Part I

From theory to practice in cryptography

FROM CRYPTO-THEORY to CRYPTO-PRACTICE

In this chapter we deal with several applied cryptography methods, systems and problems that have played very important role in applications.

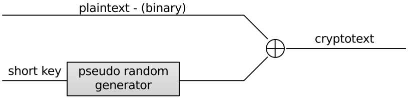
I. SHIFT REGISTERS

The first practical approach to ONE-TIME PAD cryptosystem.

I. SHIFT REGISTERS

The first practical approach to ONE-TIME PAD cryptosystem.

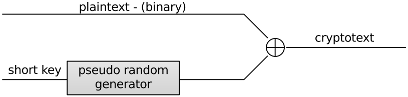
Basic idea: to use a short key, called "seed", and a pseudorandom generator to generate long pseudorandom key.



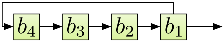
I. SHIFT REGISTERS

The first practical approach to ONE-TIME PAD cryptosystem.

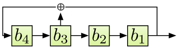
Basic idea: to use a short key, called "seed", and a pseudorandom generator to generate long pseudorandom key.



Shift registers as pseudorandom generators

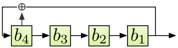


linear shift register



period

0	1	0	0
1	0	1	0
0	1	0	1
0	0	1	0
0	0	0	1
1	0	0	0



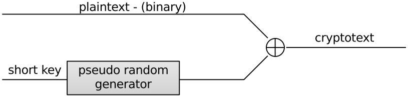
1	1	0	0
1	1	1	0
1	1	1	1
0	1	1	1
1	0	1	1
0	1	0	1
1	0	1	0
1	1	0	1
0	1	1	1
0	0	1	1
1	0	0	1
0	1	0	0
0	0	1	0
0	0	0	1
1	0	0	0

period

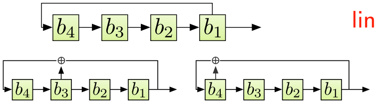
I. SHIFT REGISTERS

The first practical approach to ONE-TIME PAD cryptosystem.

Basic idea: to use a short key, called "seed", and a pseudorandom generator to generate long pseudorandom key.



Shift registers as pseudorandom generators



linear shift register

period

b_4	b_3	b_2	b_1
0	1	0	0
1	0	1	0
0	1	0	1
0	0	1	0
0	0	0	1
1	0	0	0

b_4	b_3	b_2	b_1
1	1	0	0
1	1	1	0
1	1	1	1
0	1	1	1
1	0	1	1
0	1	0	1
1	0	1	0
1	1	0	1
0	1	1	1
0	0	1	1
1	0	0	1
0	1	0	0
0	0	1	0
0	0	0	1
1	0	0	0

period

Theorem For every $n > 0$ there is a linear shift register of maximal period $2^n - 1$.

CRYPTANALYSIS of linear feedback shift registers

Sequences generated by linear shift registers have excellent statistical properties, but they are not resistant to a known plaintext attack.

CRYPTANALYSIS of linear feedback shift registers

Sequences generated by linear shift registers have excellent statistical properties, but they are not resistant to a known plaintext attack.

Example Let us have a 4-bit shift register and let us assume we know 8 bits of a plaintext and of the corresponding cryptotext.

CRYPTANALYSIS of linear feedback shift registers

Sequences generated by linear shift registers have excellent statistical properties, but they are not resistant to a known plaintext attack.

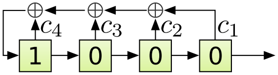
Example Let us have a 4-bit shift register and let us assume we know 8 bits of a plaintext and of the corresponding cryptotext. By XOR-ing these two bit sequences we get 8 bits of the output of the register (of the key), say 00011110

CRYPTANALYSIS of linear feedback shift registers

Sequences generated by linear shift registers have excellent statistical properties, but they are not resistant to a known plaintext attack.

Example Let us have a 4-bit shift register and let us assume we know 8 bits of a plaintext and of the corresponding cryptotext. By XOR-ing these two bit sequences we get 8 bits of the output of the register (of the key), say 00011110

The task is to determine c_4, c_3, c_2, c_1 such that the above sequence is outputted by the shift register

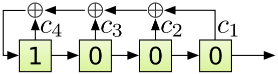


CRYPTANALYSIS of linear feedback shift registers

Sequences generated by linear shift registers have excellent statistical properties, but they are not resistant to a known plaintext attack.

Example Let us have a 4-bit shift register and let us assume we know 8 bits of a plaintext and of the corresponding cryptotext. By XOR-ing these two bit sequences we get 8 bits of the output of the register (of the key), say 00011110

The task is to determine c_4, c_3, c_2, c_1 such that the above sequence is outputted by the shift register



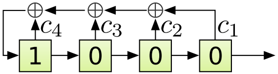
states of cell 4	states of cell 3	states of cell 2	states of cell 1
c_4	1	0	0
$c_4 \oplus c_3$	c_4	1	0
$c_2 \oplus c_4$	$c_4 \oplus c_3$	c_4	1
$c_1 \oplus c_3(c_4 \oplus c_3) \oplus c_4$	$c_2 \oplus c_4$	$c_4 \oplus c_3$	c_4

CRYPTANALYSIS of linear feedback shift registers

Sequences generated by linear shift registers have excellent statistical properties, but they are not resistant to a known plaintext attack.

Example Let us have a 4-bit shift register and let us assume we know 8 bits of a plaintext and of the corresponding cryptotext. By XOR-ing these two bit sequences we get 8 bits of the output of the register (of the key), say 00011110

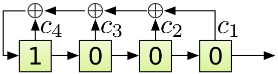
The task is to determine c_4, c_3, c_2, c_1 such that the above sequence is outputted by the shift register



states of cell 4	states of cell 3	states of cell 2	states of cell 1
c_4	1	0	0
$c_4 \oplus c_3$	c_4	1	0
$c_2 \oplus c_4$	$c_4 \oplus c_3$	c_4	1
$c_1 \oplus c_3(c_4 \oplus c_3) \oplus c_4$	$c_2 \oplus c_4$	$c_4 \oplus c_3$	c_4

$$\begin{aligned} c_4 &= 1 \\ c_4 \oplus c_3 &= 1 \\ c_2 \oplus c_4 &= 1 \\ c_1 \oplus c_3 \oplus c_4 \oplus c_3 \cdot c_4 &= 0 \end{aligned} \quad \Rightarrow \quad \begin{aligned} c_4 &= 1 \\ c_3 &= 0 \\ c_2 &= 0 \\ c_1 &= 1 \end{aligned}$$

