

- IP complement final FP step of encryption
- 1st round with K_{16} undoes 16th encrypt round
- 16th round with K_1 undoes 1st encrypt round
- Then final FP undoes initial encryption IP

DES Properties

The avalanche effect

- A change of **one** input bit or key bit should result in changing approx **half** of output bits!
- Making attempts to guess the key by using different Plaintext – Ciphertext pairs should be impossible
- DES exhibits strong avalanche

Plaintext: 0000000000000000	Key: 22234512987ABB23
Ciphertext: 4789FD476E82A5F1	
Plaintext: 000000000000000<u>**1**</u>	Key: 22234512987ABB23
Ciphertext: 0A4ED5C15A63FEA3	

The completeness

- Each bit of the ciphertext depend on many bits on the plaintext

DES Security

Among the attempted attacks, three are of interest:

- Brute-force/Exhaustive search
  - Short cipher key
  - Key complement weakness
- Differential cryptanalysis
  - Designers of DES already knew about this type of attack
  - Designed S-boxes and 16 as the number of rounds to make DES specifically resistant to this type of attack
- Linear cryptanalysis
  - S-boxes are not very resistant to linear cryptanalysis
  - DES can be broken using 2^{43} pairs of known plaintexts

Breaking DES

Time to break DES

- Number of keys: $2^{56} = 7.2 \; x \; 10^{16}$ keys
  - On the average you need to search through 2^{55} keys
  - In the worst case you need to search all 2^{56} keys
- If one encryption/decryption in 1 clock cycle @ 500 MHz
  - Time taken to check ONE key = $1/(500 \; x \; 10^6) \; s$
  - Time taken to check 2^{55} keys = $\frac{2^{55}}{500 \; x \; 10^6} s = 834$ days

Cost to break DES

- At $20 per chip, to break DES in one day
- Need to spend $16,680

Breaking DES

Key Size (bits)	Number of Alternative Keys	Time required at 1 decryption/μs	Time required at 10^6 decryptions/μs
32	$2^{32} = 4.3 \times 10^9$	$2^{31}\ \mu s = 35.8$ minutes	2.15 ms
56	$2^{56} = 7.2 \times 10^{16}$	$2^{55}\ \mu s = 1142$ years	10.01 hours
128	$2^{128} = 3.4 \times 10^{38}$	$2^{127}\ \mu s = 5.4 \times 10^{24}$ years	5.4×10^{18} years
168	$2^{168} = 3.7 \times 10^{50}$	$2^{167}\ \mu s = 5.9 \times 10^{36}$ years	5.9×10^{30} years
26 characters (permutation)	$26! = 4 \times 10^{26}$	$2 \times 10^{26}\ \mu s = 6.4 \times 10^{12}$ years	6.4×10^6 years

Breaking DES

Weak Keys

- Symmetry of bits in the 32 bit halves makes the key weak
- Roughly 64 weak keys, e.g.:
  - Alternating ones + zeros (0x0101010101010101)
  - Alternating 'F' + 'E' (0xFEFEFEFEFEFEFEFE)
  - '0xE0E0E0E0F1F1F1F1' or '0x1F1F1F1F0E0E0E0E'
- A complement of key will encrypt the complement of a plaintext into the complement of the ciphertext

Number of rounds

- Six round DES was broken very early on
- Less than 16 rounds makes DES less secure

Breaking DES

Some weaknesses in DES

- Weaknesses in S-boxes
- Weaknesses in P-boxes
- Weaknesses in Key

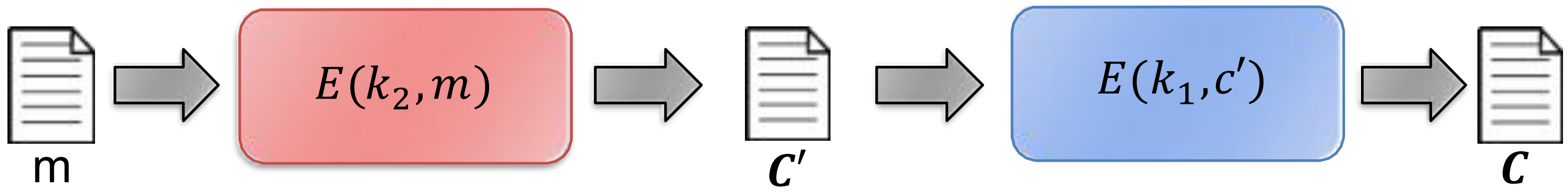
The major criticism of DES regards its key length

- We can use double or triple DES to increase the key size
  - 2-DES (Double)
  - 3-DES (Triple)
- We could then preserve the existing software and hardware

Double DES

Apply two iterations of DES

- Using two different keys k_1 and k_2
- $2DES(k_1, k_2, m) = DES(k_1, DES(k_2, m))$



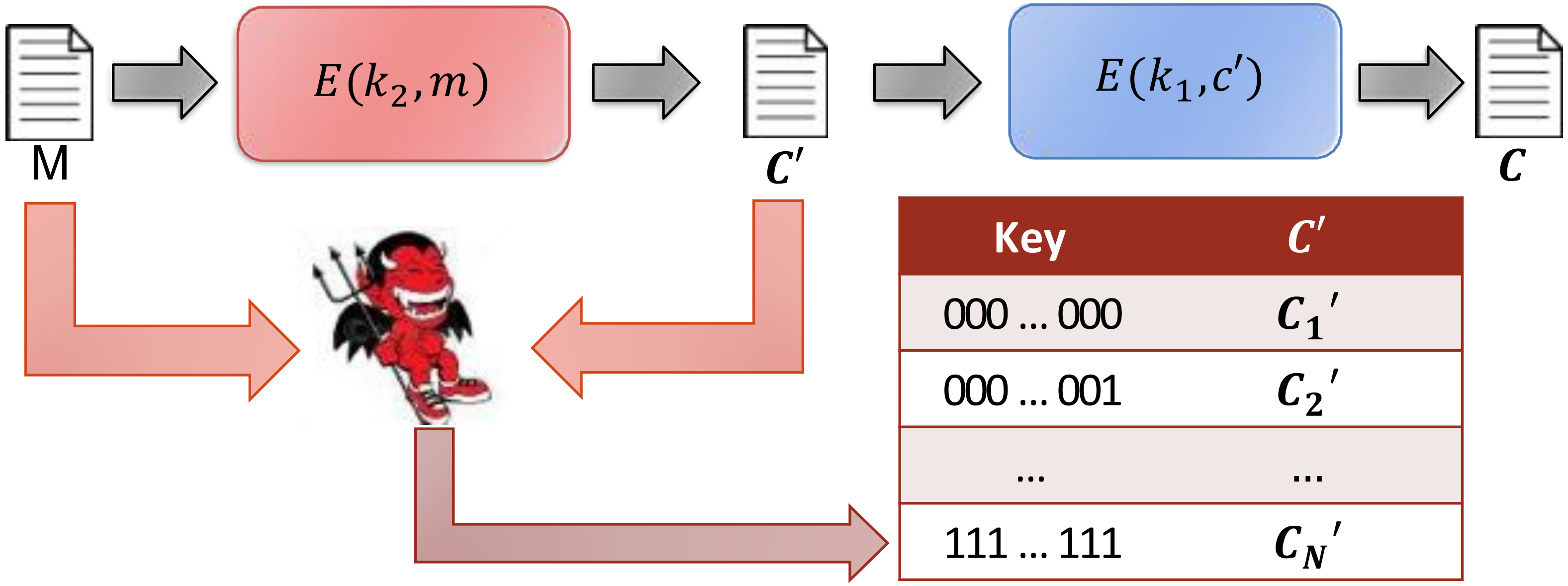
Known-plaintext attack

- 1992: Meet-in-the-middle attack
- Double DES improves this vulnerability slightly
  - 2^{57} trials, but not tremendously to 2^{112}

