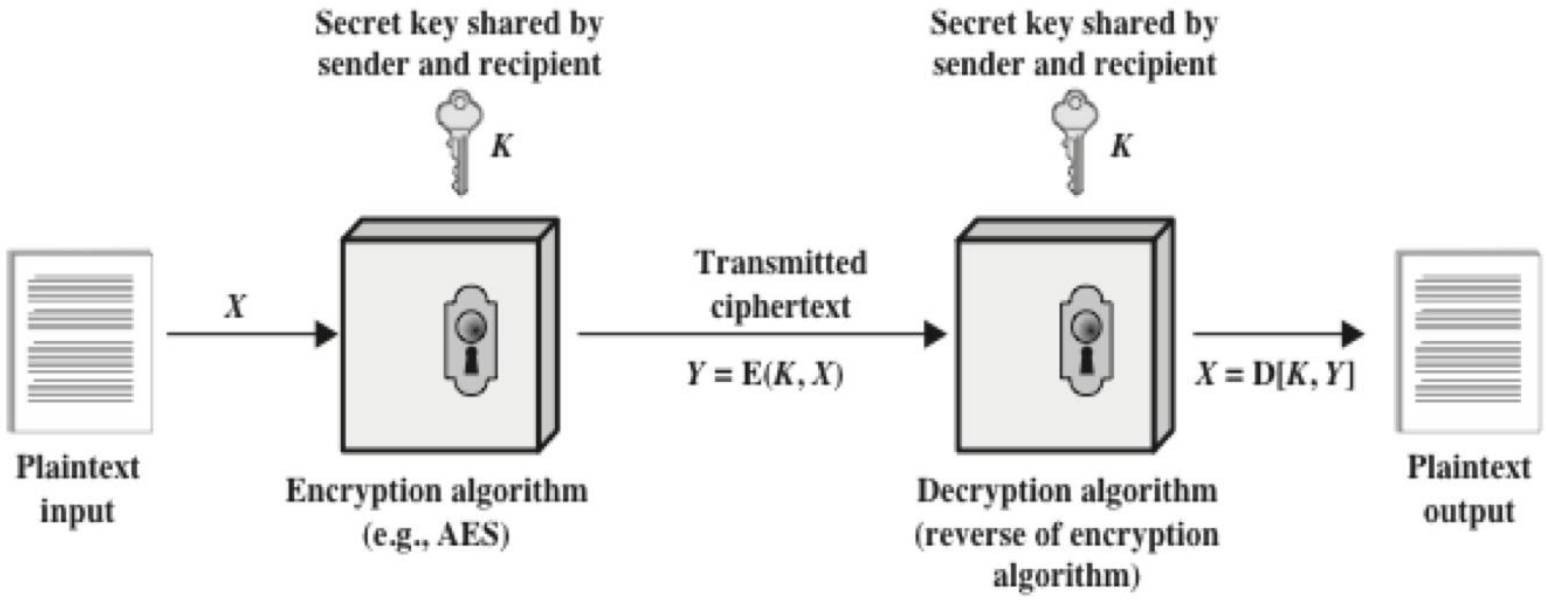
Network Security
Symmetric Crypto

Amir Rezapour
Institute of Information Security,
National Tsing Hua University

Symmetric Crypto

- The classic task of cryptography is to encrypt data for secrecy.
- E/D: Encryption/Decryption method
- k: a key from key space K
- Plaintext $m \xrightarrow{E,k}$ ciphertext c
- Ciphertext $c \xrightarrow{D,k}$ plaintext m
- For each $k \in K$ and $m \in M, D\big(k, E(k, m)\big) = m$
- $E_k^{-1} = D_k$
- E and D are poly-time computable.

Symmetric Crypto



Basic Principles

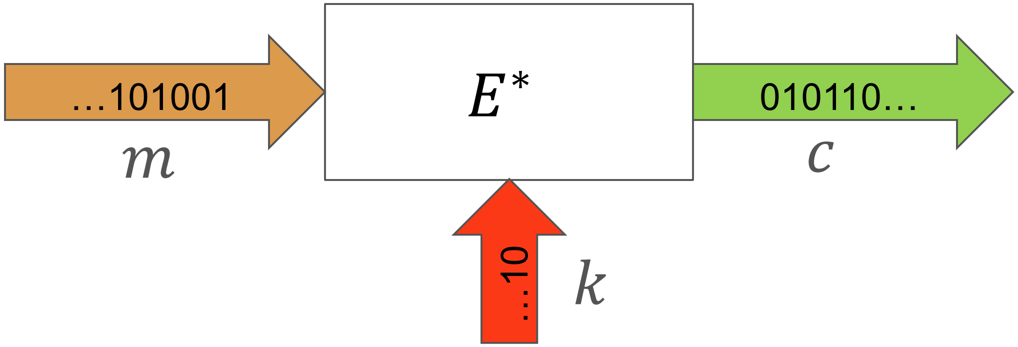
- Kerckhoff’s Principle:
  - A cryptosystem should be secure even if everything about the system, except the key, are publicly known.
- Shannon's Maxim:
  - Your enemy knows your system!
- The security of symmetric encryption depends on the secrecy of the key, not the secrecy of the algorithm

Symmetric Crypto Types

- Block cipher
  - Block by block
  - Each block is fixed-length groups of bits
  - Each block is encrypted <u>with the same key</u>

32	88	31	e0
43	5a	31	37
f6	30	98	07
a8	8d	a2	34

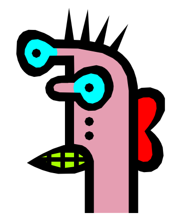
- Stream cipher
  - Bit by bit, character by character
  - Each bit/character is encrypted <u>with a different key</u>
  - $E^*(m,k) = c_1c_2$ ..., where $m = m_1m_2$ ..., $k = k_1k_2$ ..., and $E(m_i,k_i) = c_i$



Cryptanalysis

- Ciphertext-Only Attack (COA)
  - Attackers have access only to a set of ciphertexts
  - Given E, D, $\{c_1, c_2, \ldots, c_n\}$, and $c = E(k, m)$, compute m.


Adversary


Crypto Scheme

$c = E(k, m)$ ←

$c_1 = E(k, m_1)$ ←

$c_2 = E(k, m_2)$ ←

$\vdots$

$c_n = E(k, m_n)$ ←

$m = ?$

Cryptanalysis

- Ciphertext-Only Attack (COA)
  - *Example: Mono-alphabetic Cipher:* encrypts English text by mapping the alphabets to a chosen permutation

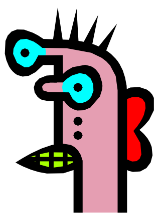
Plaintext Alphabet	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
Ciphertext Alphabet	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A

  - Relatively difficult to break based on exhaustive key search (26! – 1)
  - Easy to break based on letter frequencies of English alphabets

Cryptanalysis

- Known-Plaintext Attack (KPA)
  - Attackers have samples of both the plaintext, and its encrypted version (ciphertext)
  - Given $E, D, \{(m_1, c_1), \ldots, (m_n, c_n)\}$, and $c = E(k, m)$, compute m.

Adversary



Crypto Scheme

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

$$m_1, c_1 = E(k, m_1)$$

$$m_2, c_2 = E(k, m_2)$$

$$\vdots$$

$$m_n, c_n = E(k, m_n)$$

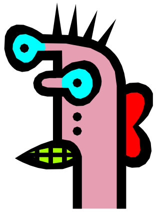
$m =?$

Cryptanalysis

- Known-Plaintext Attack (KPA)
  - Example: Easy to break *Mono-alphabetic Cipher,* if known plaintext-ciphertext pairs contain all alphabets

Plaintext Alphabet	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
Ciphertext Alphabet	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A

Adversary



Mono-alphabetic

$IFMMP = E(k, hello)$

$whom, XIPN = E(k, whom)$

$lamp, MBNQ = E(k, lamp)$

$\vdots$

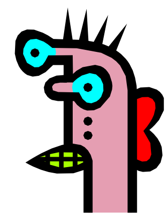
$red, SFE = E(k, red)$

$m =$hello

Cryptanalysis

- Chosen-Plaintext Attack (CPA)
  - Attackers have the capability to choose arbitrary plaintexts to be encrypted and obtain the corresponding ciphertexts
  - Given $E, D, \{(m_1, c_1), \ldots, (m_n, c_n) | \, m_i \; is \; choosen \; by \; attacker\}$, and $c = E(k, m)$, compute m.

Adversary



Crypto Scheme

$c = E(k, m)$ (Crypto Scheme → Adversary)

m_1 (Adversary → Crypto Scheme)

$c_1 = E(k, m_1)$ (Crypto Scheme → Adversary)

Plaintext query

$\vdots$

m_n (Adversary → Crypto Scheme)

$c_n = E(k, m_n)$ (Crypto Scheme → Adversary)

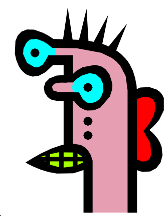
$m = ?$

Cryptanalysis

- Chosen-Plaintext Attack (CPA)
  - Easy to break *Mono-alphabetic Cipher* by having the corresponding ciphertext of plaintext *abcd…xyz* or any sub-string of 25 alphabets

Plaintext Alphabet	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
Ciphertext Alphabet	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A

Adversary



Mono-alphabetic

$IFMMP = E(k, hello)$

$m_1 = abcd..xyz$

$BCDE \ldots YZA = E(k, m_1)$

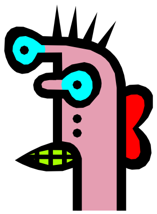
Plaintext query

$m =$ hello

Cryptanalysis

- Chosen-Ciphertext Attack (CCA): Lunch time attack
  - Attackers have the capability to choose a ciphertext and obtaining its plaintext
  - Given $E, D, \{(m_1, c_1), \dots, (m_n, c_n) | \; c_i \; is \; choosen \; by \; attacker\}$, and $c = E(k, m)$, compute m.

Adversary



Crypto Scheme

$c = E(k, m)$ ←

c_1 →

$m_1 = D(k, c_1)$ ←

Ciphertext query

⋮

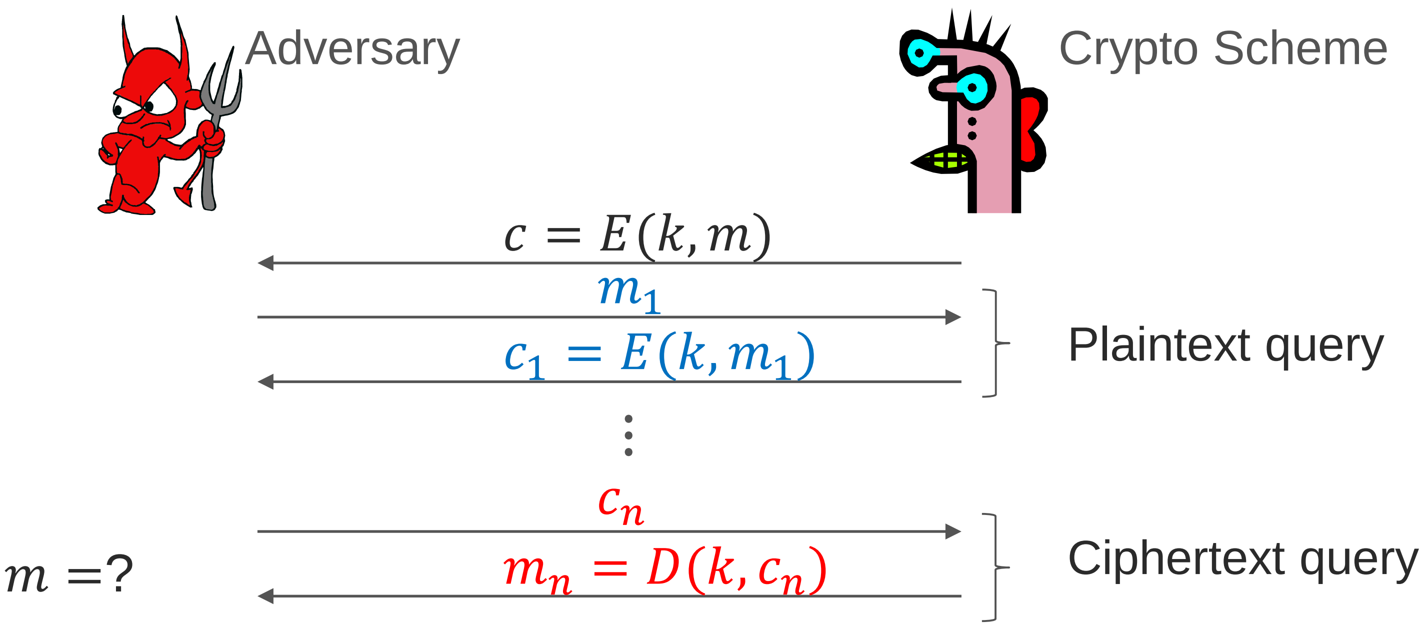
c_n →

$m_n = D(k, c_n)$ ←

$m = ?$

Cryptanalysis

- Chosen text
  - Combination of CPA and CCA
  - Give $E, D, \{(m_1, c_1), \ldots, (m_n, c_n) | \, m_i \; or \; c_i \; is \; choosen \; by \; attacker\}$, and $c = E(k, m)$, compute m.