COMPUTATIONS



states of cell 4	states of cell 3	states of cell 2	states of cell 1
c_4	1	0	0
$c_4 \oplus c_3$	c_4	1	0
$c_2 \oplus c_4$	$c_4 \oplus c_3$	c_4	1
$c_1 \oplus c_3(c_4 \oplus c_3) \oplus c_4$	$c_2 \oplus c_4$	$c_4 \oplus c_3$	c_4

After the second step new value of the first register is

$$N = (c_4 \cdot c_4) \oplus c_3 = c_4 \oplus c_3$$

After the third step new value of the first register is

$$N = ((c_4 \oplus c_3) \cdot c_4) \oplus (c_4 \cdot c_3) \oplus c_2$$

If $c_4 = 1$, then

$$N = \bar{c_3} \oplus c_3 \oplus c_2 = \bar{c_2}$$

and therefore $N = c_4 \oplus c_2$.
If $c_4 = 0$, then

$$N = c_2$$

and therefore $N = c_4 \oplus c_2$

LINEAR RECURRENCES

Linear feedback shift registers are an efficient way to realize recurrence relations of the type

$$x_{n+m} = c_0 x_n + c_1 x_{n+1} + \cdots + c_{m-1} x_{n+m-1} \text{ (mod n)}$$

that can be specified by 2m bits: $c_0, \ldots, c_{m-1}$ and $x_1, \ldots, x_m$.

LINEAR RECURRENCES

Linear feedback shift registers are an efficient way to realize recurrence relations of the type

$$x_{n+m} = c_0 x_n + c_1 x_{n+1} + \cdots + c_{m-1} x_{n+m-1} \text{ (mod n)}$$

that can be specified by 2m bits: $c_0, \ldots, c_{m-1}$ and $x_1, \ldots, x_m$.

Recurrences realized by shift registers on previous slides are:

$$x_{n+4} = x_n; \quad x_{n+4} = x_{n+2} + x_n; \quad x_{n+4} = x_{n+3} + x_n.$$

LINEAR RECURRENCES

Linear feedback shift registers are an efficient way to realize recurrence relations of the type

$$x_{n+m} = c_0 x_n + c_1 x_{n+1} + \cdots + c_{m-1} x_{n+m-1} \text{ (mod n)}$$

that can be specified by 2m bits: $c_0, \ldots, c_{m-1}$ and $x_1, \ldots, x_m$.

Recurrences realized by shift registers on previous slides are:

$$x_{n+4} = x_n; \quad x_{n+4} = x_{n+2} + x_n; \quad x_{n+4} = x_{n+3} + x_n.$$

The main advantage of such recurrences is that a pseudo-random key of a very large period can be generated using a few initial bits.

LINEAR RECURRENCES

Linear feedback shift registers are an efficient way to realize recurrence relations of the type

$$x_{n+m} = c_0 x_n + c_1 x_{n+1} + \cdots + c_{m-1} x_{n+m-1} \text{ (mod n)}$$

that can be specified by 2m bits: $c_0, \ldots, c_{m-1}$ and $x_1, \ldots, x_m$.

Recurrences realized by shift registers on previous slides are:

$$x_{n+4} = x_n; \quad x_{n+4} = x_{n+2} + x_n; \quad x_{n+4} = x_{n+3} + x_n.$$

The main advantage of such recurrences is that a pseudo-random key of a very large period can be generated using a few initial bits.

For example, the recurrence $x_{n+31} = x_n + x_{n+3}$, and any non-zero initial vector, produces sequences with period $2^{31} - 1$, what is more than two billions.

LINEAR RECURRENCES

Linear feedback shift registers are an efficient way to realize recurrence relations of the type

$$x_{n+m} = c_0 x_n + c_1 x_{n+1} + \cdots + c_{m-1} x_{n+m-1} \text{ (mod n)}$$

that can be specified by 2m bits: $c_0, \ldots, c_{m-1}$ and $x_1, \ldots, x_m$.

Recurrences realized by shift registers on previous slides are:

$$x_{n+4} = x_n; \quad x_{n+4} = x_{n+2} + x_n; \quad x_{n+4} = x_{n+3} + x_n.$$

The main advantage of such recurrences is that a pseudo-random key of a very large period can be generated using a few initial bits.

For example, the recurrence $x_{n+31} = x_n + x_{n+3}$, and any non-zero initial vector, produces sequences with period $2^{31} - 1$, what is more than two billions.

Encryption using one-time pad and a key generated by a linear feedback shift register succumbs easily to a known plaintext attack.

LINEAR RECURRENCES

Linear feedback shift registers are an efficient way to realize recurrence relations of the type

$$x_{n+m} = c_0 x_n + c_1 x_{n+1} + \cdots + c_{m-1} x_{n+m-1} \text{ (mod n)}$$

that can be specified by 2m bits: $c_0, \ldots, c_{m-1}$ and $x_1, \ldots, x_m$.

Recurrences realized by shift registers on previous slides are:

$$x_{n+4} = x_n; \quad x_{n+4} = x_{n+2} + x_n; \quad x_{n+4} = x_{n+3} + x_n.$$

The main advantage of such recurrences is that a pseudo-random key of a very large period can be generated using a few initial bits.

For example, the recurrence $x_{n+31} = x_n + x_{n+3}$, and any non-zero initial vector, produces sequences with period $2^{31} - 1$, what is more than two billions.

Encryption using one-time pad and a key generated by a linear feedback shift register succumbs easily to a known plaintext attack. As our main example illustrated, if we know few bits of the plaintext and of the corresponding cryptotext, one can easily determine the initial part of the key and then the corresponding linear recurrence, as already shown.

FINDING LINEAR RECURRENCES - A METHOD - I.

To test whether a given portion of a bit sequence was generated by a recurrence of a length m, if we know the sequence prefix $x_1, \ldots, x_{2m}$, we need to solve the matrix equation

$$\begin{pmatrix} x_1 & x_2 & \ldots & x_m \\ x_2 & x_3 & \ldots & x_{m+1} \\ \vdots & \vdots & \ddots & \vdots \\ x_m & x_{m+1} & \ldots & x_{2m-1} \end{pmatrix} \begin{pmatrix} c_0 \\ c_1 \\ \vdots \\ c_{m-1} \end{pmatrix} = \begin{pmatrix} x_{m+1} \\ x_{m+2} \\ \vdots \\ x_{2m} \end{pmatrix}$$

and then to verify whether the remaining available bits of the sequence, $x_{2m+1}, \ldots$, are really generated by the recurrence just obtained.

III. How to make cryptanalyst's task harder?

Two general methods to achieve the above goal are called diffusion and confusion.