Meet In The Middle

For given M and C

- Search only O(2^{56}) pairs of keys K_1 and K_2
  - At the intermediate message C'
- Encrypt M under all 2^{56} options for K_1
  - Denote the results by $C'_1, C'_2, \ldots, C'_N$

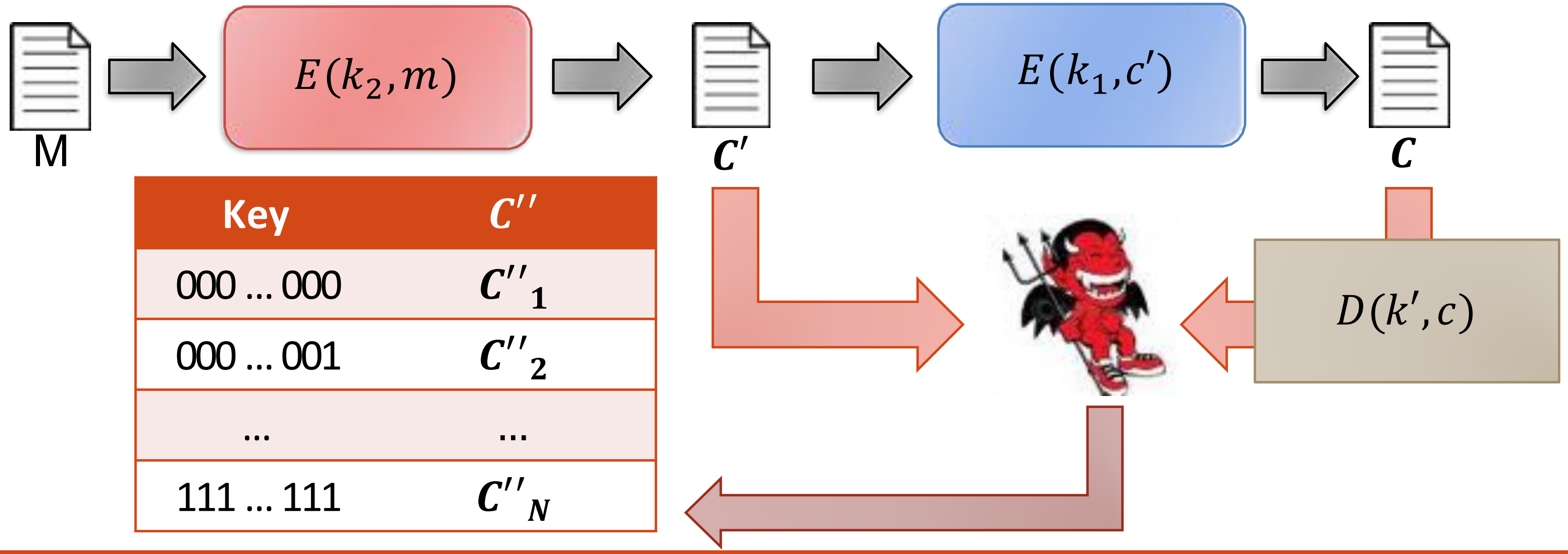


Key	C'
000 … 000	$C_1{}'$
000 … 001	$C_2{}'$
…	…
111 … 111	$C_N{}'$

Meet In The Middle

For given M and C

- Search only $O(2^{56})$ pairs of keys K_1 and K_2
  - At the intermediate message C'
- Decrypt C under all 2^{56} options for K_2
  - Denote the results by $C''_1, C''_2, \ldots, C''_N$



Key	C''
000 … 000	C''_1
000 … 001	C''_2
…	…
111 … 111	C''_N

Meet In The Middle

At least one match of C_i with two keys (k_1 and k_2)

- If there is only match → found the key
- If there is more than one → take another pair
- This is repeated until a unique pair found

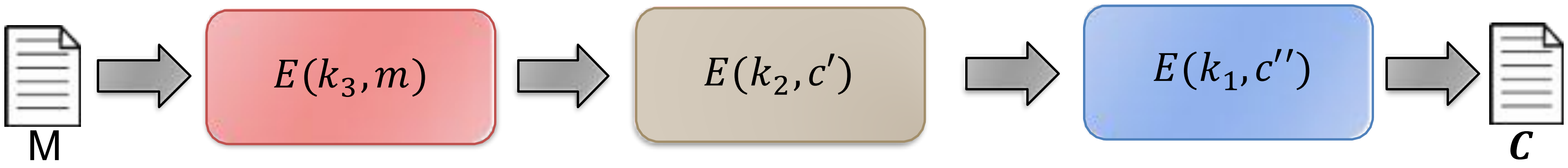
Key	C''
000 … 000	C''_1
000 … 001	C''_2
…	…
111 … 111	C''_N

Key	C'
000 … 000	$C_1{}'$
000 … 001	$C_2{}'$
…	…
111 … 111	$C_N{}'$

Triple DES

DES Encrypt-Encrypt-Encrypt Mode:

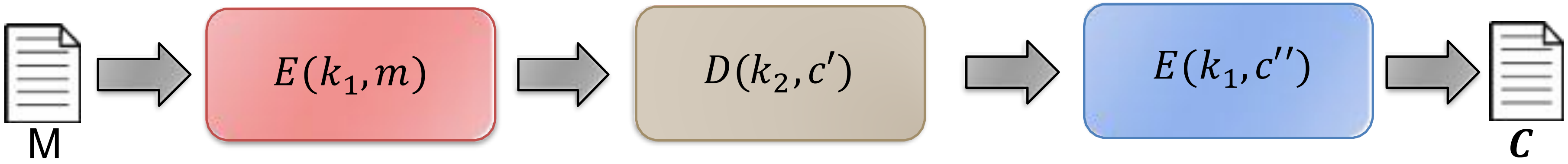
- Three keys K_1, K_2, K_3 (168 bits)
- Strength $O(2^{110})$ against Meet-in-the-Middle
- Not compatible with regular DES



Triple DES

DES Encrypt-Decrypt-Encrypt Mode:

- Two keys K_1 and K_2(112 bits)
- Two keys Strength $O(2^{110})$ against Meet-in-the-Middle
- Compatible with regular DES when $K_1 = K_2$



Double vs Triple DES

Double DES

- Meet in the middle weakness
- Time $\approx 2^{56} * 2^{56} \approx 2^{56} + 2^{56} = 2^{57}$

Triple DES

- Meet in the middle weakness
  - But still secure
- Time $\approx 2^{56} * 2^{56} = 2^{112}$ (… not 2^{168})

Why E-D-E?

