# Symmetric Key Encryption

**Jiuqin ZHOU**

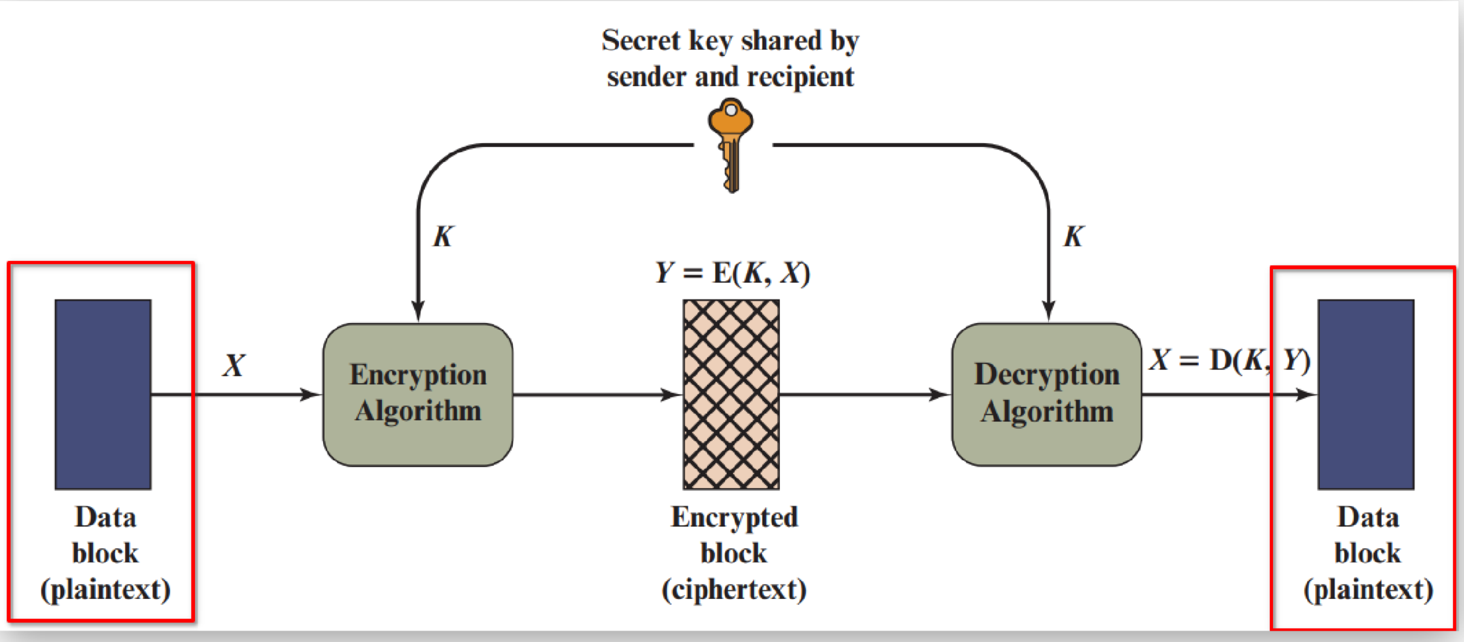
# Outline

1. Classical Ciphers
    1. Symmetric Cipher Model →
    2. Substitution Techniques →
    3. Transposition Techniques →
2. Block Ciphers
    1. DES: Data Encryption Standard →
    2. AES: Advanced Encryption Standard →
3. Stream Ciphers
    1. Pseudorandom Number Generation →
    2. RC4: Rivest Cipher →