Cryptanalysis

- An encryption scheme is **computationally secure** if the ciphertext generated by the scheme meets one or both of the following criteria:
  - The cost of breaking the cipher exceeds the value of the encrypted information
  - The time required to break the cipher exceeds the useful lifetime of the information

Brute Force attack

- Involves trying every possible key until an *intelligible* translation of the ciphertext into plaintext is obtained.
- On average, half of all possible keys must be tried to achieve success
  - $|K| = n$, 2^n possible keys. On average 2^{n-1} tries!
- Unless known plaintext is provided, the analyst must be able to recognize plaintext as plaintext
- To supplement the brute-force approach
  - Some degree of knowledge about the expected plaintext is needed
  - Some means of automatically distinguishing plaintext from garble is also needed

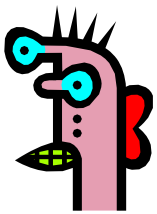
Example: Vernam's one-time pad

- Alice and Bob share a secret key k.
  - k is truly random and used only once
- $E(k, m) = m \oplus k$
- Why use key only once?
  - $c_1 \oplus c_2 = m_1 \oplus k \oplus m_2 \oplus k = m_1 \oplus m_2$
  - $1010 \oplus 0101$=1111, each bit of m_1 and m_2 in every position is different!
- Why is it secure?
  - $m \oplus k = c$, what is m?
  - Secure against ciphertext only

Example: Vernam's one-time pad

- Secure against chosen-plaintext attack?

Adversary



One-time pad

m =?

$c = 1001$

KeyGen()

$m_1 = 1010$

k_1 =0111

$c_1 = 1101$

Plaintext query

$m_2 = 0100$

k_2 =0011

$c_1 = 0111$

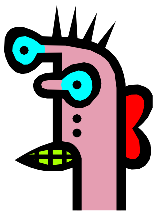
⋮

Outputs $m = 0101$, correct/wrong?

Example: Vernam's one-time pad

- Secure against chosen-ciphertext attack?

Adversary



One-time pad

$m = ?$ $\xleftarrow{c\, =\, 1001}$

KeyGen()

$k_1 = 1001$

$\xrightarrow{c_1\, =\, 0100}$

$\xleftarrow{m_1\, =\, 1101}$

Ciphertext query

$k_2 = 1010$

$\xrightarrow{c_2\, =\, 1111}$

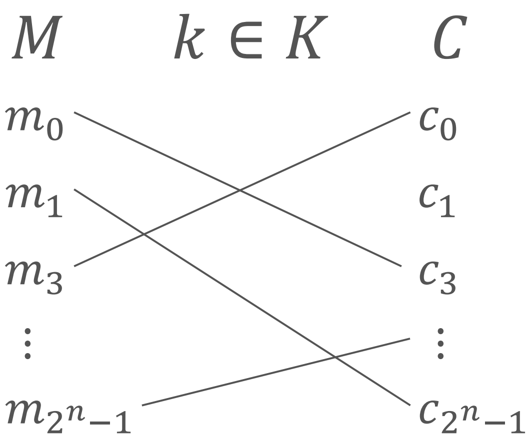
$\xleftarrow{m_2\, =\, 0101}$

$\vdots$

Outputs $m = 1010$, correct/wrong?

Block Ciphers: in abstract

- $M = C = \{0,1\}^n$
- K is a set of permutation from $\{0,1\}^n$ to $\{0,1\}^n$



- $k \in K$ is a 1-1 mapping
- # of permutations: $2^n \times (2^n - 1) \times \cdots \times (1) = 2^n!$

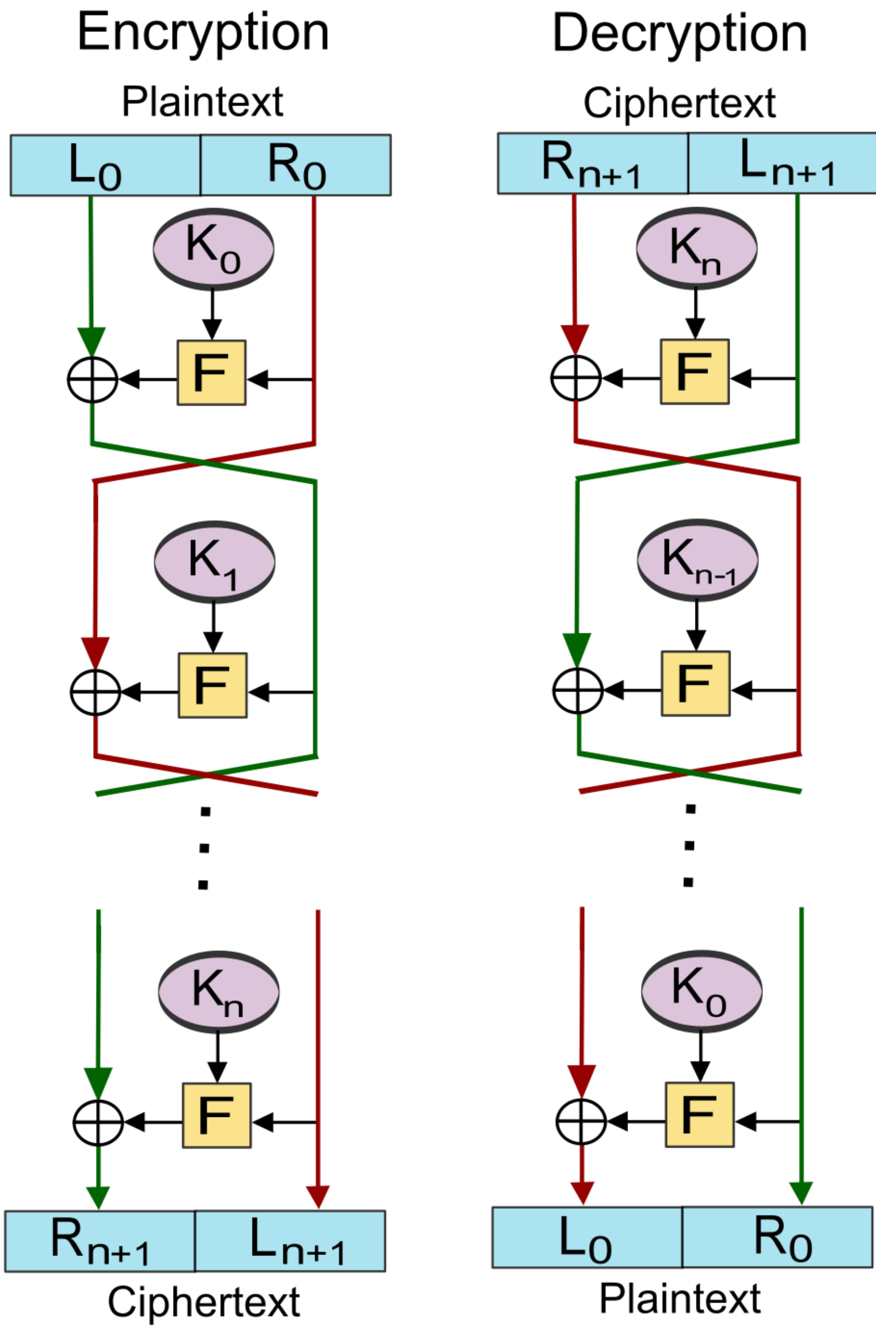
Block Ciphers: in abstract

- Ideal block cipher: truly random permutation
  - K is a set of all permutation from $\{0,1\}^n$ to $\{0,1\}^n$
  - $|K| = (2^n)! \approx 2^{(n-1.44)2^n}$
  - It takes $(n - 1.44)2^n$ bits to represent a key on average!
    - E.g. in AES, $n = 128$ and $|k \in K| = (128 - 1.44)2^{128}$ Impractical!

- Practical block cipher: pseudorandom random permutation
  - K is a subset of all permutation from $\{0,1\}^n$ to $\{0,1\}^n$
  - $|k \in K|$ is typically 64, 128, 256 bits
  - Design a symmetric-key block cipher is art!

Feistel Structure

- $L_0, R_0 \leftarrow m_0$
- $K_0, K_1, \dots, K_n \leftarrow SubkeyGen(K)$
- round function F



Feistel Cipher Elements

Block size
- Larger block sizes mean greater security but reduced encryption/decryption speed

Key size
- Larger key size means greater security but may decrease encryption/decryption speed

Number of rounds
- The essence of a symmetric block cipher is that a single round offers inadequate security but that multiple rounds offer increasing security

Subkey generation algorithm
- Greater complexity in this algorithm should lead to greater difficulty of cryptanalysis

Round function
- Greater complexity generally means greater resistance to cryptanalysis

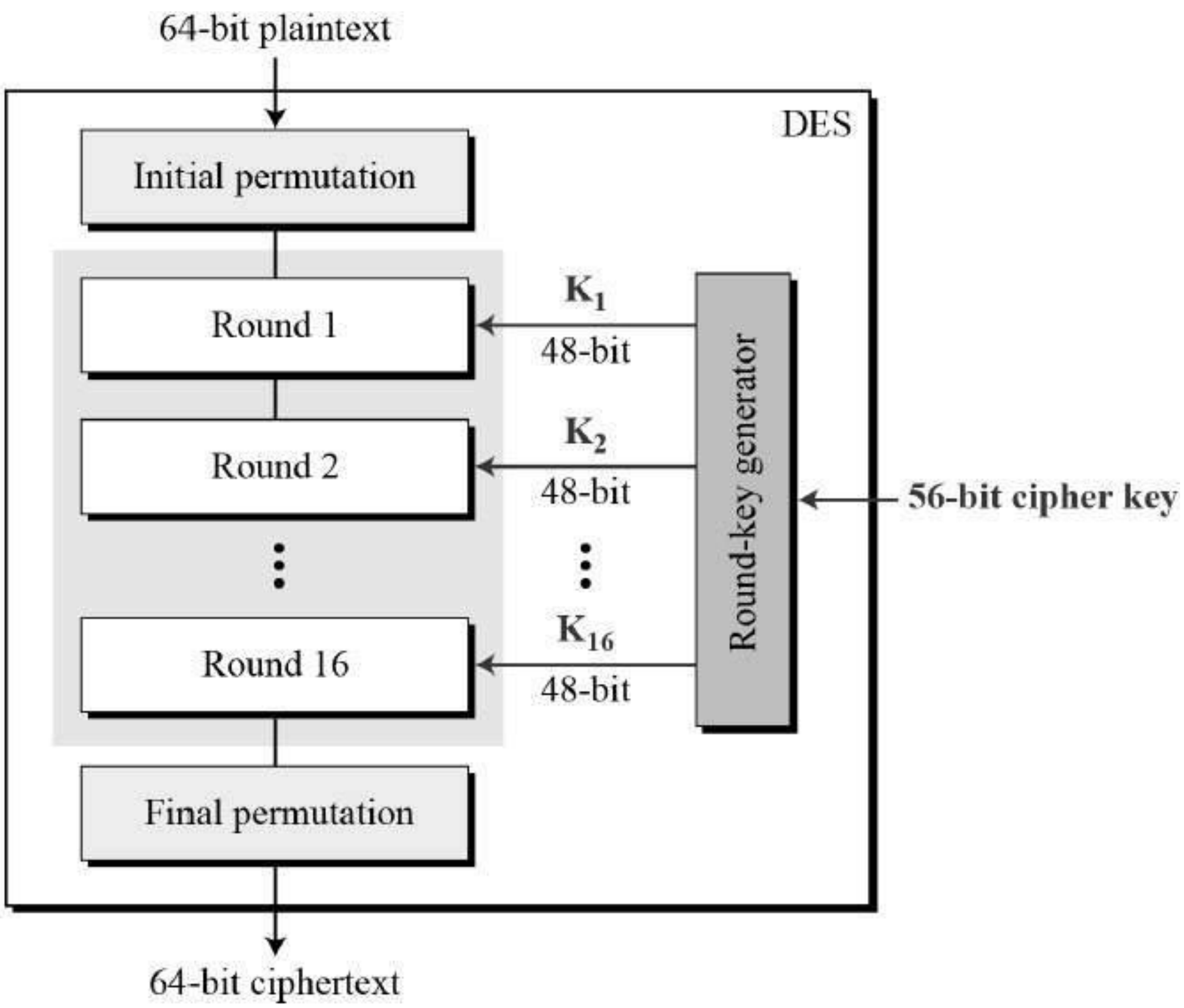
Fast software encryption/decryption
- In many cases, encryption is embedded in applications or utility functions in such a way as to preclude a hardware implementation; accordingly, the seed of execution of the algorithm becomes a concern

Ease of analysis
- If the algorithm can be concisely and clearly explained, it is easier to analyze that algorithm for cryptanalytic vulnerabilities and therefore develop a higher level of assurance as to its strength

