Lec 09: Cryptography (1)

CSED415: Computer Security

Spring 2024

Seulbae Kim

POSTECH

POHANG UNIVERSITY OF SCIENCE AND TECHNOLOGY

Administrivia

- Lab 02 deadline is fast approaching
  - Due Sunday, March 24

Cryptography – Definitions and Setting

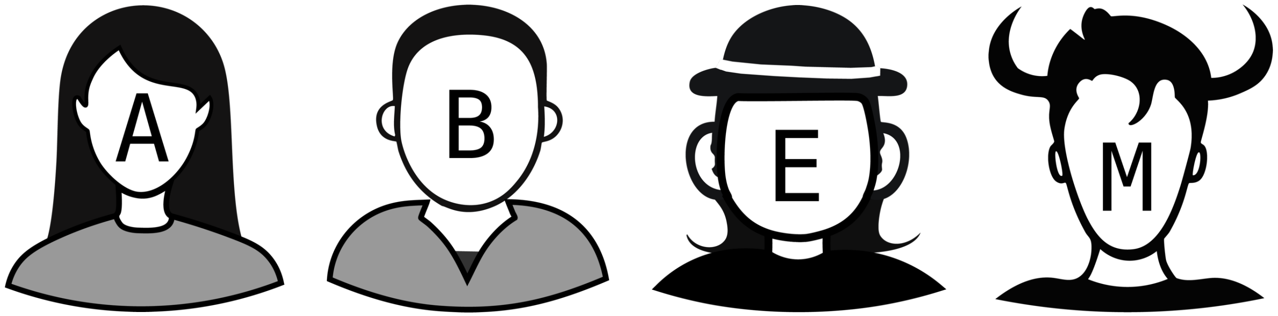
What is cryptography?

- A means to enable parties to maintain privacy of the information they send to each other, even in the presence of an adversary with access to the communication channel

Cryptography allows secure communication over insecure channels

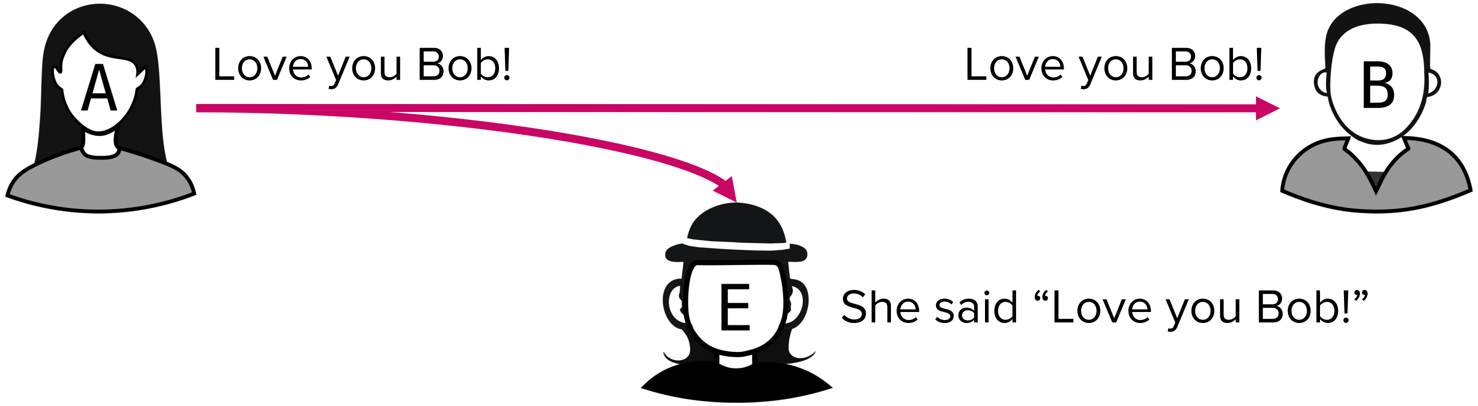
Main characters

- **Alice** and **Bob**: Two people trying to send messages to each other over an insecure communication channel
- **Eve**: An eavesdropper who can read any data on the channel
- **Mallory**: A malicious attacker who can read and modify any data on the channel

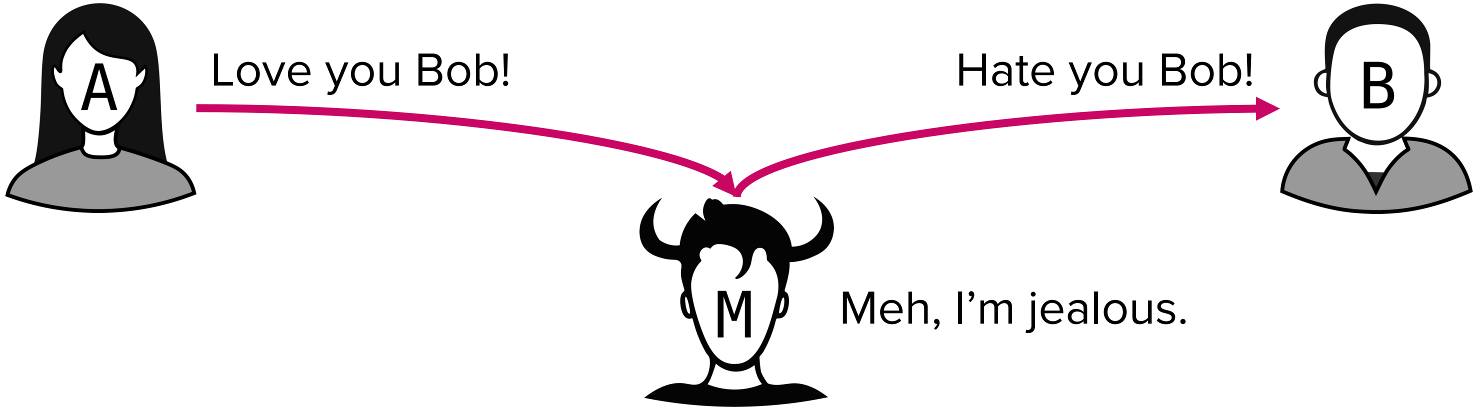


Cryptographic scenarios

- **Alice** & **Bob** against **Eve**



- **Alice** & **Bob** against **Mallory**



Goal: Preserving CI+A

- Three primary objectives
  - **Confidentiality**: Ensuring that only authorized parties can access the content of messages
  - **Integrity**: Guaranteeing that messages remain unchanged and unaltered during transmission
  - **Authenticity**: Verifying the origin of messages, confirming they were indeed sent by the claimed sender

Keys: The key to cryptography

- The basic building block of any cryptographic primitive
- Key controls encryption and decryption
- Two key models:
    - Symmetric key model
        - Alice and Bob share the same key
    - Asymmetric key model
        - A user has a secret key and a public key
        - Public key is shared to anyone
        - Secret key is kept to oneself

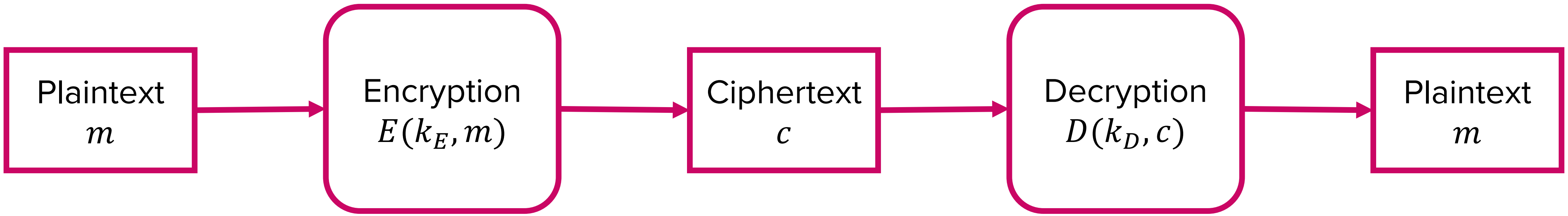
Kerckhoff’s principle

- “Security of a cryptosystem should not rely on the secrecy of the mechanism”
  - Cryptosystem should remain secure even when an attacker knows **all internal details** of the algorithm
  - The key should be the only thing that must be kept secret
  - Encourages the “Open Design” principle (ref: Lec 02)
    - Security through obscurity is discouraged

We assume that an attacker knows everything except the secret key

Terms and notations

- Plaintext: Original message m
- Ciphertext: Encrypted message c
- Encryption: Process of generating c from m
- Decryption: Process of generating m from c



Cryptography roadmap

Goal \ Scheme	Symmetric Key	Asymmetric Key
Confidentiality	• One Time Pad (OTP) • Block ciphers (DES, AES) • Stream ciphers	• ElGamal encryption • RSA encryption
Integrity & Authentication	• Message Authentication Code (MAC)	• Digital signature