- Initial and final permutations would cancel each other out with EEE (minor advantage to EDE)
- EDE compatible with single DES if same keys.
- Only 2 different Keys needed with E-D-E

Block Ciphers

ADVANCED ENCRYPTION STANDARD

The AES Standardization

1997

- NIST publishes request for proposal for DES successor
- Three selection criteria
- Security, Cost and Implementation

1998-1999

- 15 submissions – 5 finalists
- Rijndael: 86 positive, 10 negative
- Serpent: 59 positive, 7 negative
- Twofish: 31 positive, 21 negative
- RC6: 23 positive, 37 negative
- MARS: 13 positive, 84 negative

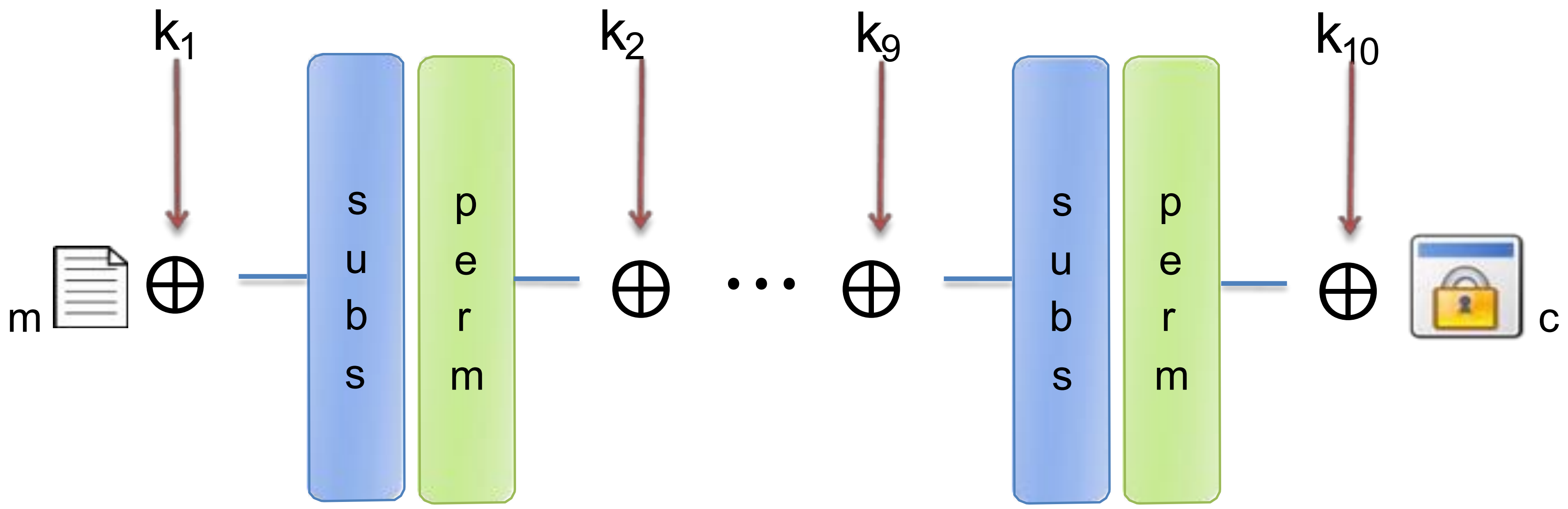
2001

- NIST chooses Rijndael as AES (designed in Belgium)

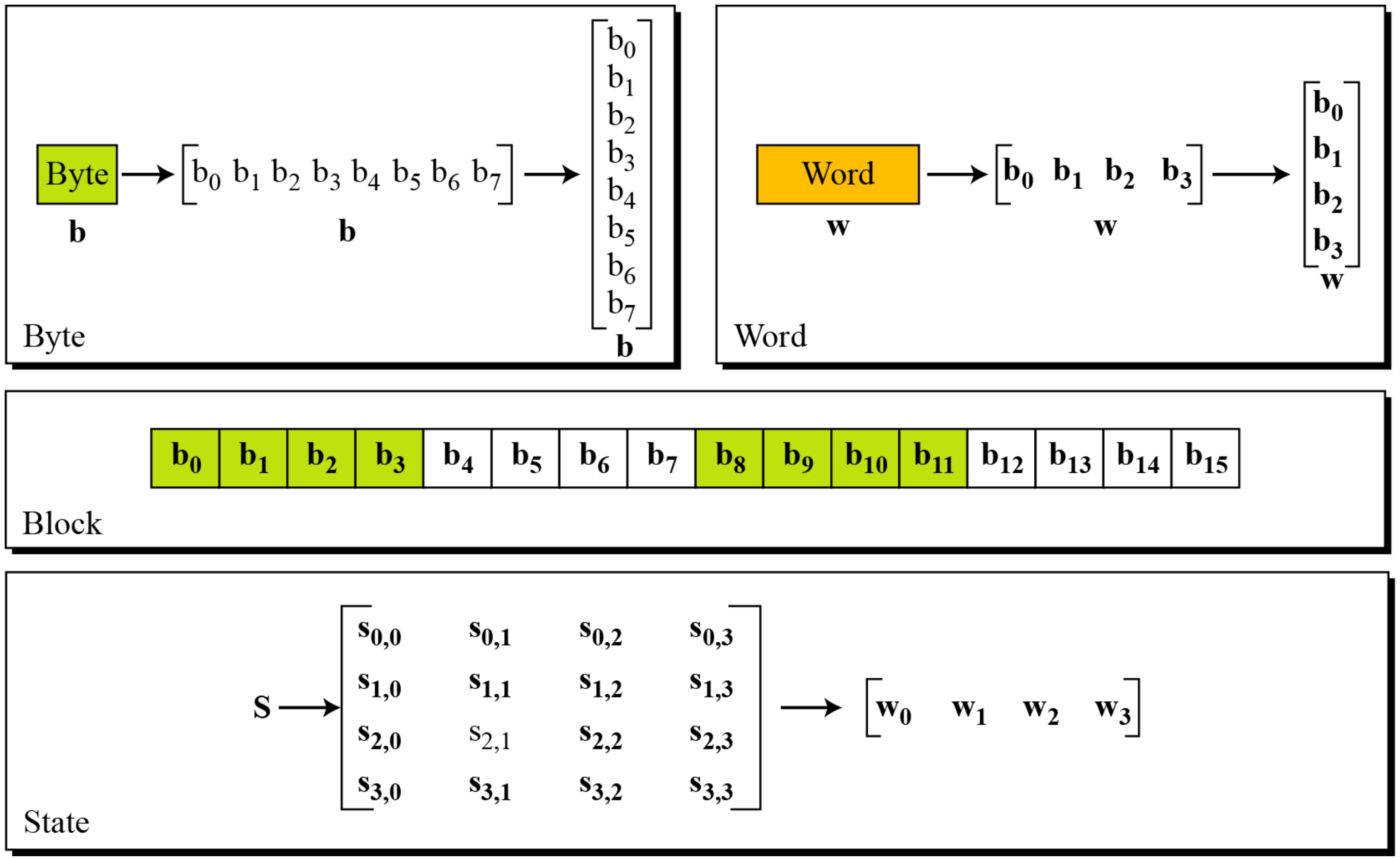
AES Overview

AES is a non-Feistel cipher

- Encrypts/Decrypts a data block of 128 bits
- Uses 10, 12, or 14 rounds
- Key size of 128, 192, or 256 bits
- Round sub-keys are always 128 bits