DES

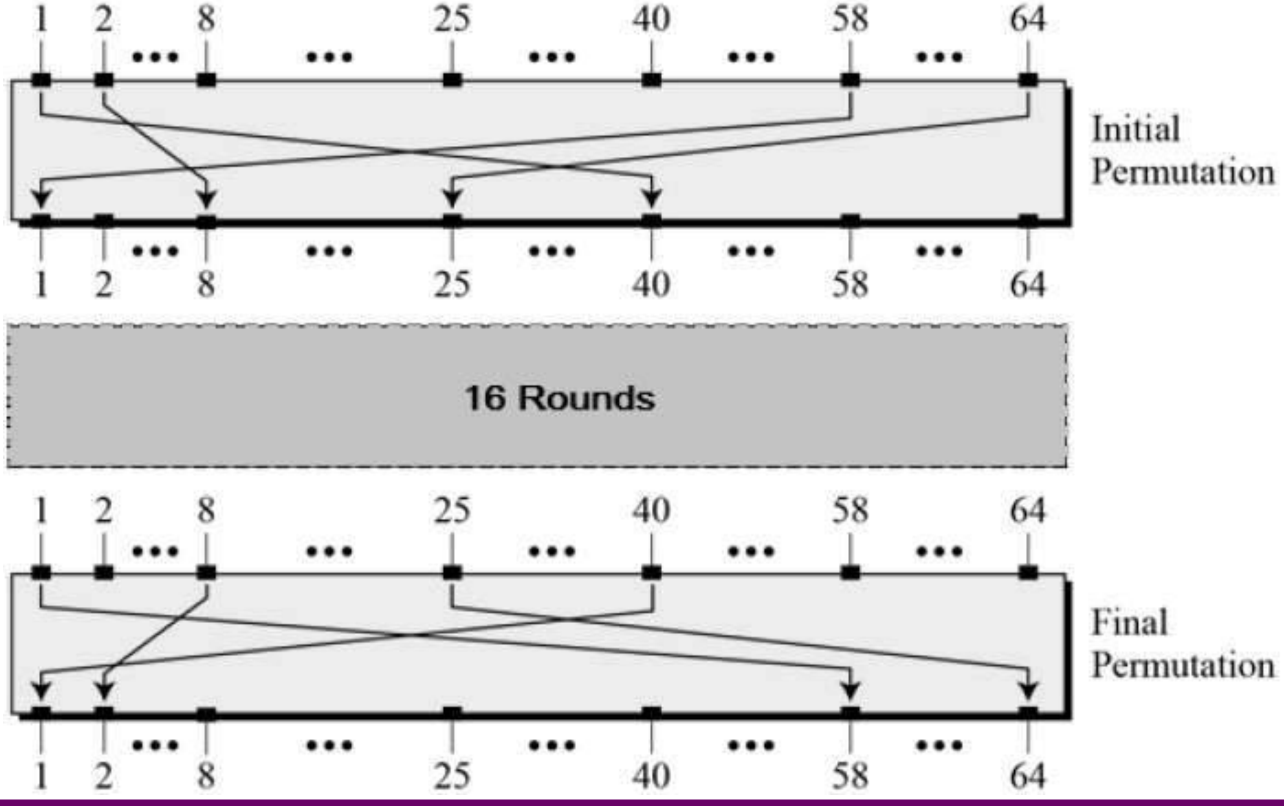
- Data Encryption Standard: USA standard,1977
- $DES: \{0,1\}^{56}, \{0,1\}^{64} \rightarrow \{0,1\}^{64}$
- DES is a pseudorandom random permutation from $\{0,1\}^{64} \rightarrow \{0,1\}^{64}$
- Structure is a minor variation of the Feistel network
- There are 16 rounds of processing
- Process of decryption is essentially the same as the encryption process

DES: Structure



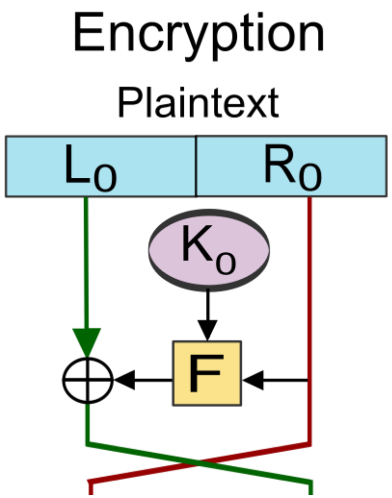
DES: Initial and Final Permutation

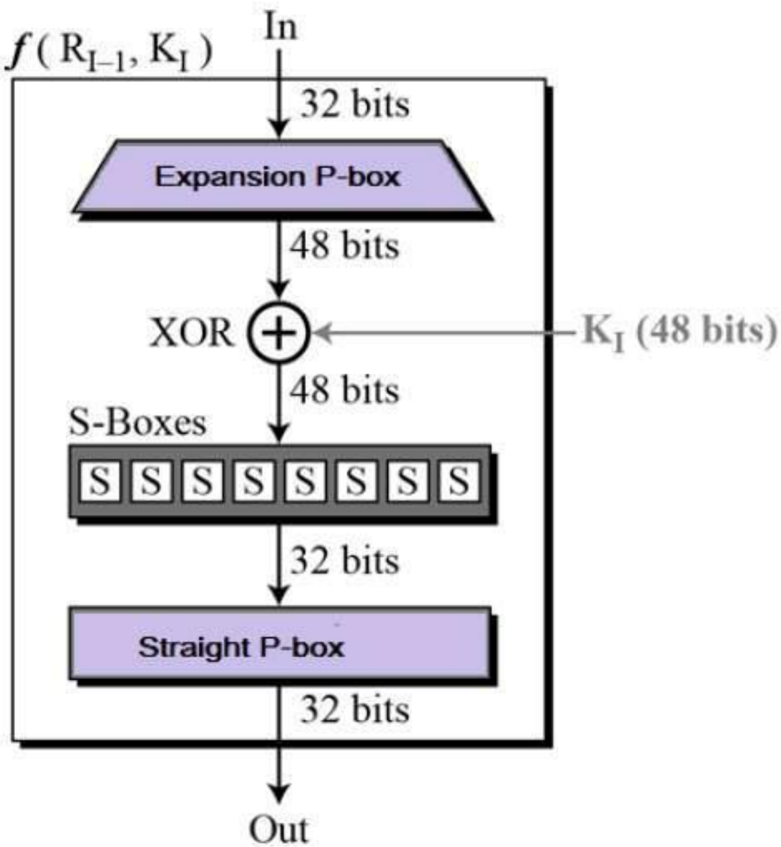
- They are inverses of each other.
- They have no cryptography significance in DES



DES: Round Function

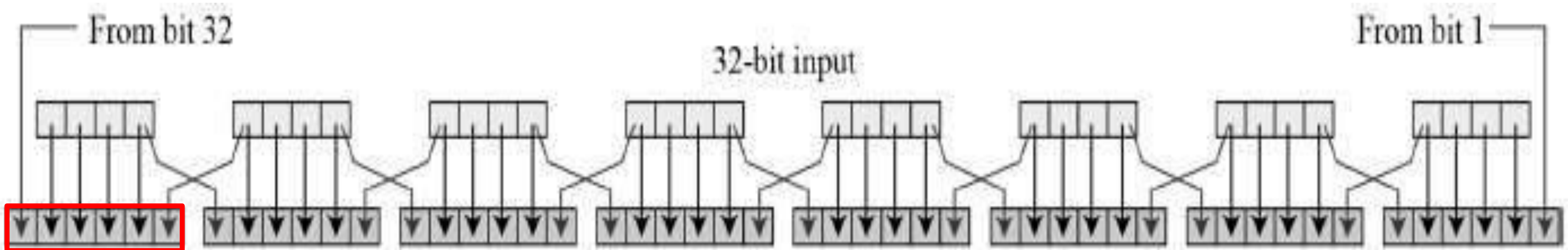
- The heart of this cipher is the DES function F
  - **Expansion Permutation Box**
  - **XOR**
  - **Substitution Boxes**





DES: Expansion Permutation Box

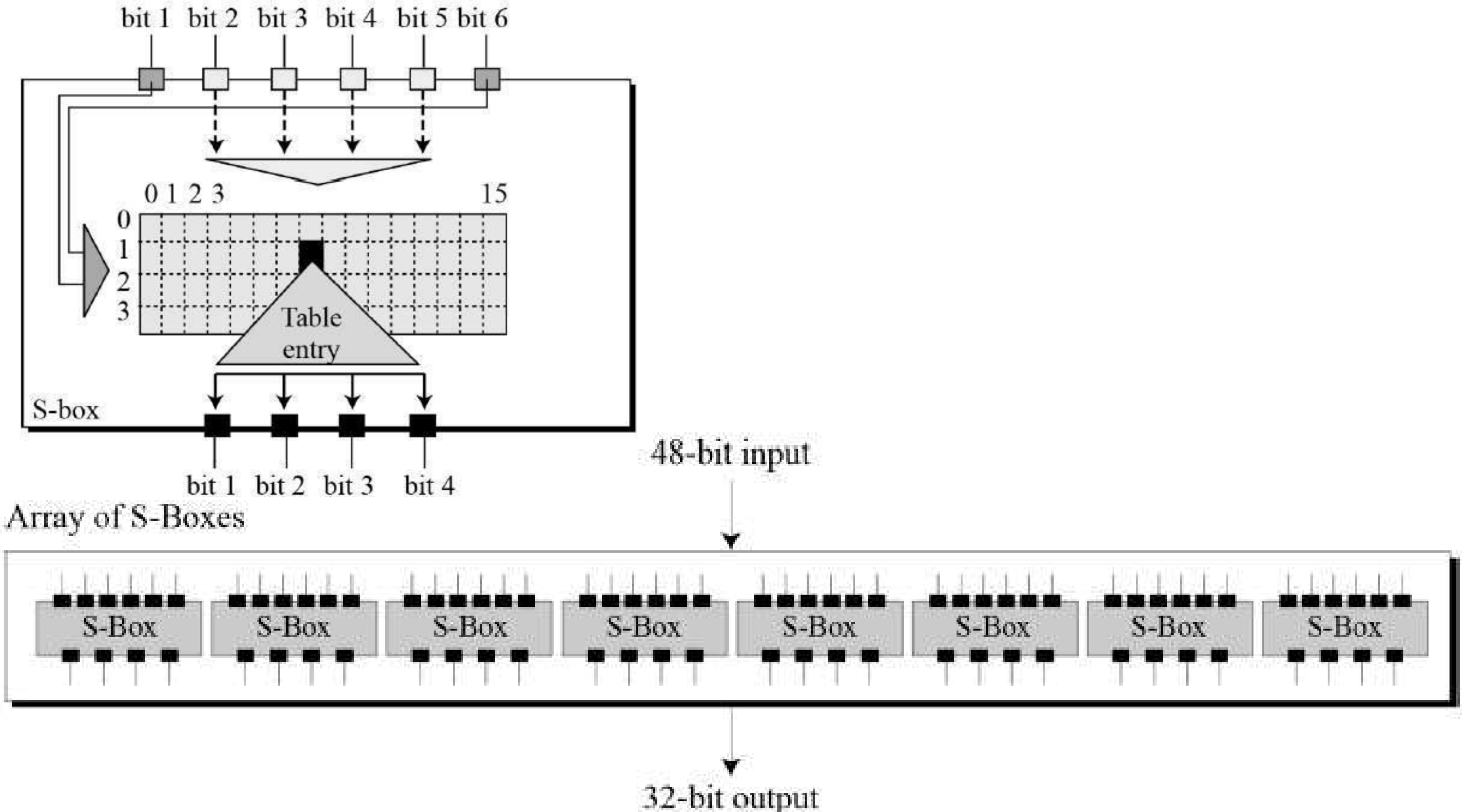
- We first need to expand right input to 48 bits



32	01	02	03	04	05
04	05	06	07	08	09
08	09	10	11	12	13
12	13	14	15	16	17
16	17	18	19	20	21
20	21	22	23	24	25
24	25	26	27	28	29
28	29	31	31	32	01

DES: Substitution Boxes

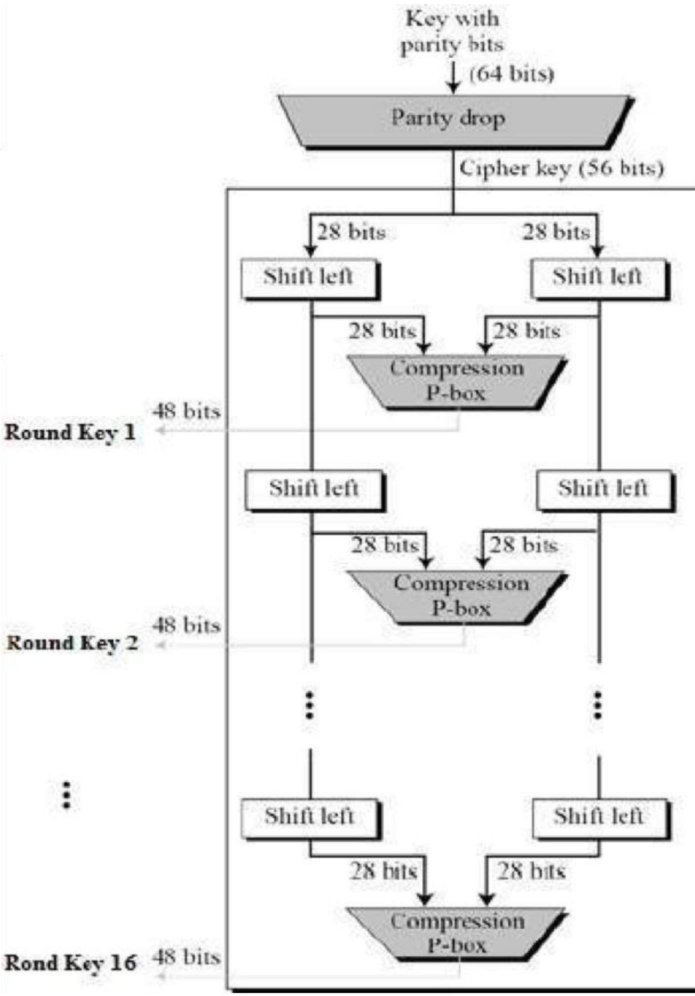
- S-boxes carry out the real mixing (confusion).
  - There are 8 different S-Boxes



DES: KeyGen

- Expand 56-bit key to 16 × 48-bit keys
- Parity drop
  - Discard the following bit positions:
  - 8, 16, 24, 32, 40, 48, 56 and 64
- Circular Shifted Left
  - Rounds 1,2,9, 16 one bit shift
  - Two bit shift for the other rounds
- Compression P-box

14	17	11	24	1	5	3	28	15	6	21	10
23	19	12	4	26	8	16	7	27	20	13	2
41	52	31	37	47	55	30	40	51	45	33	48
44	49	39	56	34	53	46	42	50	36	29	32



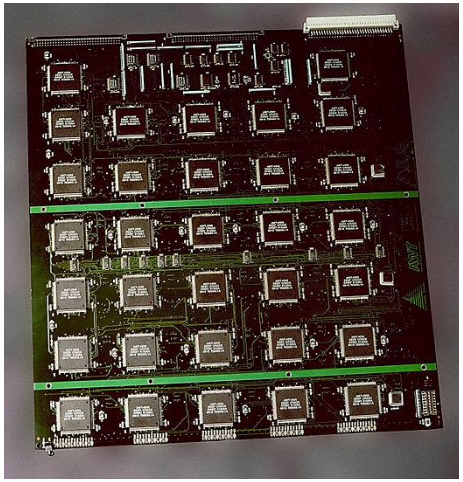
DES

- The strength of DES:
  - **Diffusion (a.k.a. Avalanche effect):** A small change in plaintext results in the very great change in the ciphertext.
    - It hides the relationship between the ciphertext and the plain text.
  - **Confusion:** Each bit of ciphertext depends on many bits of key.
    - It hides the relationship between the ciphertext and the key.
    - This property makes it difficult to find the key from the ciphertext.