# Classical Ciphers » Symmetric Cipher Model

## Simplified Model

Two Requirements

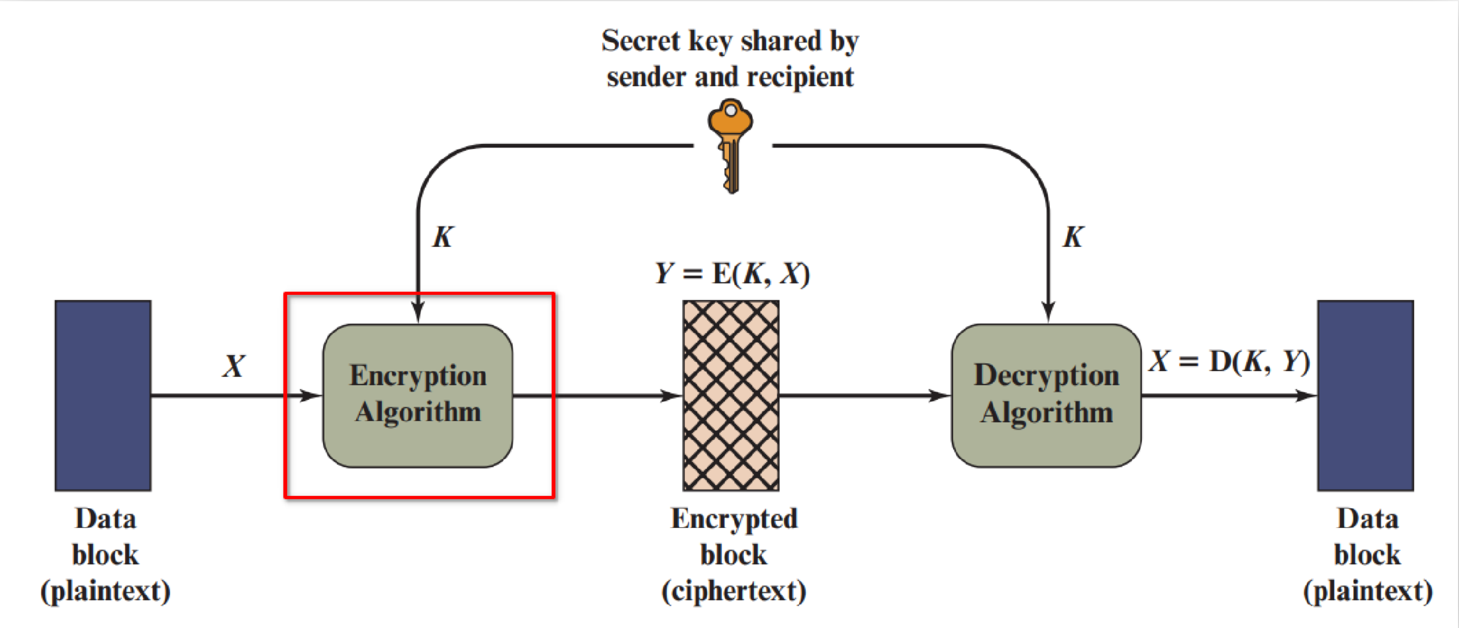


**Plaintext**: This is the original intelligible message or data that is fed into the algorithm as input.

# Classical Ciphers » Symmetric Cipher Model

## Simplified Model

Two Requirements

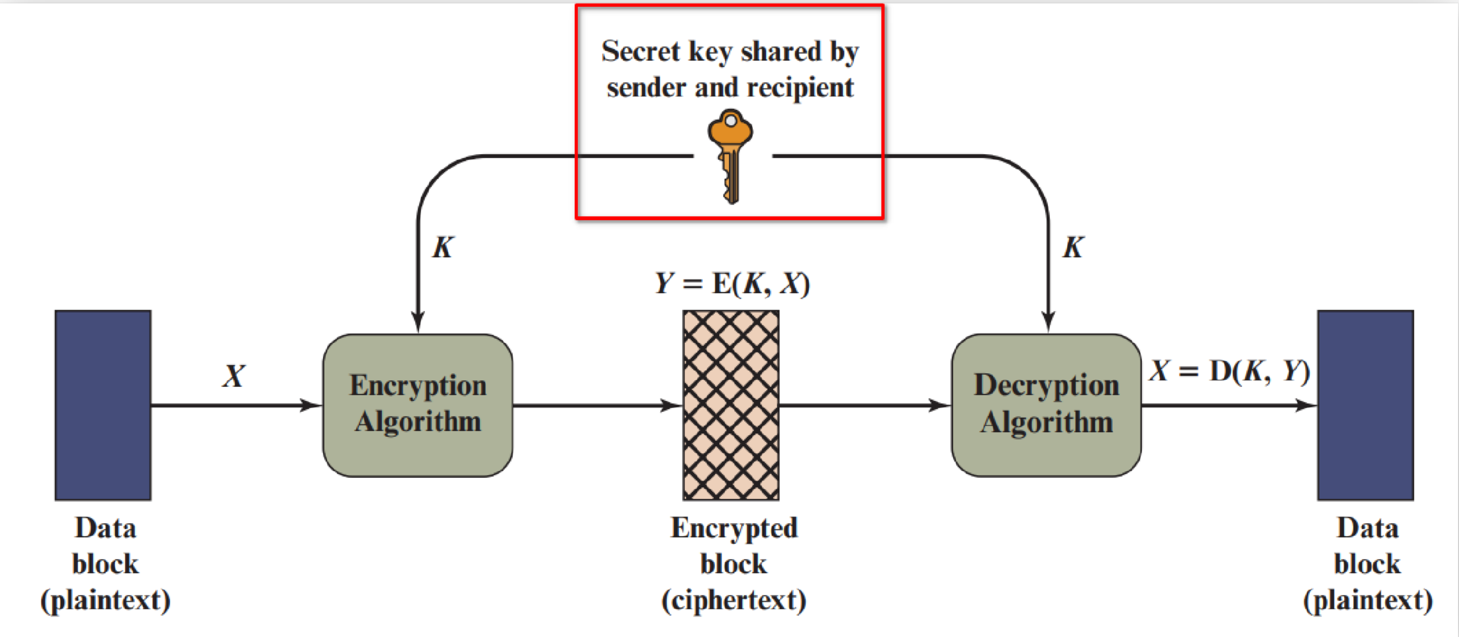


**Encryption algorithm**: The encryption algorithm performs various substitutions and transformations on the plaintext.

# Classical Ciphers » Symmetric Cipher Model

## Simplified Model

Two Requirements

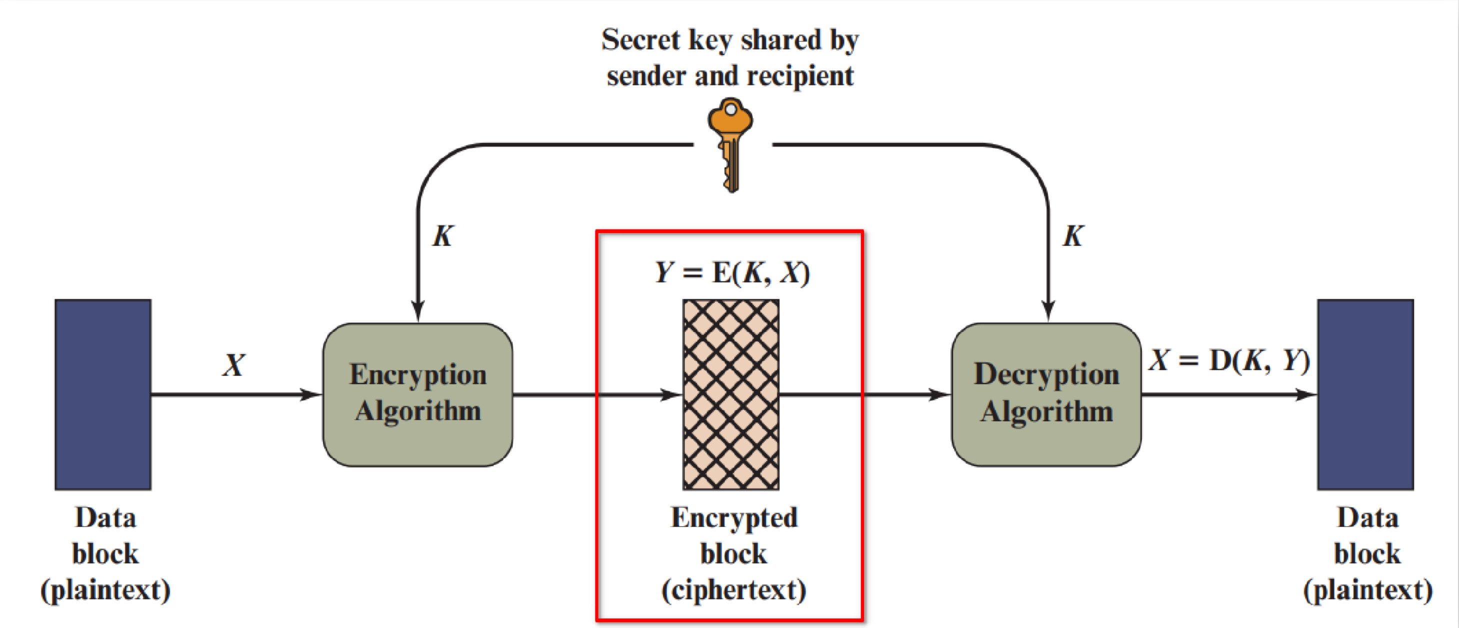


**Secret key**: The secret key is the input to the encryption algorithm as well as the decryption algorithm.

# Classical Ciphers » Symmetric Cipher Model

## Simplified Model

Two Requirements

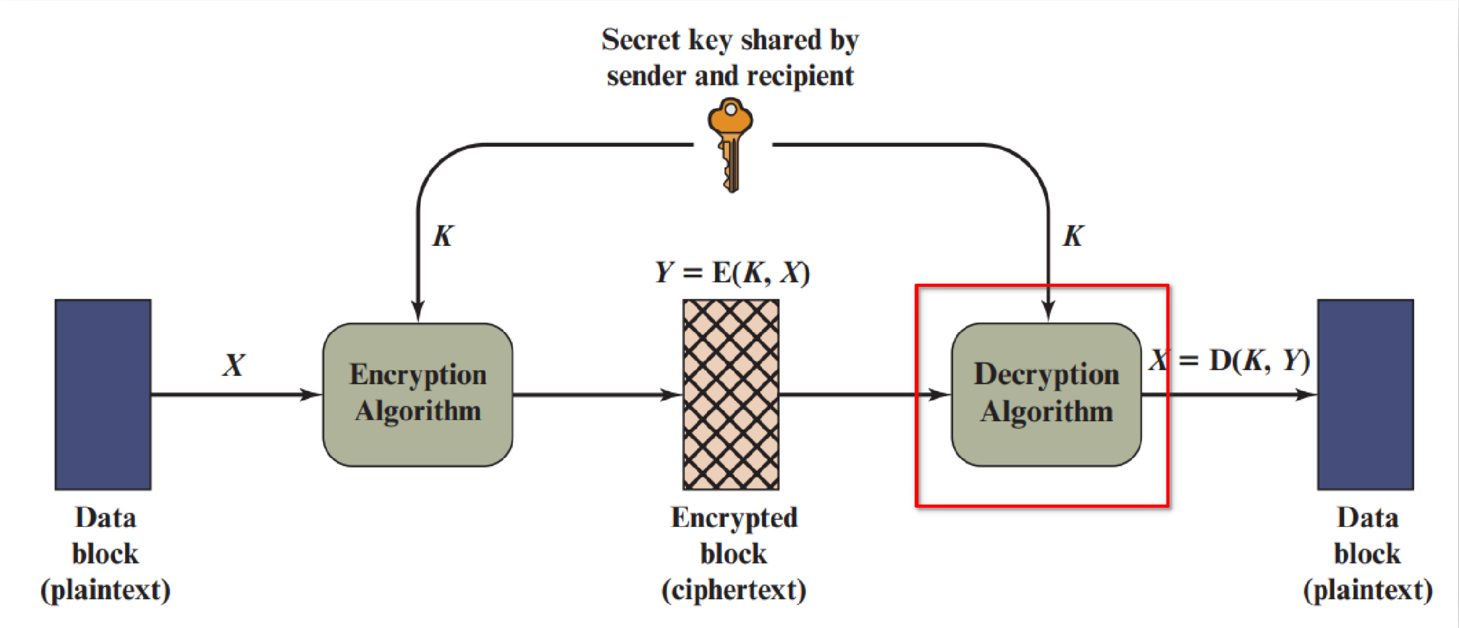


**Ciphertext**: This is the scrambled message produced as output of the encryption algorithm.

# Classical Ciphers » Symmetric Cipher Model

## Simplified Model

Two Requirements



**Decryption algorithm**: This is essentially the encryption algorithm run in reverse.

# Classical Ciphers » Symmetric Cipher Model

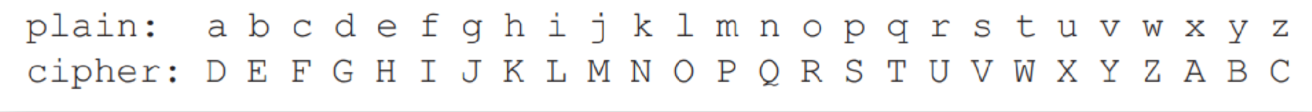
Simplified Model

**Two Requirements**

1. **Strong Encryption Algorithm**: An opponent who knows the algorithm and has access to one or more ciphertexts would be unable to decipher the ciphertext or figure out the key.
2. **Shared Secret Key**: Sender and receiver must have obtained copies of the secret key in a secure fashion and must keep the key secure.

# Classical Ciphers » Substitution Techniques

- **Caesar Cipher**
- Monoalphabetic Cipher
- Playfair Cipher
- Hill Cipher
- Vigenère Cipher
- One-Time Padding



```
plain:   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
cipher:  D E F G H I J K L M N O P Q R S T U V W X Y Z A B C
```

**Encryption Algorithm**:

$$C_i = E(k, M_i) = (M_i + k) \mod 26$$

**Decryption Algorithm**:

$$M_i = E(k, M_i) = (C_i - k) \mod 26$$

# Classical Ciphers » Substitution Techniques

Caesar Cipher

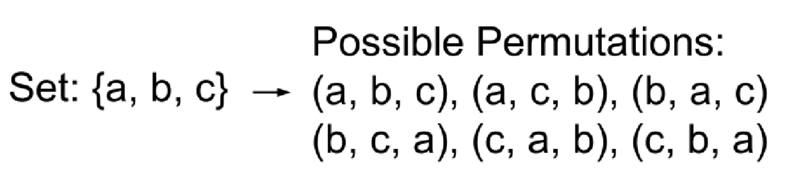
**Monoalphabetic Cipher**

Playfair Cipher

Hill Cipher

Vigenère Cipher

One-Time Padding



Set: {a, b, c} → Possible Permutations:
(a, b, c), (a, c, b), (b, a, c)
(b, c, a), (c, a, b), (c, b, a)

- A **permutation** of a finite set of elements $S$ is an ordered sequence of all the elements of $S$, with each element appearing exactly once.
- **Encryption Algorithm**: $C_i = E(S^+, M_i) = S^+_{M_i}$.
- **Decryption Algorithm**: $M_i = E(S^-, C_i) = S^-_{C_i}$.

