Block Cipher Principles

Presentation by:
V. Balasubramanian
SSN College of Engineering

Agenda

- Block cipher Principles of DES
- Strength of DES
- Differential and linear cryptanalysis
- Block cipher design principles

Feistel Cipher Design Features

- Block size
  - Larger block sizes mean greater security but reduced encryption/decryption speed for a given algorithm
- Key size
  - Larger key size means greater security but may decrease encryption/decryption speeds
- Number of rounds
  - The essence of the Feistel 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 F
  - Greater complexity generally means greater resistance to cryptanalysis
- Fast software encryption/decryption
  - In many cases, encrypting is embedded in applications or utility functions in such a way as to preclude a hardware implementation; accordingly, the speed 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

Avalanche Effect

- A desirable property of any encryption algorithm is that a small change in either the plaintext or the key should produce a significant change in the ciphertext.
- In particular, a change in one bit of the plaintext or one bit of the key should produce a change in many bits of the ciphertext. This is referred to as the avalanche effect.

DES Facts

- Data Encryption Standard (DES) encrypts **blocks of size 64 bit**.
- Developed by **IBM** based on the cipher *Lucifer* under influence of the *National Security Agency* (NSA), the design criteria for DES have not been published
- **Standardized 1977** by the **National Bureau of Standards** (NBS)
  today called *National Institute of Standards and Technology* (NIST)
- Most popular **block cipher** for most of the last 30 years.
- By far best studied symmetric algorithm.
- Nowadays considered insecure due to the small **key length of 56 bit.**
- **But: 3DES yields very secure cipher**, still widely used today.
- Replaced by the *Advanced Encryption Standard* (**AES**) in 2000

DES

1. Expansion E main purpose: increases diffusion

- In **Feistel ciphers** only the keyschedule has to be modified for decryption.
- Generate the same 16 round keys in reverse order.

Security of DES

- **After proposal of DES two major criticisms arose:**
  1. Key space is too small (2^{56} keys)
  2. S-box design criteria have been kept secret: Are there any hidden analytical attacks (*backdoors*), only known to the NSA?

- **Analytical Attacks:** DES is highly resistent to both *differential* and *linear cryptanalysis*, which have been published years later than the DES. This means IBM and NSA had been aware of these attacks for 15 years!
  So far there is no known analytical attack which breaks DES in realistic scenarios.

- **Exhaustive key search:** For a given pair of plaintext-ciphertext (x, y) test all 2^{56} keys until the condition $DES_k^{-1}(x)=y$ is fulfilled.
  $\Rightarrow$ Relatively easy given today's computer technology!

History of Attacks on DES

Year	Proposed/ implemented DES Attack
1977	Diffie & Hellman, (under-)estimate the costs of a key search machine
1990	Biham & Shamir propose differential cryptanalysis (2^{47} chosen ciphertexts)
1993	Mike Wiener proposes design of a very efficient key search machine: Average search requires 36h. Costs: $1.000.000
1993	Matsui proposes linear cryptanalysis (2^{43} chosen ciphertexts)
Jun. 1997	DES Challenge I broken, 4.5 months of distributed search
Feb. 1998	DES Challenge II--1 broken, 39 days (distributed search)
Jul. 1998	DES Challenge II--2 broken, key search machine *Deep Crack* built by the Electronic Frontier Foundation (EFF): 1800 ASICs with 24 search engines each, Costs: $250 000, 15 days average search time (required 56h for the Challenge)
Jan. 1999	DES Challenge III broken in 22h 15min (distributed search assisted by *Deep Crack*)
2006-2008	Reconfigurable key search machine *COPACOBANA* developed at the Universities in Bochum and Kiel (Germany), uses 120 FPGAs to break DES in 6.4 days (avg.) at a cost of $10 000.

Alternatives to DES

Algorithm	I/O Bit	key lengths	remarks
AES / Rijndael	128	128/192/256	DES ”replacement”, worldwide used standard
Triple DES	64	112 (effective)	conservative choice
Mars	128	128/192/256	AES finalist
RC6	128	128/192/256	AES finalist
Serpent	128	128/192/256	AES finalist
Twofish	128	128/192/256	AES finalist
IDEA	64	128	(Patented till 2011)

DES

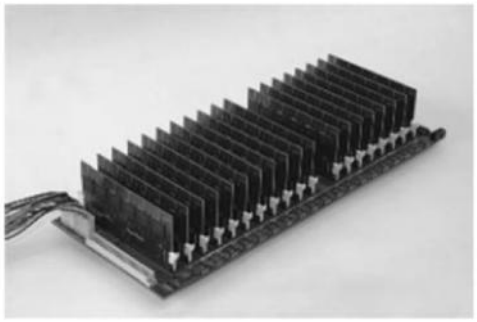
- DES was the dominant symmetric encryption algorithm from the mid-1970s to the mid-1990s. Since 56-bit keys are no longer secure, the Advanced Encryption Standard (AES) was created.
- Standard DES with 56-bit key length can be broken relatively easily nowadays through an exhaustive key search.
- DES is quite robust against known analytical attacks: In practice it is very difficult to break the cipher with differential or linear cryptanalysis.
- By encrypting with DES three times in a row, triple DES (3DES) is created, against which no practical attack is currently known.
- The “default” symmetric cipher is nowadays often AES. In addition, the other four AES finalist ciphers all seem very secure and efficient.

Exhaustive Search

Definition 3.5.1 DES Exhaustive key search
Input: *at least one pair of plaintext–ciphertext* (x, y)
Output: k, *such that* $y = DES_k(x)$
Attack: *Test all* 2^{56} *possible keys until the following condition is fulfilled:*

$$DES_{k_i}^{-1}(y) \stackrel{?}{=} x \ , \ \ i = 0, 1, \ldots, 2^{56} - 1.$$

Deep Crack



COPACOBANA — A cost-optimized parallel code breaker

Fig. 3.17 Deep Crack — the hardware exhaustive key-search machine that broke DES in 1998 (reproduced with permission from Paul Kocher)

Key Size

- A key size of 56 bits is too short to encrypt confidential data nowadays.
- Hence, single DES should only be used for applications where only short-term security is needed —say, a few hours —or where the value of the encrypted data is very low.
- However, variants of DES, in particular 3DES, are still secure.

DES Weakness

Weaknesses in Cipher Design

We will briefly mention some weaknesses that have been found in the design of the cipher.

S-boxes At least three weaknesses are mentioned in the literature for S-boxes.

1. In S-box 4, the last three output bits can be derived in the same way as the first output bit by complementing some of the input bits.
2. Two specifically chosen inputs to an S-box array can create the same output.
3. It is possible to obtain the same output in a single round by changing bits in only three neighboring S-boxes.

	0	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*	*13*	*14*	*15*
0	07	13	14	03	00	6	09	10	1	02	08	05	11	12	04	15
1	13	08	11	05	06	15	00	03	04	07	02	12	01	10	14	09
2	10	06	09	00	12	11	07	13	15	01	03	14	05	02	08	04
3	03	15	00	06	10	01	13	08	09	04	05	11	12	07	02	14

Properties

- Two desired properties of a block cipher are the
- Avalanche effect [1 bit diff plaintext gives 29]
- Completeness.

Avalanche Effect

Avalanche effect means a small change in the plaintext (or key) should create a significant change in the ciphertext. DES has been proved to be strong with regard to this property.

Example 6.7

To check the avalanche effect in DES, let us encrypt two plaintext blocks (with the same key) that differ only in one bit and observe the differences in the number of bits in each round.

Plaintext: 0000000000000000 Key: 22234512987ABB23
Ciphertext: 4789FD476E82A5F1

Plaintext: 0000000000000001 Key: 22234512987ABB23
Ciphertext: 0A4ED5C15A63FEA3

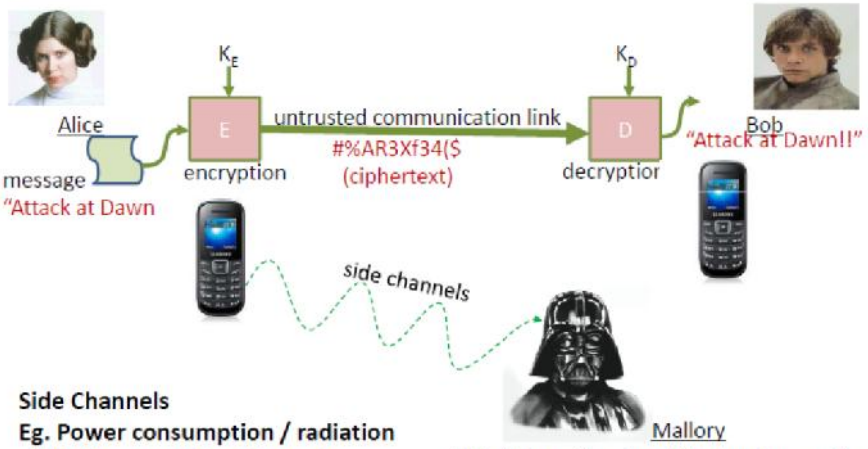
Completeness

Completeness effect means that each bit of the ciphertext needs to depend on many bits on the plaintext.
The diffusion and confusion produced by P-boxes and S-boxes in DES, show a very strong completeness effect.

Timing Attacks

- However, the issue may also be relevant for symmetric ciphers. In essence, a timing attack is one in which information about the key or the plaintext is obtained by observing how long it takes a given implementation to perform decryptions on various ciphertexts. A timing attack exploits the fact that an encryption or decryption algorithm often takes slightly different amounts of time on different inputs.