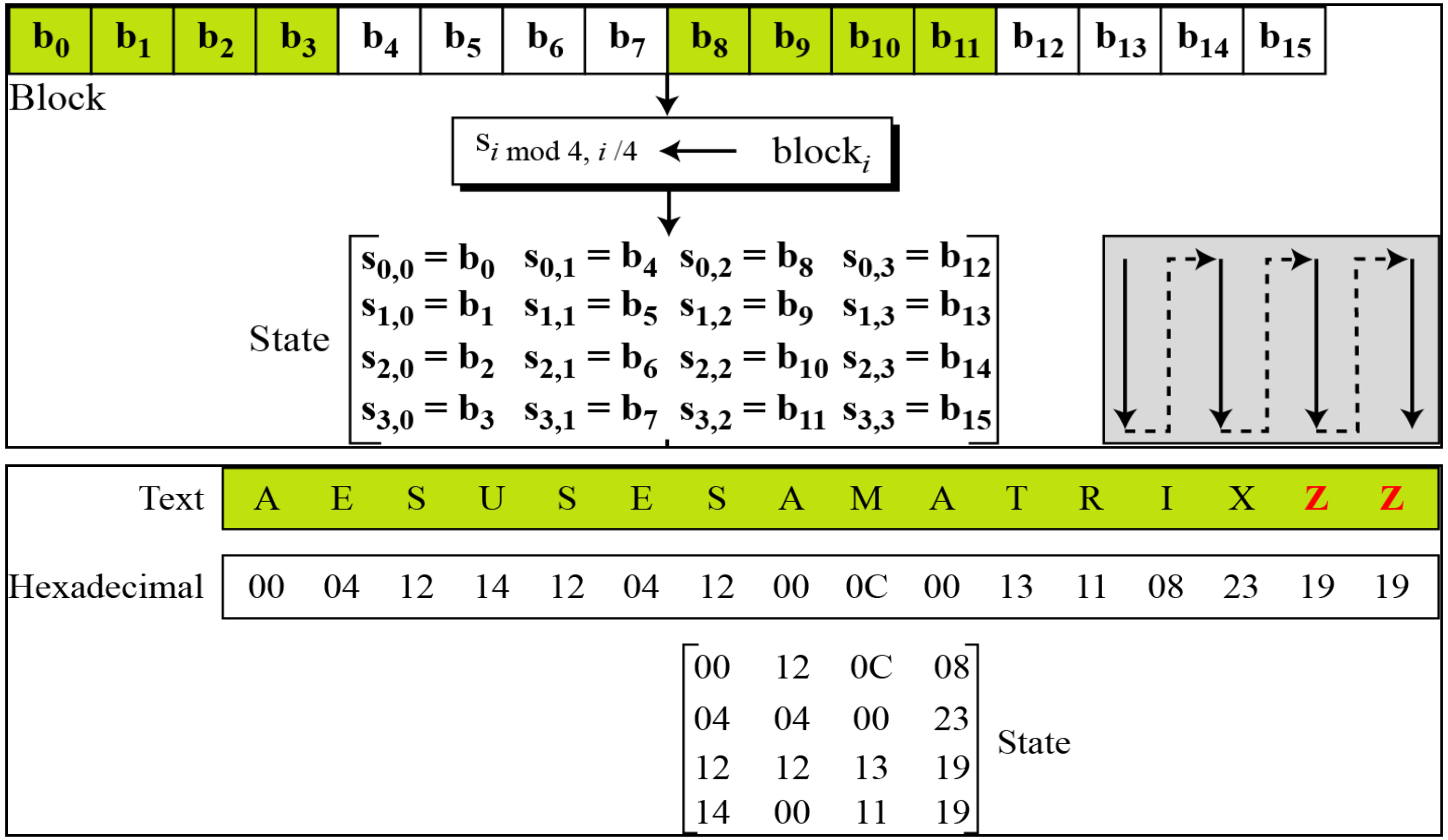


Data Units in AES

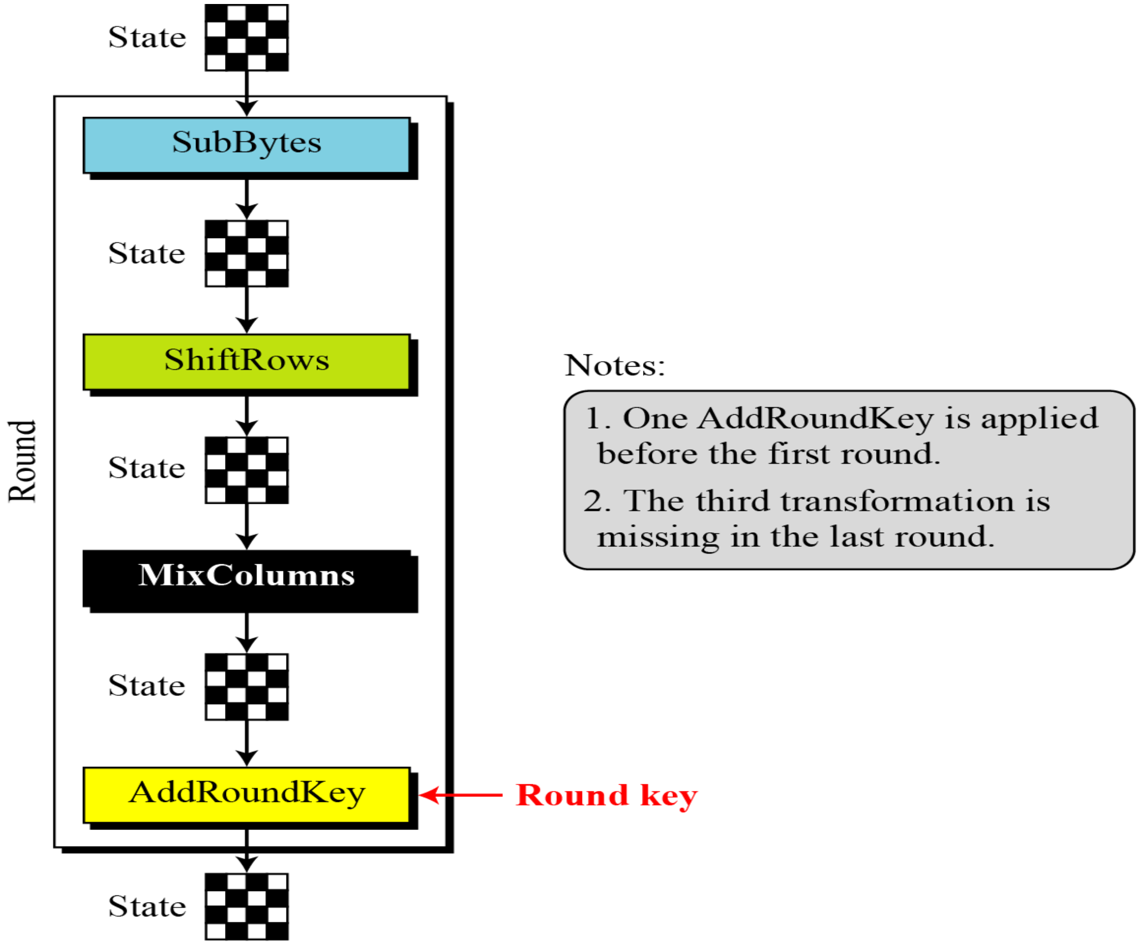


State

Block-to-state and state-to-block transformation



Round’s structure



Substitution & Permutation

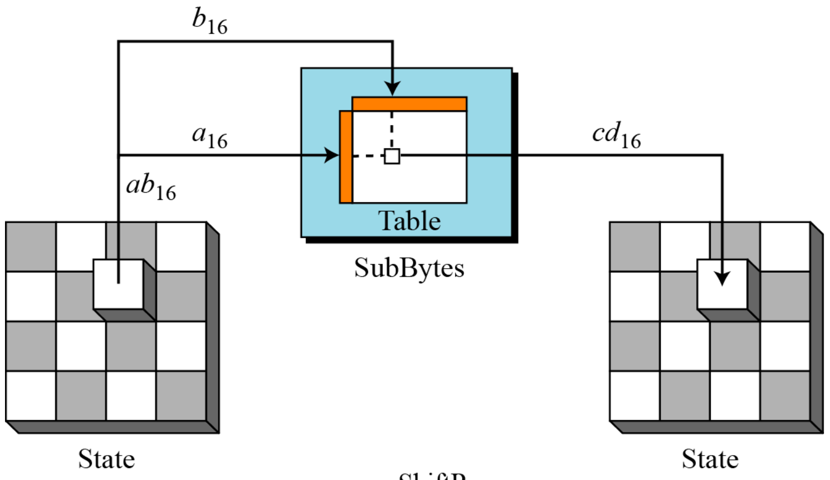
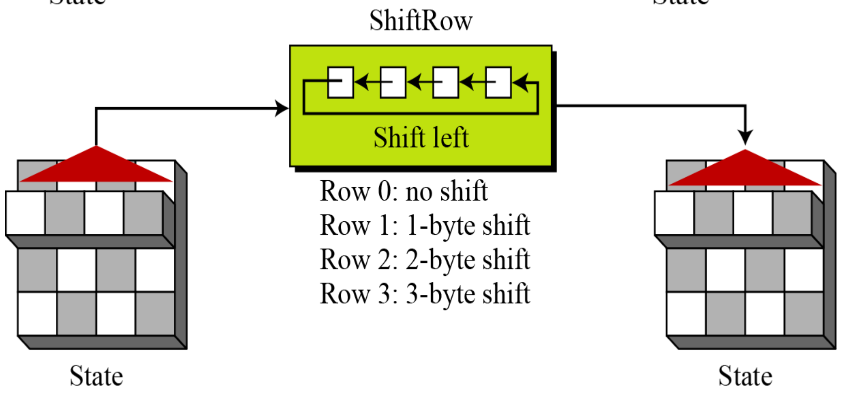
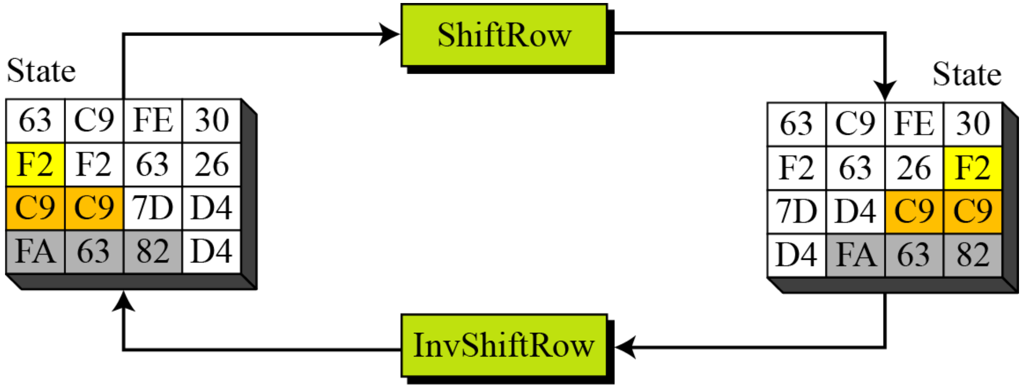


Table 7.1 *SubBytes transformation table*

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	63	7C	77	7B	F2	6B	6F	C5	30	01	67	2B	FE	D7	AB	76