# Classical Ciphers » Substitution Techniques

Caesar Cipher

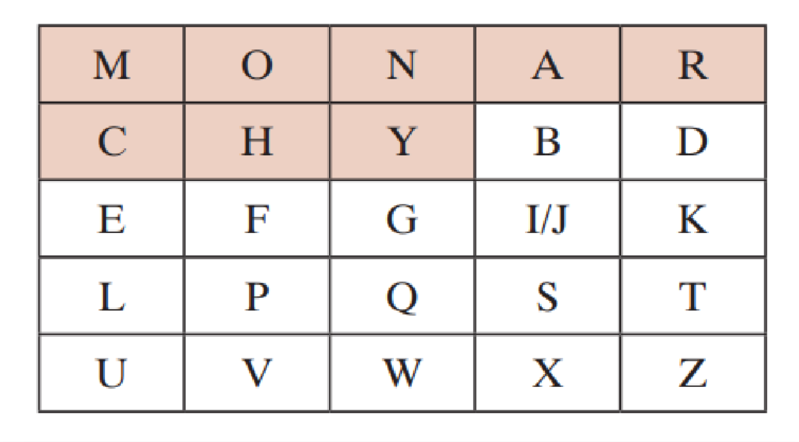
Monoalphabetic Cipher

**Playfair Cipher**

Hill Cipher

Vigenère Cipher

One-Time Padding



| | | | | |
|---|---|---|---|---|
| M | O | N | A | R |
| C | H | Y | B | D |
| E | F | G | I/J | K |
| L | P | Q | S | T |
| U | V | W | X | Z |

**Keyword**: monarchy

## Classical Ciphers » Substitution Techniques

Caesar Cipher

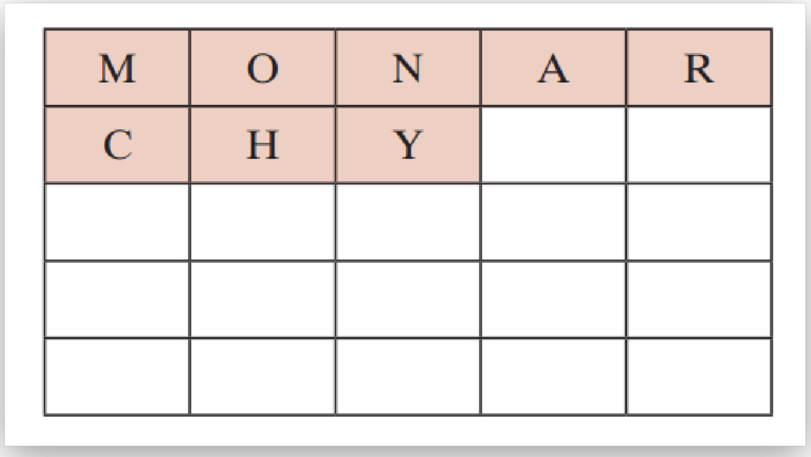
Monoalphabetic Cipher

**Playfair Cipher**

Hill Cipher

Vigenère Cipher

One-Time Padding



| | | | | |
|---|---|---|---|---|
| M | O | N | A | R |
| C | H | Y | | |
| | | | | |
| | | | | |
| | | | | |

**Matrix Construction Step 1**: Filling in the letters of the keyword (minus duplicates) from left to right and from top to bottom

# Classical Ciphers » Substitution Techniques

Caesar Cipher

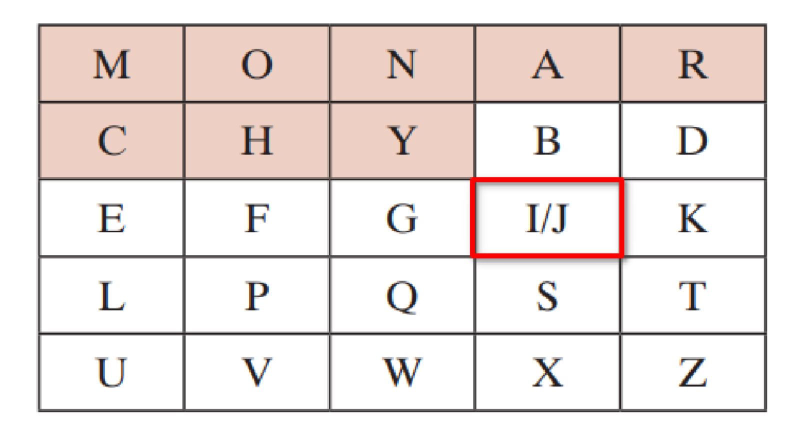
Monoalphabetic Cipher

**Playfair Cipher**

Hill Cipher

Vigenère Cipher

One-Time Padding

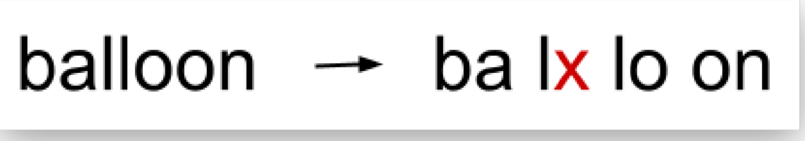


| | | | | |
|---|---|---|---|---|
| M | O | N | A | R |
| C | H | Y | B | D |
| E | F | G | I/J | K |
| L | P | Q | S | T |
| U | V | W | X | Z |

**Matrix Construction Step 2**: Filling in the remainder of the matrix with the remaining letters in alphabetic order. The letters I and J count as one letter.

# Classical Ciphers » Substitution Techniques

- Caesar Cipher
- Monoalphabetic Cipher
- **Playfair Cipher**
- Hill Cipher
- Vigenère Cipher
- One-Time Padding

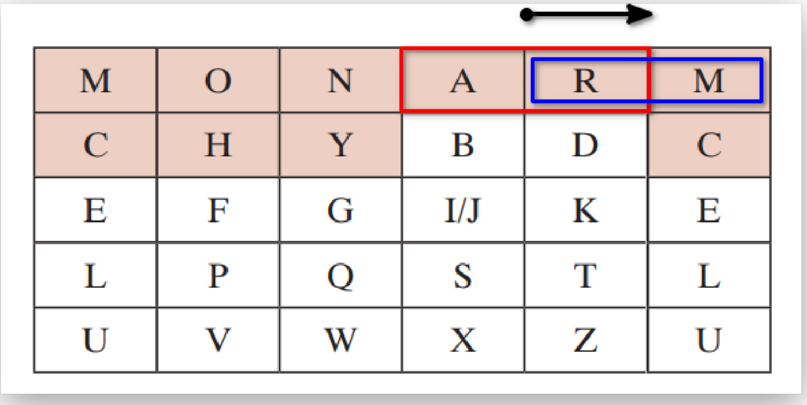


balloon → ba lx lo on

**Encryption Algorithm Rule 1**: Repeating plaintext letters that are in the same pair are separated with a filler letter, such as x, so that balloon would be treated as ba lx lo on.

# Classical Ciphers » Substitution Techniques

- Caesar Cipher
- Monoalphabetic Cipher
- **Playfair Cipher**
- Hill Cipher
- Vigenère Cipher
- One-Time Padding



| | | | | | |
|---|---|---|---|---|---|
| M | O | N | A | R | M |
| C | H | Y | B | D | C |
| E | F | G | I/J | K | E |
| L | P | Q | S | T | L |
| U | V | W | X | Z | U |

**Encryption Algorithm Rule 2**: Two plaintext letters that fall in the same row of the matrix are each replaced by the letter to the right, with the first element of the row circularly following the last. For example, AR is encrypted as RM.

# Classical Ciphers » Substitution Techniques

Caesar Cipher

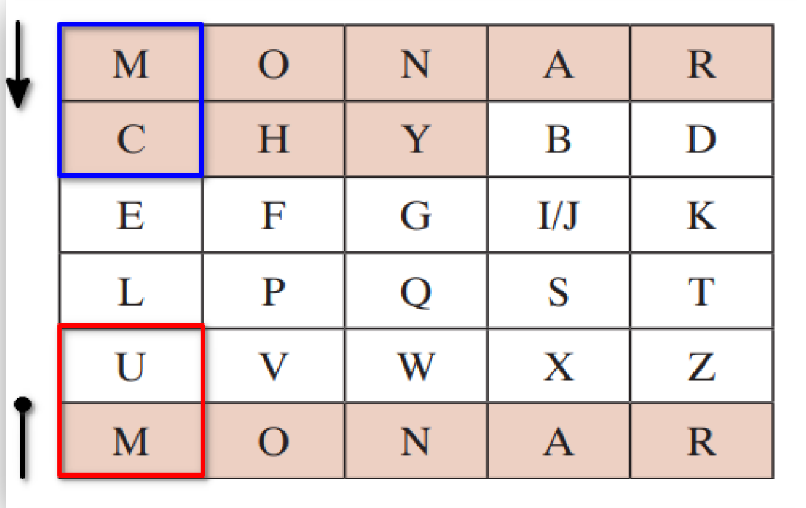
Monoalphabetic Cipher

**Playfair Cipher**

Hill Cipher

Vigenère Cipher

One-Time Padding



| | | | | |
|---|---|---|---|---|
| M | O | N | A | R |
| C | H | Y | B | D |
| E | F | G | I/J | K |
| L | P | Q | S | T |
| U | V | W | X | Z |
| M | O | N | A | R |

**Encryption Algorithm Rule 3**: Two plaintext letters that fall in the same column are each replaced by the letter beneath, with the top element of the column circularly following the last. For example, MU is encrypted as CM.

# Classical Ciphers » Substitution Techniques

Caesar Cipher

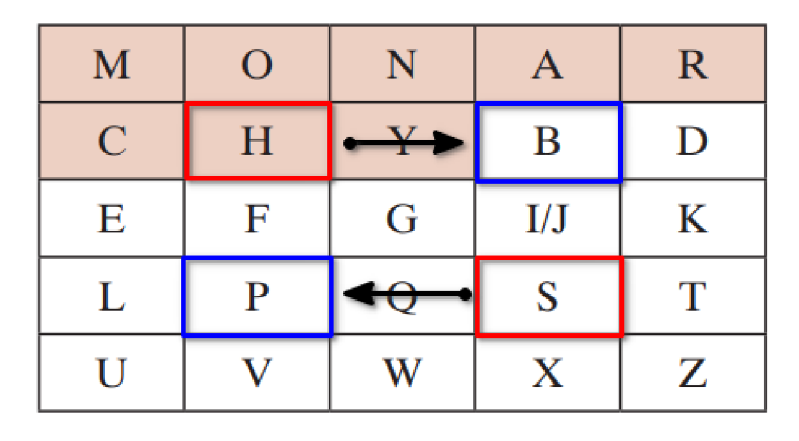
Monoalphabetic Cipher

**Playfair Cipher**

Hill Cipher

Vigenère Cipher

One-Time Padding



| | | | | |
|---|---|---|---|---|
| M | O | N | A | R |
| C | H | Y | B | D |
| E | F | G | I/J | K |
| L | P | Q | S | T |
| U | V | W | X | Z |

**Encryption Algorithm Rule 4**: Otherwise, each plaintext letter in a pair is replaced by the letter that lies in its own row and the column occupied by the other plaintext letter. Thus, HS becomes BP and EA becomes IM (or JM, as the encipherer wishes).