Side Channel Attack



Side Channels
Eg. Power consumption / radiation of device, execution time, etc.

Gets information about the keys by monitoring Side channels of the device

Timing Attack

What can you tell from the execution time of this function?

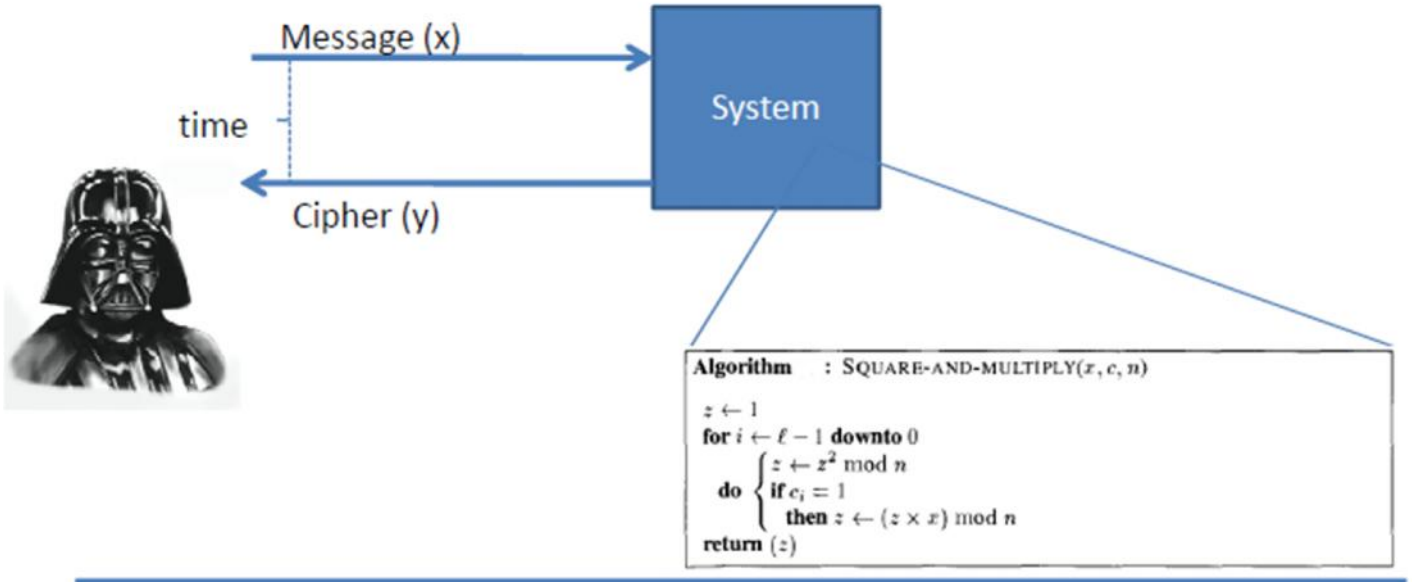
```
unsigned int Divide(unsigned int a, unsigned int b){
  if (b==0)
    return ERROR;
  else
    return a/b;
}
```

Finding N/D

```
while N ≥ D do
  N := N - D
end
return N
```

- Execution time depends on values of a and b
  - Fastest when b=0
  - Varies depending a / b
- Thus information can be inferred from execution time.
  - Can we get secret information from the timing?

Timing attack



Timing Attack

Naïve Canonical Verification Algorithm

Input: m,t'

```
t=Mac_K(m)
for i=1 to tag-length
  if t[i] != t'[i] then
    return 0
return 1
```

Example

t= 1 0 1 0 1 1 1 0

t'= 1 0 1 0 1 0 1 1

Returns 0 after 8 steps

Improved verification

Input: m,t'

B=1
t=$Mac_K(m)$
for i=1 to tag-length
  if t[i] != t'[i] **then**
    B=0
  else (dummy op)
return B

Example

t= 1 0 1 0 1 1 1 0
t'= 0 0 1 0 1 0 1 0

Returns 0 after 8 steps

DES Timing Attacks

- One in which information about the key or the plaintext is obtained by observing how long it takes a given implementation to perform decryptions on various ciphertexts
- Exploits the fact that an encryption or decryption algorithm often takes slightly different amounts of time on different inputs
- So far it appears unlikely that this technique will ever be successful against DES or more powerful symmetric ciphers such as triple DES and AES

Design principles

Block Cipher Design Principles: Number of Rounds

The greater the number of rounds, the more difficult it is to perform cryptanalysis

In general, the criterion should be that the number of rounds is chosen so that known cryptanalytic efforts require greater effort than a simple brute-force key search attack

If DES had 15 or fewer rounds, differential cryptanalysis would require less effort than a brute-force key search

Number of Rounds

Number of Rounds

DES uses sixteen rounds of Feistel ciphers. It has been proved that after eight rounds, each ciphertext is a function of every plaintext bit and every key bit; the ciphertext is thoroughly a random function of plaintext and ciphertext. Therefore, it looks like eight rounds should be enough. However, experiments have found that DES versions with less than sixteen rounds are even more vulnerable to known-plaintext attacks than brute-force attack, which justifies the use of sixteen rounds by the designers of DES.

Fiestel Function

- Strict avalanche criterion
- Bit independence criterion
- The SAC and BIC criteria appear to
- strengthen the effectiveness of the confusion function

Key Schedule

- we would like to select subkeys to maximize the difficulty of deducing individual subkeys and the difficulty of working back to the main key.

Weak Keys

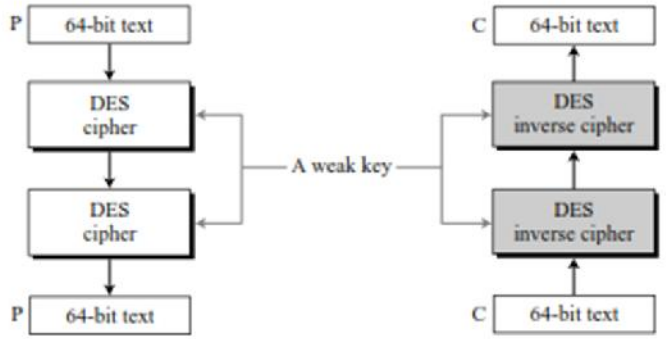
Weak Keys Four out of 2^{56} possible keys are called **weak keys.** A weak key is the one that, after parity drop operation (using Table 6.12), consists either of all 0s, all 1s, or half 0s and half 1s. These keys are shown in Table 6.18.

Table 6.18 *Weak keys*

Keys before parities drop (64 bits)	*Actual key (56 bits)*
0101 0101 0101 0101	0000000 0000000
1F1F 1F1F 0E0E 0E0E	0000000 FFFFFFF
E0E0 E0E0 F1F1 F1F1	FFFFFFF 0000000
FEFE FEFE FEFE FEFE	FFFFFFF FFFFFFF

Weak Keys

Figure 6.11 *Double encryption and decryption with a weak key*



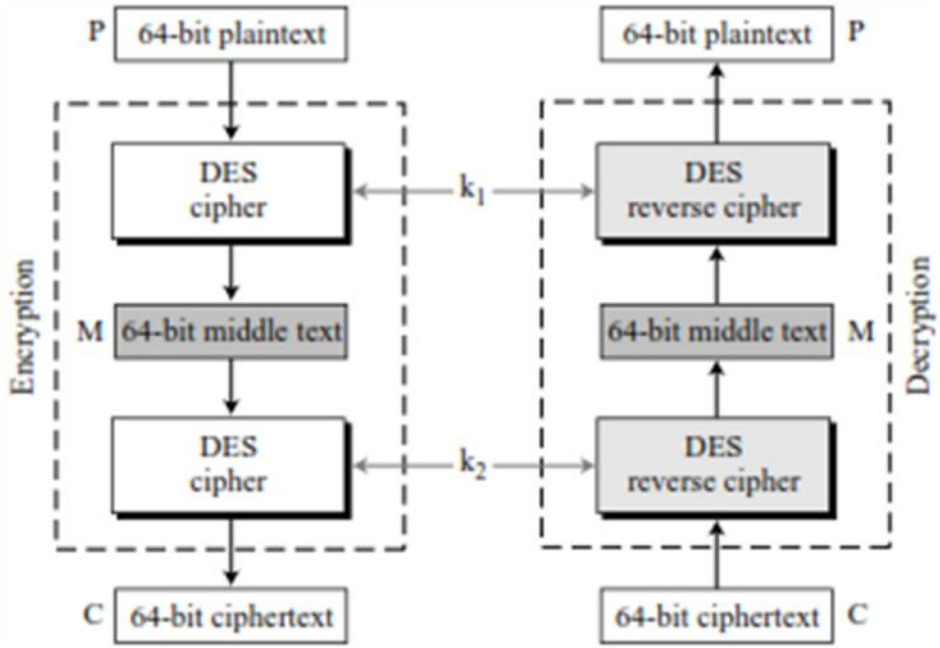
$$E_k(E_k(P)) = P$$

Key: 0x0101010101010101
Plaintext: *0x1234567887654321* Ciphertext: 0x814FE938589154F7

Key: 0x0101010101010101
Plaintext: 0x814FE938589154F7 Ciphertext: *0x1234567887654321*