Classical Ciphers

Caesar cipher

- One of the earliest cryptographic schemes
  - Used by Julius Caesar around 58 BC
- Scheme: Substitution cipher
  - Key k: An integer within the range [0:25]
  - $E(k, m)$: Substitutes each letter in m with the letter k positions forward in the alphabet
  - $D(k, c)$: Substitutes each letter in c with the letter k positions backward in the alphabet

Caesar cipher

- Example
  - $k = 3$
  - $m =$ `HELLO WORLD`
  - $E(k, m)$
    - H → K
    - E → H
    - L → O
    - …
  - c becomes `KHOOR ZRUOG`

m	c
A	D
B	E
C	F
D	G
E	H
F	I
G	J
H	K
I	L
J	M
K	N
L	O
M	P

m	c
N	Q
O	R
P	S
Q	T
R	U
S	V
T	W
U	X
V	Y
W	Z
X	A
Y	B
Z	C

Cryptanalysis of Caesar cipher

- Setting
  - Eve can see c = `ORYH BRX ERE`
  - Eve doesn't know k

- Possible attacks (1)
  - Brute-force attack: Try decryption with all 26 possible keys

```
k=0  m=ORYH BRX ERE     k=8  m=GJQZ TJP WJW     k=16 m=YBIR LBH OBO     k=24 m=QTAJ DTZ GTG
k=1  m=NQXG AQW DQD     k=9  m=FIPY SIO VIV     k=17 m=XAHQ KAG NAN     k=25 m=PSZI CSY FSF
k=2  m=MPWF ZPV CPC     k=10 m=EHOX RHN UHU     k=18 m=WZGP JZF MZM
k=3  m=LOVE YOU BOB     k=11 m=DGNW QGM TGT     k=19 m=VYFO IYE LYL
k=4  m=KNUD XNT ANA     k=12 m=CFMV PFL SFS     k=20 m=UXEN HXD KXK
k=5  m=JMTC WMS ZMZ     k=13 m=BELU OEK RER     k=21 m=TWDM GWC JWJ
k=6  m=ILSB VLR YLY     k=14 m=ADKT NDJ QDQ     k=22 m=SVCL FVB IVI
k=7  m=HKRA UKQ XKX     k=15 m=ZCJS MCI PCP     k=23 m=RUBK EUA HUH
```

Cryptanalysis of Caesar cipher

- Setting
  - Eve can see c = `ORYH BRX ERE`
  - Eve doesn't know k

- Possible attacks (2)
  - Chosen-plaintext attack: Eve can choose arbitrary plaintexts and obtain their corresponding ciphertexts
    - e.g., by tricking Alice into encrypting m that Eve chose
  - Eve chooses m = `ABCD` and receives c = `DEFG`
    - Eve can readily deduce $k = 3$

Rail Fence cipher

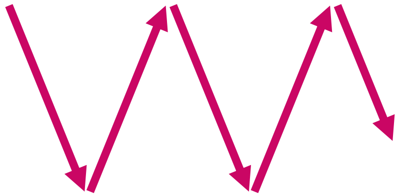
- A simple permutation cipher
  - Permutation cipher encrypts m by rearranging the letter order, without altering the actual letters used
- Scheme
  - Key k: An integer smaller than the length of plaintext m
  - E(k,m): Write the first letter of the plaintext. Write the following letters downwards diagonally for $k-1$ letters, then write upwards diagonally for $k-1$ letters. Repeat until the whole plaintext is written out

Rail Fence cipher

- Example
  - $k = 3$ (3 rails)
  - $m =$ `HELLO WORLD`
  - $E(k, m)$:

```
H...O...L.
.E.L.W.R.D
..L...O...
```

→ c becomes `HOL ELWRD LO`



Cryptanalysis of Rail Fence cipher

- Vulnerable to brute-force attacks
  - k is always smaller than the length of m
  - An attacker can try decryption with all possible k's

- Vulnerable to exhaustive permutations
  - c is a permutation of m
  - Therefore, m is a permutation of c
  - An attacker can try all permutations of c to obtain m

Classical ciphers are considered weak

- Basic substitution cipher (S) and permutation cipher (P) are considered insecure
  - Letters in a natural language (e.g., English) are not uniformly distributed
  - Knowledge of letter frequencies (most frequent: e, least frequent: ?) can be used for cryptanalysis against S or P ciphers

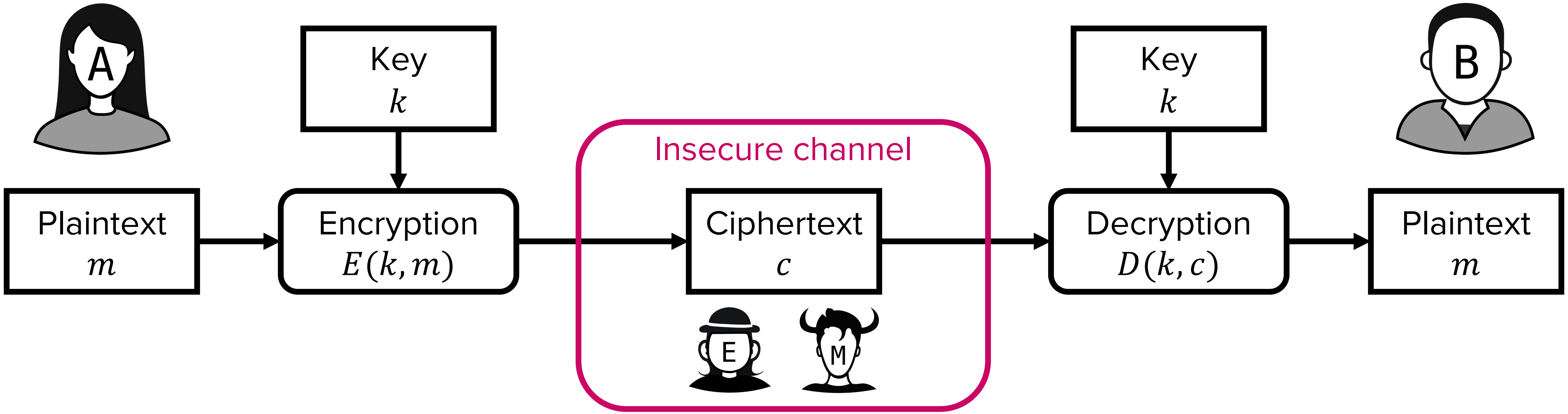
What if we combine S with P?

→ Transition into modern cryptography

Symmetric Cryptography (Shared key Scheme)

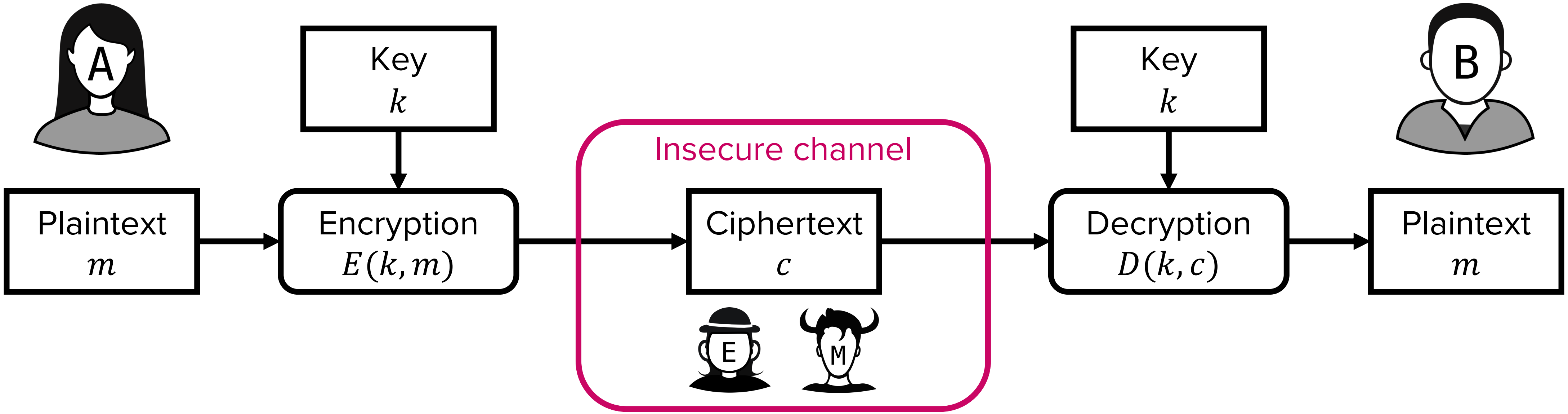
Symmetric key cryptography

- A symmetric encryption scheme consists of:
  - The key generation algorithm: Generates k
  - The encryption algorithm: $c = E(k, m)$
  - The decryption algorithm: $m = D(k, c)$