DES

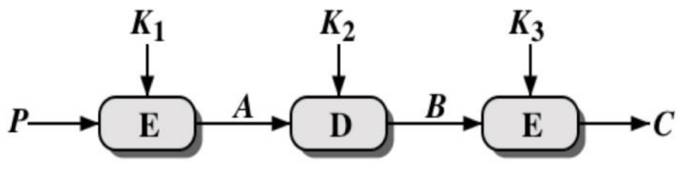
- The strength of DES:
  - The use of a 56-bit key
    - Speed of commercial, off-the-shelf processors threatens the security
    - Example, EFF DES cracker a.k.a. Deep Crack only takes 3 days!

Key size (bits)	Cipher	Number of Alternative Keys	Time Required at 10^9 decryptions/s	Time Required at 10^{13} decryptions/s
56	DES	$2^{56} \approx 7.2 \times 10^{16}$	2^{55} ns = 1.125 years	1 hour
128	AES	$2^{128} \approx 3.4 \times 10^{38}$	2^{127} ns = 5.3×10^{21} years	5.3×10^{17} years
168	Triple DES	$2^{168} \approx 3.7 \times 10^{50}$	2^{167} ns = 5.8×10^{33} years	5.8×10^{29} years
192	AES	$2^{192} \approx 6.3 \times 10^{57}$	2^{191} ns = 9.8×10^{40} years	9.8×10^{36} years
256	AES	$2^{256} \approx 1.2 \times 10^{77}$	2^{255} ns = 1.8×10^{60} years	1.8×10^{56} years

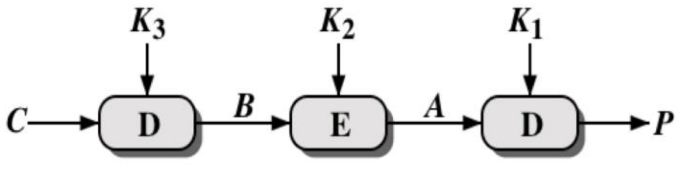


Triple DES

- Triple DES: ANSI standard,1999
- $3DES: c = E\left(k_3, D\left(k_2, E(k_1, m)\right)\right)$
- $|k| = 168$-bit long
- EDE idea is to support older single DES hardware:
  - $c = E\left(k_1, D\left(k_1, E(k_1, m)\right)\right) = E(k_1, m)$



(a) Encryption



(b) Decryption

Why not 2DES?

- $2DES: c = E(k_2, E(k_1, m))$
- $D(k_2, c) = D(k_2, E(k_2, E(k_1, m))) = E(k_1, m)$
- Meet-in-the-middle attack
  - Suppose attacker knows above c and m
  - Create Encryption table as follows.
    - Encrypt m with all 2^{56} possible keys.
  - Decrypt c with all 2^{56} possible keys and check the table for matching

$\vdots$	$\vdots$
$E(k_1, m) \rightarrow$	$\leftarrow D(k_2, c)$
$\vdots$	$\vdots$

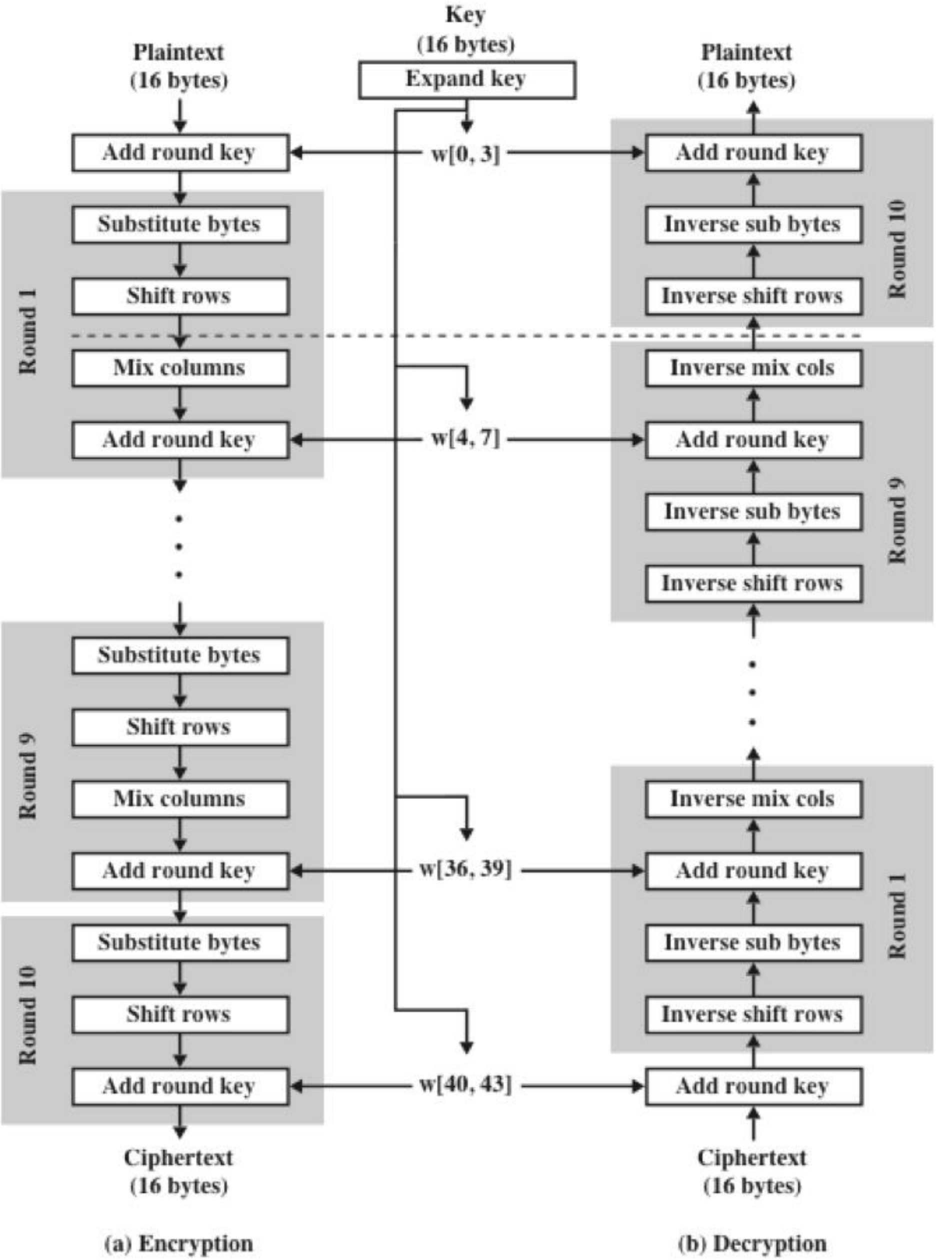
  - It only takes twice as long to break $2DES$, i.e., 2^{57}

AES

- The new US standard AES (Advanced Encryption Standard) was proposed in 2000.
- AES is the Rijndael cipher, proposed by two Belgium scientists.
- M and C are 128-bit long.
- The key length is 128-, 192-, or 256-bit long
- Unlike DES, AES is a byte-oriented cipher.

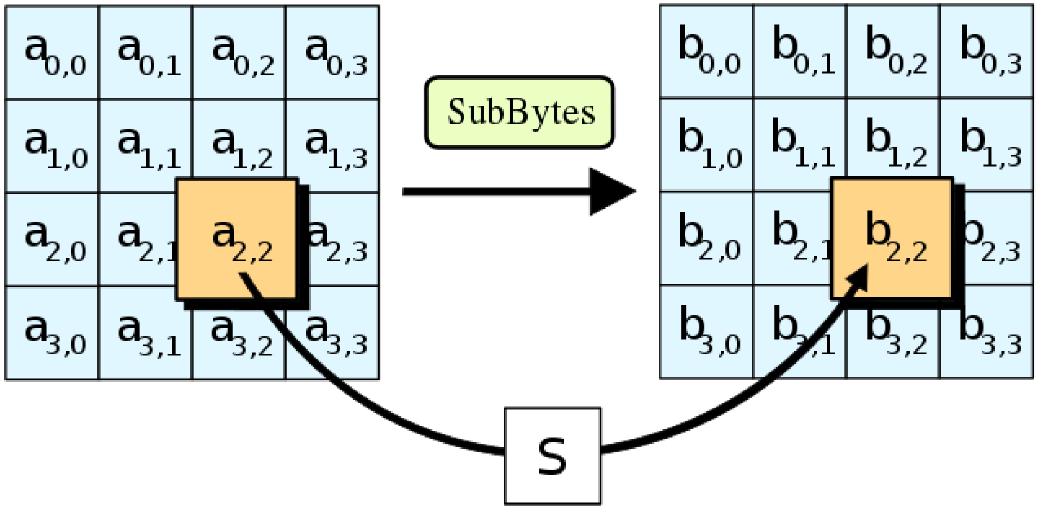
AES: Structure

- $w[0,43] \leftarrow KeyGen(K)$
  - W[i] is 32-bit
- Substitute bytes
  - Uses S-box table
  - Non-linearity properties
- Shift row
  - Permutes row by row
- Mix column
  - Column-oriented substitution
- Add round key
  - $state \leftarrow state \oplus roundkey$



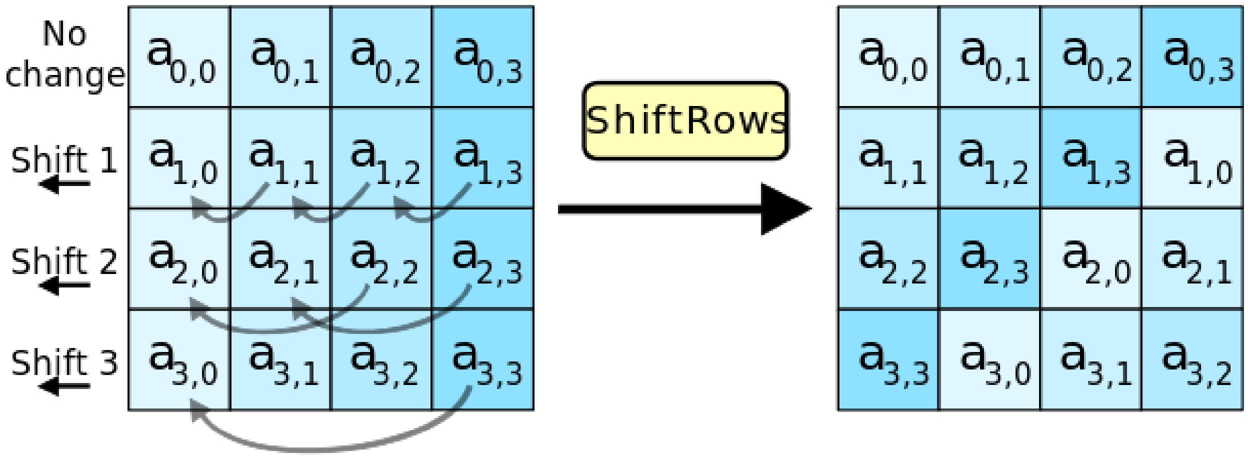
AES: Substitute bytes

- It uses a fixed table (S-box) given in design.
- This operation provides the non-linearity in the cipher.