1	CA	82	C9	7D	FA	59	47	F0	AD	D4	A2	AF	9C	A4	72	C0
2	B7	FD	93	26	36	[illegible]	F7	CC	34	A5	E5	F1	71	D8	31	15
3	04	C7	23	C3	18	96	05	9A	07	12	80	E2	EB	27	B2	75
4	09	83	2C	1A	1B	6E	5A	A0	52	3B	D6	B3	29	E3	2F	84
5	53	D1	00	ED	20	FC	B1	5B	6A	CB	BE	39	4A	4C	58	CF
6	D0	EF	AA	FB	43	4D	33	85	45	F9	02	7F	50	3C	9F	A8

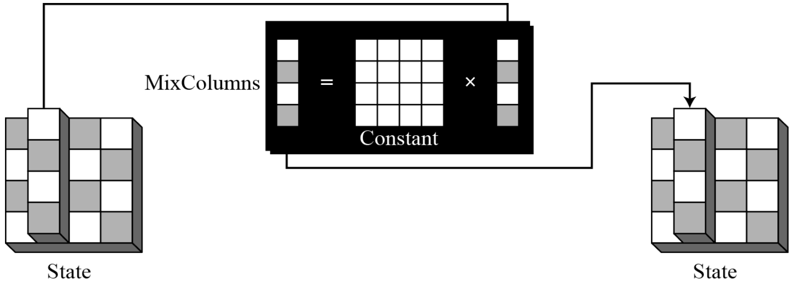




Mixing

Inter-byte transformation

- Changes the bits inside a byte
- Based on the bits inside the neighboring bytes
- Mix bytes to provide diffusion at the bit level