Symmetric key cryptography

- Required properties
  - Correctness
    - $D(k, E(k, m)) = m$ should hold for all k and m
  - Confidentiality
    - c should not give an attacker any additional information about m



One-time Pad (OTP)

- Scheme
  - Key k: Randomly selected bitstring of length n
    - n: length of the plaintext m
  - $E(k, m) = k \oplus m$: Bitwise XOR k and m
  - $D(k, c) = k \oplus c$: Bitwise XOR k and c

Review: XOR

$$0 \oplus 0 = 0 \qquad x \oplus 0 = x$$
$$0 \oplus 1 = 1 \qquad x \oplus x = 0$$
$$1 \oplus 0 = 1 \qquad x \oplus y = y \oplus x$$
$$1 \oplus 1 = 0 \qquad (x \oplus y) \oplus x = y$$

One-time Pad (OTP)

- Example
  - m = `OMW` (== bitstring $01101111\ 01101101\ 01110111$)
    - $n = 24$
  - $k = 00111101\ 01101010\ 11001101$
    - Generated at random, shared between Alice and Bob

One-time Pad (OTP)

- Example
  - Encryption (Alice)

m	0	1	1	0	1	1	1	1	0	1	1	0	1	1	0	1	0	1	1	1	0	1	1	1
	⊕	⊕	⊕	⊕	⊕	⊕	⊕	⊕	⊕	⊕	⊕	⊕	⊕	⊕	⊕	⊕	⊕	⊕	⊕	⊕	⊕	⊕	⊕	⊕
k	0	0	1	1	1	1	0	1	0	1	1	0	1	0	1	0	1	1	0	0	1	1	0	1
	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓
c	0	1	0	1	0	0	1	0	0	0	0	0	0	1	1	1	1	0	1	1	1	0	1	0

  - Alice transmits c to Bob

One-time Pad (OTP)

- Example
  - Decryption (Bob)

c	0	1	0	1	0	0	1	0	0	0	0	0	0	1	1	1	1	0	1	1	1	0	1	0
	⊕	⊕	⊕	⊕	⊕	⊕	⊕	⊕	⊕	⊕	⊕	⊕	⊕	⊕	⊕	⊕	⊕	⊕	⊕	⊕	⊕	⊕	⊕	⊕
k	0	0	1	1	1	1	0	1	0	1	1	0	1	0	1	0	1	1	0	0	1	1	0	1
	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓
m	0	1	1	0	1	1	1	1	0	1	1	0	1	1	0	1	0	1	1	1	0	1	1	1

  - Bob retrieves m = 01101111 01101101 01110111 = `OMW`

One-time Pad (OTP)

- Evaluation: Correctness
  - Cryptographic algorithm is correct if $D\big(k, E(k,m)\big) = m$

$$E(k,m) = k \oplus m \quad \cdots \text{ Definition of } E$$

$$\begin{aligned} D\big(k, E(k,m)\big) &= D(k, k \oplus m) && \cdots \text{ Substitution} \\ &= k \oplus (k \oplus m) && \cdots \text{ Definition of } D \\ &= m && \cdots \text{ Property of XOR} \end{aligned}$$

Thus, OTP is correct. ■

How do we evaluate the security (i.e., confidentiality)?

Theorem: Shannon's perfect secrecy (1949)

- An encryption scheme is perfectly secure if for every ciphertext c and messages m_1 and m_2,

$$Prob[E(k, m_1) = c] = Prob[E(k, m_2) = c]$$

- In plain English, even if an attacker has infinite time and computational powers in the world, he or she cannot crack your ciphertext if your scheme is Shannon-secure

OTP ensures perfect secrecy

- Theorem

$$\forall c, \forall m_1, \forall m_2$$

$$Prob[E(k, m_1) = c] = Prob[E(k, m_2) = c]$$

- Proof
  - Fix any ciphertext $c \in \{0,1\}^n$ (i.e., a bitstring of length n)
  - For every m, $Prob[E(k, m) = c] = Prob[k = m \oplus c] = 2^{-n}$
    - Constraint: For every new message m, a new key k is generated

OTP ensures perfect secrecy

- Example
  - m = `SEE YOU AT 8PM TOMORROW`
  - $c = 001010001 \ldots$
  - Attacker tries **all** possible $k \in \{0,1\}^n$ and decrypt the given c
    - What the attacker gets:

```
SEE YOU AT 2PM TOMORROW
EAT HIM BY 4PM TOMORROW
THE CAT IN THE HOSPITAL
WAS JIM AT THE VINEYARD
           ...
```

→ Can NEVER guess the correct m

Why not use OTP everywhere?

- Practical limitations exist
  - Key generation: k should be used only once
    - k needs to be randomly generated for each message (expensive)
  - Key management: k needs to be as long as m
    - Storage complexity increases for longer m
  - Key distribution: k needs to be shared
    - n-bit k needs to be shared securely first before we can send c securely

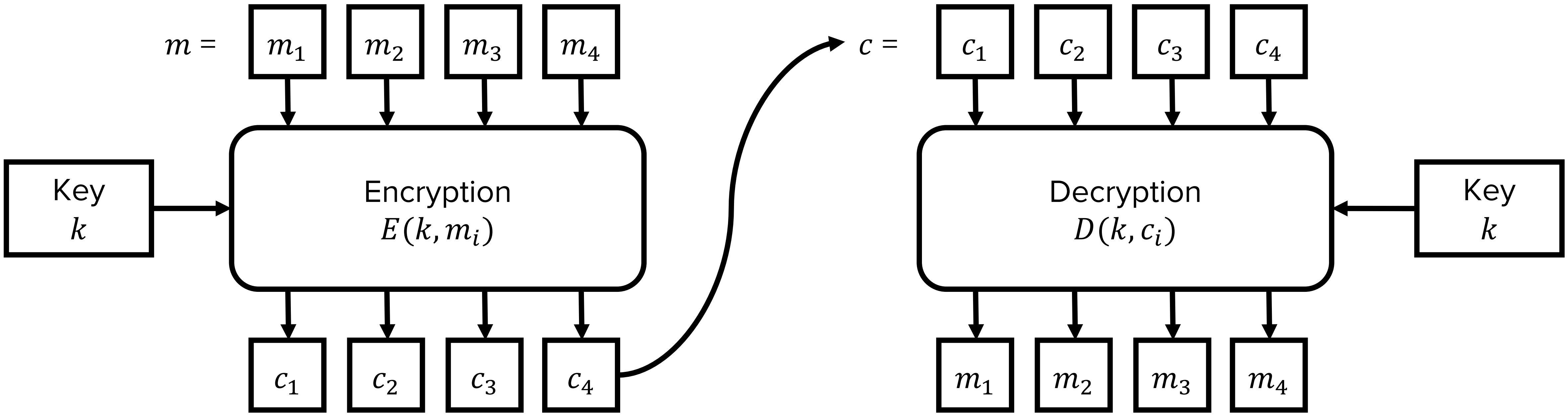
OTP is impractical for real-world usage

Cryptography roadmap

Goal \ Scheme	Symmetric Key	Asymmetric Key
Confidentiality	✅ One Time Pad (OTP) • Block ciphers (DES, AES) • Stream ciphers	• ElGamal encryption • RSA encryption
Integrity & Authentication	• Message Authentication Code (MAC)	• Digital signature

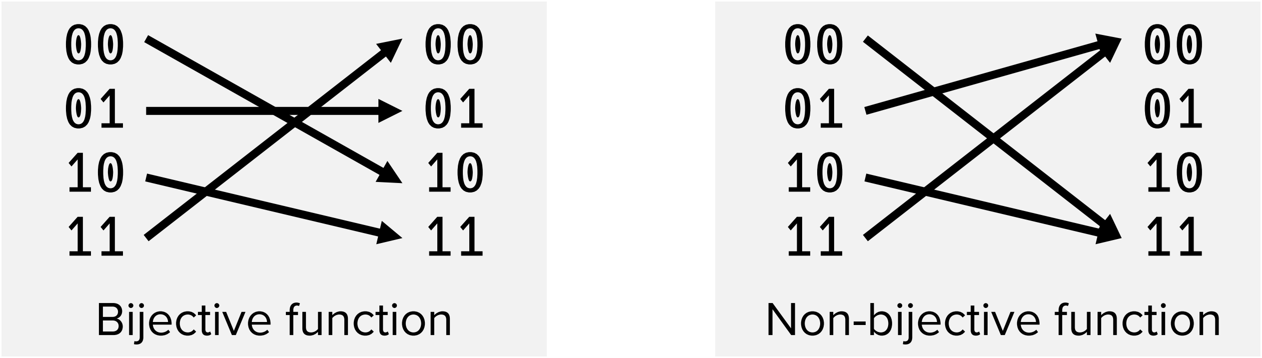
Block ciphers

- A scheme consisting of encode/decode algorithms for a **fixed-sized block** of bits