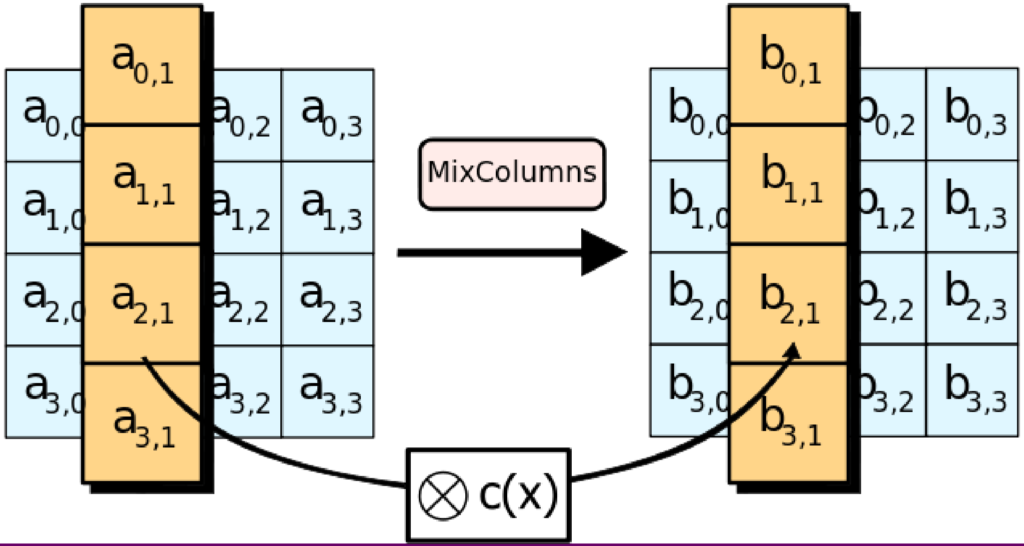
AES: Shift row

- It cyclically shifts the bytes in each row by a certain offset.



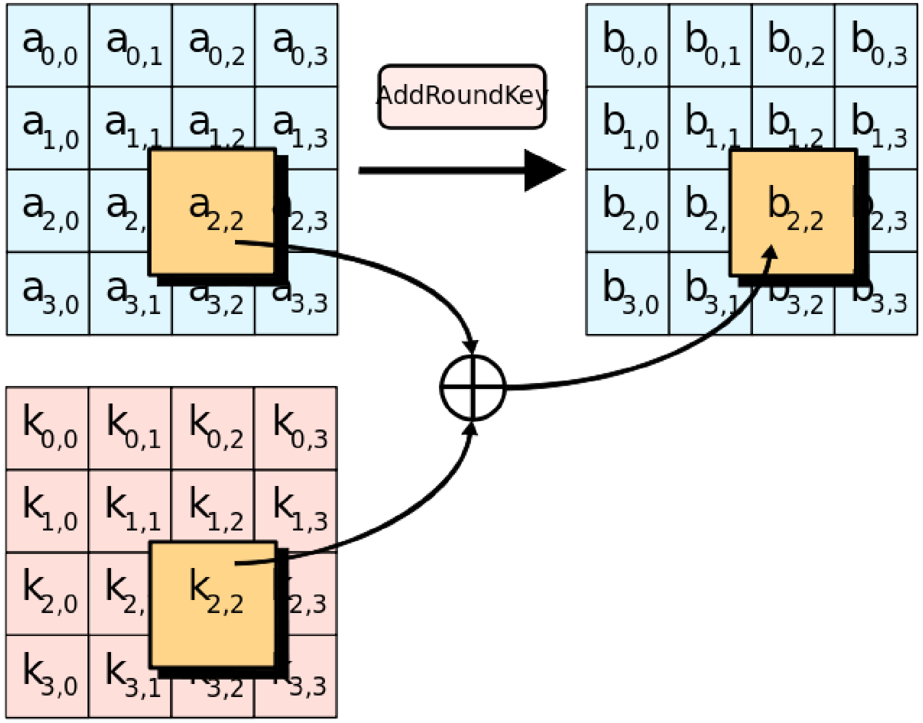
AES: Mix column

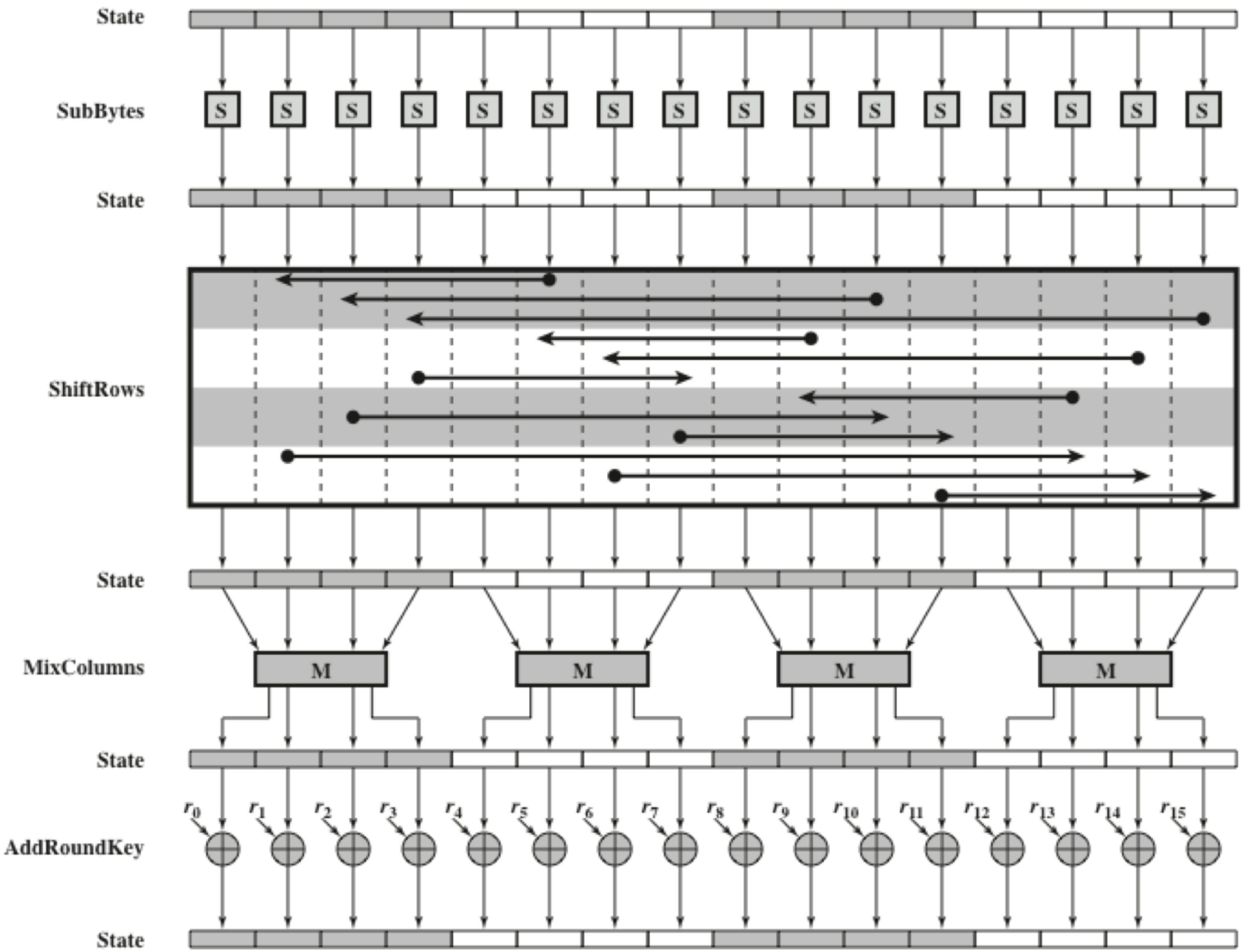
- This function transforms a column with completely new column.
- shift-rows step + the mix-column step causes
  - each bit of the ciphertext to depend on every bit of the plaintext after 10 rounds of processing



AES: Add round key

- The subkey is combined with the state.




State
SubBytes
S
State
ShiftRows
State
MixColumns
M
State
AddRoundKey
r_0 r_1 r_2 r_3 r_4 r_5 r_6 r_7 r_8 r_9 r_{10} r_{11} r_{12} r_{13} r_{14} r_{15}
State

AES

- The operations in AES are simple and can be implemented using cheap processors with a small amount of memory.
- Attacks
  - AES is definitely more secure than DES due to large-key size.
  - Best know attacks need 2^{126} for AES-128, $2^{189.9}$ for AES-192 and $2^{254.3}$ for AES-256.
- https://formaestudio.com/rijndaelinspector/archivos/rijndaelanimation.html

Random and pseudorandom Numbers

- A number of network security algorithms based on cryptography make use of random numbers
  - Examples:
    - Generation of keys for public-key encryption algorithms
    - Generation of a symmetric key for use as a temporary session key
    - In a number of key distribution scenarios, such as Kerberos, random numbers are used for handshaking to prevent replay attacks
- Two distinct and not necessarily compatible requirements for a sequence of random numbers are:
  - Randomness
  - Unpredictability

Randomness

- The following criteria are used to validate that a sequence of numbers is random:

Uniform distribution

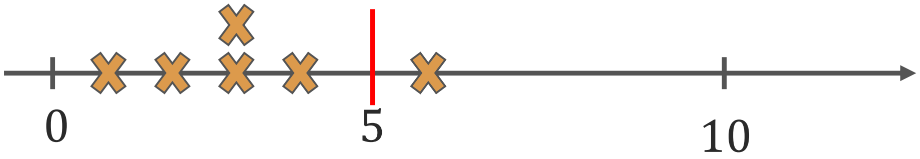
- The distribution of bits in the sequence should be uniform
- Frequency of occurrence of ones and zeros should be approximately the same

Independence

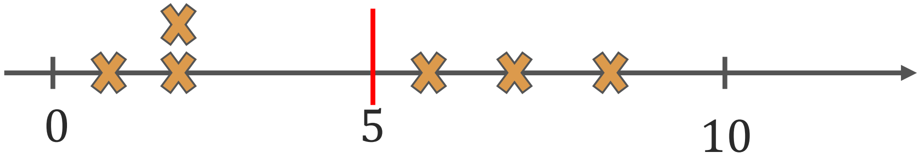
- No one subsequence in the sequence can be inferred from the others
- There is no test to “prove” independence
- The general strategy is to apply a number of tests until the confidence that independence exists is sufficiently strong

Randomness: Test

- Suppose function F is a random number generator in range $[1,10]$
  - We sample $N = 6$ numbers from F → $1,3,4,6,2,3$.



  - Now compare F with a uniform S → $1,2,6,2,7,8$



  - Only uniform distribution is truly random. For any distribution other than S, we use a test to determine its randomness.

Unpredictability

- With “true” random sequences, each number is statistically independent of other numbers in the sequence and therefore unpredictable
- An opponent should not be able to predict future elements of the sequence on the basis of earlier elements

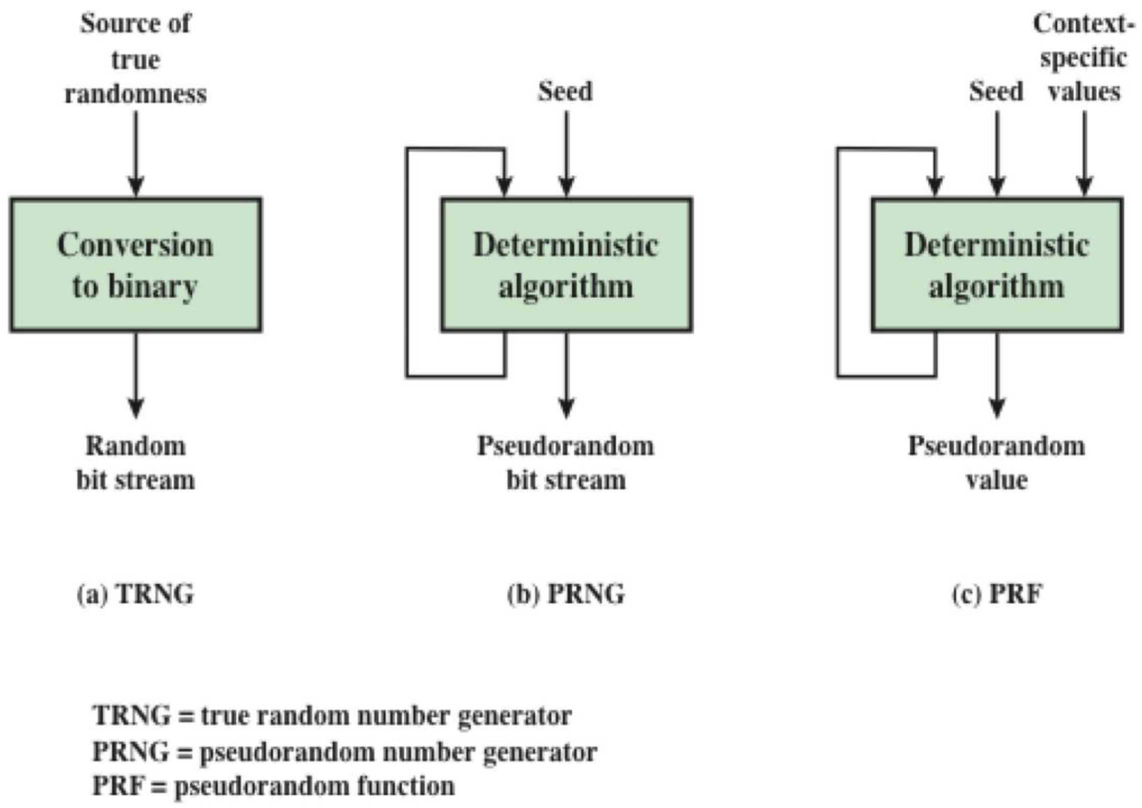
H

- HTHHT

T

- A PRNG should be unpredictable!