Double DES

Figure 6.14 *Meet-in-the-middle attack for double DES*



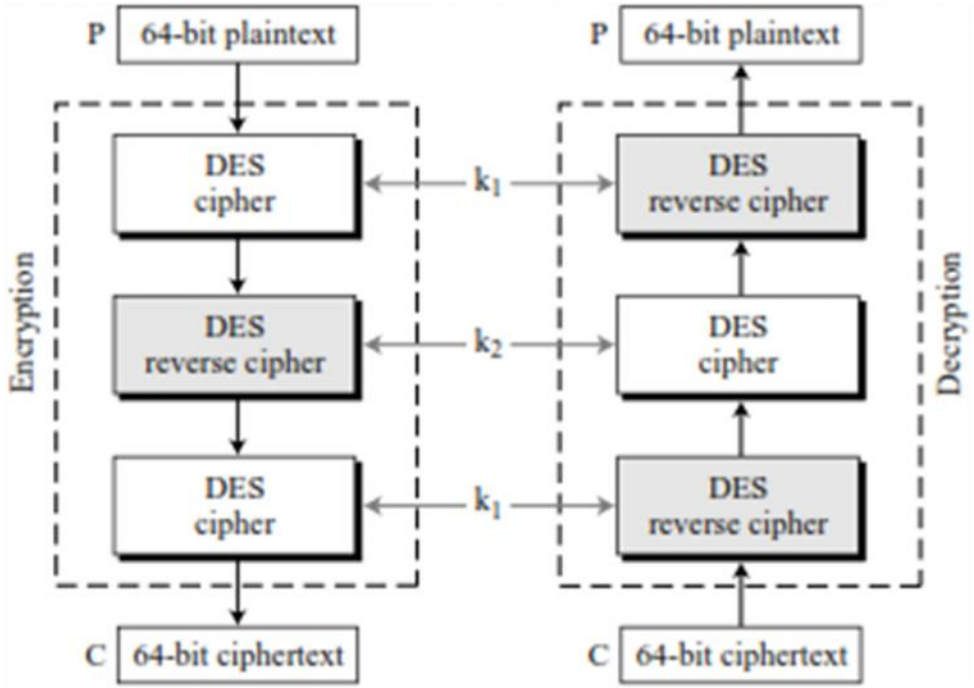
Meet in the Middle attack

$$M = E_{k_1}(P) \qquad \text{and} \qquad M = D_{k_2}(C)$$

Assume that Eve has intercepted a previous pair P and C (known-plaintext attack). Based on the first relationship mentioned above, Eve encrypts P using all possible values (2^{56}) of k_1 and records all values obtained for M. Based on the second relationship mentioned above, Eve decrypts C using all possible values (2^{56}) of k_2 and records all values obtained for M. Eve creates two tables sorted by M values. She then compares the values for M until she finds those pairs of k_1 and k_2 for which the value of M is the

Triple DES

Figure 6.16 *Triple DES with two keys*



Analytical Attacks

- Differential Cryptanalysis
  - Chosen Plain Text
- Linear Cryptanalysis
  - Known Plaintext

Differential Cryptanalysis

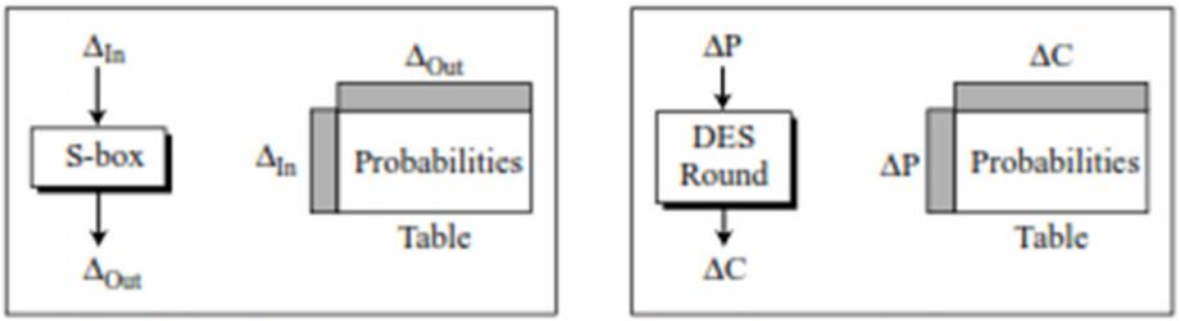
Differential cryptanalysis for DES was invented by Biham and Shamir. In this cryptanalysis, the intruder concentrates on *chosen-plaintext* attacks. The analysis uses the propagation of input differences through the cipher. The term *difference* here is used to refer to the exclusive-or of two different inputs (plaintexts). In other words, the intruder analyzes how $P \oplus P'$ is propagated through rounds.

Differential Cryptanalysis

Probabilistic Relations

The idea of differential cryptanalysis is based on the probabilistic relations between input differences and output differences. Two relations are of particular interest in the analysis: *differential profiles* and *round characteristics*, as shown in Figure N.1.