III. How to make cryptanalyst's task harder?

Two general methods to achieve the above goal are called diffusion and confusion.

Diffusion: dissipate the source language redundancy found in the plaintext by spreading it out over the whole cryptotext.

III. How to make cryptanalyst's task harder?

Two general methods to achieve the above goal are called diffusion and confusion.

Diffusion: dissipate the source language redundancy found in the plaintext by spreading it out over the whole cryptotext.

Example 1: A permutation of the plaintext rules out possibility to use frequency tables for digrams, trigrams,....

III. How to make cryptanalyst's task harder?

Two general methods to achieve the above goal are called diffusion and confusion.

Diffusion: dissipate the source language redundancy found in the plaintext by spreading it out over the whole cryptotext.

Example 1: A permutation of the plaintext rules out possibility to use frequency tables for digrams, trigrams,....
Example 2: Make each letter of cryptotext to depend on so many letters of the plaintext as possible

III. How to make cryptanalyst's task harder?

Two general methods to achieve the above goal are called diffusion and confusion.

Diffusion: dissipate the source language redundancy found in the plaintext by spreading it out over the whole cryptotext.

Example 1: A permutation of the plaintext rules out possibility to use frequency tables for digrams, trigrams,....
Example 2: Make each letter of cryptotext to depend on so many letters of the plaintext as possible

Illustration: Let letters of English be encoded by integers from $\{0, \ldots, 25\}$.

III. How to make cryptanalyst's task harder?

Two general methods to achieve the above goal are called diffusion and confusion.

Diffusion: dissipate the source language redundancy found in the plaintext by spreading it out over the whole cryptotext.

Example 1: A permutation of the plaintext rules out possibility to use frequency tables for digrams, trigrams,....
Example 2: Make each letter of cryptotext to depend on so many letters of the plaintext as possible

Illustration: Let letters of English be encoded by integers from $\{0, \ldots, 25\}$. Let the key $k = k_1, \ldots, k_s$ be a sequence of such integers.
Let

$$p_1, \ldots, p_n$$

be a plaintext.
Define, for $0 \leq i < s$, $p_{-i} = k_{s-i}$, and construct the cryptotext by

$$c_i = \left(\sum_{j=0}^{s} p_{i-j} \right) \bmod 26, \; 1 \leq i \leq n$$

III. How to make cryptanalyst's task harder?

Two general methods to achieve the above goal are called diffusion and confusion.

Diffusion: dissipate the source language redundancy found in the plaintext by spreading it out over the whole cryptotext.

Example 1: A permutation of the plaintext rules out possibility to use frequency tables for digrams, trigrams,....
Example 2: Make each letter of cryptotext to depend on so many letters of the plaintext as possible

Illustration: Let letters of English be encoded by integers from $\{0, \ldots, 25\}$. Let the key $k = k_1, \ldots, k_s$ be a sequence of such integers.
Let

$$p_1, \ldots, p_n$$

be a plaintext.
Define, for $0 \leq i < s$, $p_{-i} = k_{s-i}$, and construct the cryptotext by

$$c_i = \left(\sum_{j=0}^{s} p_{i-j} \right) \mod 26, \ 1 \leq i \leq n$$

Confusion makes the relation between the cryptotext and plaintext as complex as possible.

III. How to make cryptanalyst's task harder?

Two general methods to achieve the above goal are called diffusion and confusion.

Diffusion: dissipate the source language redundancy found in the plaintext by spreading it out over the whole cryptotext.

Example 1: A permutation of the plaintext rules out possibility to use frequency tables for digrams, trigrams,....

Example 2: Make each letter of cryptotext to depend on so many letters of the plaintext as possible

Illustration: Let letters of English be encoded by integers from $\{0, \ldots, 25\}$. Let the key $k = k_1, \ldots, k_s$ be a sequence of such integers.

Let

$$p_1, \ldots, p_n$$

be a plaintext.

Define, for $0 \leq i < s$, $p_{-i} = k_{s-i}$, and construct the cryptotext by

$$c_i = \left(\sum_{j=0}^{s} p_{i-j}\right) \mod 26,\ 1 \leq i \leq n$$

Confusion makes the relation between the cryptotext and plaintext as complex as possible.

Example: polyalphabetic substitutions.

CONFUSION and DIFFUSION - a more DETAILED VIEW

As already mentioned, two fundamental cryptographic techniques, introduced already by Shannon, are confusion and diffusion.

CONFUSION and DIFFUSION - a more DETAILED VIEW

As already mentioned, two fundamental cryptographic techniques, introduced already by Shannon, are confusion and diffusion.

Confusion **obscures much the relationship between the plaintext and the cryptotext**, to make much more difficult cryptanalyst's attempts to study cryptotext by looking for redundancies and statistical patterns. (The best way to cause confusion is through complicated substitutions.)

CONFUSION and DIFFUSION - a more DETAILED VIEW

As already mentioned, two fundamental cryptographic techniques, introduced already by Shannon, are confusion and diffusion.

Confusion obscures much the relationship between the plaintext and the cryptotext, to make much more difficult cryptanalyst's attempts to study cryptotext by looking for redundancies and statistical patterns. (The best way to cause confusion is through complicated substitutions.)

Diffusion dissipates redundancy of the plaintext by spreading it over cryptotext – that again makes much more difficult a cryptanalyst's attempts to search for redundancy in the plaintext through observation of cryptotext.

CONFUSION and DIFFUSION - a more DETAILED VIEW

As already mentioned, two fundamental cryptographic techniques, introduced already by Shannon, are confusion and diffusion.

Confusion obscures much the relationship between the plaintext and the cryptotext, to make much more difficult cryptanalyst's attempts to study cryptotext by looking for redundancies and statistical patterns. (The best way to cause confusion is through complicated substitutions.)

Diffusion dissipates redundancy of the plaintext by spreading it over cryptotext – that again makes much more difficult a cryptanalyst's attempts to search for redundancy in the plaintext through observation of cryptotext. (The best way to achieve it is through transformations that cause that bits from different positions in plaintext contribute to the same bit of cryptotext.)

CONFUSION and DIFFUSION - a more DETAILED VIEW

As already mentioned, two fundamental cryptographic techniques, introduced already by Shannon, are confusion and diffusion.

Confusion obscures much the relationship between the plaintext and the cryptotext, to make much more difficult cryptanalyst's attempts to study cryptotext by looking for redundancies and statistical patterns. (The best way to cause confusion is through complicated substitutions.)