TRNG, PRNG, PRF



Algorithm design

Purpose-built algorithms

- Designed specifically and solely for the purpose of generating pseudorandom bit streams

Algorithms based on existing cryptographic algorithms

- Cryptographic algorithms have the effect of randomizing input
- Can serve as the core of PRNGs

Three broad categories of cryptographic algorithms are commonly used to create PRNGs:

- Symmetric block ciphers
- Asymmetric ciphers
- Hash functions and message authentication codes

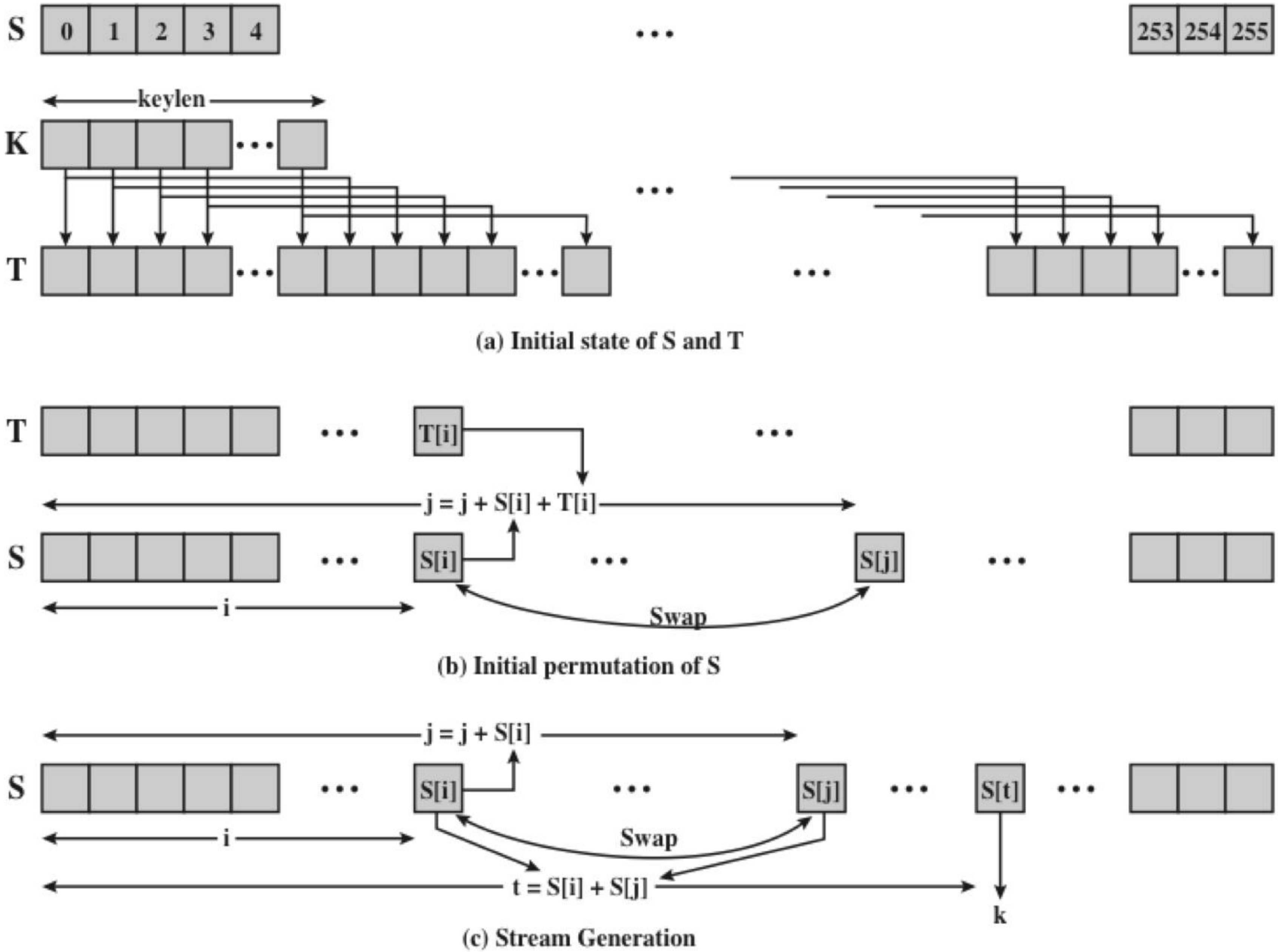
Why is it PRNG not TRNG?

- Suppose input seed $S = \{00,01,10,11\}$ with $p(00) = 0.2, p(01) = 0.3, p(10) = 0.1, p(11) = 0.4$
- Let $\mathrm{PRNG}(s = s_0 s_1) = (s_0)(s_0 \oplus s_1)(s_1)(s_0 \cdot s_1)$

$p(s)$	s	Output	Probability of PRNG	Probability of TRNG
0.2	00	0000	0.2	$1/16$
0.3	01	0110	0.3	$1/16$
0.1	10	1100	0.1	$1/16$
0.4	11	1011	0.4	$1/16$
		1000	0	$1/16$
		0001	0	$1/16$
		$\ldots$	0	$1/16$

RC4

- A stream cipher designed in 1987 by Ron Rivest for RSA Security
- It is a variable key-size (40 to 2048 bits) stream cipher with byte-oriented operations
- The algorithm is based on the use of a random permutation
- It mainly uses swap, XOR, and mod operations


S
0
1
2
3
4
253
254
255
keylen
K
T
(a) Initial state of S and T
T[i]
j = j + S[i] + T[i]
S[i]
S[j]
i
Swap
(b) Initial permutation of S
j = j + S[i]
S[t]
t = S[i] + S[j]
k
(c) Stream Generation

RC4: Toy Example (mod 32)

S

0	1	2	3	4		29	30	31

T

31	26	11	25	30	0	22	29	9	7	22	27		16	25	22

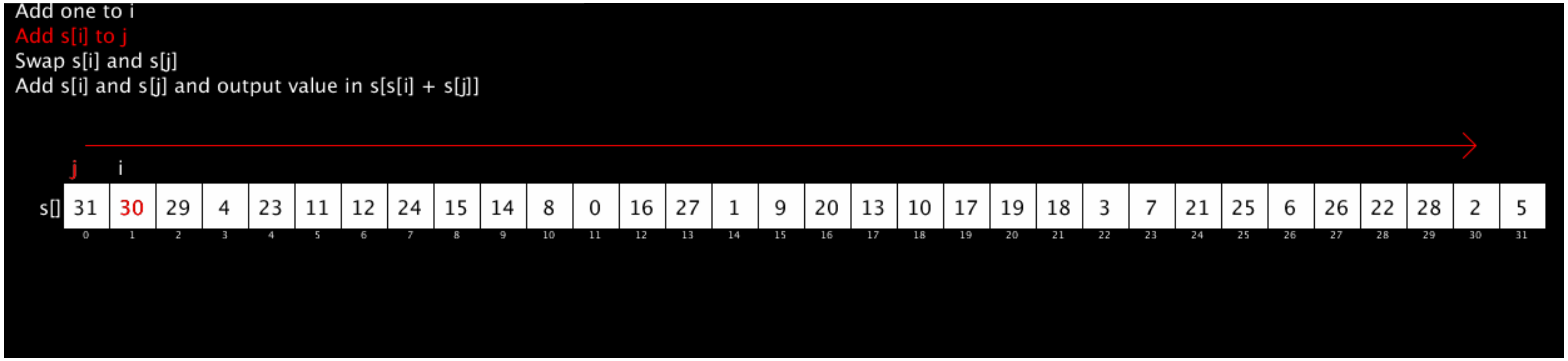
```
j = 0;
for
    i = 0 to 31 do
    {
        j = (j + S[i] + T[i])mod 32;
        Swap(S[i], S[j]);
    }
```

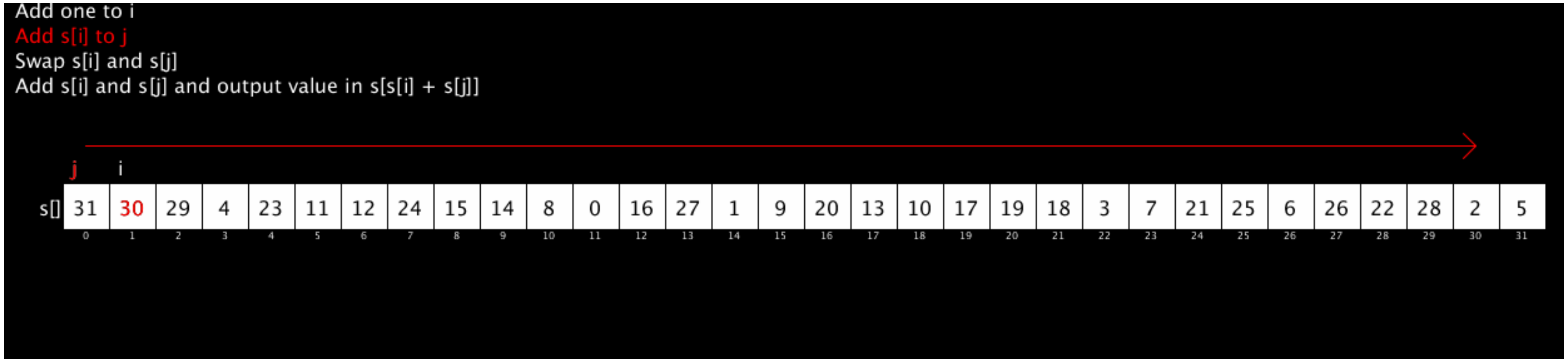
S

31	30	29	4	23	11	12	24	15	14	8	0		28	2	5

RC4: Toy Example (mod 32)

```
i, j = 0;
while (true):
    i = (i + 1)mod 32;
    j = (j + S[i])mod 32;
    Swap(S[i], S[j]);
    t = (S[i] + S[j])mod 32;
    k = S[t];
```





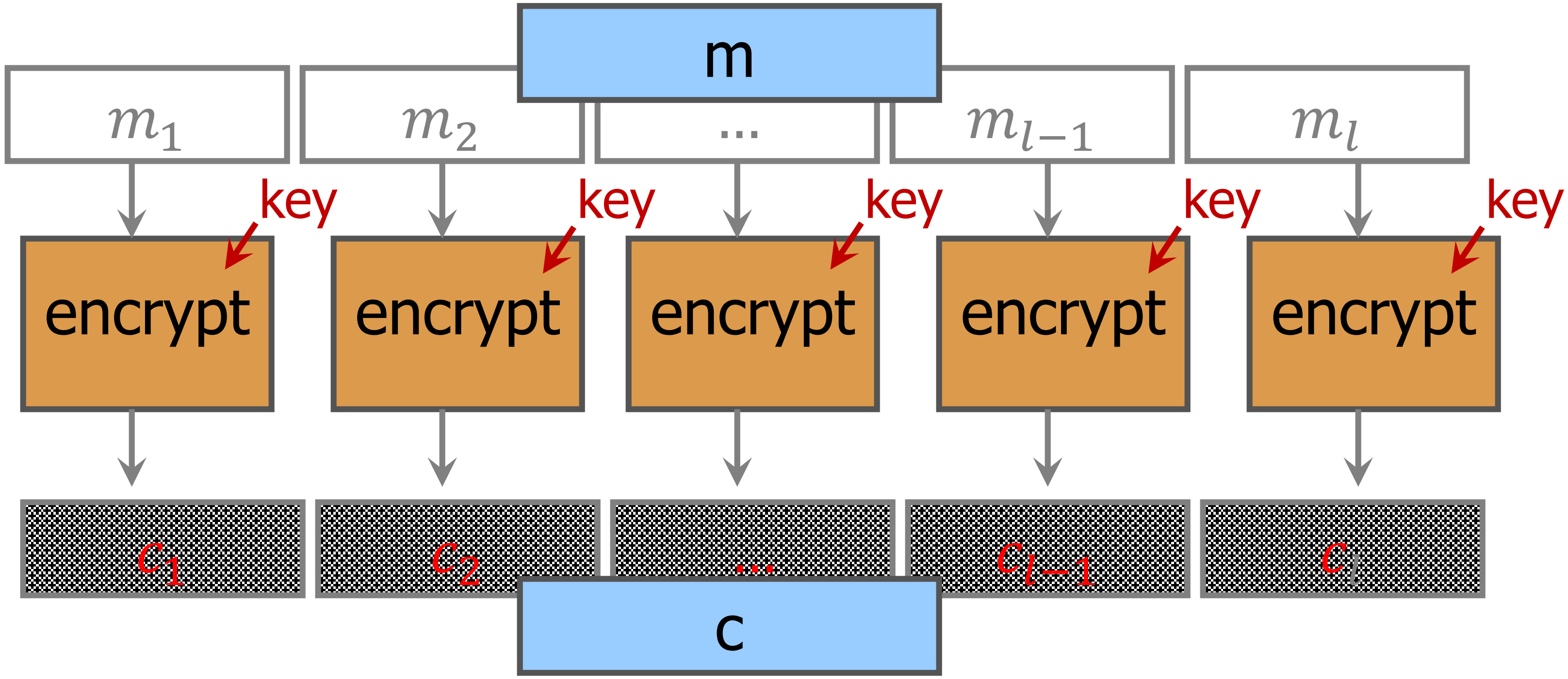
Encrypting a Large Message

- AES is a good block cipher, but our plaintext is larger than 128-bit block size
- Split plaintext into blocks, encrypt each one separately using the block cipher
  - $m = m_1 m_2 \ldots m_l$, where $|m_i| = 128$
  - $c = c_1 c_2 \ldots c_l$, where $\mathrm{c_i} = E(k, m_i)$
- Or it should be more complicated?
  - Modes of Operation

Cipher block Modes of Operation

- A symmetric block cipher processes one block of data at a time
  - In the case of DES and 3DES, the block length is $n = 64$ bits
  - For AES, the block length is $n = 128$
  - For longer amounts of plaintext, it is necessary to break the plaintext into n-bit blocks, padding the last block if necessary
- Five modes of operation have been defined by NIST
  - Intended to cover virtually all of the possible applications of encryption for which a block cipher could be used
  - Intended for use with any symmetric block cipher, including triple DES and AES