Figure N.1 *Differential profile and round characteristic for DES*

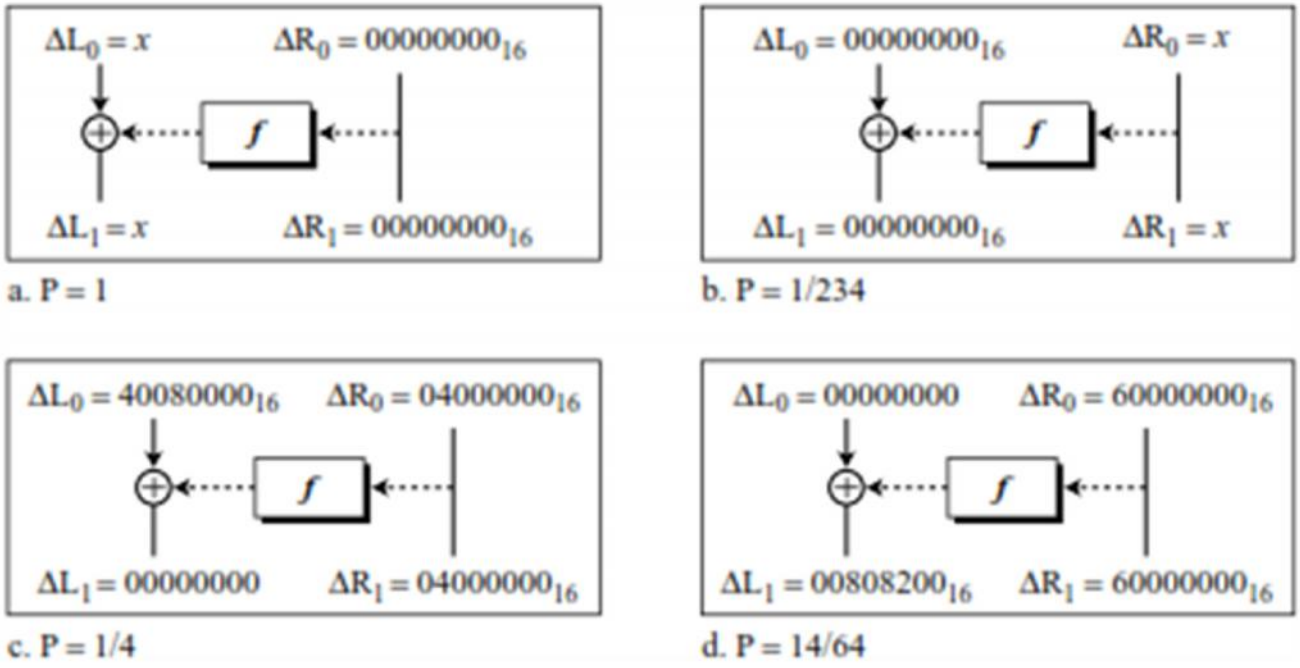


a. Differential Profile

b. Round Characteristic

Differential Cryptanalysis

Figure N.2 *Some round characteristics for differential cryptanalysis*

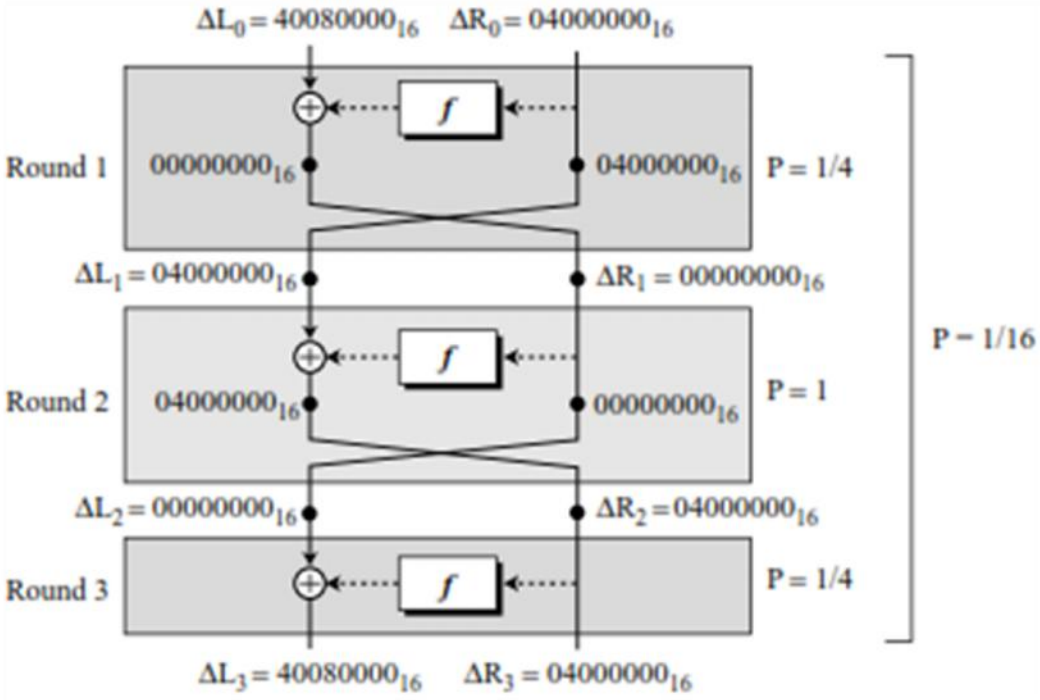


DC

Although we can have many characteristics for a round, Figure N.2 shows only four of them. In each characteristic, we have divided the input differences and the output differences into the left and right sections. Each left or right difference is made of 32 bits or eight hexadecimal digits. All of these characteristics can be proved using a program that finds the input/output relation in a round of DES. Figure N.2a shows that the input difference of (x, 00000000_{16}) produces the output difference of (x, 00000000_{16}) with probability 1. Figure N.2b shows the same characteristic as Figure N.2a except that the left and right inputs and outputs are swapped; the probability will change tremendously. Figure N.2c shows that input difference of (40080000_{16}, 04000000_{16}) produces the output difference (00000000_{16}, 04000000_{16}) with probability 1/4. Finally, Figure N.2d shows that the input difference (00000000_{16}, 60000000_{16}) produces the output difference (00808200_{16}, 6000000_{16}) with probability 14/64.

DC

Figure N.3 *A three-round characteristic for differential cryptanalysis*



Attacks

Attack

For the sake of example, let us assume that Eve uses the characteristic of Figure N.4 to attack a sixteen-round DES. Eve somehow lures Alice to encrypt a lot of plaintexts in the form $(x, 0)$, in which the left half is x (different values) and the right half is 0. Eve then keeps all ciphertexts received from Alice in the form $(0, x)$. Note that 0 here means 00000000_{16}.

Cipher Key finding

Finding the Cipher Key

The ultimate goal of the intruder in differential cryptanalysis is to find the cipher key. This can be done by finding the round keys from the bottom to the top (K_{16} to K_1).

Finding the Last Round Key

If the intruder has enough plaintext/ciphertext pairs (each with different values of x), she can use the relationship in the last round, $0 = f(K_{16}, x)$, to find some of the bits in K_{16}. This can be done by finding the most probable values that make this relation more likely.

Finding Other Round Keys

The keys for other rounds can be found using other characteristics or using brute-force attacks.

Security of DES

Security

It turned out that 2^{47} chosen plaintext/ciphertext pairs are needed to attack a 16-round DES. Finding such a huge number of chosen pairs is extremely difficult in real-life situations. This means that DES is not vulnerable to this type of attack.

DES Strength

- It is a major triumph for the designers of DES that no weakness was found until 1990.
- In 1990, Eli Biham and Adi Shamir discovered what is called differential cryptanalysis (DC). This is a powerful attack which is in principle applicable to any block cipher.
- It turned out that the DES S-boxes are particularly resistant against this attack. In fact, one member of the original IBM design team declared after the discovery of DC that they had been aware of the attack at the time of design.

DES

- The reason why the S-box design criteria were not made public was that the design team did not want to make such a powerful attack public.
- If this claim is true — and all circumstances support it — it means that the IBM and NSA team was 15 years ahead of the research community.

Linear cryptanalysis

- In 1993 a related but distinct analytical attack was published by Mitsuru Matsui,which was named linear cryptanalysis (LC).
- Similar to differential cryptanalysis, the effectiveness of this attack also heavily depends on the structure of the S-boxes.

Linear Cryptanalysis

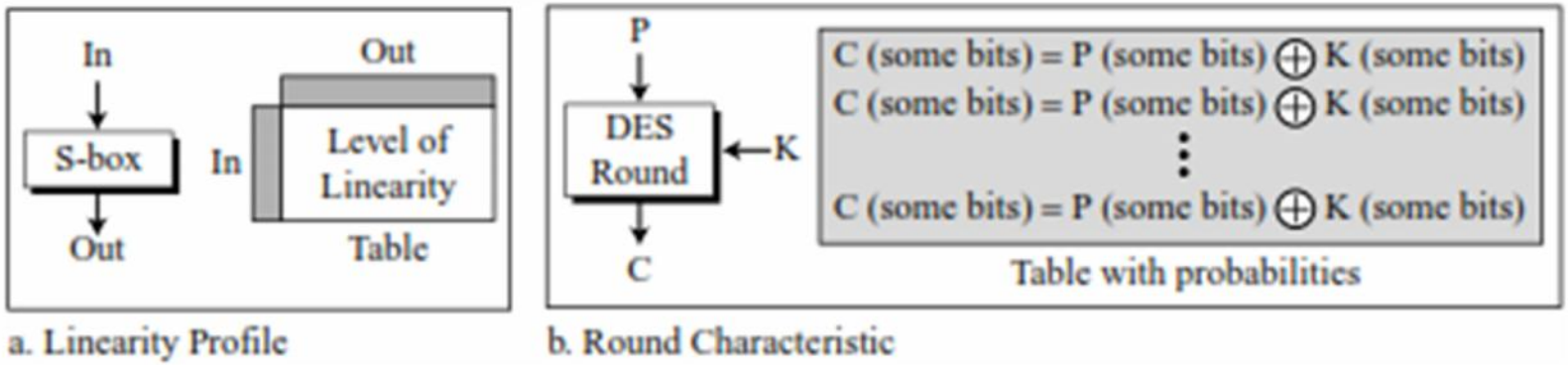
- Linear cryptanalysis for DES was developed by Matsui. It is a Known Plaintext attack.

Attacks

What is the practical relevance of these two analytical attacks against DES? It turns out that an attacker needs 2^{47} plaintext–ciphertext pairs for a successful DC attack. This assumes particularly chosen plaintext blocks; for random plaintext 2^{55} pairs are needed! In the case of LC, an attacker needs 2^{43} plaintext–ciphertext pairs. All these numbers seem highly impractical for several reasons. First, an attacker needs to know an extremely large number of plaintexts, i.e., pieces of data which are supposedly encrypted and thus hidden from the attacker. Second, collecting and storing such an amount of data takes a long time and requires considerable memory resources. Third, the attack only recovers one key. (This is actually one of many arguments for introducing key freshness in cryptographic applications.) As a result of all these arguments, it does not seem likely that DES can be broken with either DC or LC in real-world systems. However, both DC and LC are very powerful attacks which are applicable to many other block ciphers. Table 3.13 provides an

LC

Figure N.5 *Linear profile and round characteristic for DES*



a. Linearity Profile

b. Round Characteristic

LC attack

Attack

After finding and storing many relationship between some plaintext bits, ciphertext bits, and round-key bits. Eve can access some plaintext/ciphertext pairs (known-plaintext attack) and use the corresponding bits in the stored characteristics to find bits in the round keys.

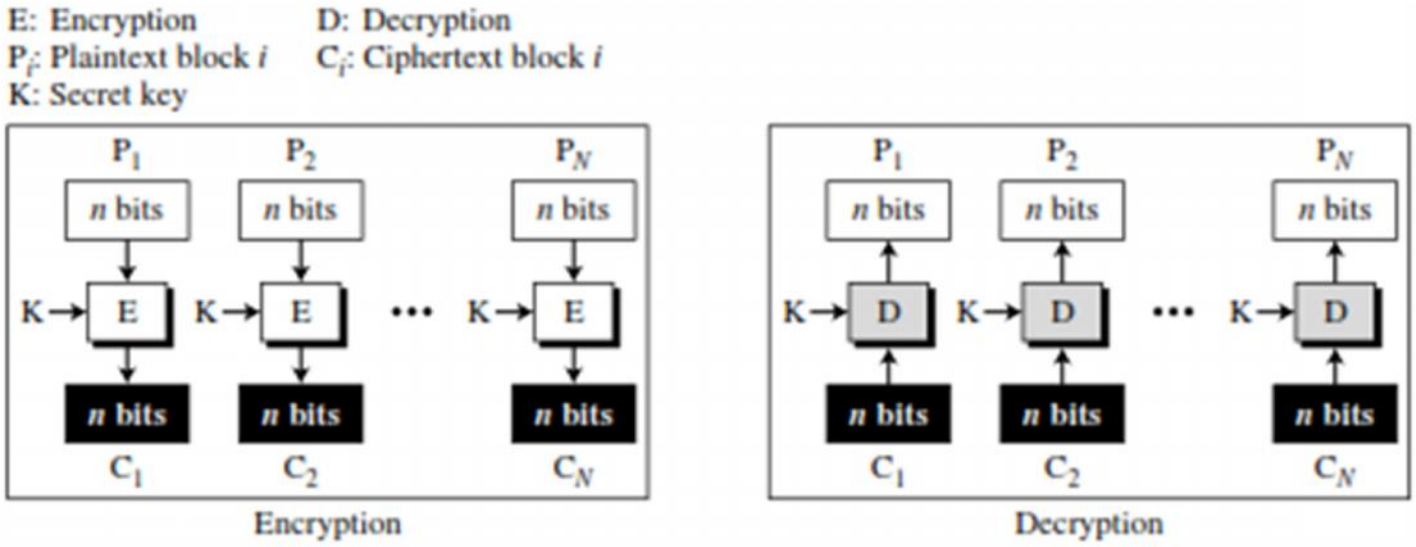
Security

It turned out that 2^{43} known plaintext/ciphertext pairs are needed to attack a 16-round DES. Linear cryptanalysis looks more probable than differential cryptanalysis for two reasons. First, the number of steps is smaller. Second it is easier to launch a known plaintext attack than a chosen-plaintext attack. However, the attack is still far from being a serious treat to DES.