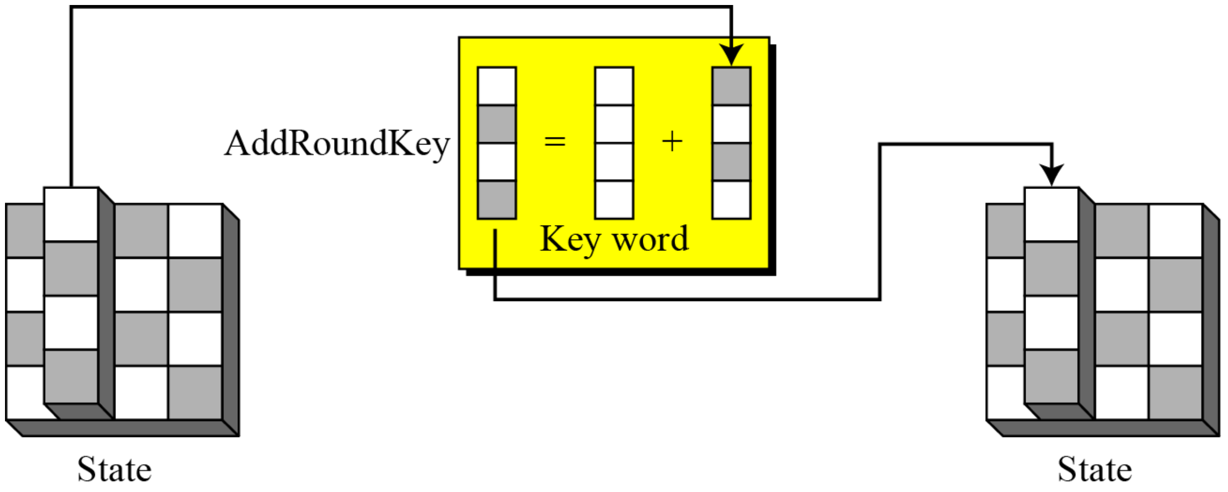
$$\begin{bmatrix} a\mathbf{x} + b\mathbf{y} + c\mathbf{z} + d\mathbf{t} \\ e\mathbf{x} + f\mathbf{y} + g\mathbf{z} + h\mathbf{t} \\ i\mathbf{x} + j\mathbf{y} + k\mathbf{z} + l\mathbf{t} \\ m\mathbf{x} + n\mathbf{y} + o\mathbf{z} + p\mathbf{t} \end{bmatrix} = \begin{bmatrix} a & b & c & d \\ e & f & g & h \\ i & j & k & l \\ m & n & o & p \end{bmatrix} \times \begin{bmatrix} \mathbf{x} \\ \mathbf{y} \\ \mathbf{z} \\ \mathbf{t} \end{bmatrix}$$

New matrix | **Constant matrix** | Old matrix

Key Adding

AddRoundKey proceeds one column at a time

- Adds a round key word with each state column matrix
- The operation is a matrix addition



AES Security

AES was designed after DES
- AES can be easily implemented
  - Cheap processors and minimum amount of memory
- Known attacks on DES were already tested on AES

Brute-Force Attack
- AES is definitely more secure than DES
- The key is larger

Statistical Attacks
- Many tests failed to do statistical analysis of the ciphertext

Differential and Linear Attacks
- There are no differential and linear attacks on AES as yet

Block Ciphers

MODES OF OPERATIONS

Encryption Modes Motivation

What if the message size shorter or larger than the block size?

- Say, message Size = 224-bit
- Block Cipher Supported = 64-bit DES
- Block Cipher Supported = 128-bit AES

- Adapt cryptographic algorithm to applications
- Increase the strength of a cryptographic algorithm
- It is necessary to divide bigger plaintext into fixed sized blocks so that cipher can work on it (i.e.DES-64bit)

Conventional Modes of Operations

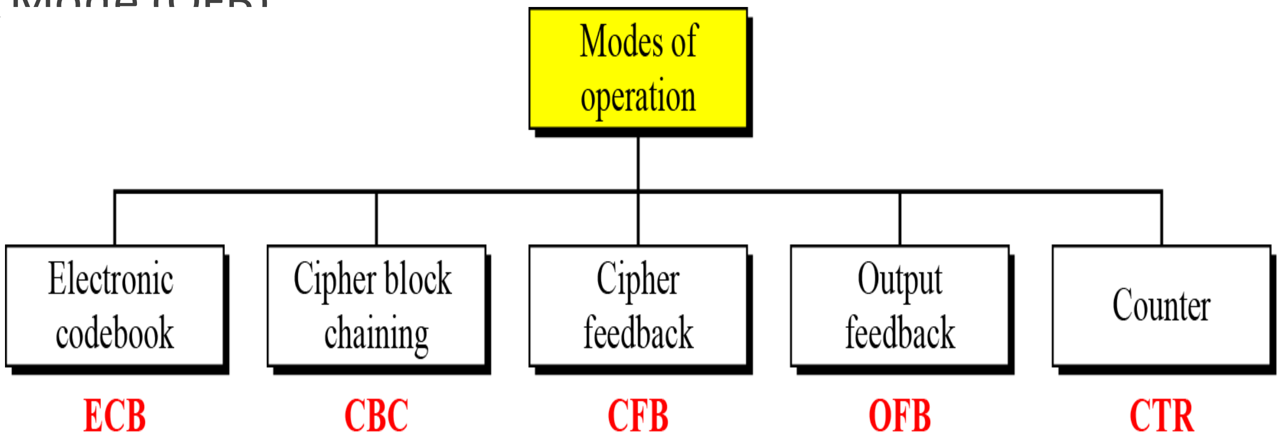
Electronic Codebook Mode (ECB)

Cipher Block Chaining Mode (CBC)

Counter Mode (CTR)

Cipher Feedback Mode (CFB)

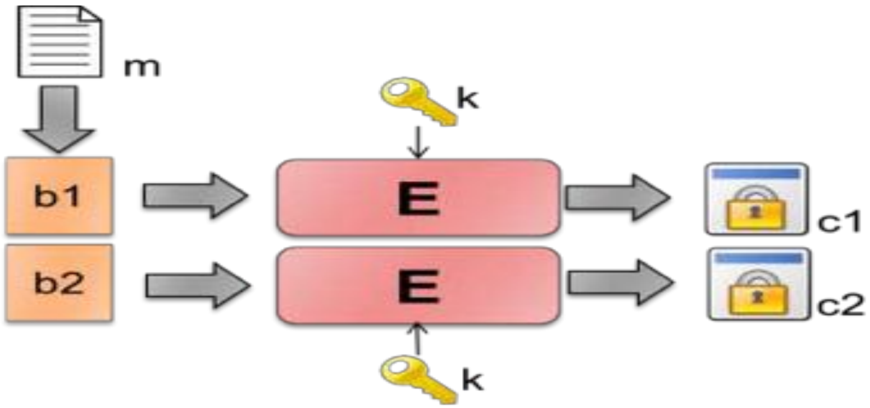
Output Feedback Mode (OFB)