Electronic Codebook Mode (ECB)

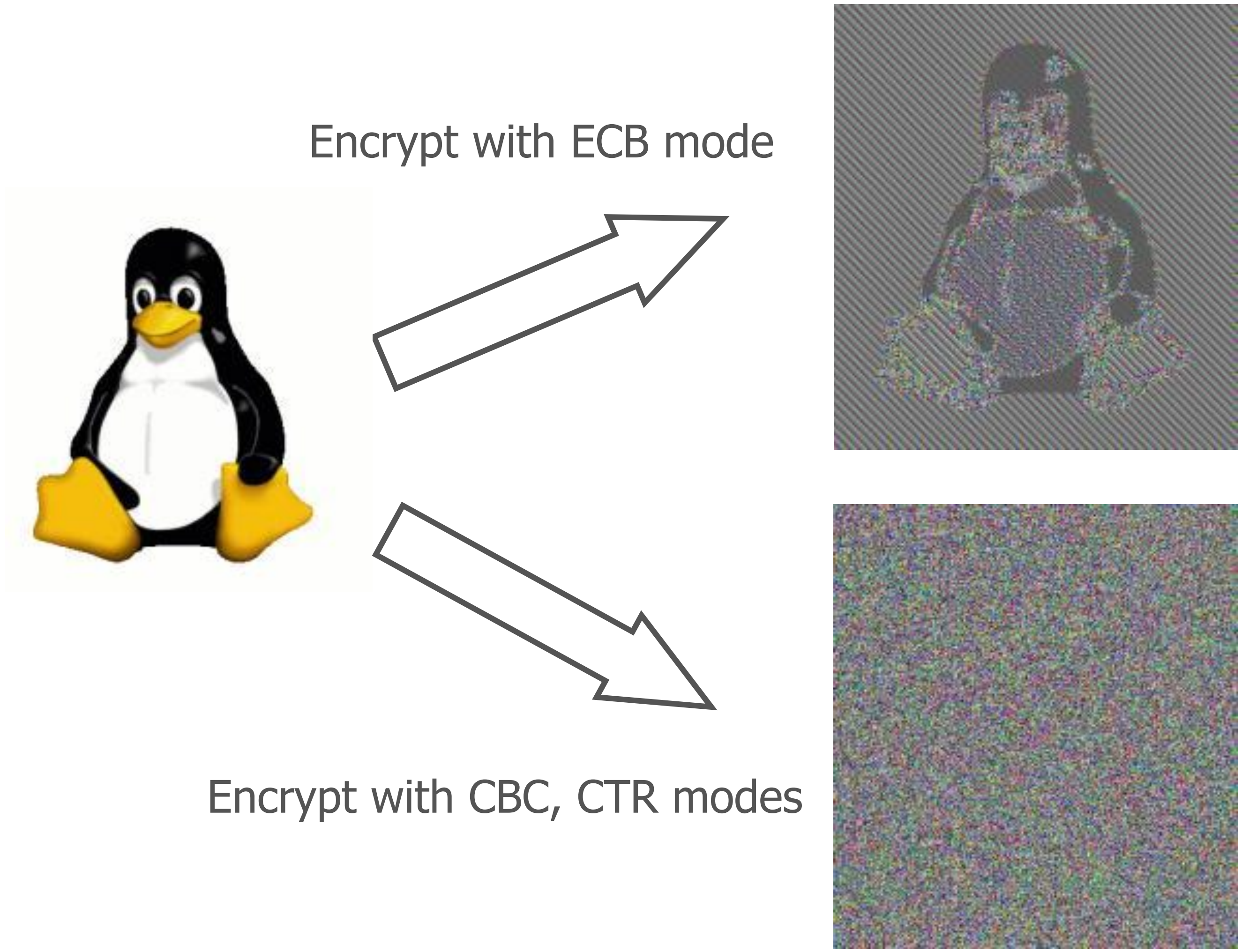


- $ecbE(k,m) = E(k,m_1)E(k,m_2)\dots E(k,m_l)$
- $ecbD(k,c) = D(k,c_1)D(k,c_2)\dots D(k,c_l)$

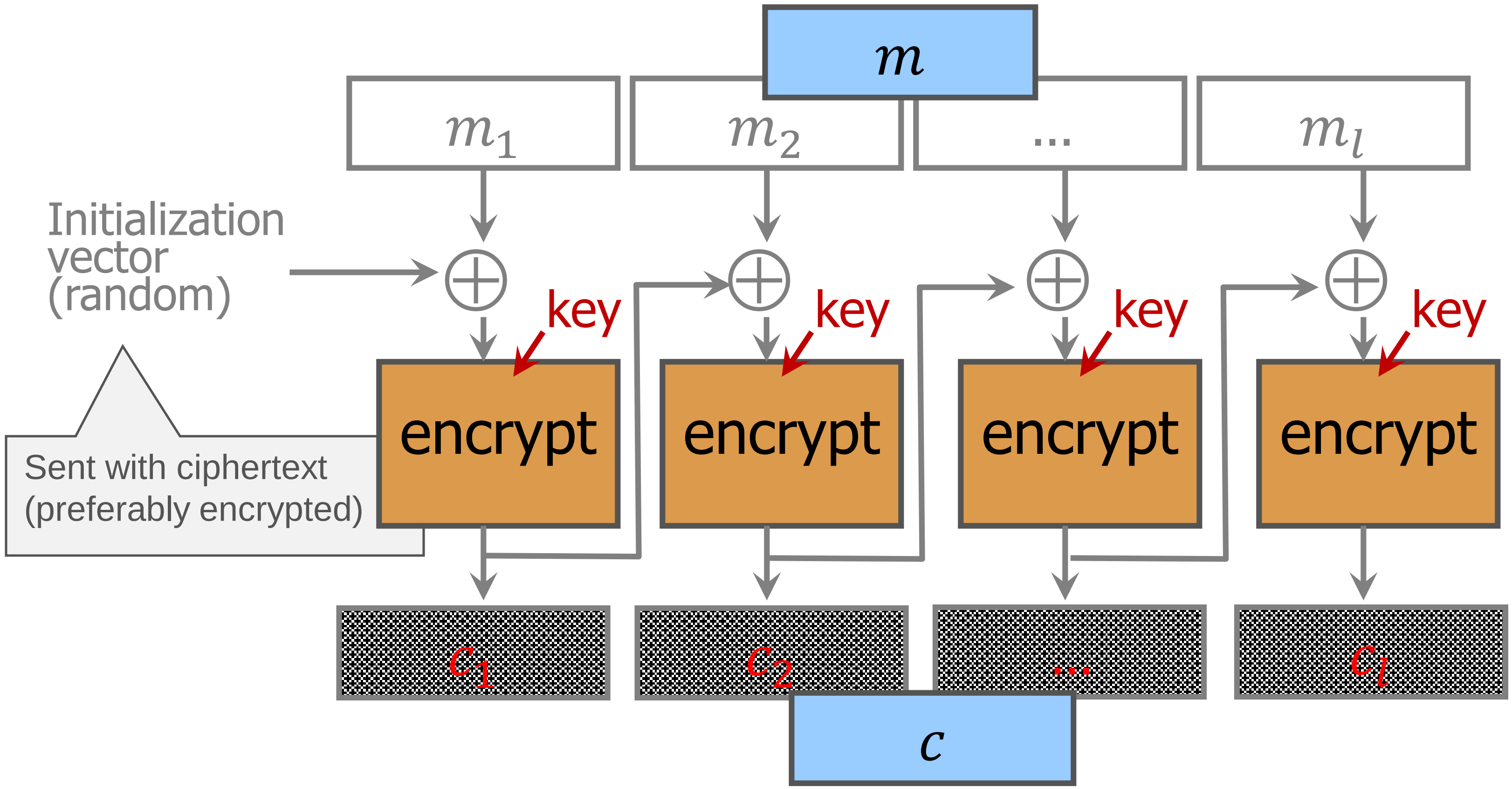
Information Leakage in ECB Mode

- With ECB, if the same n-bit block of plaintext appears more than once in the message, it always produces the same ciphertext
  - Because of this, for lengthy messages, the ECB mode may not be secure
  - If the message is highly structured, it may be possible for a cryptanalyst to exploit these regularities

Information Leakage in ECB Mode

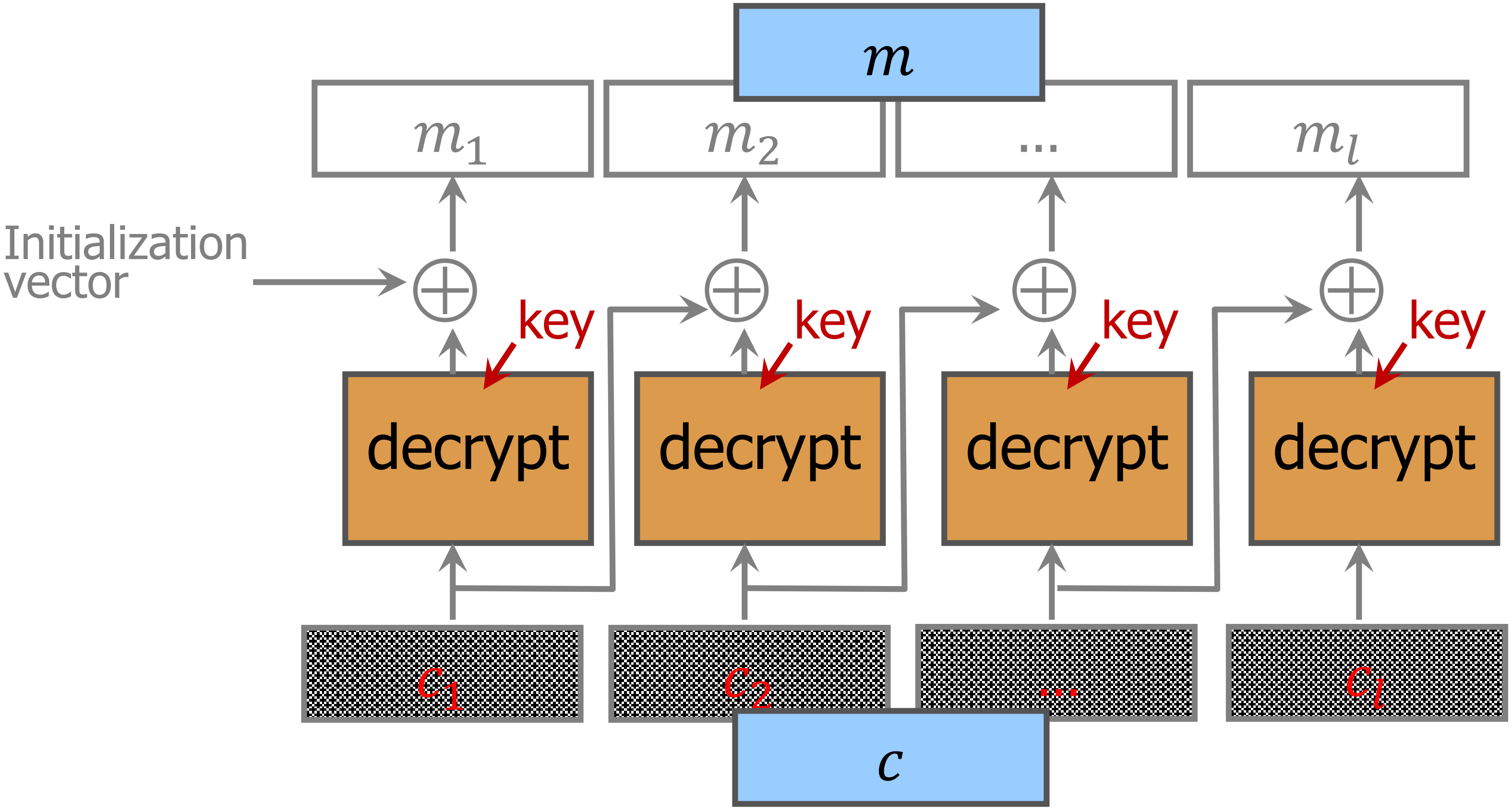


Cipher Block Chaining (CBC): Encryption



- $c_0 = IV$ is chosen randomly for each message
- $cbcE(k, m) = c_0 c_1 c_2 \ldots c_l$
- $c_i = E(k, m_i \oplus c_{i-1})$

Cipher Block Chaining (CBC): Decryption



- $cbcD(k, c_0c_1c_2 \ldots c_n) = m_1m_2 \ldots m_l$
- $m_i = D(k, c_i) \oplus c_{i-1}$