$$C(\text{some bits}) = P(\text{some bits}) \oplus K_1(\text{some bits}) \oplus \cdots \oplus K_{16}(\text{some bits})$$

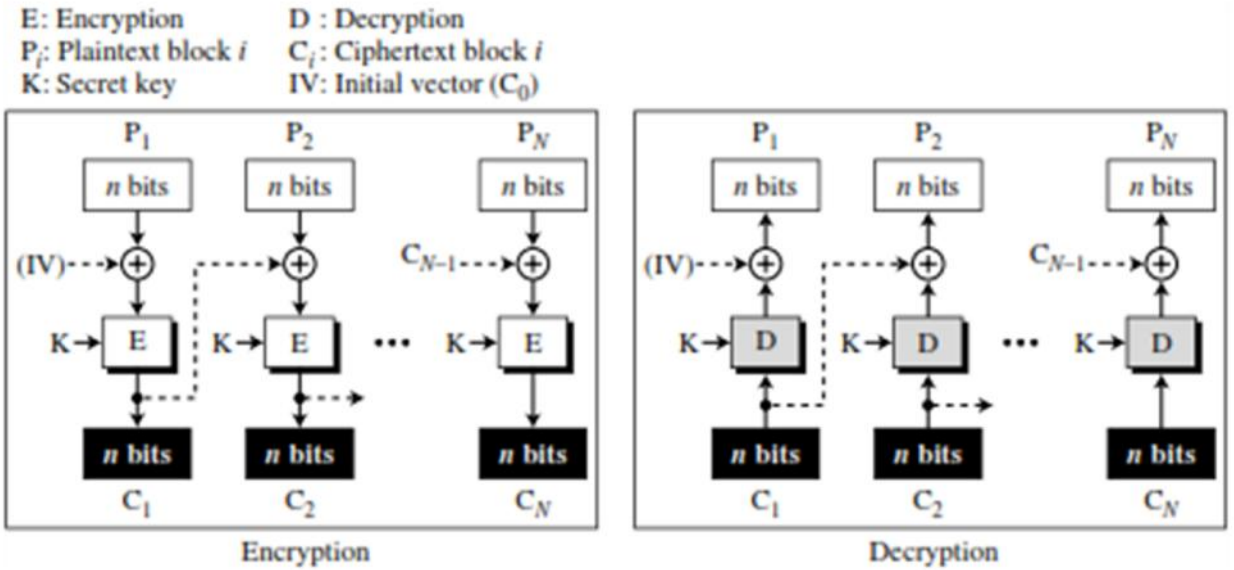
ECB

Figure 8.2 *Electronic codebook (ECB) mode*



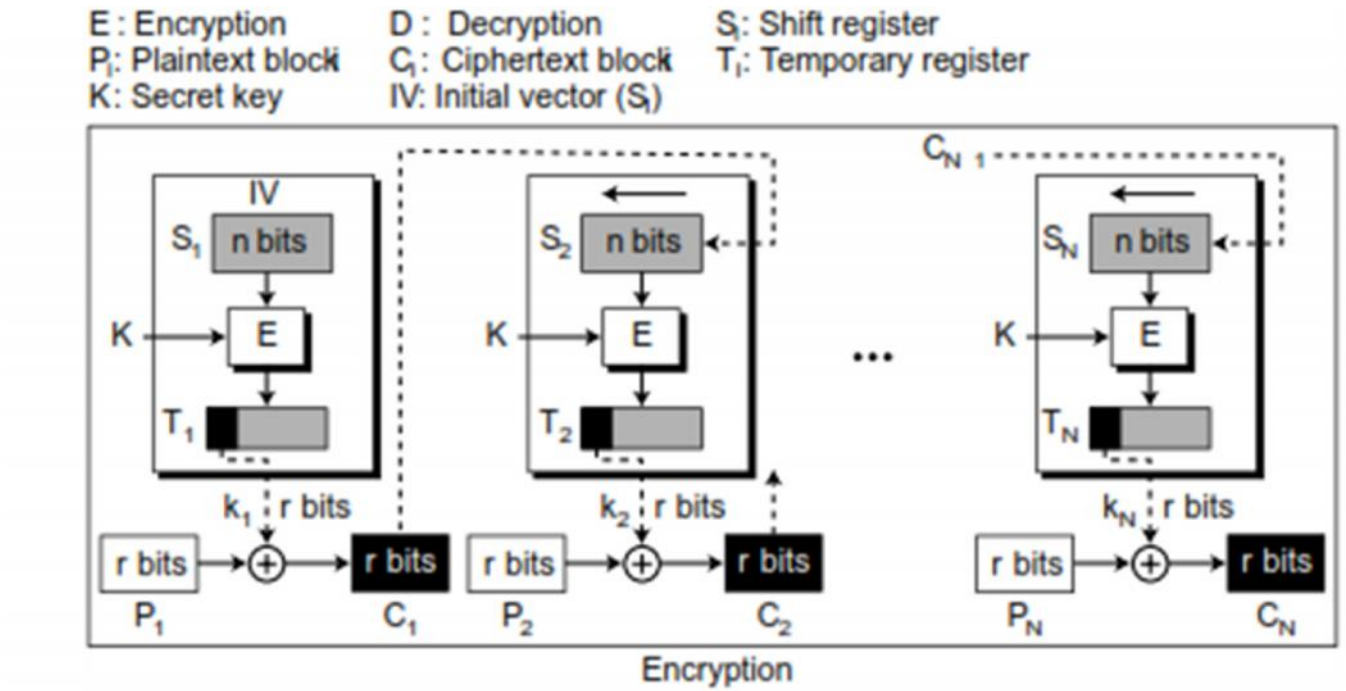
CBC

Figure 8.3 *Cipher block chaining (CBC) mode*



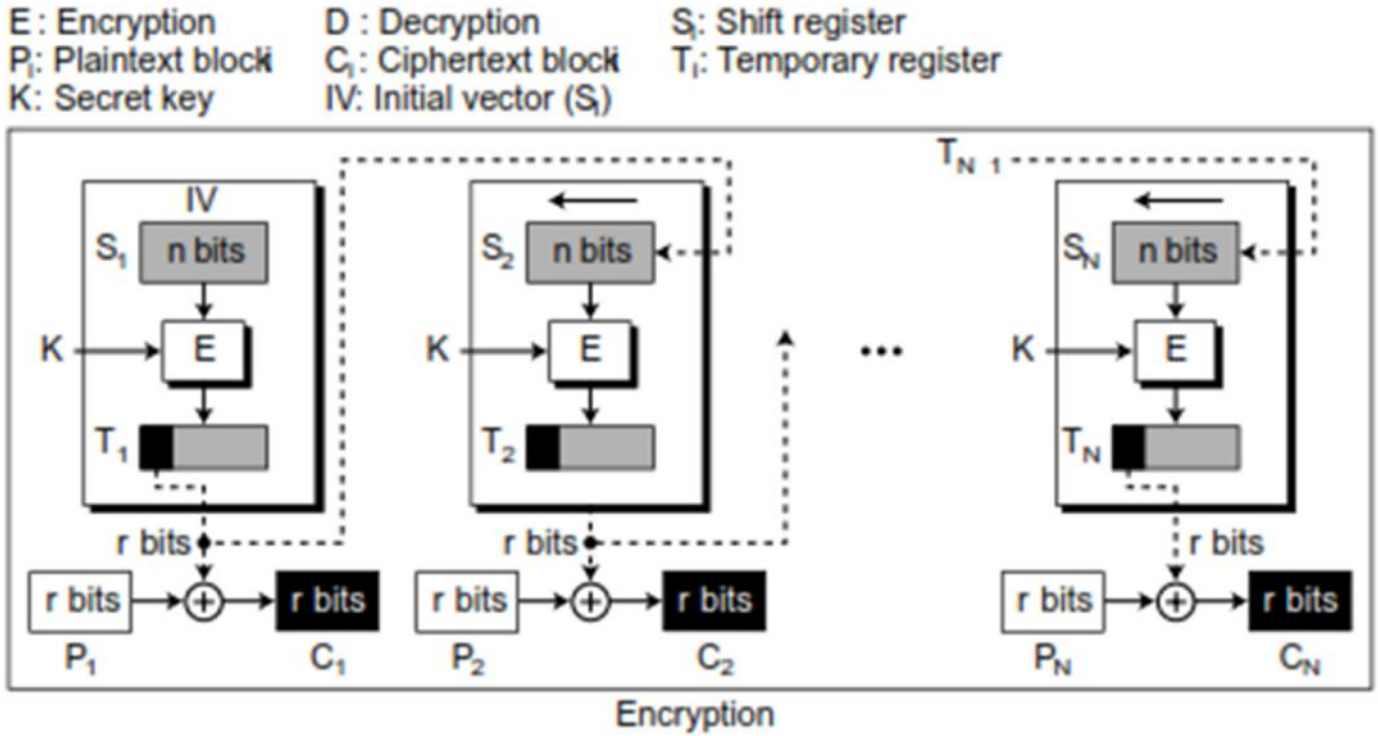
CFB

Figure 8.4 Encryption in cipher feedback (CFB) mode



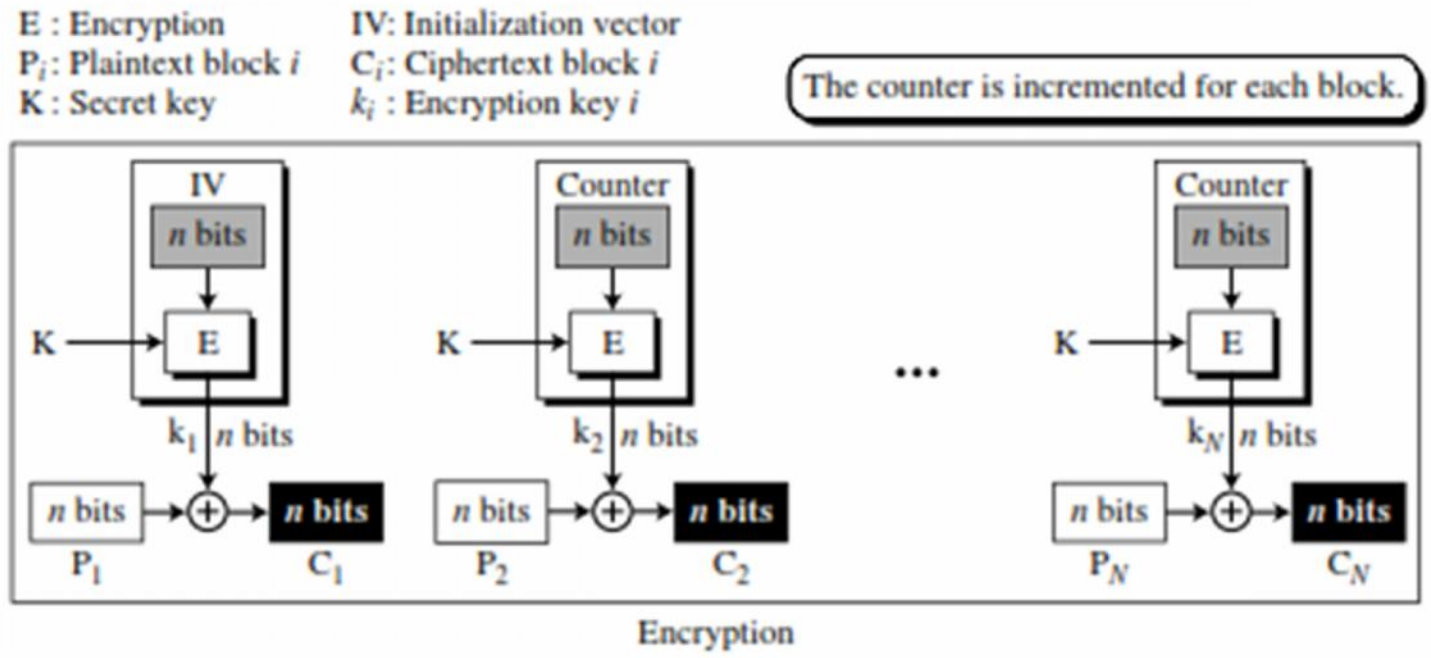
OFB

Figure 8.6 Encryption in output feedback (OFB) mode



CTR mode

Figure 8.8 *Encryption in counter (CTR) mode*



RC4

- RC4 is a stream cipher that was designed in 1984 by Ronald Rivest for RSA Data Security.
- RC4 is used in many data communication and networking protocols, including SSL/TLS and the IEEE802.11 wireless LAN standard.
- RC4 is a byte-oriented stream cipher in which a byte (8 bits) of a plaintext is exclusive-ored with a byte of key to produce a byte of a ciphertext. The secret key, from which the one-byte keys in the key stream are generated, can contain anywhere from 1 to 256 bytes.

RC4