Correctness requirement of block ciphers

- E: A permutation (bijective function) and D: E^{-1} (inverse of E)
  - Every input is uniquely mapped to a single output



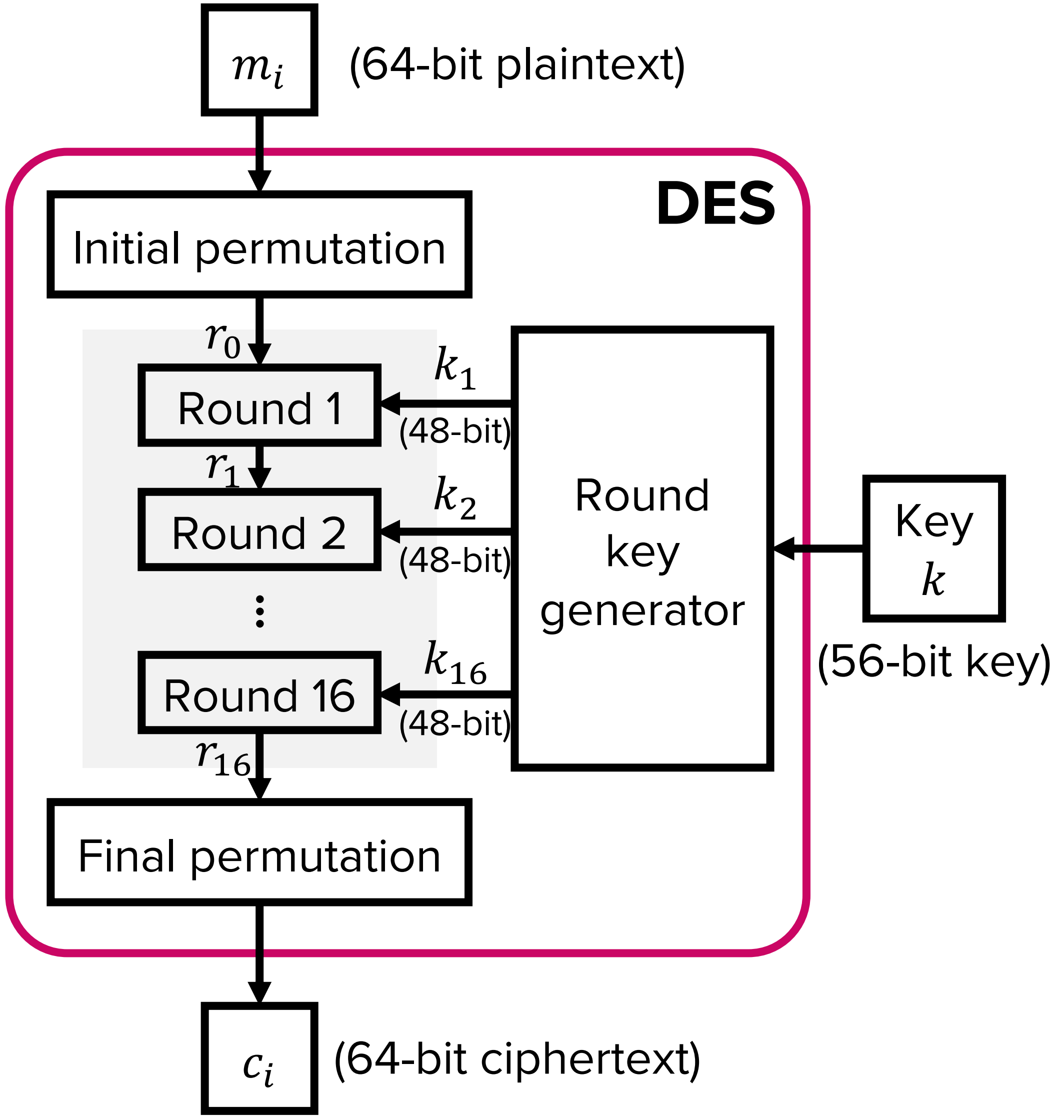
  - If E is not bijective, there may exist m_1 and m_2 such that

$$E(k, m1) = E(k, m2) = c$$

  - Then, we cannot decode c and obtain a unique plaintext

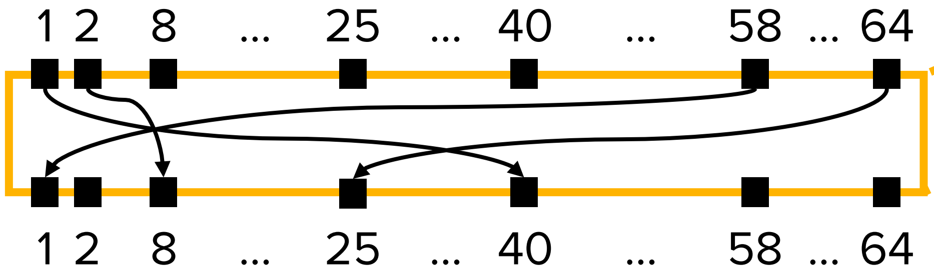
DES (Data Encryption Standard) (1975)

- Setting
  - Key size: 56 bits
  - Block size: 64 bits
    - In: 64-bit plaintext
    - Out: 64-bit ciphertext

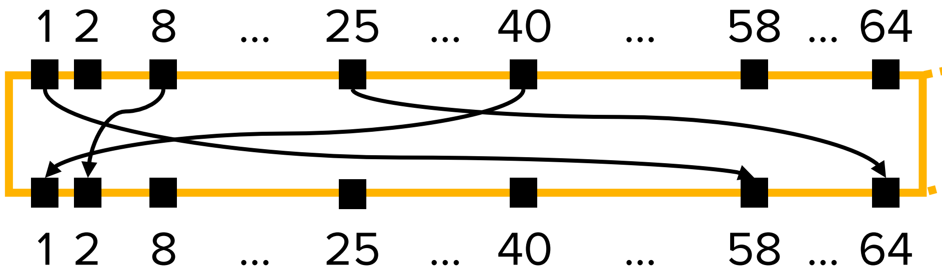


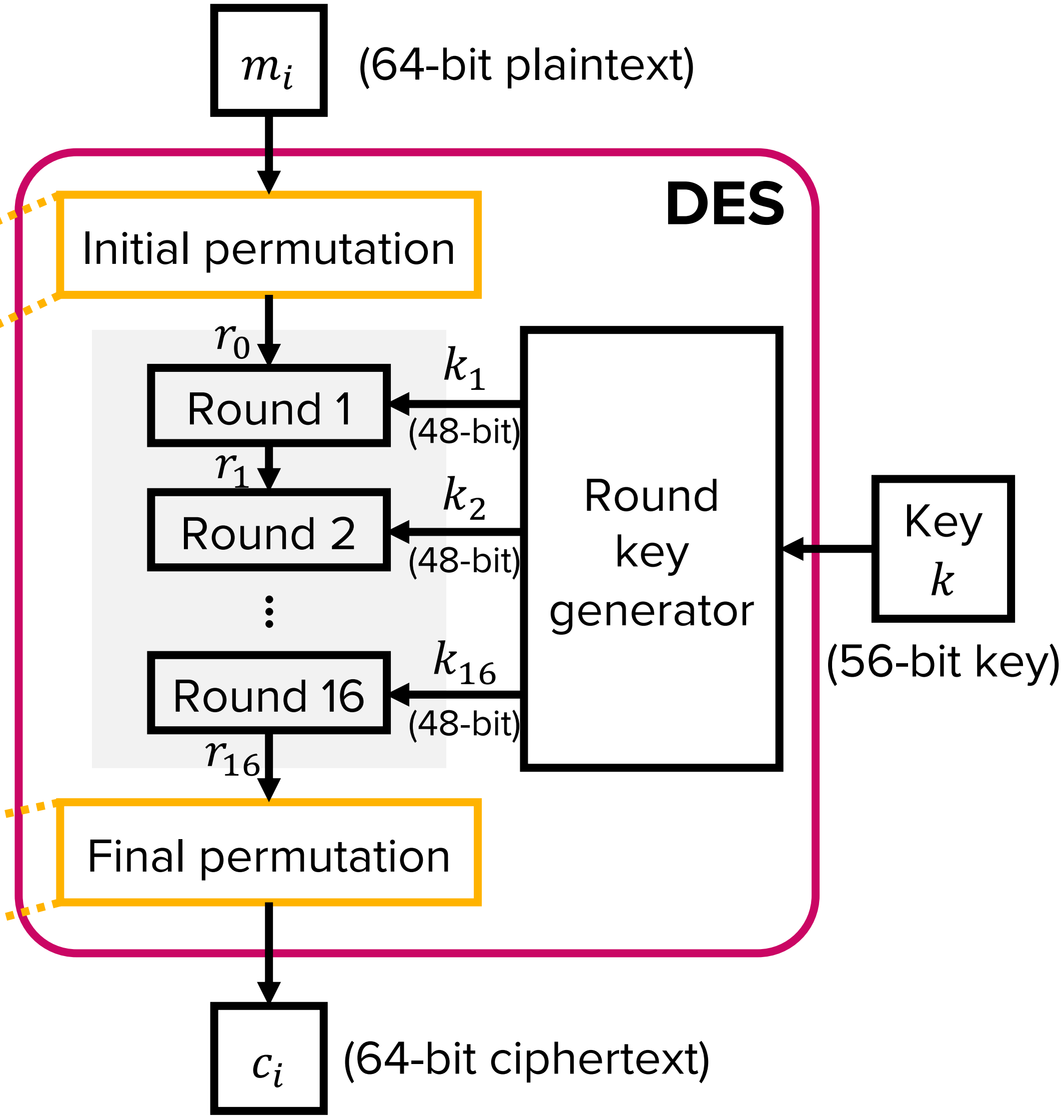
DES (Data Encryption Standard) (1975)