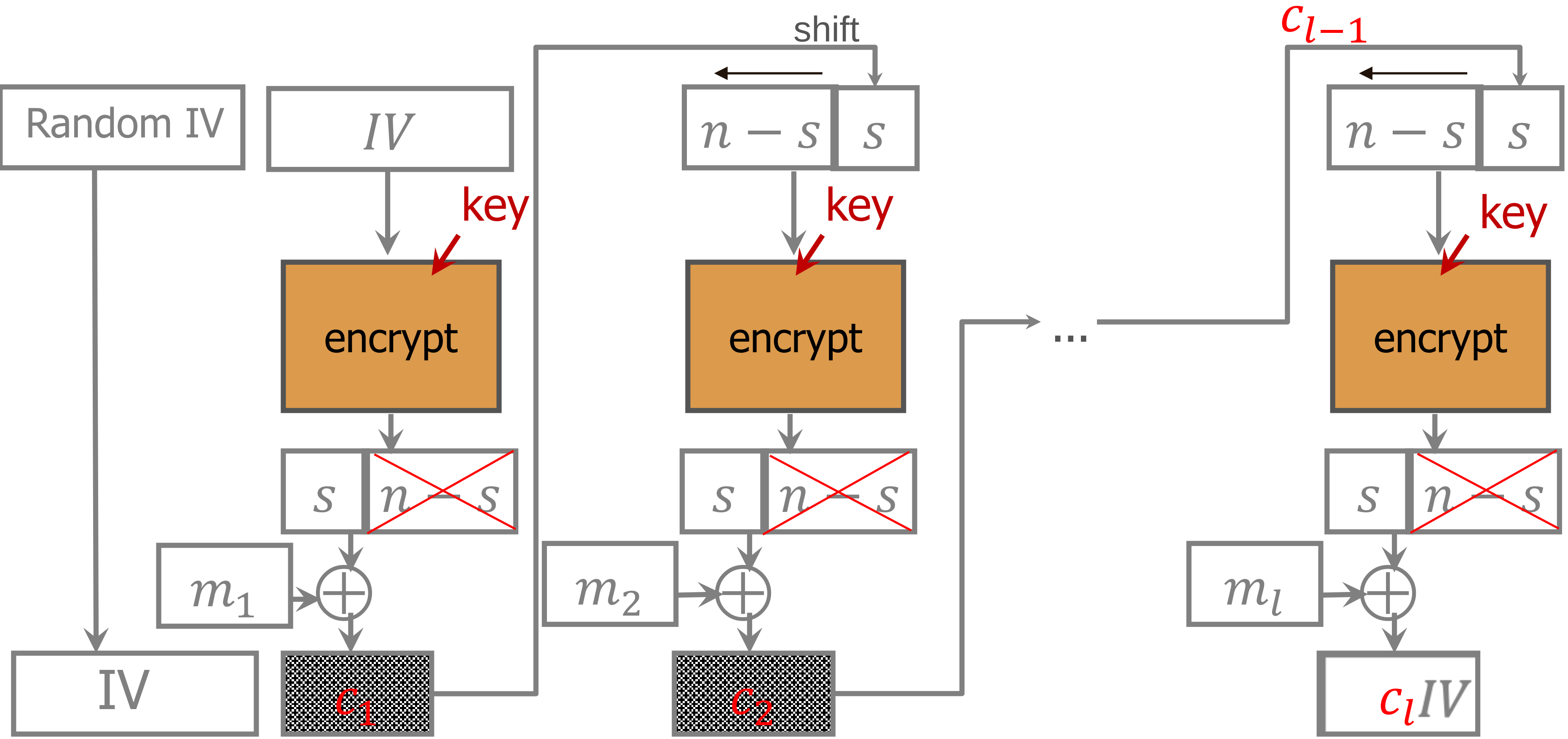
Cipher Block Chaining (CBC)

- It is a block cipher
- Identical blocks of plaintext encrypted differently
- A randomized encryption algorithm since c_0 is random.
- A transmission bit error in c_i affects correctness of m_i and m_{i+1}.
- Self-synchronized after two blocks if an entire block is lost.

Choosing the Initialization Vector

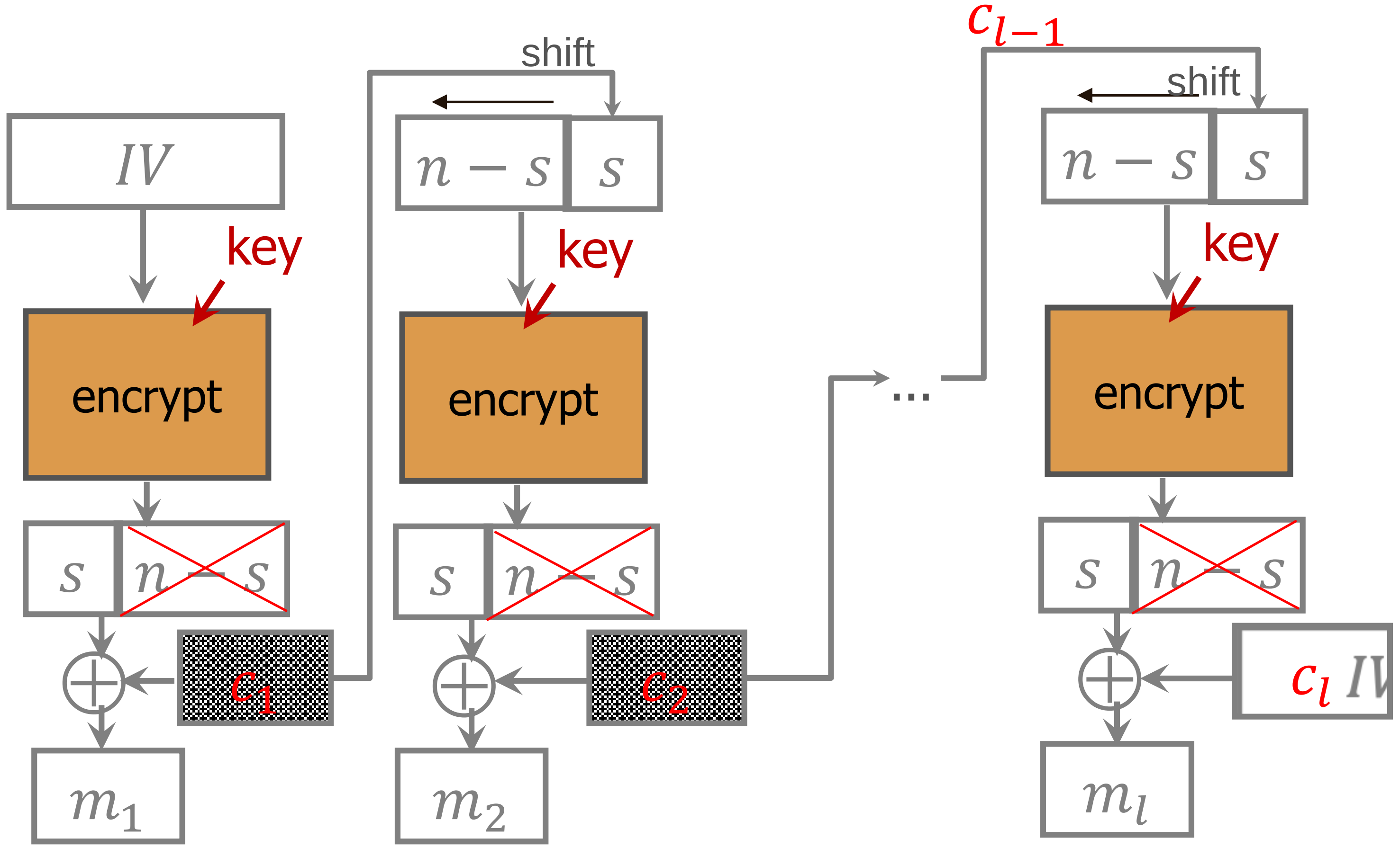
- Key used only once
  - No IV needed (can use IV=0)
- Key used multiple times
  - Best: fresh, random IV for every message
  - Can also use unique IV (eg, counter), but then the first step in CBC mode must use $IV' \leftarrow E(k, IV)$
    - Example: Windows BitLocker uses a function of sector number
    - May not need to transmit IV with the ciphertext

Cipher Feedback (CFB): Encryption



- $|m_i| = |c_i| = s$-bits
- $|IV| = n$-bits

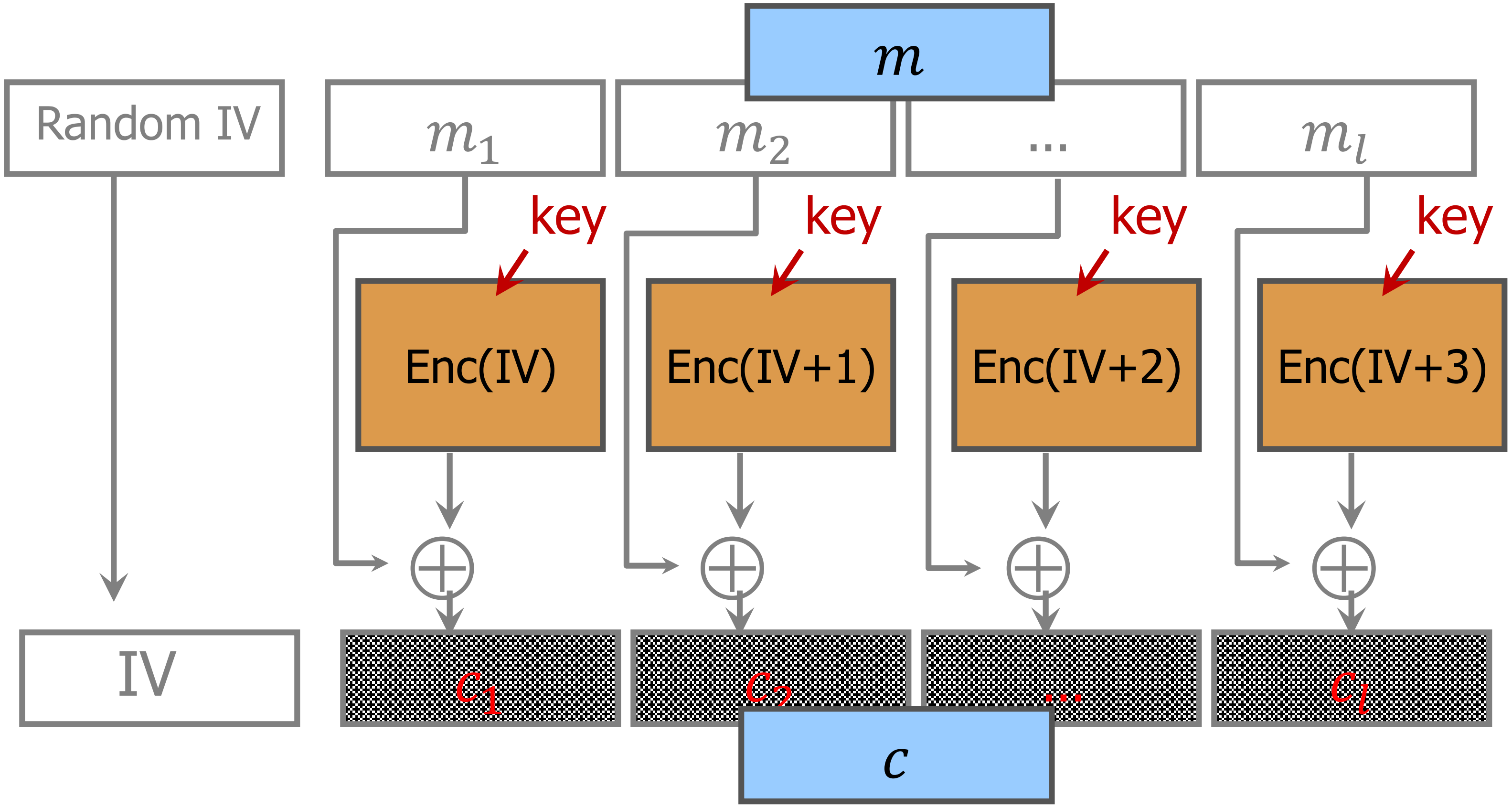
Cipher Feedback (CFB): Decryption



Cipher Feedback (CFB)

- $cfbE(k, m, IV) = IV c_1 c_2 \dots c_l$
  - $x_1 = IV$ is chosen randomly
  - $c_i = m_i \oplus msb_s\big(E(k, x_i)\big)$
  - $x_{i+1} = lsb_{n-s}(x_i) || c_i$
- It is a stream cipher
- A transmission error in c_i affects correctness of m_i and the next $\lceil n/s \rceil$ plaintext blocks.
- Self-synchronized after $\lceil n/s \rceil$ if an entire block is lost.

Counter mode (CTR)



- IV is chosen randomly for each message
- $ctrE(k, m) = IV c_1 c_2 \dots c_l$ $\qquad$ $ctrD(k, c) = m_1 m_2 \dots m_l$
- $c_i = E(k, IV + i - 1) \oplus m_i$ $\qquad$ $m_i = E(k, IV + i - 1) \oplus c_i$

Advantages of CTR mode

- It is a stream cipher
- Hardware efficiency
  - Encryption/decryption can be done in parallel on multiple blocks of plaintext or ciphertext
  - Throughput is only limited by the amount of parallelism that is achieved
- Software efficiency
  - Because of the opportunities for parallel execution, processors that support parallel features can be effectively utilized
- Preprocessing
  - The execution of the underlying encryption algorithm does not depend on input of the plaintext or ciphertext
  - the only computation is a series of XORs, greatly enhancing throughput

Question

- Which one is a good mode? ECB, CBC, CFB, or CTR?
  - Security?
  - Speed? Parallelizable?
  - Error propagation?
  - Self-synchronized?

Summary

- Symmetric encryption principles
  - Cryptography
  - Cryptanalysis
  - Feistel cipher structure
- Symmetric block encryption algorithms
  - Data encryption standard
  - Triple DES
  - Advanced encryption standard
- Random and pseudorandom numbers
  - The use of random numbers
  - TRNGs, PRNGs, PRFs
  - Algorithm design
- Stream ciphers and RC4
  - Stream cipher structure
  - RC4 algorithm
- Cipher block modes of operation
  - ECB
  - CBC
  - CFB
  - CTR