Diffusion dissipates redundancy of the plaintext by spreading it over cryptotext – that again makes much more difficult a cryptanalyst's attempts to search for redundancy in the plaintext through observation of cryptotext. (The best way to achieve it is through transformations that cause that bits from different positions in plaintext contribute to the same bit of cryptotext.)

Mono-alphabetic cryptosystems use no confusion and no diffusion.

CONFUSION and DIFFUSION - a more DETAILED VIEW

As already mentioned, two fundamental cryptographic techniques, introduced already by Shannon, are confusion and diffusion.

Confusion obscures much the relationship between the plaintext and the cryptotext, to make much more difficult cryptanalyst's attempts to study cryptotext by looking for redundancies and statistical patterns. (The best way to cause confusion is through complicated substitutions.)

Diffusion dissipates redundancy of the plaintext by spreading it over cryptotext – that again makes much more difficult a cryptanalyst's attempts to search for redundancy in the plaintext through observation of cryptotext. (The best way to achieve it is through transformations that cause that bits from different positions in plaintext contribute to the same bit of cryptotext.)

Mono-alphabetic cryptosystems use no confusion and no diffusion. Polyalphabetic cryptosystems use only confusion.

CONFUSION and DIFFUSION - a more DETAILED VIEW

As already mentioned, two fundamental cryptographic techniques, introduced already by Shannon, are confusion and diffusion.

Confusion obscures much the relationship between the plaintext and the cryptotext, to make much more difficult cryptanalyst's attempts to study cryptotext by looking for redundancies and statistical patterns. (The best way to cause confusion is through complicated substitutions.)

Diffusion dissipates redundancy of the plaintext by spreading it over cryptotext – that again makes much more difficult a cryptanalyst's attempts to search for redundancy in the plaintext through observation of cryptotext. (The best way to achieve it is through transformations that cause that bits from different positions in plaintext contribute to the same bit of cryptotext.)

Mono-alphabetic cryptosystems use no confusion and no diffusion. Polyalphabetic cryptosystems use only confusion. In permutation cryptosystems only diffusion step is used.

CONFUSION and DIFFUSION - a more DETAILED VIEW

As already mentioned, two fundamental cryptographic techniques, introduced already by Shannon, are confusion and diffusion.

Confusion obscures much the relationship between the plaintext and the cryptotext, to make much more difficult cryptanalyst's attempts to study cryptotext by looking for redundancies and statistical patterns. (The best way to cause confusion is through complicated substitutions.)

Diffusion dissipates redundancy of the plaintext by spreading it over cryptotext – that again makes much more difficult a cryptanalyst's attempts to search for redundancy in the plaintext through observation of cryptotext. (The best way to achieve it is through transformations that cause that bits from different positions in plaintext contribute to the same bit of cryptotext.)

Mono-alphabetic cryptosystems use no confusion and no diffusion. Polyalphabetic cryptosystems use only confusion. In permutation cryptosystems only diffusion step is used. DES cryptosyste, introduced later, uses essentially a sequence of confusion and diffusion steps.

IV. DES CRYPTOSYSTEM and its FOLLOWERS

PREHISTORY of DES - LUCIFER

During the years 1966-1972 a need grew up to develop an encryption standard so that many users can easily communicate among themselves using encrypted messages.

PREHISTORY of DES - LUCIFER

During the years 1966-1972 a need grew up to develop an encryption standard so that many users can easily communicate among themselves using encrypted messages.

The idea was that people should be able to communicate secretly using identical encryption and decryption machines and/or software systems.

PREHISTORY of DES - LUCIFER

During the years 1966-1972 a need grew up to develop an encryption standard so that many users can easily communicate among themselves using encrypted messages.

The idea was that people should be able to communicate secretly using identical encryption and decryption machines and/or software systems.

On May 15, 1973 American National Bureau of Standards formally requested to make proposals for a standard encryption system.

PREHISTORY of DES - LUCIFER

During the years 1966-1972 a need grew up to develop an encryption standard so that many users can easily communicate among themselves using encrypted messages.

The idea was that people should be able to communicate secretly using identical encryption and decryption machines and/or software systems.

On May 15, 1973 American National Bureau of Standards formally requested to make proposals for a standard encryption system.

The main candidate was the cryptosystem LUCIFER developed in IBM by Horst Feistel.

PREHISTORY of DES - LUCIFER

During the years 1966-1972 a need grew up to develop an encryption standard so that many users can easily communicate among themselves using encrypted messages.

The idea was that people should be able to communicate secretly using identical encryption and decryption machines and/or software systems.

On May 15, 1973 American National Bureau of Standards formally requested to make proposals for a standard encryption system.

The main candidate was the cryptosystem LUCIFER developed in IBM by Horst Feistel.

After 3 years of arguing of experts, a 56-bit key version of Lucifer was accepted (supposedly only for the next 5 years) as the standard called DES (Data Encryption Standard) on November 23, 1976.

DES was a revolutionary step in the secret-key cryptography history because:

DES was a revolutionary step in the secret-key cryptography history because:

1. Both encryption and decryption algorithms were made public!!!!!!

The same algorithms, software systems or hardware could be used for both encyption and decryption.

DES ALGORITHM – CONCISE DESCRIPTION

Preprocessing: A secret 56-bit key k_{56} is chosen.

DES ALGORITHM – CONCISE DESCRIPTION

Preprocessing: A secret 56-bit key k_{56} is chosen.
A fixed+public permutation ϕ_{56} is applied to get $\phi_{56}(k_{56})$.

DES ALGORITHM – CONCISE DESCRIPTION

Preprocessing: A secret 56-bit key k_{56} is chosen.

A fixed+public permutation ϕ_{56} is applied to get $\phi_{56}(k_{56})$. The first (second) part of the resulting string is taken to get a 28-bit block $C_0(D_0)$.

DES ALGORITHM – CONCISE DESCRIPTION

Preprocessing: A secret 56-bit key k_{56} is chosen.
A fixed+public permutation ϕ_{56} is applied to get $\phi_{56}(k_{56})$.The first (second) part of the resulting string is taken to get a 28-bit block $C_0(D_0)$. Using a fixed+public sequence $s_1, \ldots, s_{16}$ of integers, 16 pairs of 28-bit blocks (C_i, D_i), i = 1,. . . ,16 are obtained as follows:

- $C_i(D_i)$ is obtained from $C_{i-1}(D_{i-1})$ by s_i left shifts.

DES ALGORITHM – CONCISE DESCRIPTION

Preprocessing: A secret 56-bit key k_{56} is chosen.