- Initial permutation (IP)
  - Rearranges the bits of m (diffusion)



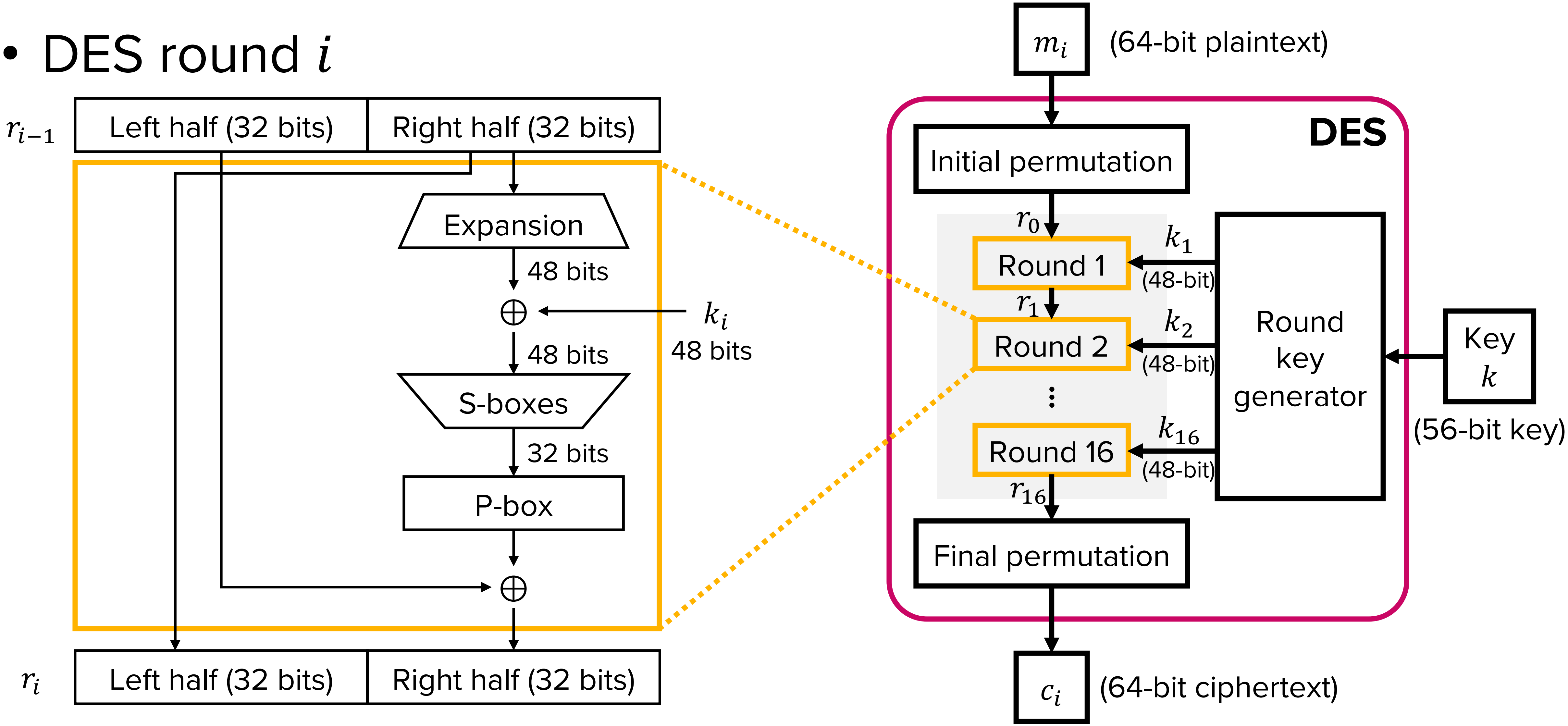
- Final permutation
  - Inverse of the IP





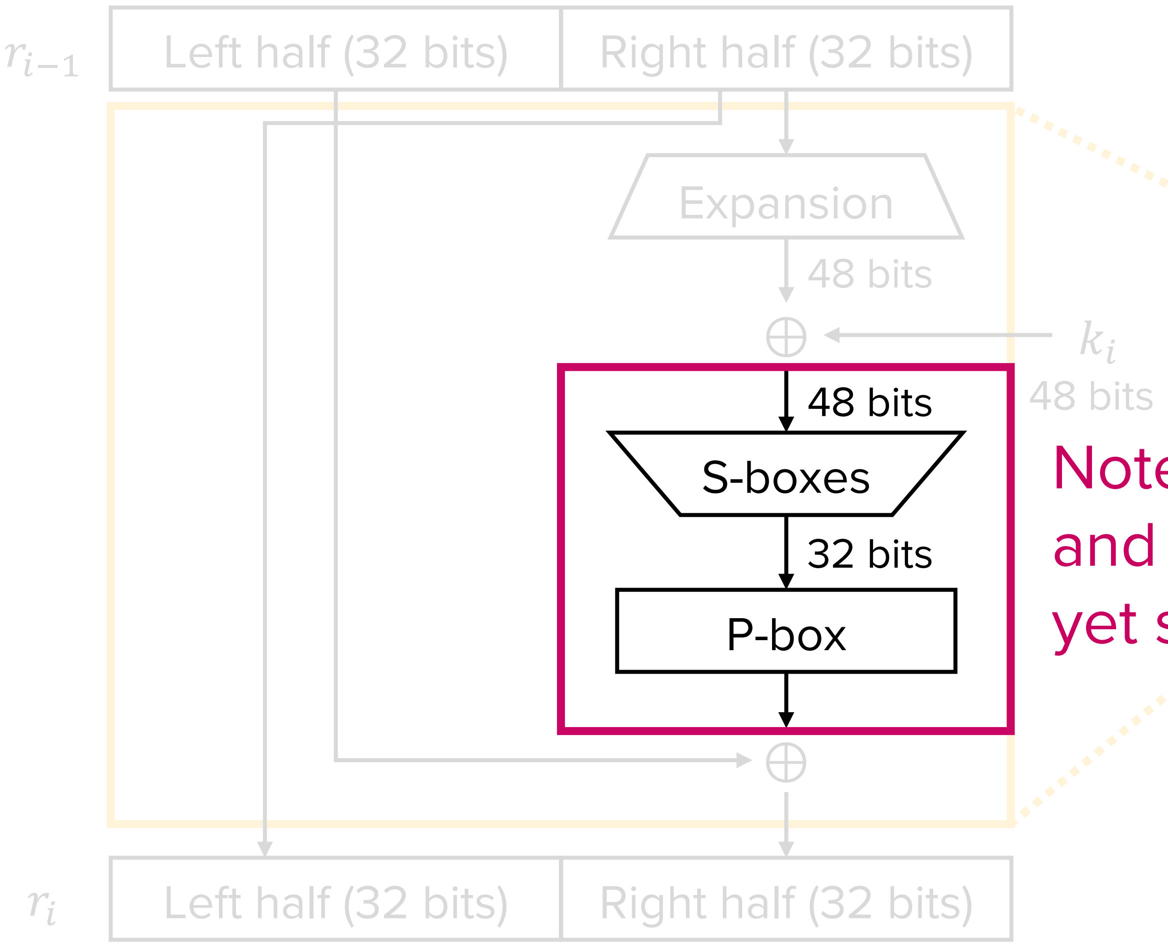
DES (Data Encryption Standard) (1975)