- RC4 is a stream cipher and variable length key algorithm. This algorithm encrypts one byte at a time (or larger units on a time).
- A key input is pseudorandom bit generator that produces a stream 8-bit number that is unpredictable without knowledge of input key, The output of the generator is called key-stream, is combined one byte at a time with the plaintext stream cipher using X-OR operation.

Algorithm

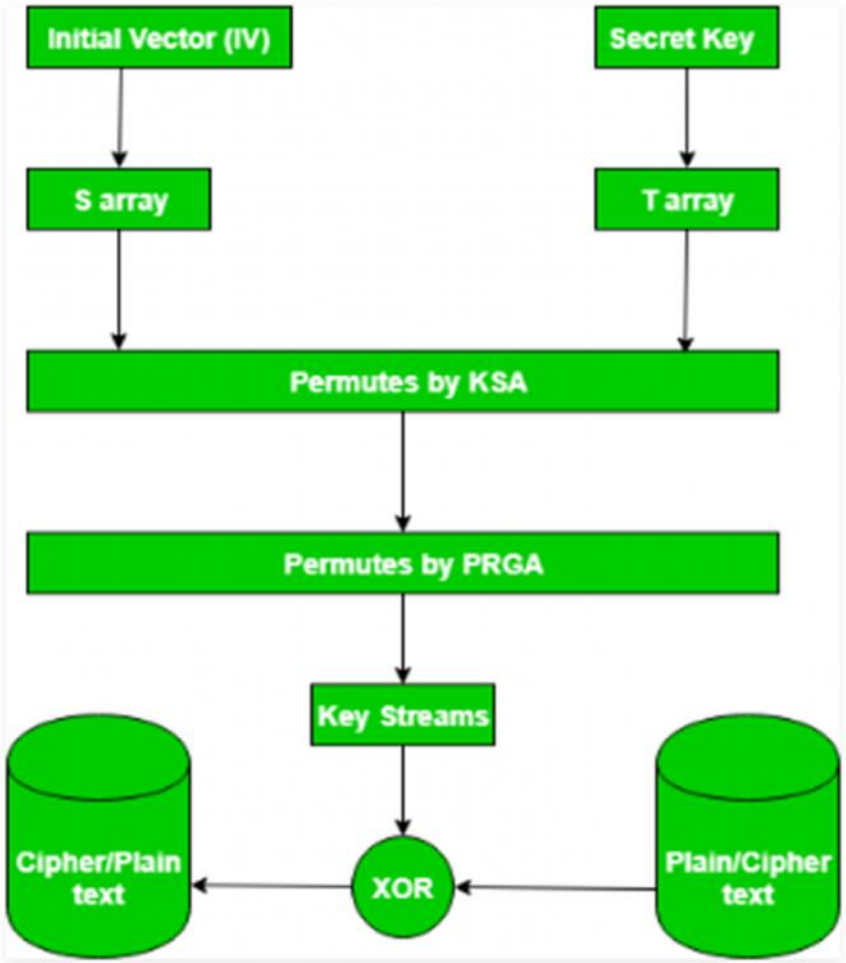
```
RC4 Encryption
10011000 ? 01010000 = 11001000

RC4 Decryption
11001000 ? 01010000 = 10011000
```

Key-Generation Algorithm –

A variable-length key from 1 to 256 byte is used to initialize a 256-byte state vector S, with elements S[0] to S[255]. For encryption and decryption, a byte k is generated from S by selecting one of the 255 entries in a systematic fashion, then the entries in S are permuted again.

Algorithm


Initial Vector (IV)
Secret Key
S array
T array
Permutes by KSA
Permutes by PRGA
Key Streams
Cipher/Plain text
XOR
Plain/Cipher text

RC4

- In September 2015, Microsoft announced the end of using RC4 in Microsoft edge and internet explorer 11.