A fixed+public permutation ϕ_{56} is applied to get $\phi_{56}(k_{56})$.The first (second) part of the resulting string is taken to get a 28-bit block $C_0(D_0)$. Using a fixed+public sequence $s_1, \ldots, s_{16}$ of integers, 16 pairs of 28-bit blocks (C_i, D_i), i = 1,...,16 are obtained as follows:

- $C_i(D_i)$ is obtained from $C_{i-1}(D_{i-1})$ by s_i left shifts.
- Using a fixed and public order, a 48-bit block K_i is created from each pair C_i and D_i.

DES ALGORITHM – CONCISE DESCRIPTION

Preprocessing: A secret 56-bit key k_{56} is chosen.

A fixed+public permutation ϕ_{56} is applied to get $\phi_{56}(k_{56})$.The first (second) part of the resulting string is taken to get a 28-bit block $C_0(D_0)$. Using a fixed+public sequence $s_1, \ldots, s_{16}$ of integers, 16 pairs of 28-bit blocks (C_i, D_i), i = 1,...,16 are obtained as follows:

- $C_i(D_i)$ is obtained from $C_{i-1}(D_{i-1})$ by s_i left shifts.
- Using a fixed and public order, a 48-bit block K_i is created from each pair C_i and D_i.

Encryption A fixed+public permutation ϕ_{64} is applied to a 64-bits long plaintext w to get $w' = L_0R_0$, where each of the strings L_0 and R_0 has 32 bits.

DES ALGORITHM – CONCISE DESCRIPTION

Preprocessing: A secret 56-bit key k_{56} is chosen.
A fixed+public permutation ϕ_{56} is applied to get $\phi_{56}(k_{56})$.The first (second) part of the resulting string is taken to get a 28-bit block $C_0(D_0)$. Using a fixed+public sequence $s_1, \ldots, s_{16}$ of integers, 16 pairs of 28-bit blocks (C_i, D_i), i = 1,...,16 are obtained as follows:

- $C_i(D_i)$ is obtained from $C_{i-1}(D_{i-1})$ by s_i left shifts.
- Using a fixed and public order, a 48-bit block K_i is created from each pair C_i and D_i.

Encryption A fixed+public permutation ϕ_{64} is applied to a 64-bits long plaintext w to get $w' = L_0 R_0$, where each of the strings L_0 and R_0 has 32 bits.16 pairs of 32-bit blocks L_i, R_i, $1 \leq i \leq 16$, are designed using the recurrence:

$$L_i = R_{i-1}$$
$$R_i = L_{i-1} \oplus f(R_{i-1}, K_i),$$

where f is a fixed+public and easy-to-implement function.
The cryptotext $c = \phi_{64}^{-1}(L_{16}, R_{16})$

DES cryptosystem – Data Encryption Standard – 1977

Encryption A fixed+public permutation ϕ_{64} is applied to a 64-bits long plaintext w to get $w' = L_0 R_0$, where each of the strings L_0 and R_0 has 32 bits.

16 pairs of 32-bit blocks L_i, R_i, $1 \leq i \leq 16$, are designed using the recurrence:

$$L_i = R_{i-1}$$
$$R_i = L_{i-1} \oplus f(R_{i-1}, K_i),$$

where f is a fixed+public and easy-to-implement function.
The cryptotext $c = \phi_{64}^{-1}(L_{16}, R_{16})$

DES cryptosystem – Data Encryption Standard – 1977

Encryption A fixed+public permutation ϕ_{64} is applied to a 64-bits long plaintext w to get $w' = L_0R_0$, where each of the strings L_0 and R_0 has 32 bits.

16 pairs of 32-bit blocks L_i, R_i, $1 \leq i \leq 16$, are designed using the recurrence:

$$L_i = R_{i-1}$$
$$R_i = L_{i-1} \oplus f(R_{i-1}, K_i),$$

where f is a fixed+public and easy-to-implement function.
The cryptotext $c = \phi_{64}^{-1}(L_{16}, R_{16})$

Decryption $\phi_{64}(c) = L_{16}R_{16}$ is computed and then the recurrence

$$R_{i-1} = L_i$$
$$L_{i-1} = R_i \oplus f(L_i, K_i),$$

is used to get L_i, R_i i = 15,...,1,0, w = $\phi_{64}^{-1}(L_0, R_0)$.

DES ALGORITHM – DETAILS - PREPROCESSING

A secret 56-bit key k_{56} is chosen.

DES ALGORITHM – DETAILS - PREPROCESSING

A secret 56-bit key k_{56} is chosen.

Eight bits in (new) positions 8, 16,..., 64 are added to the key, to make each byte of odd parity.

DES ALGORITHM – DETAILS - PREPROCESSING

A secret 56-bit key k_{56} is chosen.

Eight bits in (new) positions 8, 16,..., 64 are added to the key, to make each byte of odd parity.

This step is useful for errors detection in the key distribution and in storage.

A secret 56-bit key k_{56} is chosen.
Eight bits in (new) positions 8, 16,..., 64 are added to the key, to make each byte of odd parity.
This step is useful for errors detection in the key distribution and in storage.
56 bits of the key are now subject to the following fixed+public permutation ϕ_{56}:

57	49	41	33	25	17	9
1	58	50	42	34	26	18
10	2	59	51	43	35	27
19	11	3	60	52	44	36
63	55	47	39	31	23	15
7	62	54	46	38	30	22
14	6	61	53	45	37	29
21	13	5	28	20	12	4

The first (second) part of the resulting string is taken to get a 28-bit block $C_0(D_0)$.

DES ALGORITHM – DETAILS - PREPROCESSING

A secret 56-bit key k_{56} is chosen.

Eight bits in (new) positions 8, 16,..., 64 are added to the key, to make each byte of odd parity.

This step is useful for errors detection in the key distribution and in storage.

56 bits of the key are now subject to the following fixed+public permutation ϕ_{56}:

57	49	41	33	25	17	9
1	58	50	42	34	26	18
10	2	59	51	43	35	27
19	11	3	60	52	44	36
63	55	47	39	31	23	15
7	62	54	46	38	30	22
14	6	61	53	45	37	29
21	13	5	28	20	12	4

The first (second) part of the resulting string is taken to get a 28-bit block $C_0(D_0)$.

Blocks C_i, D_i for $i = 1, 2, \ldots, 16$ are now constructed from blocks C_{i-1}, D_{i-1} by one or two left shifts according the following table

ble:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

shift means a rotation of the bits one place to the left; after one left

NEXT STEP II

Using a fixed and publicly known order,

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

16 subkeys k_i, each of 48 bits, are then created, each k_i from blocks C_i, D_i

DES -ENCRYPTION DETAILS

A fixed+public initial permutation ϕ_{64}

58	50	42	34	26	18	10	2
60	52	44	36	28	20	12	4
62	54	46	38	30	22	14	6
64	56	48	40	32	24	16	8
57	49	41	33	25	17	9	1
59	51	43	35	27	19	11	3
61	53	45	37	29	21	13	5
63	55	47	39	31	23	15	7

is applied to a 64-bits long plaintext w to get $w' = L_0 R_0$, where L_0 (R_0) has 32 bits.