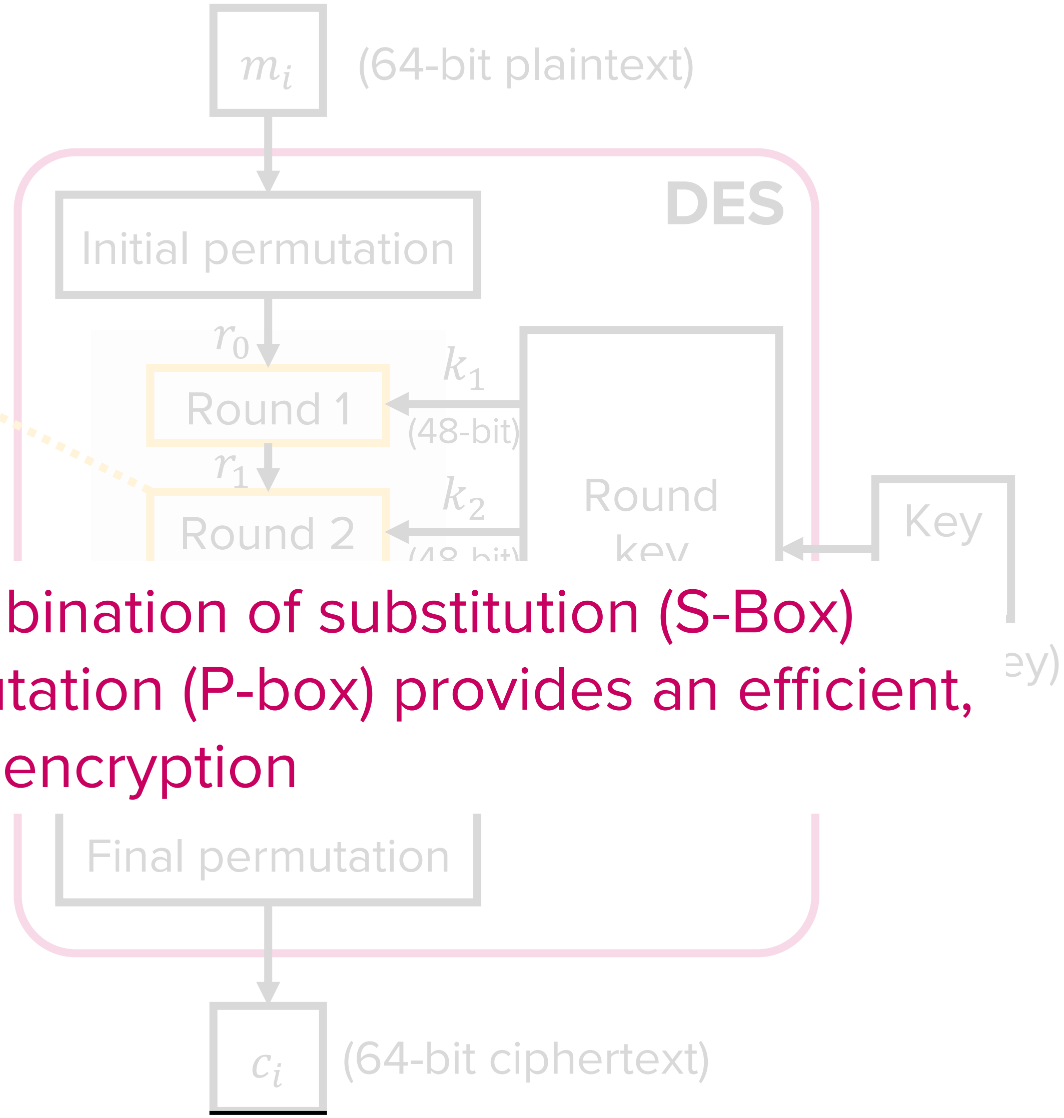
- DES round i



DES (Data Encryption Standard) (1975)

- DES round i

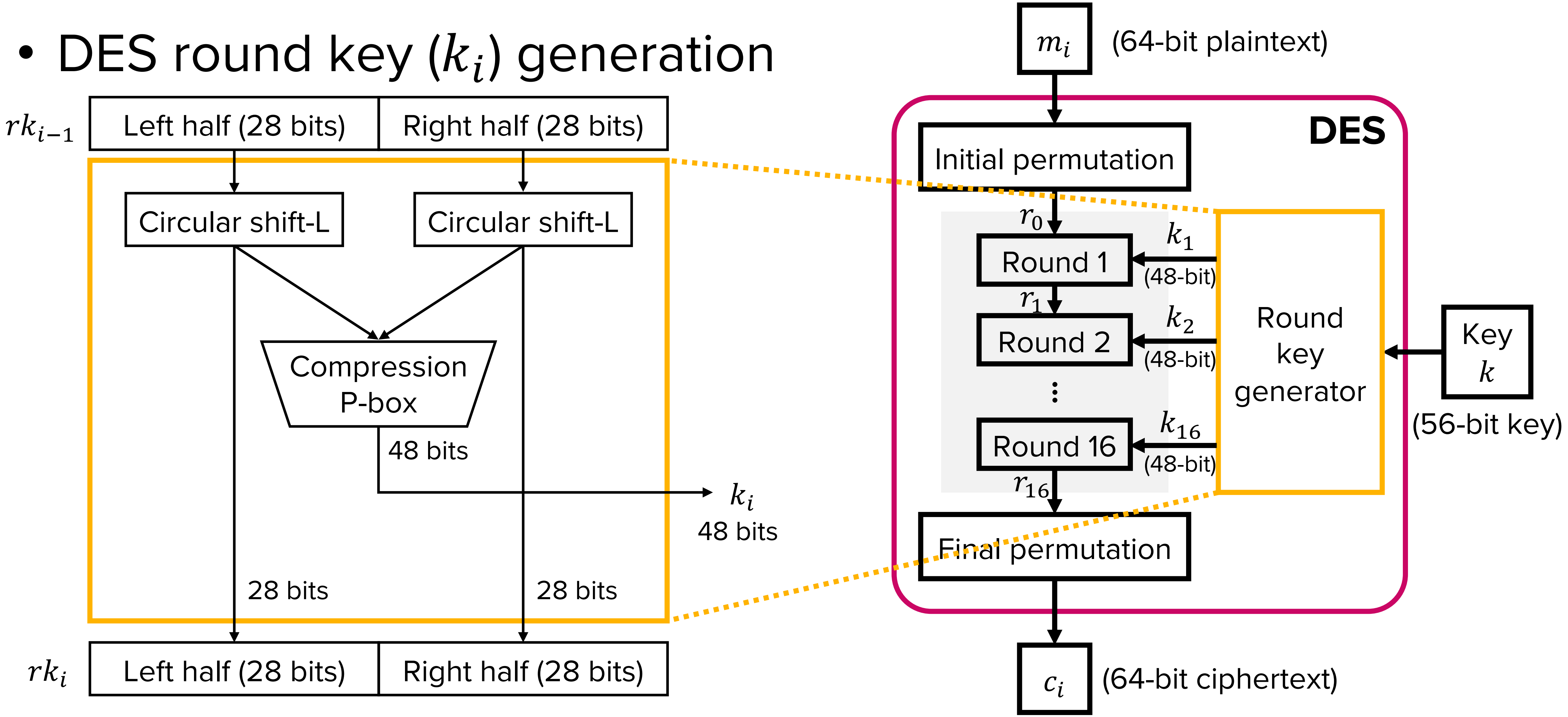




Note: Combination of substitution (S-Box) and permutation (P-box) provides an efficient, yet strong encryption

DES (Data Encryption Standard) (1975)

- DES round key (k_i) generation



Cryptanalysis of DES

- DES algorithm remains unbroken even now
  - No algorithmic weakness has been identified yet
- However, DES is considered unsafe due to its small key size
  - The entire keyspace of a 56-bit key can be searched within days on modern computers
  - In 1999, a dedicated machine brute-forced DES key in 22 hours (ref: Lec 01)
  - A new block cipher was needed

Triple-DES (3DES)

- Setting
  - Use two keys: k_1 and k_2 (Key size is 56*2 = 112 bits)
  - $3DES(k_1 k_2, m) = DES\left(k_2, DES^{-1}(k_1, DES(k_2, m))\right)$

- Cryptanalysis
  - Underlying encryption algorithm is the same
  - Key size is larger, making brute-force attacks infeasible
  - However, 3DES requires three DES computations (inefficient)