## Classical Ciphers » Substitution Techniques

Caesar Cipher

Monoalphabetic Cipher

Playfair Cipher

**Hill Cipher**

Vigenère Cipher

One-Time Padding

**Encryption Algorithm**:

$$C = E(K, M) = MK \mod 26$$

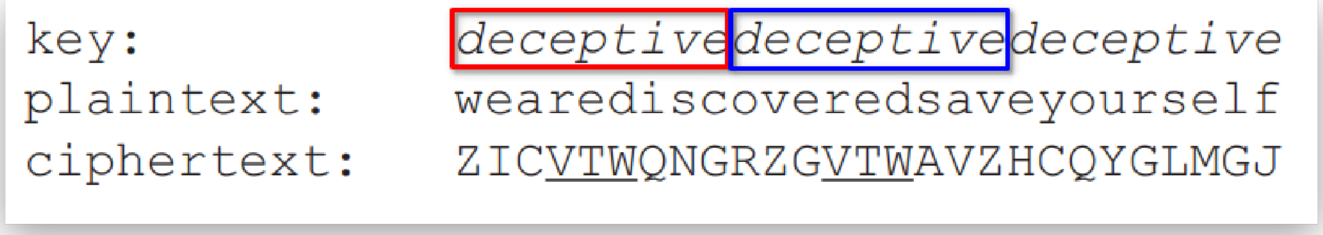
**Decryption algorithm**:

$$M = D(K, C) = CK^{-1} \mod 26$$

- $C$ and $M$ are row vectors of length $n$ representing the plaintext and ciphertext
- $K$ is a $n \times n$ matrix representing the encryption key.

# Classical Ciphers » Substitution Techniques

- Caesar Cipher
- Monoalphabetic Cipher
- Playfair Cipher
- Hill Cipher
- **Vigenère Cipher**
- One-Time Padding



```
key:          deceptivedeceptivedeceptive
plaintext:    wearediscoveredsaveyourself
ciphertext:   ZICVTWQNGRZGVTWAVZHCQYGLMGJ
```

**Encryption Algorithm**:

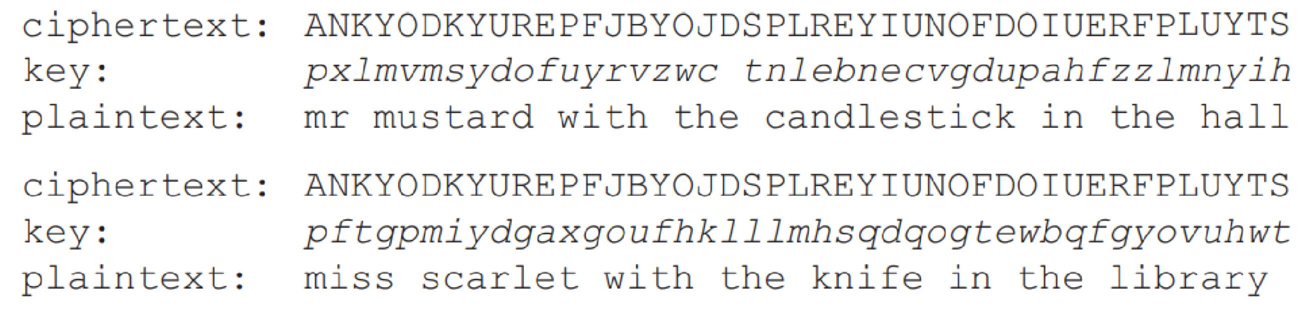
$$C_i = E(K, M_i) = (M_i + K_{(i \mod len(K))}) \mod 26$$

**Decryption Algorithm**:

$$C_i = E(K, M_i) = (M_i - K_{(i \mod len(K))}) \mod 26$$

# Classical Ciphers » Substitution Techniques

- Caesar Cipher
- Monoalphabetic Cipher
- Playfair Cipher
- Hill Cipher
- Vigenère Cipher
- **One-Time Padding**



```
ciphertext: ANKYODKYUREPFJBYOJDSPLREYIUNOFDOIUERFPLUYTS
key:        pxlmvmsydofuyrvzwc tnlebnecvgdupahfzzlmnyih
plaintext:  mr mustard with the candlestick in the hall

ciphertext: ANKYODKYUREPFJBYOJDSPLREYIUNOFDOIUERFPLUYTS
key:        pftgpmiydgaxgoufhklllmhsqdqogtewbqfgyovuhwt
plaintext:  miss scarlet with the knife in the library
```

Using a random key that is as long as the message, so that the key need not be repeated.

# Classical Ciphers » Substitution Techniques

Caesar Cipher

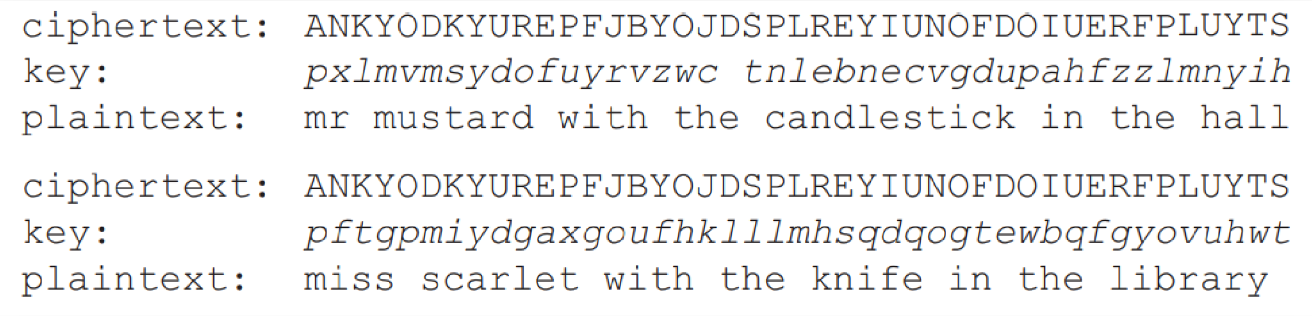
Monoalphabetic Cipher

Playfair Cipher

Hill Cipher

Vigenère Cipher

**One-Time Padding**



```
ciphertext:  ANKYODKYUREPFJBYOJDSPLREYIUNOFDOIUERFPLUYTS
key:         pxlmvmsydofuyrvzwc tnlebnecvgdupahfzzlmnyih
plaintext:   mr mustard with the candlestick in the hall

ciphertext:  ANKYODKYUREPFJBYOJDSPLREYIUNOFDOIUERFPLUYTS
key:         pftgpmiydgaxgoufhklllmhsqdqogtewbqfgyovuhwt
plaintext:   miss scarlet with the knife in the library
```

**Encryption Algorithm**:

$$C_i = E(K, M_i) = (M_i + K_i) \mod 26$$

**Decryption ALgorithm**:

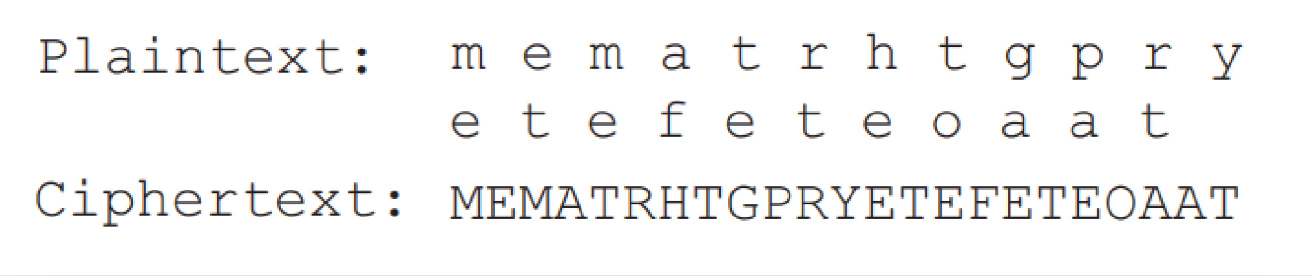
$$M_i = D(K, M_i) = (C_i - K_i) \mod 26$$

# Classical Ciphers » Transposition Techniques

**Rail Fence Cipher**

Column Order

Permutation



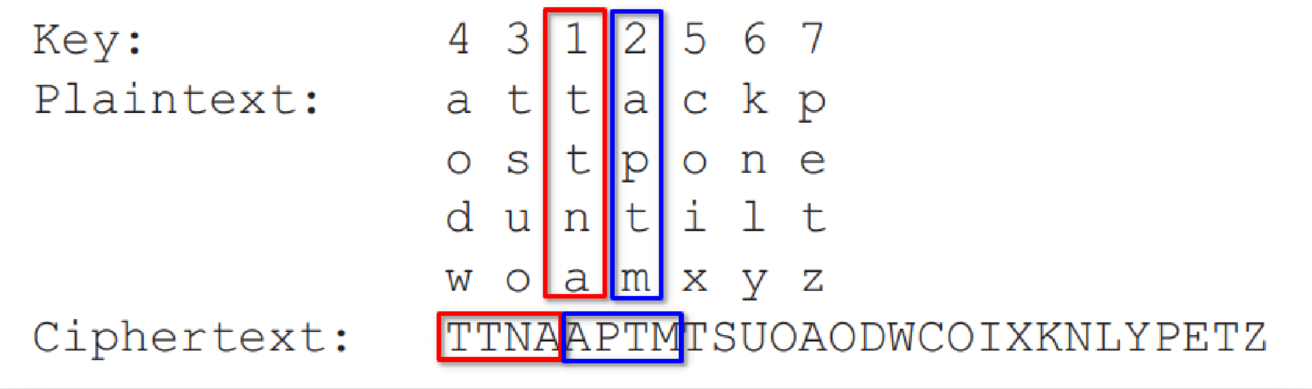
```
Plaintext:   m e m a t r h t g p r y
             e t e f e t e o a a t
Ciphertext:  MEMATRHTGPRYETEFETEOAAT
```

**Encryption Algorithm**: The plaintext is written down as a sequence of rectangle columns and then read off as a sequence of rows.