DES -ENCRYPTION DETAILS

A fixed+public initial permutation ϕ_{64}

58	50	42	34	26	18	10	2
60	52	44	36	28	20	12	4
62	54	46	38	30	22	14	6
64	56	48	40	32	24	16	8
57	49	41	33	25	17	9	1
59	51	43	35	27	19	11	3
61	53	45	37	29	21	13	5
63	55	47	39	31	23	15	7

is applied to a 64-bits long plaintext w to get $w' = L_0 R_0$, where L_0 (R_0) has 32 bits.
16 pairs of 32-bit blocks L_i, R_i, $1 \leq i \leq 16$, are then designed using the recurrences:

$$L_i = R_{i-1}$$
$$R_i = L_{i-1} \oplus f(R_{i-1}, K_i),$$

where f is a fixed+public and easy-to-implement function to be described next.

DES -ENCRYPTION DETAILS

A fixed+public initial permutation ϕ_{64}

58	50	42	34	26	18	10	2
60	52	44	36	28	20	12	4
62	54	46	38	30	22	14	6
64	56	48	40	32	24	16	8
57	49	41	33	25	17	9	1
59	51	43	35	27	19	11	3
61	53	45	37	29	21	13	5
63	55	47	39	31	23	15	7

is applied to a 64-bits long plaintext w to get $w' = L_0 R_0$, where L_0 (R_0) has 32 bits. 16 pairs of 32-bit blocks L_i, R_i, $1 \leq i \leq 16$, are then designed using the recurrences:

$$L_i = R_{i-1}$$
$$R_i = L_{i-1} \oplus f(R_{i-1}, K_i),$$

where f is a fixed+public and easy-to-implement function to be described next. The cryptotext is now $c = \phi_{64}^{-1}(L_{16}, R_{16})$

FUNCTION f

The function f produces from a 32-bit block R_{i-1} and a 48-bit subkey K_i a 32-bit block as follows:

FUNCTION f

The function f produces from a 32-bit block R_{i-1} and a 48-bit subkey K_i a 32-bit block as follows:
At first, the 32-bit block is expanded into 48-bits according the following table:

32	1	2	3	4	5
4	5	6	7	8	9
8	9	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	30	31	32	1

After this expansion two 48-bits blocks are XOR-ed - bit by bit.

DES - ENCRYPTION - CONTINUATION

The resulting block of 48 bits is now divided into eight 6-bit blocks $B_1, B_2, \ldots, B_8$ and j-th of these eight 6-bit blocks is transformed into a 4-bit block using table S_j. The first two of them are:

DES - ENCRYPTION - CONTINUATION

The resulting block of 48 bits is now divided into eight 6-bit blocks $B_1, B_2, \ldots, B_8$ and j-th of these eight 6-bit blocks is transformed into a 4-bit block using table S_j. The first two of them are:

S_1

14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13

S_2

15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10
3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5
0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15
13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9

DES - ENCRYPTION - CONTINUATION

The resulting block of 48 bits is now divided into eight 6-bit blocks $B_1, B_2, \ldots, B_8$ and j-th of these eight 6-bit blocks is transformed into a 4-bit block using table S_j. The first two of them are:

S_1

14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13

S_2

15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10
3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5
0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15
13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9

Transformation is performed as follows. For a given 8-bit string, the first and last bit determine a number $x \in \{0, 1, 2, 3\}$ and the middle four bits a number y. The number in x-row and y-column is then written in binary.

DES - ENCRYPTION - CONTINUATION

The resulting block of 48 bits is now divided into eight 6-bit blocks $B_1, B_2, \dots, B_8$ and j-th of these eight 6-bit blocks is transformed into a 4-bit block using table S_j. The first two of them are:

S_1

14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13

S_2

15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10
3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5
0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15
13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9

Transformation is performed as follows. For a given 8-bit string, the first and last bit determine a number $x \in \{0, 1, 2, 3\}$ and the middle four bits a number y. The number in x-row and y-column is then written in binary.

Example: for the string 110010, we have $x = 2$, $y = 9$ and the resulting number defined by S_1 is 15. Therefore the resulting string is 1111.

KEY ELEMENTS of DES

- A cryptosystem is called linear if each bit of cryptotext is a linear combination of bits of plaintext.
- For linear cryptosystems there is a powerful decryption method - so-called linear cryptanalysis.
- The only components of DES that are non-linear are S-boxes.
- Some of original requirements for S-boxes:
  - Each row of an S-box should include all possible output bit combinations;
  - It two inputs to an S-box differ in precisely one bit, then the output must differ in a minimum of two bits;
  - If two inputs to an S-box differ in their first two bits, but have identical last two bits, the two outputs have to be distinct.
- There have been many other very technical requirements.

How fast is DES?

200 megabits can be encrypted per second using a special hardware.

How fast is DES?

200 megabits can be encrypted per second using a special hardware.

How safe is DES?

Pretty good.

How fast is DES?

200 megabits can be encrypted per second using a special hardware.

How safe is DES?

Pretty good.

How to increase security when using DES?

How fast is DES?

200 megabits can be encrypted per second using a special hardware.

How safe is DES?

Pretty good.

How to increase security when using DES?

1. DOUBLE DES: Use two keys, for a double encryption.

How fast is DES?

200 megabits can be encrypted per second using a special hardware.

How safe is DES?

Pretty good.

How to increase security when using DES?

1. DOUBLE DES: Use two keys, for a double encryption.
2. TRIPLE-DES: Use three keys, k_1, k_2 and k_3 to compute

How fast is DES?

200 megabits can be encrypted per second using a special hardware.

How safe is DES?

Pretty good.

How to increase security when using DES?

1. DOUBLE DES: Use two keys, for a double encryption.
2. TRIPLE-DES: Use three keys, k_1, k_2 and k_3 to compute

$$c = DES_{k_1}(DES_{k_2}^{-1}(DES_{k_3}(w)))$$

How to increase security when encrypting long plaintexts?

$$w = m_1 m_2 \dots m_n$$

where each m_i has 64-bits.

How fast is DES?

200 megabits can be encrypted per second using a special hardware.

How safe is DES?

Pretty good.

How to increase security when using DES?