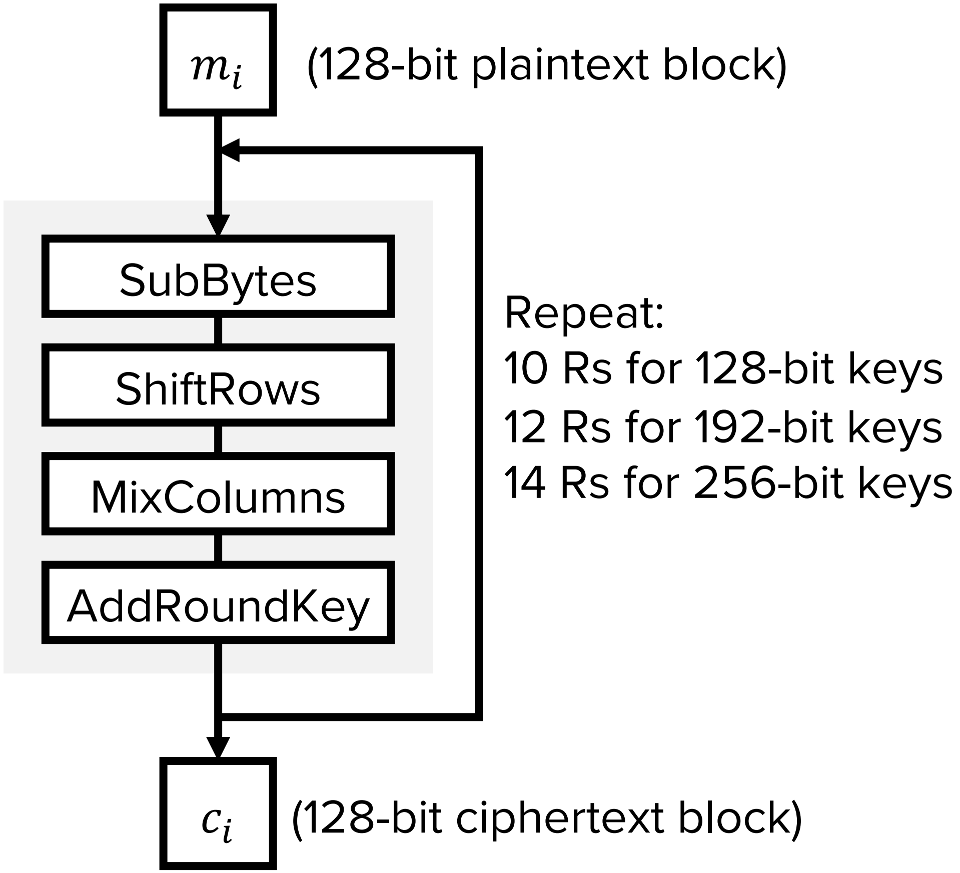
AES (Advanced Encryption Standard) (2001)

- A new encryption standard replacing DES
  - 15 algorithms from different countries were submitted to NIST
  - Rijndael algorithm by John Daemen and Vincent Rijmen was selected as the Advanced Encryption Standard

- Setting
  - Key size: 128, 192, or 256 bits
  - Block size: 128 bits

AES (Advanced Encryption Standard) (2001)

- Scheme
  - m_i: A 16-byte block (4x4)
  - SubBytes: Substitute bytes within block
  - ShiftRows: Shift bytes in each row
  - MixColumns: Multiply columns
  - AddRoundKey: XOR with round key



*You do not need to know all details of AES

Cryptanalysis of AES

- AES has not been broken
  - No algorithmic weakness
  - Exhaustive key search is **believed** to be infeasible
    - Nor formally proven, but empirically, no practical attack has been discovered
    - 128-bit key is large enough to prevent brute-force attacks

- Stronger and faster than DES/3DES

→ AES is the modern standard block cipher algorithm

Cryptography roadmap

POSTECH

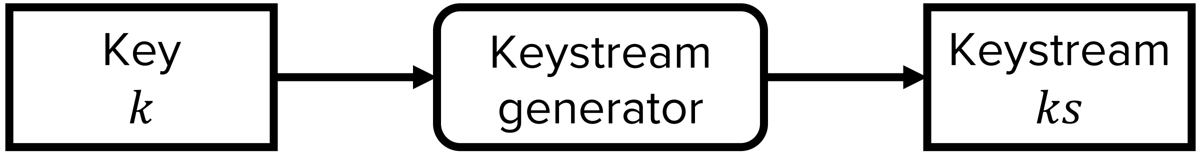
Goal \ Scheme	Symmetric Key	Asymmetric Key
Confidentiality	✅ One Time Pad (OTP) ✅ Block ciphers (DES, AES) • Stream ciphers	• ElGamal encryption • RSA encryption
Integrity & Authentication	• Message Authentication Code (MAC)	• Digital signature

Stream Cipher

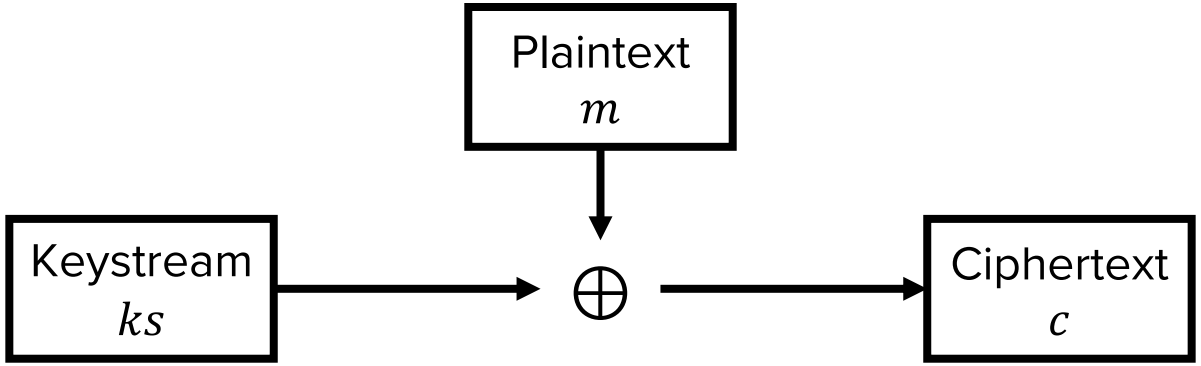
- Block ciphers break plaintext message in equal-sized blocks and encrypt each block as a unit
  - Overhead introduced for block-granularity processing (e.g., need to add padding for messages smaller than the block size)
- Stream ciphers encrypt one bit at a time
  - Better efficiency in real-time communications

Stream cipher – Approach

- Generate a pseudorandom keystream ks



- $E(ks, m)$: Bitwise XOR keystream ks with plaintext m



Background: Randomness

- Randomness is essential for symmetric key cryptography
  - e.g., random keystream for stream cipher
- If an attacker can predict a random number, many cryptographic schemes will be broken
- How can we securely generate random numbers?
  - Can computers generate random numbers?

Background: Randomness

- Entropy: A measure of uncertainty
  - High entropy means the outcomes are more unpredictable, which is desirable in cryptography
  - The uniform distribution has the highest entropy
    - e.g., Every output of a coin toss is equally likely

- In cryptography, randomness indicates uncertainty

Background: Randomness

- Keystream generator scenario
  - We want a keystream for stream cipher that attacker cannot guess
  - We can generate every bit of ks by tossing a fair (50-50) coin
  - Attacker cannot feasibly guess ks due to high entropy
    - “This ks is truly random”
  - Problem?

How would a computer do this?

Background: True randomness

- True randomness requires a physical source of entropy
  - A physical coin toss
  - Chaotic systems with complex dynamics, e.g., weather patterns
  - Atmospheric noise
  - Human activity