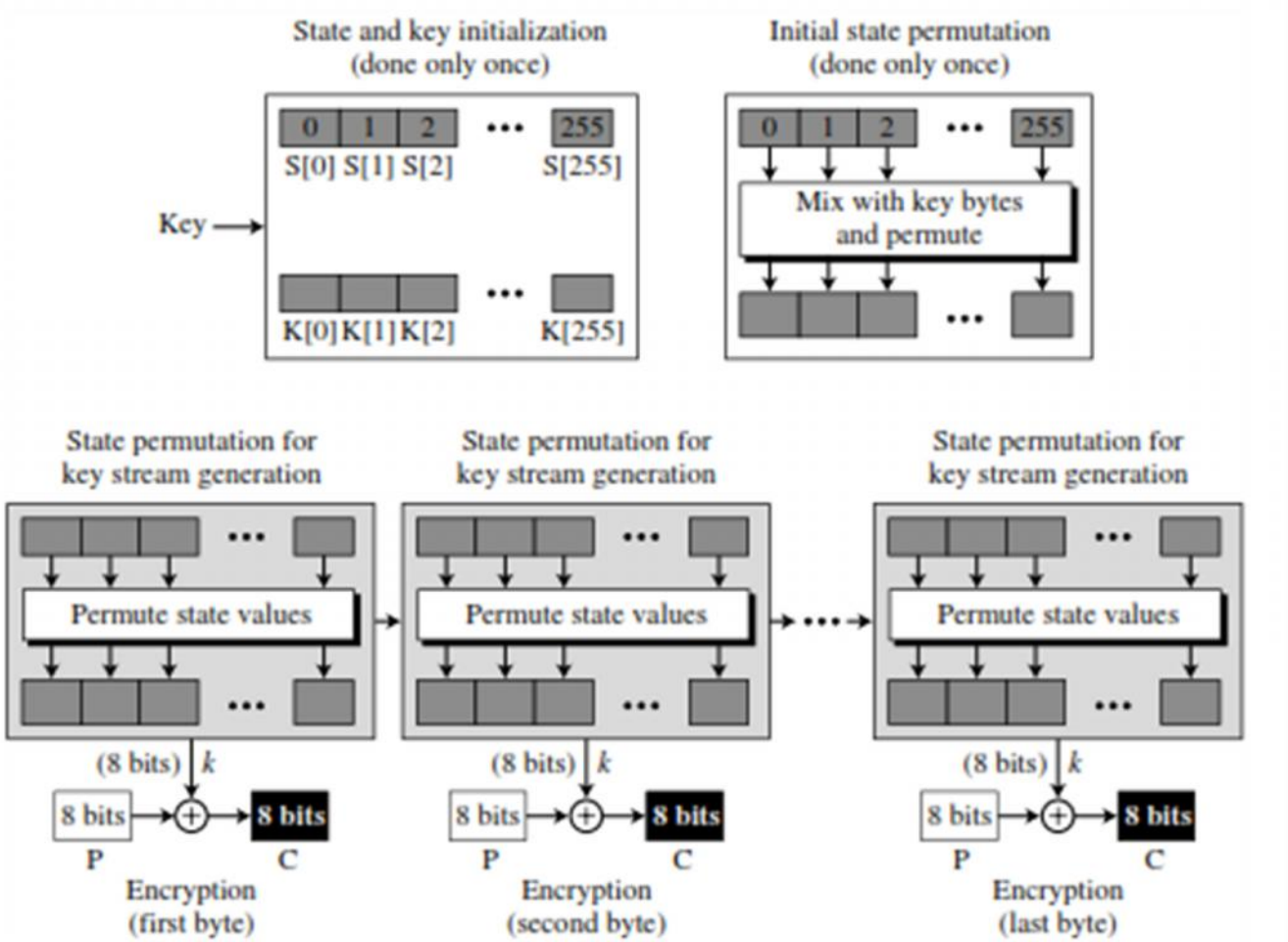
RC4

State

RC4 is based on the concept of a state. At each moment, a state of 256 bytes is active, from which one of the bytes is randomly selected to serve as the key for encryption. The idea can be shown as an array of bytes:

S[0]	S[1]	S[2]	...	S[255]

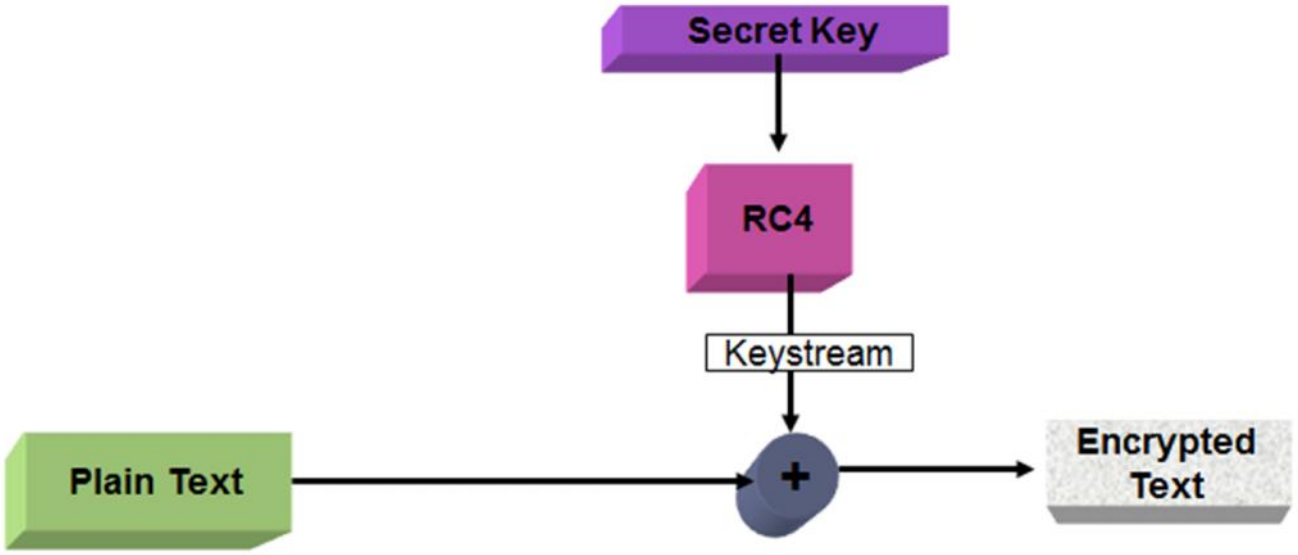
Encryption


State and key initialization
(done only once)
Initial state permutation
(done only once)
0
1
2
255
S[0] S[1] S[2]
S[255]
Key
K[0] K[1] K[2]
K[255]
Mix with key bytes
and permute
State permutation for
key stream generation
State permutation for
key stream generation
State permutation for
key stream generation
Permute state values
Permute state values
Permute state values
(8 bits)
k
8 bits
8 bits
P
C
Encryption
(first byte)
Encryption
(second byte)
Encryption
(last byte)

RC4

- A symmetric key encryption algorithm invented by Ron Rivest
  - A proprietary cipher owned by RSA, kept secret
  - Code released at the sites of Cyberpunk remailers
- **Variable key size**, **byte-oriented** stream cipher
  - Normally uses 64 bit and 128 bit key sizes.
- Used in
  - SSL/TLS (Secure socket, transport layer security) between web browsers and servers,
  - IEEE 802.11 wirelss LAN std: WEP (Wired Equivalent Privacy), WPA (WiFi Protocol Access) protocol

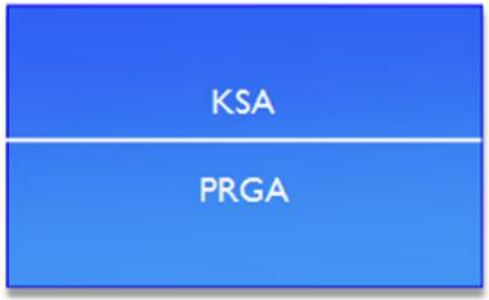
Algorithm



Parts

- Consists of 2 parts:
  - Key Scheduling Algorithm (KSA)
  - Pseudo-Random Generation Algorithm (PRGA)

- KSA
  - Generate State array
- PRGA on the KSA
  - Generate keystream
  - XOR keystream with the data to generated encrypted stream



The KSA

- Use the secret key to initialize and permutation of state vector S, done in two steps

1

```
for i = 0 to 255 do
   S[i] = i;
   T[i] = K[i mod(|K|)]);
```

[S], S is set equal to the values from 0 to 255
S[0]=0, S[1]=1,…, S[255]=255
[T], A temporary vector
[K], Array of bytes of secret key
|K| = Keylen, Length of (K)

2

```
j = 0;
for i = 0 to 255 do
  j = (j+S[i]+T[i])(mod 256)
swap (S[i], S[j])
```

- Use T to produce initial permutation of S
- The only operation on S is a swap;
  S still contains number from 0 to 255

After KSA, the input key and the temporary vector T will be no longer used

The PRGA

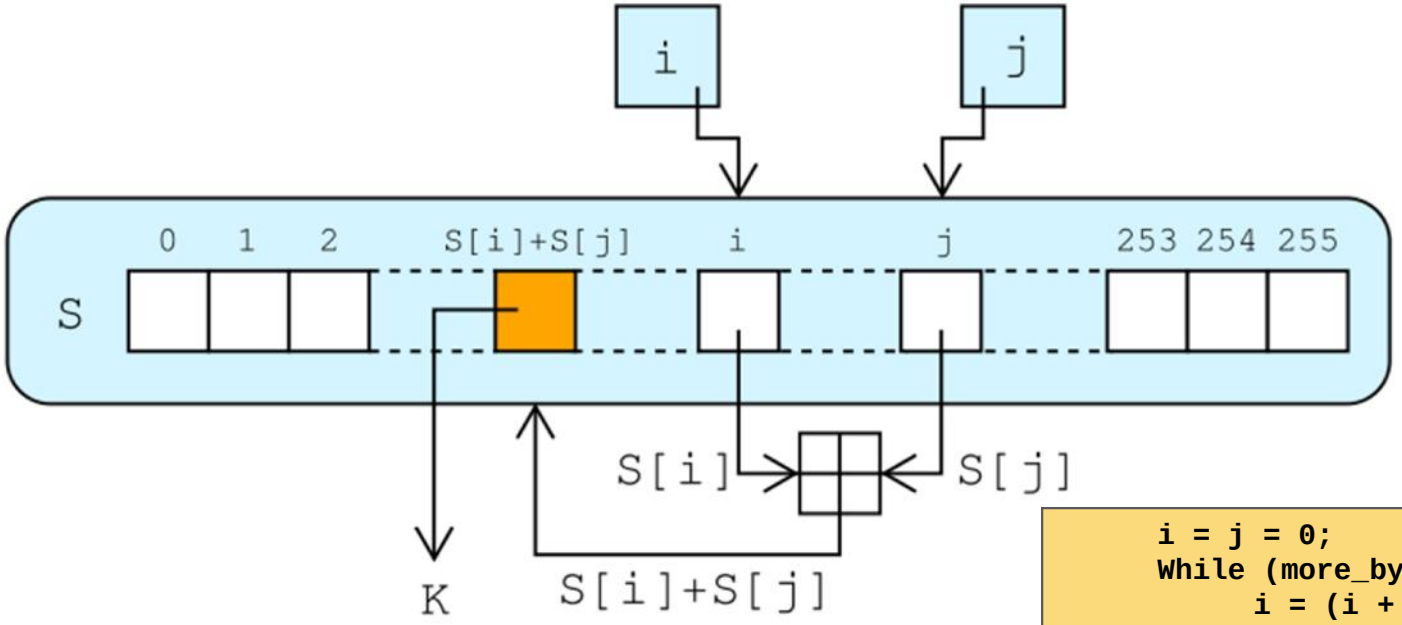
- Generate key stream *k* , one by one
- XOR S[k] with next byte of message to encrypt/decrypt

```
i = j = 0;
While (more_byte_to_encrypt)
    i = (i + 1) (mod 256);
    j = (j + S[i]) (mod 256);
    swap(S[i], S[j]);
    k = (S[i] + S[j]) (mod 256);
    C_i = M_i XOR S[k];
```