1. DOUBLE DES: Use two keys, for a double encryption.
2. TRIPLE-DES: Use three keys, k_1, k_2 and k_3 to compute

$$c = DES_{k_1}(DES_{k_2}^{-1}(DES_{k_3}(w)))$$

How to increase security when encrypting long plaintexts?

$$w = m_1 m_2 \dots m_n$$

where each m_i has 64-bits.
Choose a 56-bit key k and a 64-bit block c_0 and compute

$$c_i = DES(m_i \oplus c_{i-1})$$

for i = 1,...,n.

MEET in THE MIDDLE ATTACK

MEET in THE MIDDLE ATTACK

This attack will be illustrated on **DOUBLE-DES**, but can be used to attack also other cryptosystems.

MEET in THE MIDDLE **ATTACK**

This attack will be illustrated on **DOUBLE-DES**, but can be used to attack also other cryptosystems.

Let DES uses an (unknown) set K_1 of keys for the first encryption and a set K_2 of keys for second encryption.

MEET in THE MIDDLE ATTACK

This attack will be illustrated on **DOUBLE-DES**, but can be used to attack also other cryptosystems.

Let DES uses an (unknown) set K_1 of keys for the first encryption and a set K_2 of keys for second encryption.

Assume that the attacker knows a set of pairs (p, c) - plaintext and corresponding cryptotext, such that

$$c = \text{ENCR-DES}_{k_2}(\text{ENCR-DES}_{k_1}(p))$$
$$p = \text{DECR-DES}_{k_1}(\text{DECR-DES}_{k_2}(c))$$

where $k_1 \in K_1, k_2 \in K_2$.

MEET in THE MIDDLE ATTACK

This attack will be illustrated on **DOUBLE-DES**, but can be used to attack also other cryptosystems.

Let DES uses an (unknown) set K_1 of keys for the first encryption and a set K_2 of keys for second encryption.

Assume that the attacker knows a set of pairs (p, c) - plaintext and corresponding cryptotext, such that

$$c = \text{ENCR-DES}_{k_2}(\text{ENCR-DES}_{k_1}(p))$$
$$p = \text{DECR-DES}_{k_1}(\text{DECR-DES}_{k_2}(c))$$

where $k_1 \in K_1, k_2 \in K_2$.

ATTACK:

- Compute $\text{ENCR-DES}_{k_1}(p)$ for all $k_1 \in K_1$;
- Compute $\text{DECR-DES}_{k_2}(c)$ for all $k_2 \in K_2$;

MEET in THE MIDDLE **ATTACK**

This attack will be illustrated on **DOUBLE-DES**, but can be used to attack also other cryptosystems.

Let DES uses an (unknown) set K_1 of keys for the first encryption and a set K_2 of keys for second encryption.

Assume that the attacker knows a set of pairs (p, c) - plaintext and corresponding cryptotext, such that

$$c = \text{ENCR-DES}_{k_2}(\text{ENCR-DES}_{k_1}(p))$$
$$p = \text{DECR-DES}_{k_1}(\text{DECR-DES}_{k_2}(c))$$

where $k_1 \in K_1, k_2 \in K_2$.

ATTACK:

- Compute $\text{ENCR-DES}_{k_1}(p)$ for all $k_1 \in K_1$;
- Compute $\text{DECR-DES}_{k_2}(c)$ for all $k_2 \in K_2$;

Any matches between these two resulting sets is likely to reveal the correct keys.

MEET in THE MIDDLE ATTACK

This attack will be illustrated on **DOUBLE-DES**, but can be used to attack also other cryptosystems.

Let DES uses an (unknown) set K_1 of keys for the first encryption and a set K_2 of keys for second encryption.

Assume that the attacker knows a set of pairs (p, c) - plaintext and corresponding cryptotext, such that

$$c = \text{ENCR-DES}_{k_2}(\text{ENCR-DES}_{k_1}(p))$$
$$p = \text{DECR-DES}_{k_1}(\text{DECR-DES}_{k_2}(c))$$

where $k_1 \in K_1, k_2 \in K_2$.

ATTACK:

- Compute $\text{ENCR-DES}_{k_1}(p)$ for all $k_1 \in K_1$;
- Compute $\text{DECR-DES}_{k_2}(c)$ for all $k_2 \in K_2$;

Any matches between these two resulting sets is likely to reveal the correct keys.

Complexity of attacks

Brute force: $2^{56} \times 2^{56} = 2^{2\times 56} = 2^{112}$;

MITM: $2 \times 2^{56} = 2^{1+56} = 2^{57}$.

MEET in THE MIDDLE ATTACK

This attack will be illustrated on **DOUBLE-DES**, but can be used to attack also other cryptosystems.

Let DES uses an (unknown) set K_1 of keys for the first encryption and a set K_2 of keys for second encryption.

Assume that the attacker knows a set of pairs (p, c) - plaintext and corresponding cryptotext, such that

$$c = \text{ENCR-DES}_{k_2}(\text{ENCR-DES}_{k_1}(p))$$
$$p = \text{DECR-DES}_{k_1}(\text{DECR-DES}_{k_2}(c))$$

where $k_1 \in K_1, k_2 \in K_2$.

ATTACK:

- Compute $\text{ENCR-DES}_{k_1}(p)$ for all $k_1 \in K_1$;
- Compute $\text{DECR-DES}_{k_2}(c)$ for all $k_2 \in K_2$;

Any matches between these two resulting sets is likely to reveal the correct keys.

Complexity of attacks

Brute force: $2^{56} \times 2^{56} = 2^{2\times 56} = 2^{112}$;
MITM: $2 \times 2^{56} = 2^{1+56} = 2^{57}$.

MITM attack has been generalized for the case on n-multiple encodings are used for DES and some other cryptosystems.

The DES CONTROVERSY

1. There have been suspicions that the design of DES might contain hidden “trapdoors‘‘ what allows NSA to decrypt messages.

1. There have been suspicions that the design of DES might contain hidden “trapdoors'‘ what allows NSA to decrypt messages.
2. The main criticism has been that the size of the key-space, 2^{56}, is too small for DES to be really secure.

The DES CONTROVERSY

1. There have been suspicions that the design of DES might contain hidden “trapdoors‘‘ what allows NSA to decrypt messages.
2. The main criticism has been that the size of the key-space, 2^{56}, is too small for DES to be really secure.
3. In 1977 Diffie+Hellamn suggested that for \$ 20 millions one could build a VLSI chip that could search the entire key space within 1 day.