→ Very expensive and slow to generate

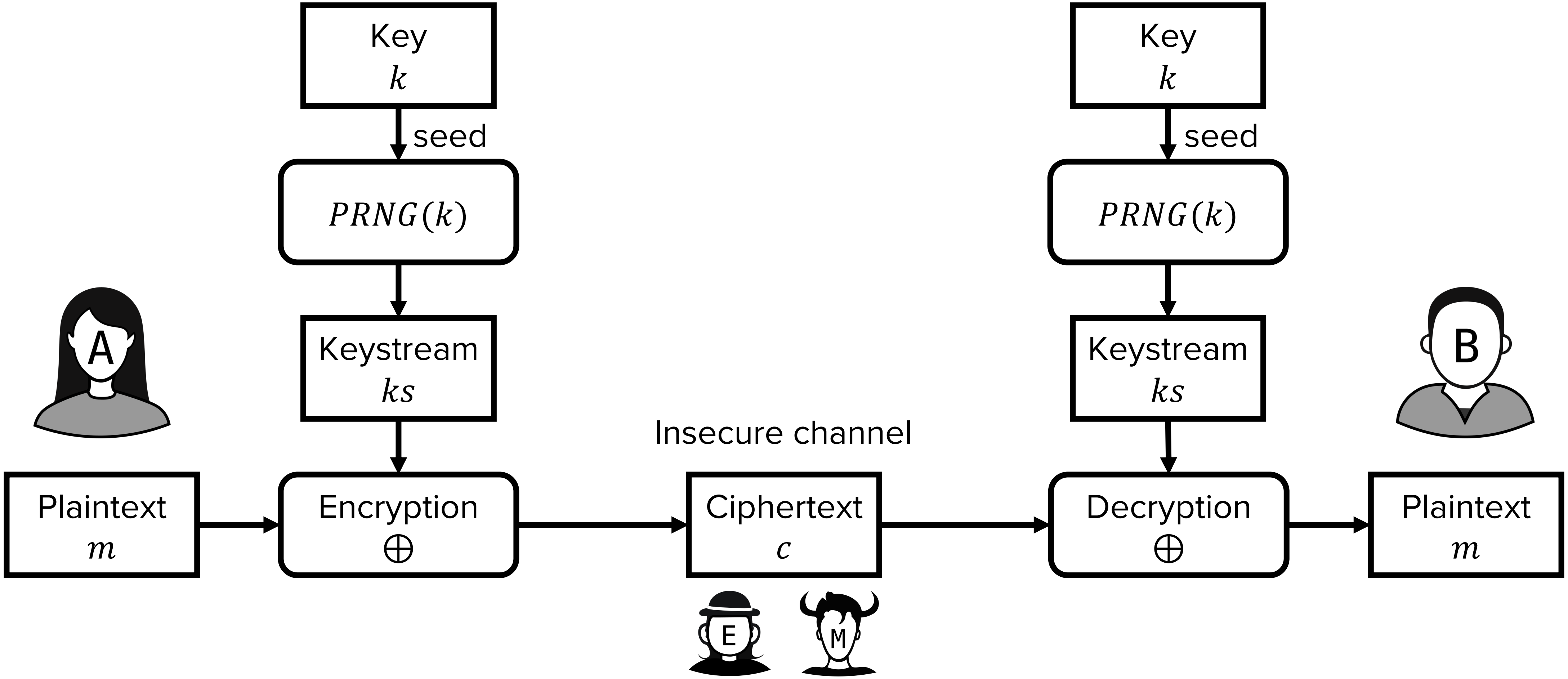
Again, how would a computer do this?

Background: PRNG

- Pseudorandom Number Generator (PRNG): An angorithm that utilizes a small seed of true randomness to produce outputs that appear random
  - Generate seed from expensive true randomness (e.g., environmental noise from device drivers, such as keystroke intervals)
  - Seed a PRNG algorithm
  - Generate pseudorandom numbers quickly and cheaply
- PRNG outputs are <u>deterministic</u>
  - However, it is computationally indistinguishable from true randomness

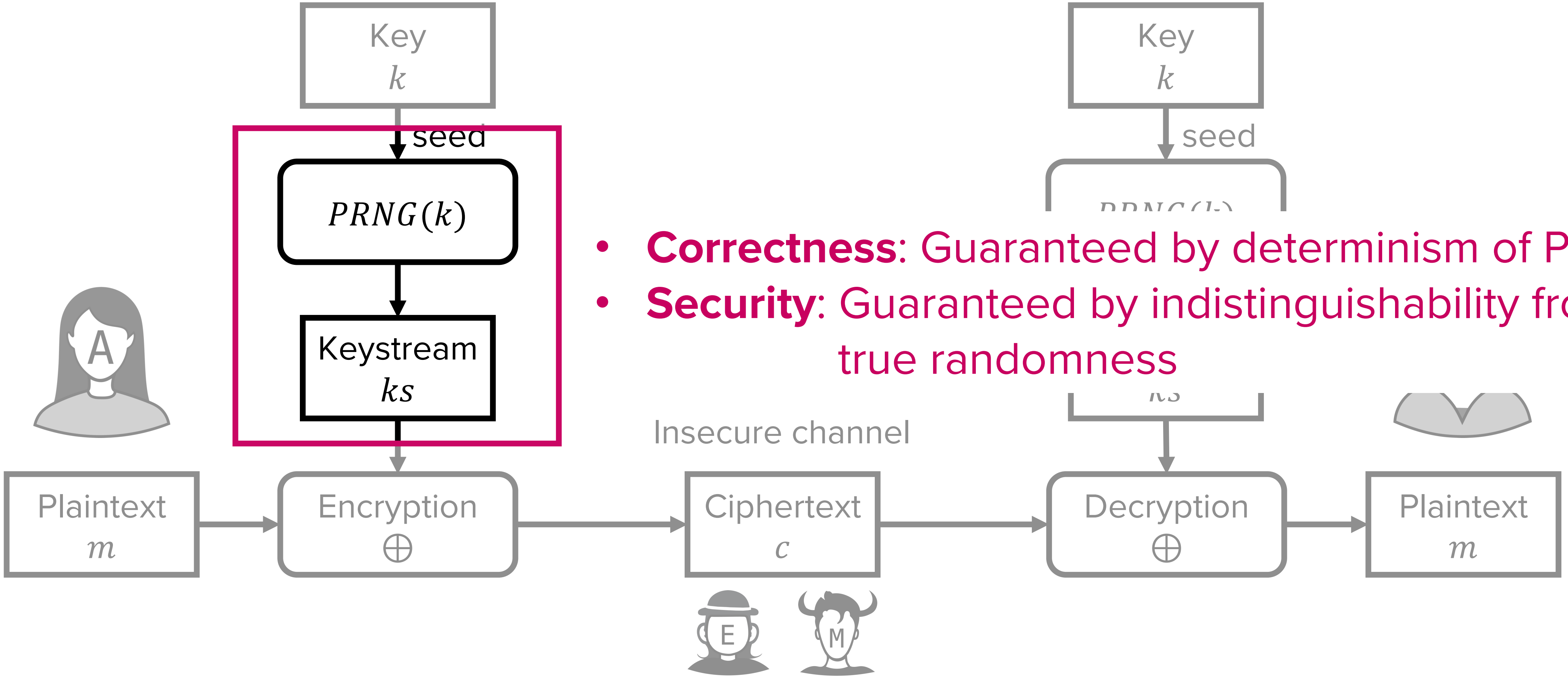
Back to stream cipher…

- Setting



Evaluating stream cipher

- Setting



- **Correctness**: Guaranteed by determinism of PRNG
- **Security**: Guaranteed by indistinguishability from true randomness

Example: Rivest Cipher (RC4) (1987)

- Encrypts one byte at a time (stream cipher)
- Key k: 1 to 256 bytes
  - k[0], …, k[255]
- Consists of a Key Scheduling Algorithm (KSA) and Pseudo-Random Generation Algorithm (PRGA)

Example: Rivest Cipher (RC4) (1987)

- KSA
  - Key array: k[0], …, k[255]
  - Initial S array: S[0] = 0, S[1] = 1, …, S[255] = 255

```
j = 0
for i in range(256):
    j = (j + S[i] + k[i]) mod 256
    swap S[i] and S[j]
```

→ initializes the S-box array

Example: Rivest Cipher (RC4) (1987)

POSTECH

- PRGA
  - Key array: k[0], ..., k[255]
  - S-box array: S[0], ..., S[255] (initialized by KSA)

```
i, j, l = 0
for each byte in plaintext m:
    i = (i + 1) mod 256
    j = (j + S[i]) mod 256
    swap S[i] and S[j]
    t = (S[i] + S[j]) mod 256
    KS[l] = S[t]
    l += 1
```

→ Keystream ks is generated

Encryption:
c[idx] = m[idx] ⊕ KS[idx]

Example: Rivest Cipher (RC4) (1987)

- Security of RC4
  - Many known weaknesses exist
    - Biased keystream
    - Biased outputs
    - Inferrable correlation between keystream and the key
    - …
  - Despite its efficiency and simplicity, RC4 is no longer recommended for cryptographic applications
    - Secure alternatives: ChaCha20, AES-CTR, …

Cryptography roadmap

Goal \ Scheme	Symmetric Key	Asymmetric Key
Confidentiality	✅ One Time Pad (OTP) ✅ Block ciphers (DES, AES) ✅ Stream ciphers	• ElGamal encryption • RSA encryption
Integrity & Authentication	• Message Authentication Code (MAC)	• Digital signature

Coming up next

- Limitations of symmetric schemes
  - Key needs to be securely shared
  - Too many keys are needed
    - 2 keys for 2 ppl, 3 keys for 3 ppl, 6 keys for 4 ppl, 10 keys for 5 ppl, …

→ Asymmetric schemes were introduced

Questions?