## Classical Ciphers » Transposition Techniques

Rail Fence Cipher

**Column Order Permutation**



```
Key:           4 3 1 2 5 6 7
Plaintext:     a t t a c k p
               o s t p o n e
               d u n t i l t
               w o a m x y z
Ciphertext:    TTNAAPTMTSUOAODWCOIXKNLYPETZ
```

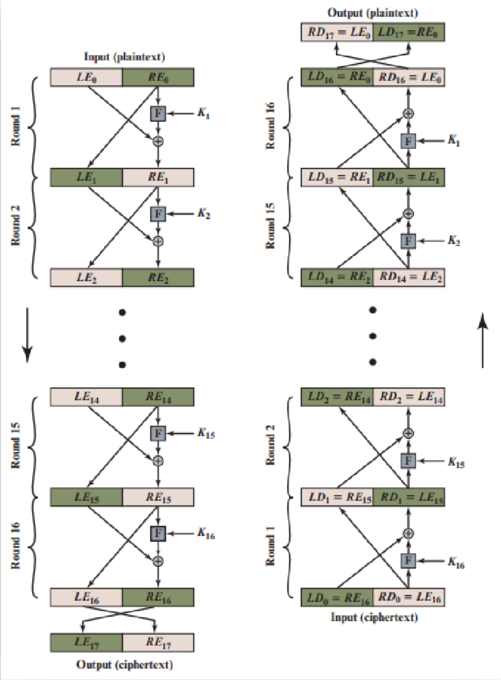
**Encryption Algorithm**: Write the message in a rectangle, row by row, and read the message off, column by column, but permute the order of the columns.

# Block Ciphers » DES: Data Encryption Standard

**Feistel Structure**

Key Generation

Round Function



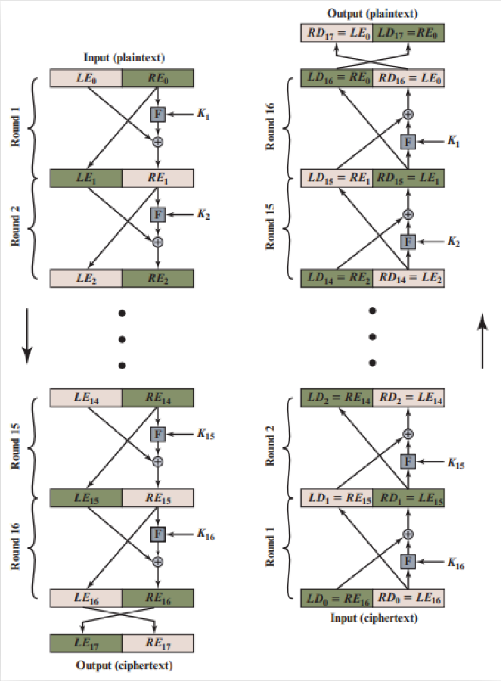
- The inputs to the encryption algorithm are a plaintext block of length $2w$ bits and a key $K$
- The plaintext block is divided into two halves, $LE_0$ and $RE_0$.

# Block Ciphers » DES: Data Encryption Standard

**Feistel Structure**

Key Generation

Round Function



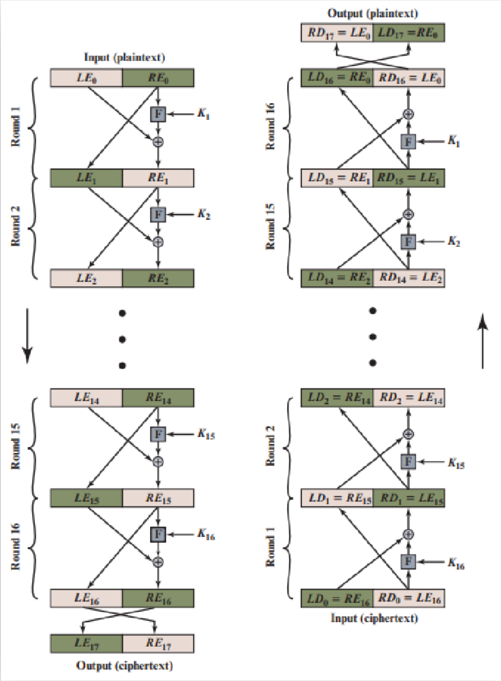
- The two halves of the data pass through $n$ rounds of processing and then combine to produce the ciphertext block.
- Each round $i$ has as inputs $LE_{i-1}$ and $RE_{i-1}$ derived from the previous round, as well as a subkey $K_i$ derived from the overall $K$.

# Block Ciphers » DES: Data Encryption Standard

**Feistel Structure**

Key Generation

Round Function



**Encryption Algorithm**:

$$LE_i = RE_{i-1}$$
$$RE_i = LE_{i-1} \oplus F(RE_{i-1}, K_i)$$

**Decryption Algorithm**:

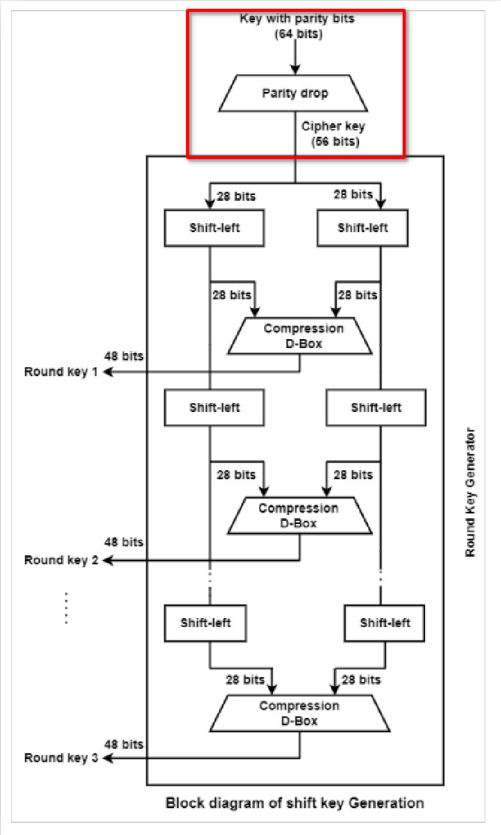
$$LD_i = RD_{i-1}$$
$$RD_i = LD_{i-1} \oplus F(RD_{i-1}, K_{n+1-i})$$

# Block Ciphers » DES: Data Encryption Standard

Feistel Structure

**Key Generation**

Round Function



Block diagram of shift key Generation

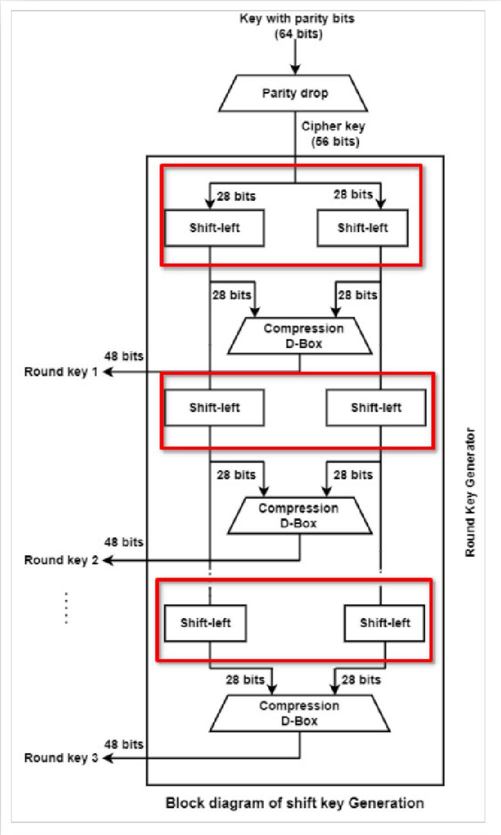
- The cipher key is provided as 64 bit key in which 8 extra bits are parity bits (bit 8, 16, 24, 32 … 64), which are discarded before the actual key generation process begins.
- Parity Drop then permutes the remaining bit according to the pre-defined of the parity bit drop table.

# Block Ciphers » DES: Data Encryption Standard

Feistel Structure

**Key Generation**

Round Function



Block diagram of shift key Generation

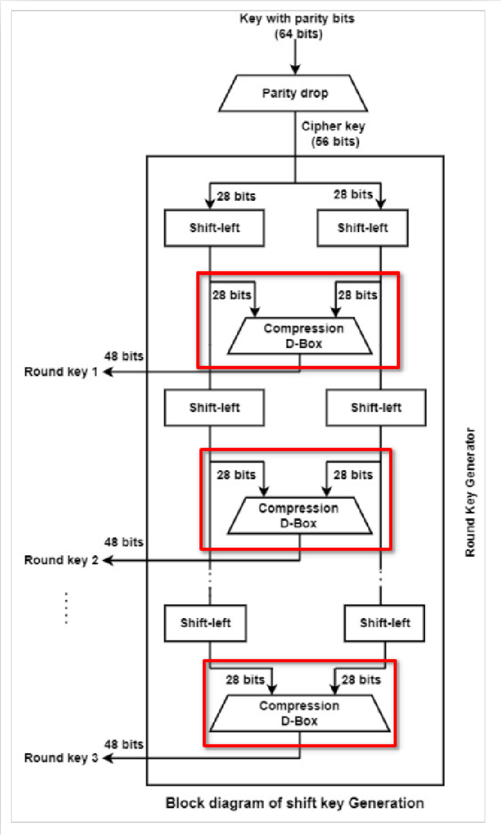
- After the permutation, the keys are divided into two 28 bits parts. Each part is changed left one or two bits is depend on the rounds.
- In round 1, 2, 9, and 16 shifting is one bit and in the other rounds it is two bits.