Electronic CodeBook

Message is broken into independent blocks

- Each block is encrypted
- each block is a value which is substituted
  - Like a codebook, hence name
- Each block is encoded independently of the other blocks
- Uses:
  - Secure transmission of single values



Electronic CodeBook: Limitations

Message repetitions may show in ciphertext

- If aligned with message block
- Particularly with data such graphics
- With messages that change very little
  - Become a code-book analysis problem

Weakness due to encrypted blocks independent

Main use is sending a few blocks of data

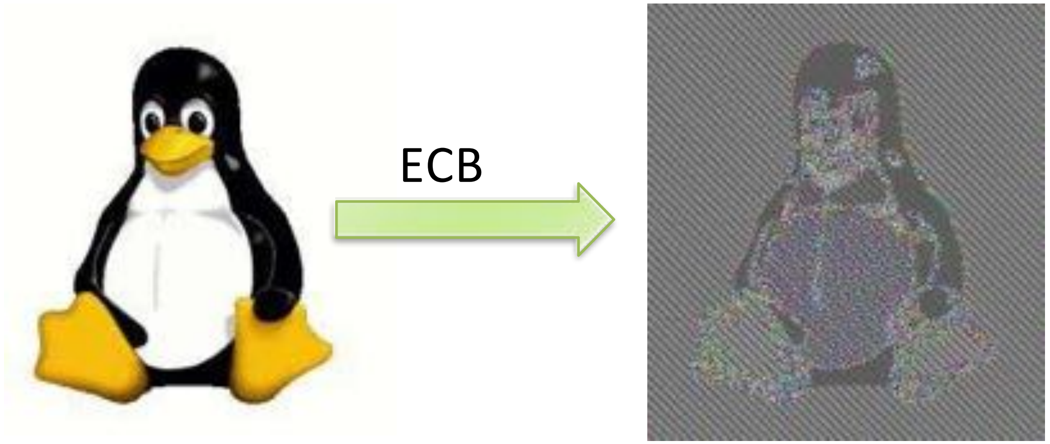
Electronic CodeBook: Limitations

Does not hide data patterns

- Unsuitable for long messages
- Wiki example: pixel map using ECB

Susceptible to replay attacks

- Example: a wired transfer transaction can be replayed by resending the original message



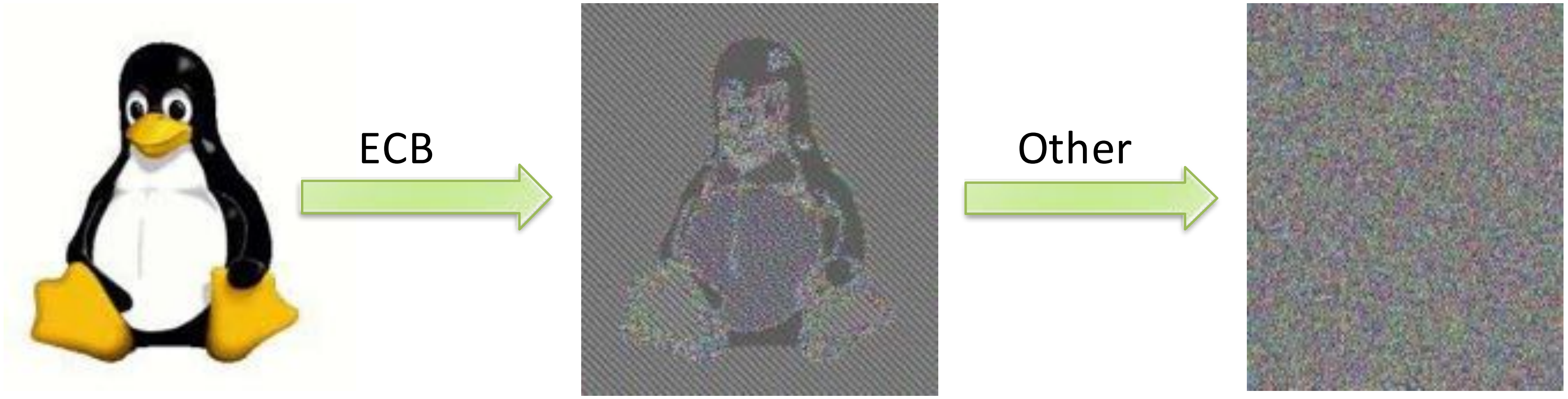
Electronic CodeBook: Limitations

Does not hide data patterns

- Unsuitable for long messages
- Wiki example: pixel map using ECB

Susceptible to replay attacks

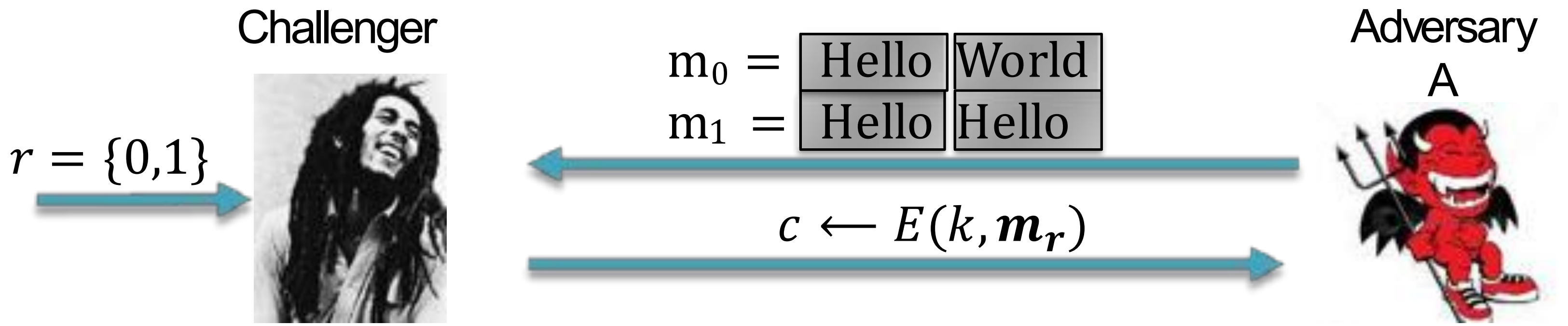
- Example: a wired transfer transaction can be replayed by resending the original message