Sum of shuffled pair selects "stream key" value from permutation

RC4 Lookup Stage

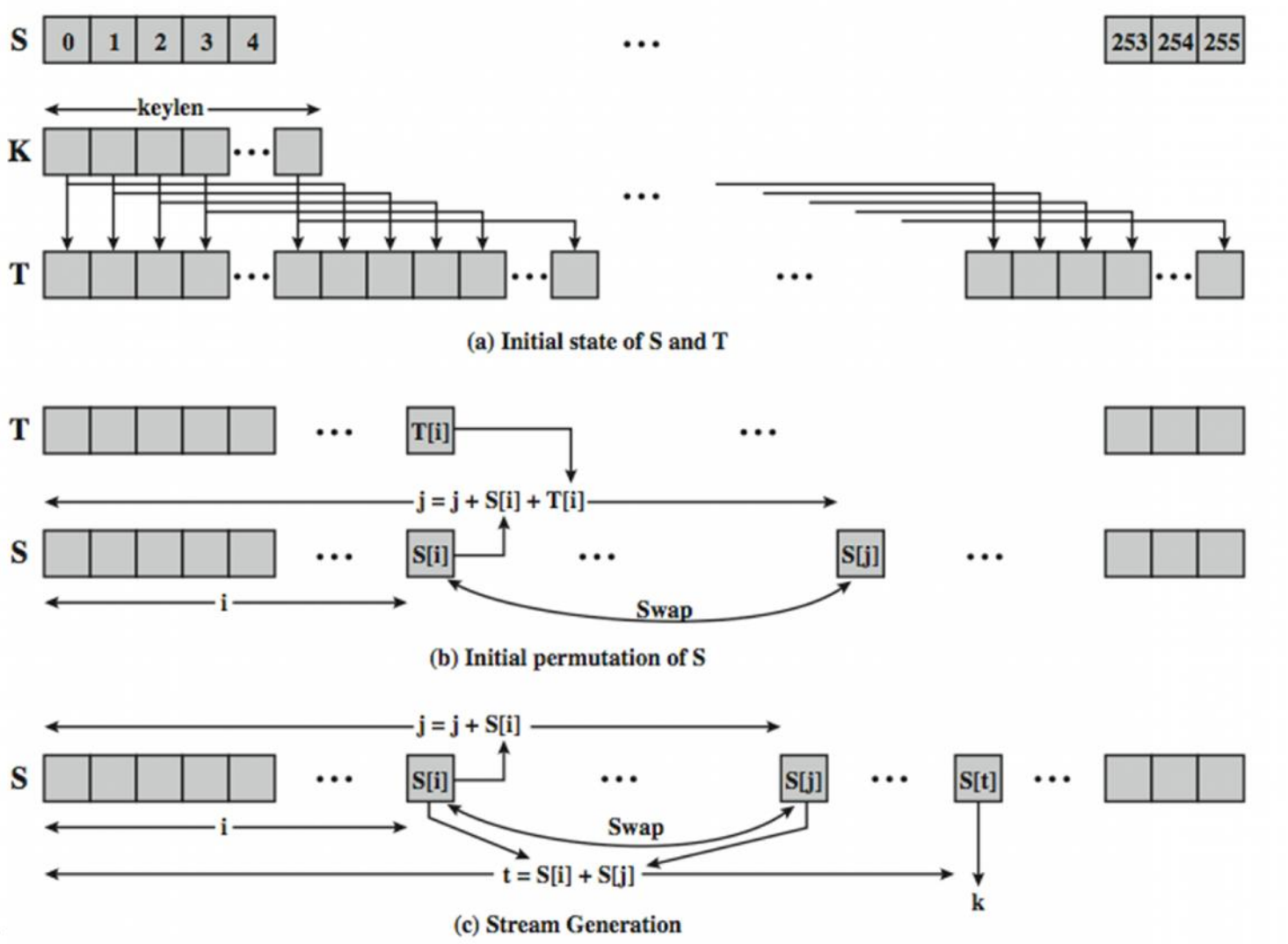
- The output byte is selected by looking up the values of S[i] and S[j], adding them together modulo 256, and then looking up the sum in S
- S [S[i] + S[j]] is used as a byte of the key stream, *K*



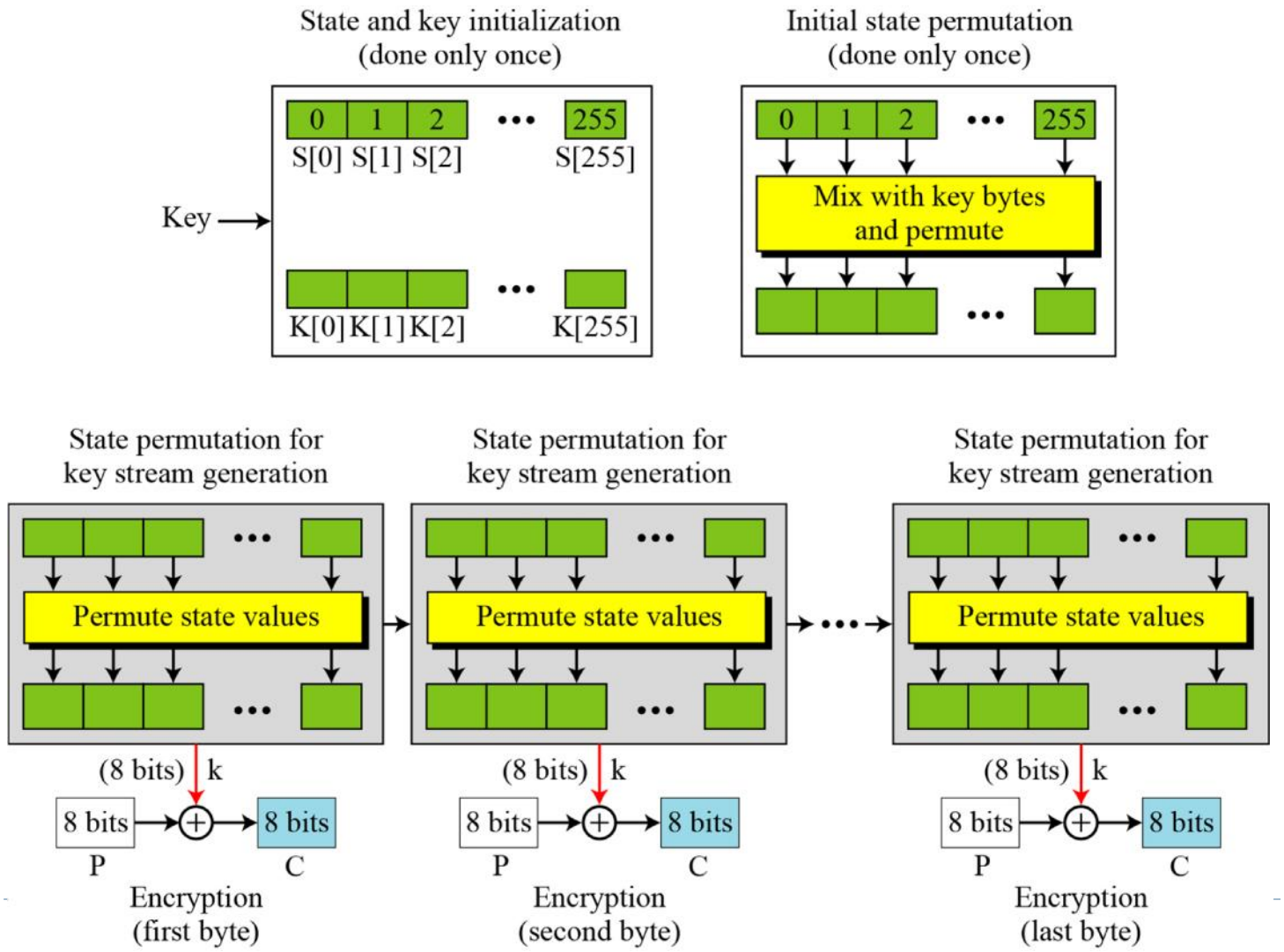
http://en.wikipedia.org/wiki/File:RC4.svg

```
i = j = 0;
While (more_byte_to_encrypt)
      i = (i + 1) (mod 256);
      j = (j + S[i]) (mod 256);
      swap(S[i], S[j]);
      k = (S[i] + S[j]) (mod 256);
      C_i = M_i XOR S[k];
```

Detailed Diagram



Overall Operation of RC4



Decryption using RC4

- Use the same secret key as during the encryption phase.
- Generate keystream by running the KSA and PRGA.
- XOR keystream with the encrypted text to generate the plain text.
- Logic is simple :

$$(A \text{ xor } B) \text{ xor } B = A$$

A = Plain Text or Data
B = KeyStream