# Block Ciphers » DES: Data Encryption Standard

Feistel Structure

**Key Generation**

Round Function



Block diagram of shift key Generation

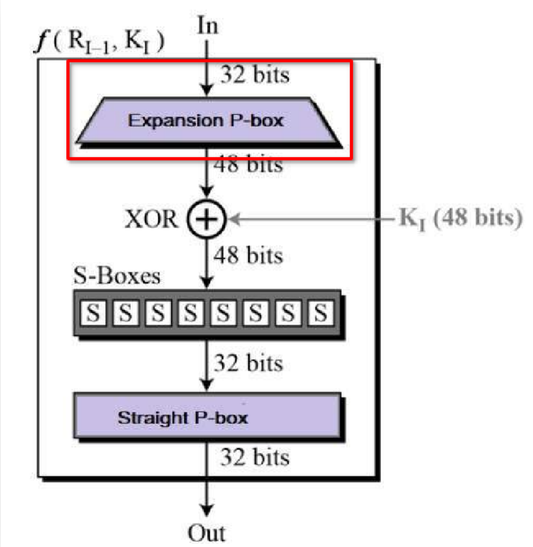
- The two parts are ingredients to build a 56 bit part. Thus the compression D-box transform it into 48 bit. These 48 bits are being utilized as a key for a round.

# Block Ciphers » DES: Data Encryption Standard

Feistel Structure

Key Generation

**Round Function**



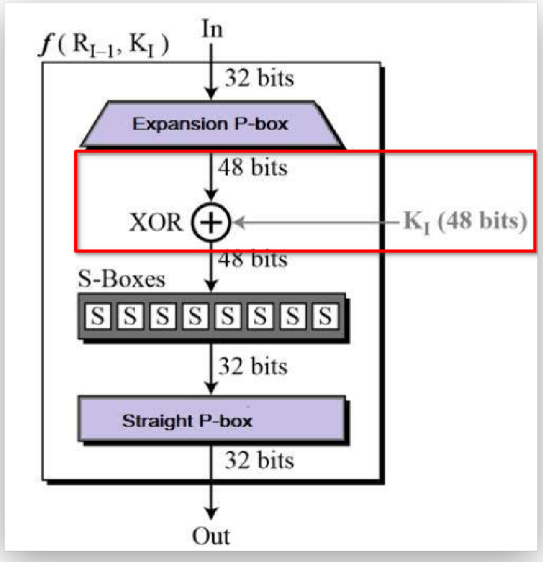
- The first step is the Expansion Permutation Box. Since right input is 32-bit and round key is a 48-bit, we first need to expand right input to 48 bits.

# Block Ciphers » DES: Data Encryption Standard

Feistel Structure

Key Generation

**Round Function**



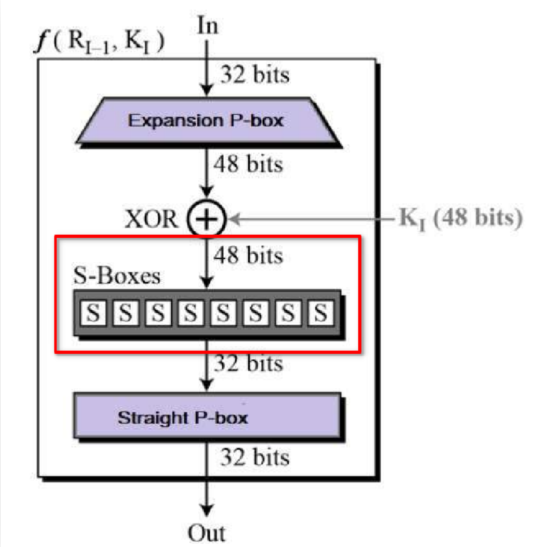
- After the expansion permutation comes the second step. DES does XOR operation on the expanded right section and the round key. The round key is used only in this operation.

# Block Ciphers » DES: Data Encryption Standard

Feistel Structure

Key Generation

**Round Function**



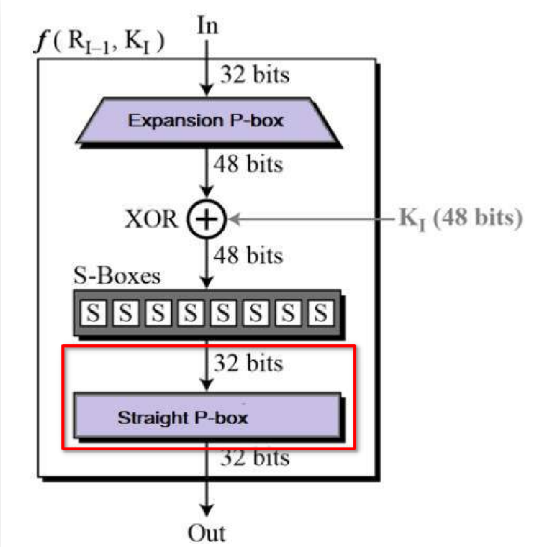
- The third step is Substitution Boxes. The S-boxes carry out the real mixing, namely confusion. DES uses 8 S-boxes, each with a 6-bit input and a 4-bit output.

# Block Ciphers » DES: Data Encryption Standard

Feistel Structure

Key Generation

**Round Function**



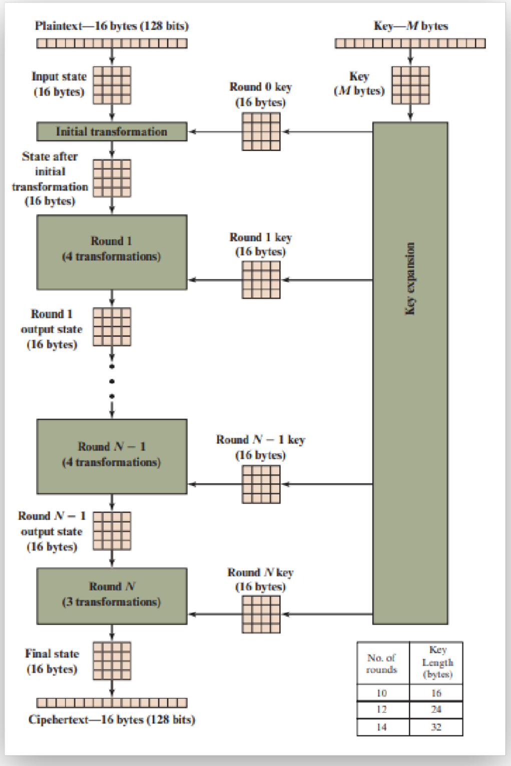
- The last step is straight permutation. The 32 bit output of S-boxes is then subjected to the straight permutation of pre-defined rules.

# Block Ciphers » AES: Advanced Encryption Standard

**AES Structure**

Transformation Functions

Key Expansion


Plaintext—16 bytes (128 bits)
Key—$M$ bytes
Input state (16 bytes)
Round 0 key (16 bytes)
Key ($M$ bytes)
Initial transformation
State after initial transformation (16 bytes)
Round 1 (4 transformations)
Round 1 key (16 bytes)
Key expansion
Round 1 output state (16 bytes)
Round $N - 1$ (4 transformations)
Round $N - 1$ key (16 bytes)
Round $N - 1$ output state (16 bytes)
Round $N$ (3 transformations)
Round $N$ key (16 bytes)
Final state (16 bytes)
Cipehertext—16 bytes (128 bits)

| No. of rounds | Key Length (bytes) |
|---|---|
| 10 | 16 |
| 12 | 24 |
| 14 | 32 |

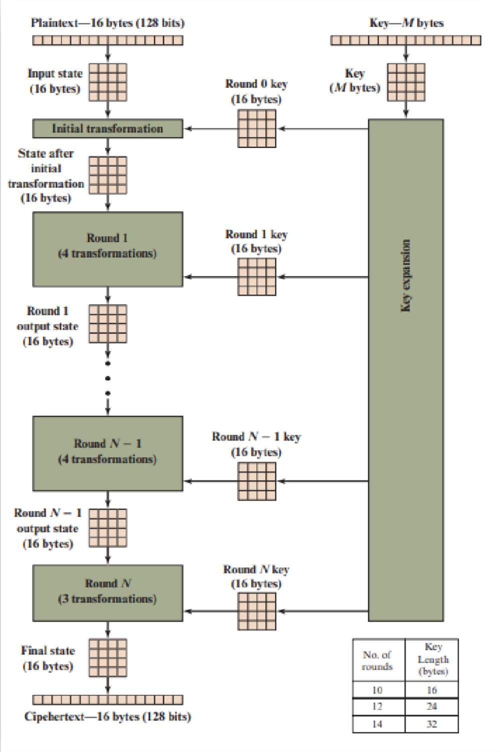
- The cipher takes a plaintext block size of 128 bits, or 16 bytes. The key length can be 16, 24, or 32 bytes (128, 192, or 256 bits). The algorithm is referred to as AES-128, AES-192, or AES-256.
- The cipher consists of $N$ rounds, where the number of rounds depends on the key length: 10 rounds for a 16-byte key, 12 rounds for a 24-byte key, and 14 rounds for a 32-byte key.