ECB: Semantic Security

Given algorithm A

- That compares ciphertexts



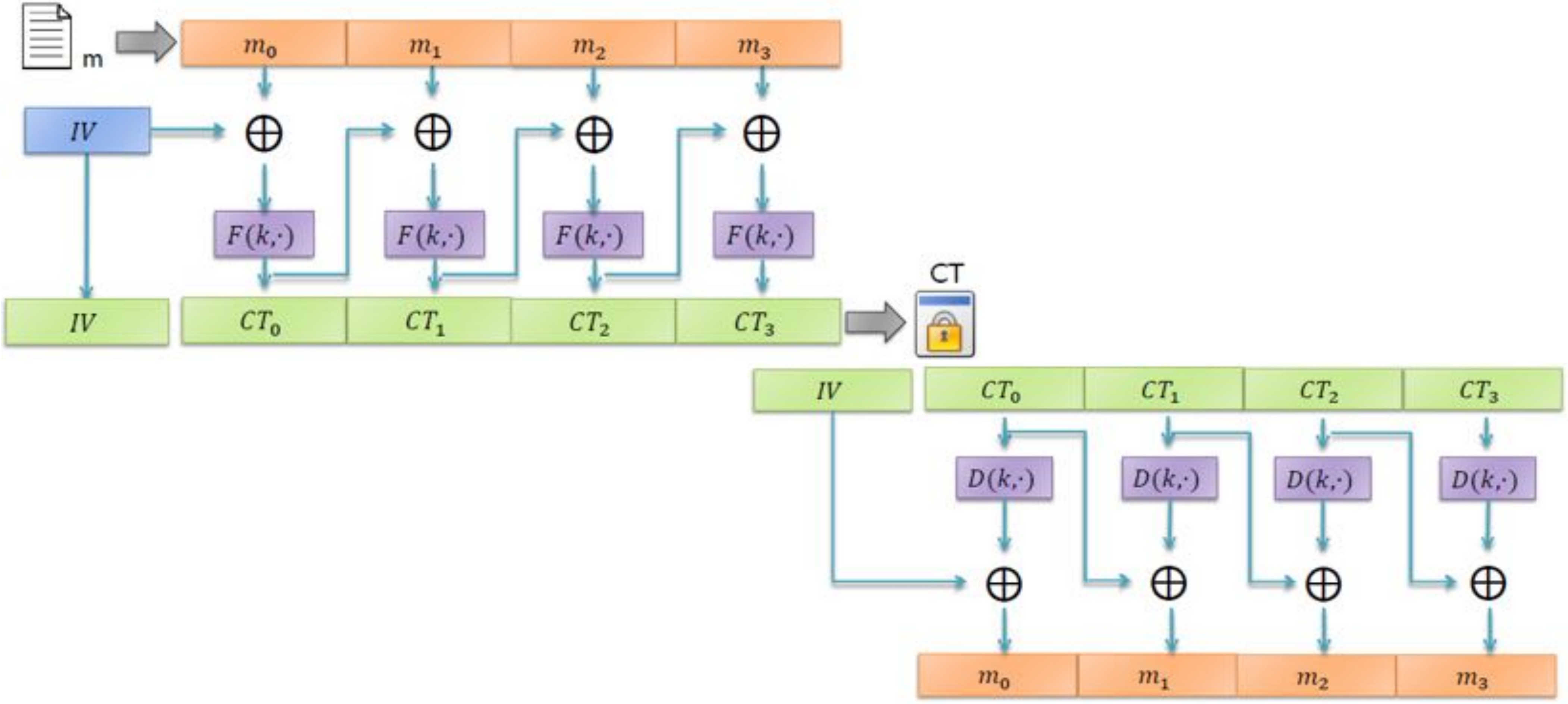
- A: if $c_1 = c_2$ output 0; if $c_1 \neq c_2$ output 1
- $Adv[A, ECB] = 1$
  - Adversary distinguishes between m_0 and m_1

Cipher Block Chaining

Message is broken into blocks

- Linked together in encryption operation
- Each previous cipher blocks is chained with current plaintext block, hence name
- Use Initial Vector (IV) to start process
  - $C_i = E_K(P_i \ XOR \ C_{i-1})$
  - $C_{-1} = IV$
- Uses:
  - Bulk data encryption
  - Authentication

CBC Encryption/Decryption



CBC: Advantages and Limitations

A ciphertext block depends on all blocks before it

- Any change to a block affects all following blocks

Need Initialization Vector (IV)

- Must be known to sender & receiver
- If sent in clear, attacker can change bits of first block and change IV to compensate
- Hence IV must either be
  - A fixed value
  - Must be sent encrypted in ECB mode before rest of message

Stream Modes

Block modes encrypt entire block

- May need to operate on smaller units

Real time data?

- Convert block cipher into stream cipher
  - Cipher Feedback (CFB) mode
  - Output Feedback (OFB) mode
  - Counter (CTR) mode

Use block cipher as pseudo-random generator

Counter Mode

A “new” mode, though proposed early on

- Similar to OFB but encrypts counter value
  - Rather than any feedback value
- Different key & counter value for every plaintext block
  - Never reused!!
- $O_i = E_K(i)$
- $C_i = P_i \, XOR \, O_i$

Uses:

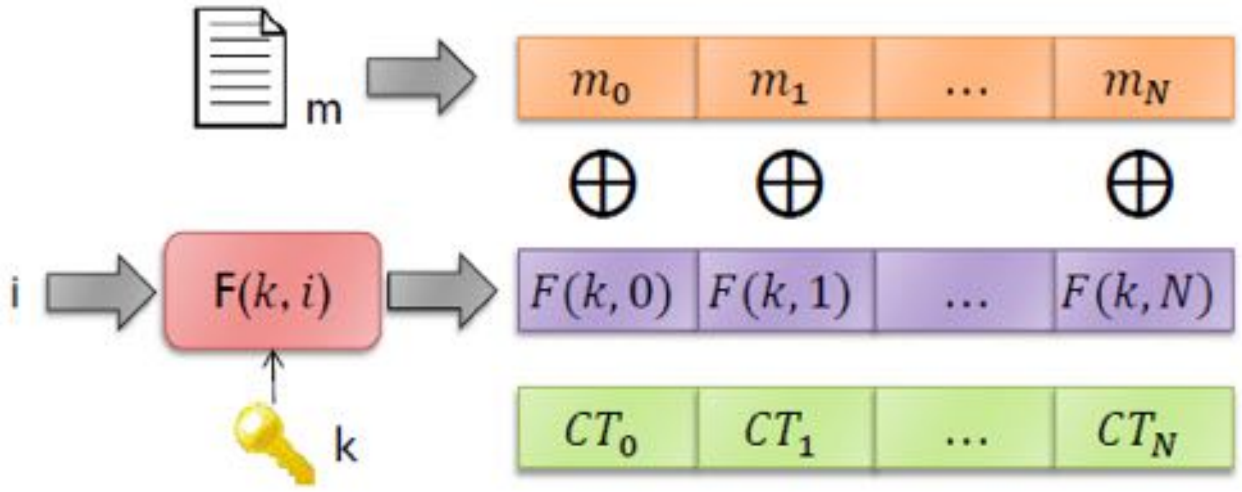
- High-speed network encryptions

CTR Structure

Deterministic Counter Mode

- Chunk the plaintext
- Encrypt a counter
- Encrypt as a stream cipher

- Secure if function $F()$ is secure!



CTR: Advantages and Limitations

Efficiency

- Can do parallel encryptions in h/w or s/w
- Can pre-process in advance of need
- Good for bursty high speed links

Random access to encrypted data blocks

Provable security (good as other modes)

But must ensure never reuse key/counter values

- Otherwise could break