# Block Ciphers » AES: Advanced Encryption Standard

**AES Structure**

Transformation Functions

Key Expansion


Plaintext—16 bytes (128 bits)
Key—$M$ bytes
Input state (16 bytes)
Key ($M$ bytes)
Round 0 key (16 bytes)
Initial transformation
State after initial transformation (16 bytes)
Round 1 (4 transformations)
Round 1 key (16 bytes)
Key expansion
Round 1 output state (16 bytes)
Round $N-1$ (4 transformations)
Round $N-1$ key (16 bytes)
Round $N-1$ output state (16 bytes)
Round $N$ (3 transformations)
Round $N$ key (16 bytes)
Final state (16 bytes)
Cipehertext—16 bytes (128 bits)

| No. of rounds | Key Length (bytes) |
|---|---|
| 10 | 16 |
| 12 | 24 |
| 14 | 32 |

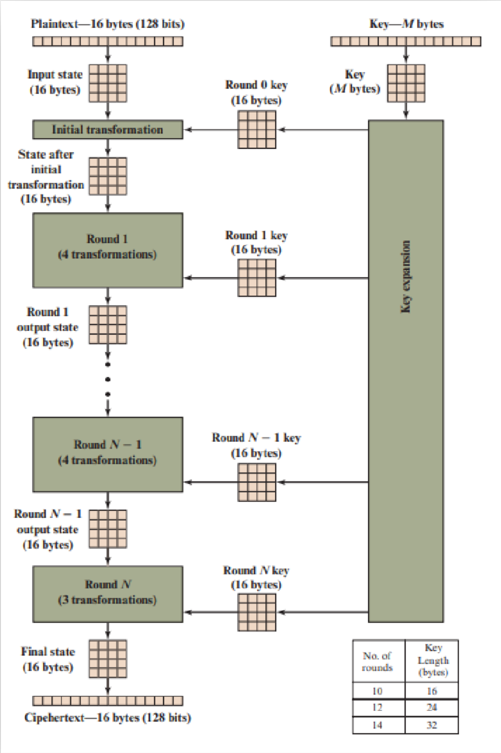
- The first $N-1$ rounds consist of four distinct transformation functions: SubBytes, ShiftRows, MixColumns, and AddRoundKey.
- The final round contains only three transformations.
- There is a initial single transformation (AddRoundKey) before the first round.

# Block Ciphers » AES: Advanced Encryption Standard

**AES Structure**

Transformation Functions

Key Expansion


Plaintext—16 bytes (128 bits)
Key—M bytes
Input state (16 bytes)
Key (M bytes)
Round 0 key (16 bytes)
Initial transformation
State after initial transformation (16 bytes)
Round 1 (4 transformations)
Round 1 key (16 bytes)
Key expansion
Round 1 output state (16 bytes)
Round N − 1 (4 transformations)
Round N − 1 key (16 bytes)
Round N − 1 output state (16 bytes)
Round N (3 transformations)
Round N key (16 bytes)
Final state (16 bytes)
Cipehertext—16 bytes (128 bits)

| No. of rounds | Key Length (bytes) |
|---|---|
| 10 | 16 |
| 12 | 24 |
| 14 | 32 |

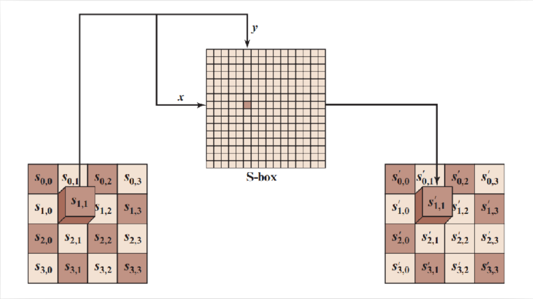
- The output of each round is a 4 * 4 matrix, with the output of the final round being the ciphertext.
- Also, the key expansion function generates N + 1 round keys, each of which is a distinct 4 * 4 matrix.

# Block Ciphers » AES: Advanced Encryption Standard

AES Structure

**Transformation Functions**

Key Expansion



y
x
S-box
$s_{0,0}$ $s_{0,1}$ $s_{0,2}$ $s_{0,3}$
$s_{1,0}$ $s_{1,1}$ $s_{1,2}$ $s_{1,3}$
$s_{2,0}$ $s_{2,1}$ $s_{2,2}$ $s_{2,3}$
$s_{3,0}$ $s_{3,1}$ $s_{3,2}$ $s_{3,3}$
$s'_{0,0}$ $s'_{0,1}$ $s'_{0,2}$ $s'_{0,3}$
$s'_{1,0}$ $s'_{1,1}$ $s'_{1,2}$ $s'_{1,3}$
$s'_{2,0}$ $s'_{2,1}$ $s'_{2,2}$ $s'_{2,3}$
$s'_{3,0}$ $s'_{3,1}$ $s'_{3,2}$ $s'_{3,3}$

**Substitute Bytes Transformation**

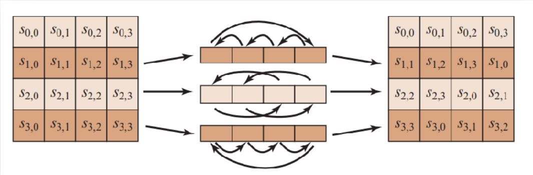
- AES defines a 16 * 16 matrix of byte values, called an S-box, that contains a permutation of all possible 256 8-bit values.
- The leftmost 4 bits of a byte are used as a row value and the rightmost 4 bits are used as a column value.
- These row and column values serve as indexes into the S-box to select a unique 8-bit output value.

# Block Ciphers » AES: Advanced Encryption Standard

AES Structure

**Transformation Functions**

Key Expansion



|  |  |  |  |
|---|---|---|---|
| $s_{0,0}$ | $s_{0,1}$ | $s_{0,2}$ | $s_{0,3}$ |
| $s_{1,0}$ | $s_{1,1}$ | $s_{1,2}$ | $s_{1,3}$ |
| $s_{2,0}$ | $s_{2,1}$ | $s_{2,2}$ | $s_{2,3}$ |
| $s_{3,0}$ | $s_{3,1}$ | $s_{3,2}$ | $s_{3,3}$ |

|  |  |  |  |
|---|---|---|---|
| $s_{0,0}$ | $s_{0,1}$ | $s_{0,2}$ | $s_{0,3}$ |
| $s_{1,1}$ | $s_{1,2}$ | $s_{1,3}$ | $s_{1,0}$ |
| $s_{2,2}$ | $s_{2,3}$ | $s_{2,0}$ | $s_{2,1}$ |
| $s_{3,3}$ | $s_{3,0}$ | $s_{3,1}$ | $s_{3,2}$ |

**Shift Rows Transformation**

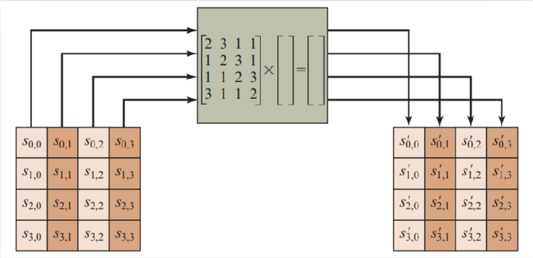
- The first row of State is not altered.
- For the second row, a 1-byte circular left shift is performed.
- For the third row, a 2-byte circular left shift is performed.
- For the fourth row, a 3-byte circular left shift is performed.

# Block Ciphers » AES: Advanced Encryption Standard

AES Structure

**Transformation Functions**

Key Expansion



2 3 1 1
1 2 3 1
1 1 2 3
3 1 1 2
×
=
$s_{0,0}$ $s_{0,1}$ $s_{0,2}$ $s_{0,3}$
$s_{1,0}$ $s_{1,1}$ $s_{1,2}$ $s_{1,3}$
$s_{2,0}$ $s_{2,1}$ $s_{2,2}$ $s_{2,3}$
$s_{3,0}$ $s_{3,1}$ $s_{3,2}$ $s_{3,3}$
$s'_{0,0}$ $s'_{0,1}$ $s'_{0,2}$ $s'_{0,3}$
$s'_{1,0}$ $s'_{1,1}$ $s'_{1,2}$ $s'_{1,3}$
$s'_{2,0}$ $s'_{2,1}$ $s'_{2,2}$ $s'_{2,3}$
$s'_{3,0}$ $s'_{3,1}$ $s'_{3,2}$ $s'_{3,3}$

**Mix Columns Transformation**

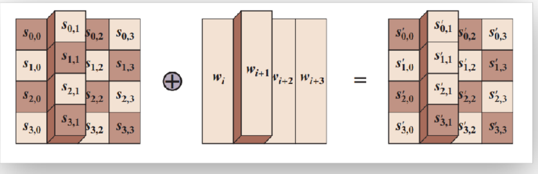
- Each byte of a column is mapped into a new value that is a function of all four bytes in that column.
- The transformation can be defined by the following matrix multiplication on State.
- Each element in the product matrix is the sum of products of elements of one row and one column.
- In this case, the individual additions and multiplications are performed in $GF(2^8)$.

# Block Ciphers » AES: Advanced Encryption Standard

AES Structure

**Transformation Functions**

Key Expansion


$s_{0,0}$ $s_{0,1}$ $s_{0,2}$ $s_{0,3}$
$s_{1,0}$ $s_{1,1}$ $s_{1,2}$ $s_{1,3}$
$s_{2,0}$ $s_{2,1}$ $s_{2,2}$ $s_{2,3}$
$s_{3,0}$ $s_{3,1}$ $s_{3,2}$ $s_{3,3}$
⊕
$w_i$ $w_{i+1}$ $w_{i+2}$ $w_{i+3}$
=
$s'_{0,0}$ $s'_{0,1}$ $s'_{0,2}$ $s'_{0,3}$
$s'_{1,0}$ $s'_{1,1}$ $s'_{1,2}$ $s'_{1,3}$
$s'_{2,0}$ $s'_{2,1}$ $s'_{2,2}$ $s'_{2,3}$
$s'_{3,0}$ $s'_{3,1}$ $s'_{3,2}$ $s'_{3,3}$

**Add Round Key Transformation**

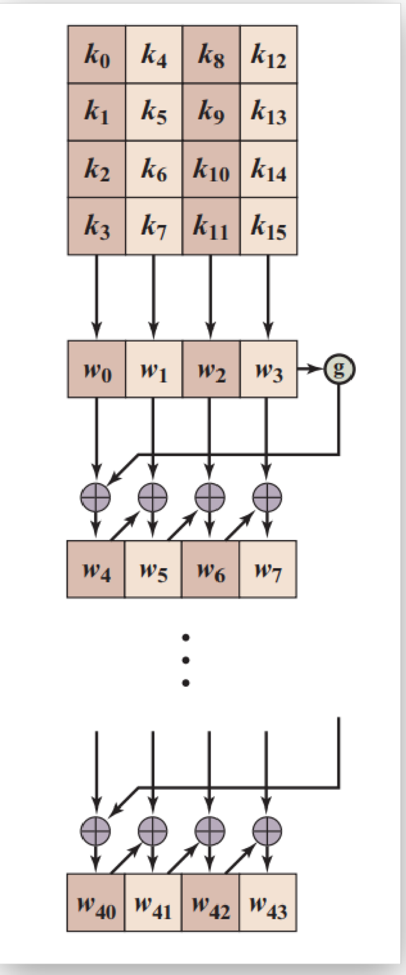
- The 128 bits of State are bitwise XORed with the 128 bits of the round key.
- The operation is viewed as a columnwise operation between the 4 bytes of a State column and one word of the round key.

# Block Ciphers » AES: Advanced Encryption Standard

AES Structure

Transformation Functions

**Key Expansion**


$k_0$ $k_4$ $k_8$ $k_{12}$
$k_1$ $k_5$ $k_9$ $k_{13}$
$k_2$ $k_6$ $k_{10}$ $k_{14}$
$k_3$ $k_7$ $k_{11}$ $k_{15}$
$w_0$ $w_1$ $w_2$ $w_3$
g
$w_4$ $w_5$ $w_6$ $w_7$
$w_{40}$ $w_{41}$ $w_{42}$ $w_{43}$

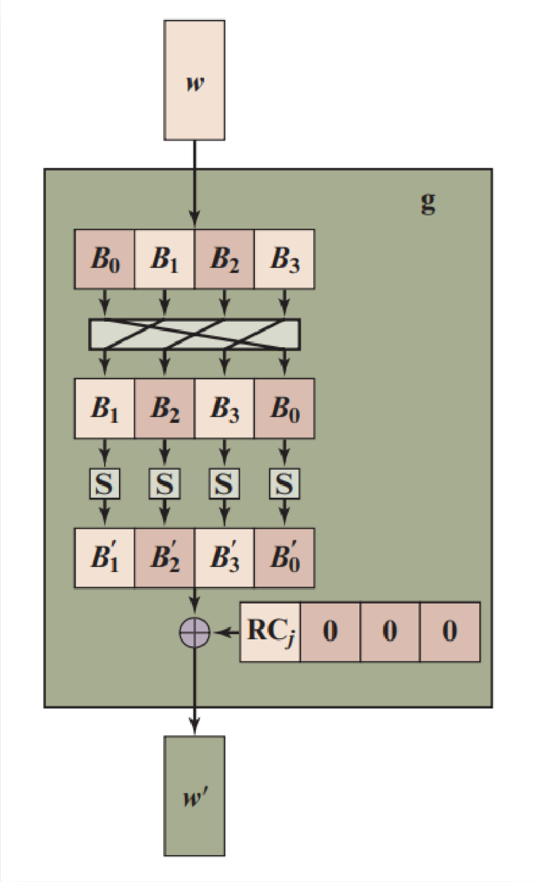
- The key is copied into the first four words of the expanded key.
- The remainder of the expanded key is filled in four words at a time. Each added word $w_i$ depends on the immediately preceding word, $w_{i-1}$, and the word four positions back, $w_{i-4}$.
- In three out of four cases, a simple XOR is used. For a word whose position in the w array is a multiple of 4, a more complex function $g$ is used.

# Block Ciphers » AES: Advanced Encryption Standard

AES Structure

Transformation Functions

**Key Expansion**



**Function g**

- **RotWord** performs a one-byte circular left shift on a word.
- **SubWord** performs a byte substitution on each byte of its input word, using the S-box.
- The result of steps 1 and 2 is XORed with a **round constant**, $Rcon_j$. The round constant is a word in which the three rightmost bytes are always 0.

## Stream Ciphers » Pseudorandom Number Generation

**Linear Congruential Generator**

Blum Blum Shub Generator

- The sequence of random numbers $X_n$ is obtained via the following iterative equation: $X_{n+1} = (aX_n + c) \mod m$. If $m$, $a$, $c$, and $X_0$ are integers, then this technique will produce a sequence of integers with each integer in the range $0 \leq Xn \leq m$.
- The selection of values for $a$, $c$, and $m$ is critical in developing a good random number generator. It is unsatisfactory if the **peroid** of a Linear Congruential Generator are short.

# Stream Ciphers » Pseudorandom Number Generation

Linear Congruential Generator

**Blum Blum Shub Generator**

- First, choose two large prime numbers, p and q, that both have a remainder of 3 when divided by 4. That is, $p = q = 3 \mod 4$. Let $n = p * q$.
- Next, choose a random number $s$, such that $s$ is relatively prime to $n$; this is equivalent to saying that neither $p$ nor $q$ is a factor of $s$.
- Then the BBS generator produces a sequence of bits $B_i$ according to the following algorithm: $X_0 = s^2 \mod n$, $X_i = (X_{i-1})^2 \mod n$, and $B_i = X_i \mod 2$. Thus, the least significant bit is taken at each iteration.

# Stream Ciphers » RC4: Rivest Cipher

**RC4 Overview**

S-box Initialization

Encryption & Decryption

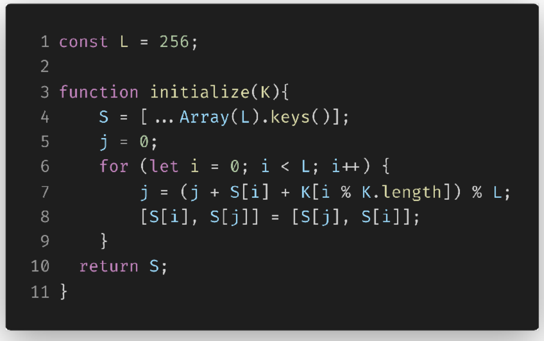
- A variable-length key of from 1 to 256 bytes (8 to 2048 bits) is used to initialize a 256-byte state vector $S$, with elements $S[0], S[1], \ldots, S[255]$. At all times, $S$ contains a permutation of all 8-bit numbers from 0 through 255.
- For encryption and decryption, a byte $k$ is generated from $S$ by selecting one of the 255 entries in a systematic fashion. As each value of $k$ is generated, the entries in $S$ are once again permuted.

# Stream Ciphers » RC4: Rivest Cipher

RC4 Overview

**S-box Initialization**

Encryption & Decryption



```javascript
const L = 256;

function initialize(K){
    S = [...Array(L).keys()];
    j = 0;
    for (let i = 0; i < L; i++) {
        j = (j + S[i] + K[i % K.length]) % L;
        [S[i], S[j]] = [S[j], S[i]];
    }
  return S;
}
```

**in Javascript**

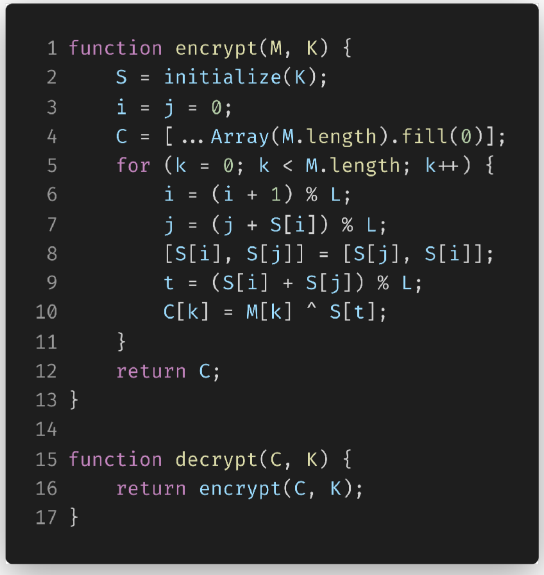
- To begin, the entries of S are set equal to the values from 0 through 255 in ascending order; that is, $S[0] = 0$, $S[1] = 1$, ... , $S[255] = 255$.
- Next we use the key $K$ to produce the initial permutation of $S$. This involves starting with $S[0]$ and going through to $S[255]$, and for each $S[i]$, swapping $S[i]$ with another byte in $S$ according to a scheme dictated by $K[i \mod 256]$.

# Stream Ciphers » RC4: Rivest Cipher

RC4 Overview

S-box Initialization

**Encryption & Decryption**



```javascript
function encrypt(M, K) {
    S = initialize(K);
    i = j = 0;
    C = [...Array(M.length).fill(0)];
    for (k = 0; k < M.length; k++) {
        i = (i + 1) % L;
        j = (j + S[i]) % L;
        [S[i], S[j]] = [S[j], S[i]];
        t = (S[i] + S[j]) % L;
        C[k] = M[k] ^ S[t];
    }
    return C;
}

function decrypt(C, K) {
    return encrypt(C, K);
}
```

**in Javascript**

- Stream generation involves cycling through all the elements of $S[i]$, and for each $S[i]$, swapping $S[i]$ with another byte in $S$ according to a scheme dictated by the current configuration of $S$. After $S[255]$ is reached, the process continues, starting over again at $S[0]$.
- To encrypt, XOR the value $k$ with the next byte of plaintext. To decrypt, XOR the value $k$ with the next byte of ciphertext.