The DES CONTROVERSY

1. There have been suspicions that the design of DES might contain hidden “trapdoors“ what allows NSA to decrypt messages.
2. The main criticism has been that the size of the key-space, 2^{56}, is too small for DES to be really secure.
3. In 1977 Diffie+Hellamn suggested that for \$ 20 millions one could build a VLSI chip that could search the entire key space within 1 day.
4. In 1993 M. Wiener suggested a machine of the cost \$ 100.000 that could find the key in 1.5 days.

WEAKNESSES of DES

- Existence of weak keys: they are such keys k that for any plaintext p,

$$E_k(E_k(p)) = p.$$

There are four such keys:
$k \in \{(0^{28}, 0^{28}), (1^{28}, 1^{28}), (0^{28}, 1^{28}), (1^{28}, 0^{28})\}$

WEAKNESSES of DES

- Existence of weak keys: they are such keys k that for any plaintext p,
$$E_k(E_k(p)) = p.$$
There are four such keys:
$k \in \{(0^{28}, 0^{28}), (1^{28}, 1^{28}), (0^{28}, 1^{28}), (1^{28}, 0^{28})\}$
- The existence of semi-weak key pairs (k_1, k_2) such that for any plaintext
$$E_{k_1}(E_{k_2}(p)) = p.$$

WEAKNESSES of DES

- Existence of weak keys: they are such keys k that for any plaintext p,
$$E_k(E_k(p)) = p.$$
There are four such keys:
$k \in \{(0^{28}, 0^{28}), (1^{28}, 1^{28}), (0^{28}, 1^{28}), (1^{28}, 0^{28})\}$
- The existence of semi-weak key pairs (k_1, k_2) such that for any plaintext
$$E_{k_1}(E_{k_2}(p)) = p.$$
- The existence of complementation property
$$E_{c(k)}(c(p)) = c(E_k(p)),$$
where c(x) is binary complement of binary string x.

MAIN DES MODES of OPERATION

DES MODES of OPERATION

ECB (Electronic Code Book) mode: to encode a sequence

$$x_1, x_2, x_3, \ldots$$

of 64-bit plaintext blocks, each x_i is encrypted with the same key.

DES MODES of OPERATION

ECB (Electronic Code Book) mode: to encode a sequence

$$x_1, x_2, x_3, \ldots$$

of 64-bit plaintext blocks, each x_i is encrypted with the same key.

CBC (Cipher Block Chaining) mode: to encode a sequence

$$x_1, x_2, x_3, \ldots$$

of 64-bit plaintext blocks, a c_0 is chosen and each x_i is encrypted to get cryptotext

$$c_i = e_k(c_{i-1} \oplus x_i).$$

DES MODES of OPERATION

ECB (Electronic Code Book) mode: to encode a sequence

$$x_1, x_2, x_3, \ldots$$

of 64-bit plaintext blocks, each x_i is encrypted with the same key.

CBC (Cipher Block Chaining) mode: to encode a sequence

$$x_1, x_2, x_3, \ldots$$

of 64-bit plaintext blocks, a c_0 is chosen and each x_i is encrypted to get cryptotext

$$c_i = e_k(c_{i-1} \oplus x_i).$$

OFB (Output Feedback) mode: to encode a sequence

DES MODES of OPERATION

ECB (Electronic Code Book) mode: to encode a sequence

$$x_1, x_2, x_3, \ldots$$

of 64-bit plaintext blocks, each x_i is encrypted with the same key.

CBC (Cipher Block Chaining) mode: to encode a sequence

$$x_1, x_2, x_3, \ldots$$

of 64-bit plaintext blocks, a c_0 is chosen and each x_i is encrypted to get cryptotext

$$c_i = e_k(c_{i-1} \oplus x_i).$$

OFB (Output Feedback) mode: to encode a sequence

$$x_1, x_2, x_3, \ldots$$

of 64-bit plaintext blocks, a z_0 is chosen, $z_i = e_k(z_{i-1})$ are computed and each x_i is encrypted to get cryptotext $c_i = x_i \oplus z_i$.

DES MODES of OPERATION

ECB (Electronic Code Book) mode: to encode a sequence

$$x_1, x_2, x_3, \ldots$$

of 64-bit plaintext blocks, each x_i is encrypted with the same key.

CBC (Cipher Block Chaining) mode: to encode a sequence

$$x_1, x_2, x_3, \ldots$$

of 64-bit plaintext blocks, a c_0 is chosen and each x_i is encrypted to get cryptotext

$$c_i = e_k(c_{i-1} \oplus x_i).$$

OFB (Output Feedback) mode: to encode a sequence

$$x_1, x_2, x_3, \ldots$$

of 64-bit plaintext blocks, a z_0 is chosen, $z_i = e_k(z_{i-1})$ are computed and each x_i is encrypted to get cryptotext $c_i = x_i \oplus z_i$.

CFB (Cipher Feedback) mode: to encode a sequence

$$x_1, x_2, x_3, \ldots$$

of 64-bit plaintext blocks a c_0 is chosen and each x_i is encrypted to get cryptotext

$$c_i = x_i \oplus z_i, \text{ where } z_i = e_k(c_{i-1}).$$

ADVANTAGES of DIFFERENT ENCRYPTION MODES

- CBC mode is used for block-encryption and also for authentication;

ADVANTAGES of DIFFERENT ENCRYPTION MODES

- CBC mode is used for block-encryption and also for authentication;
- CFB mode is used for stream-encryptions;

ADVANTAGES of DIFFERENT ENCRYPTION MODES

- CBC mode is used for block-encryption and also for authentication;
- CFB mode is used for stream-encryptions;
- OFB mode is used for stream-encryptions that require message authentication;

ADVANTAGES of DIFFERENT ENCRYPTION MODES

- CBC mode is used for block-encryption and also for authentication;
- CFB mode is used for stream-encryptions;
- OFB mode is used for stream-encryptions that require message authentication;

CTR MODE

Counter Mode – some consider it as the best one.

ADVANTAGES of DIFFERENT ENCRYPTION MODES

- CBC mode is used for block-encryption and also for authentication;
- CFB mode is used for stream-encryptions;
- OFB mode is used for stream-encryptions that require message authentication;

CTR MODE

Counter Mode – some consider it as the best one.
Key design: $k_i = E_k(n, i)$ for a nonce n;
Encryption: $c_i = x_i \oplus k_i$

ADVANTAGES of DIFFERENT ENCRYPTION MODES

- CBC mode is used for block-encryption and also for authentication;
- CFB mode is used for stream-encryptions;
- OFB mode is used for stream-encryptions that require message authentication;

CTR MODE

Counter Mode – some consider it as the best one.
Key design: $k_i = E_k(n, i)$ for a nonce n;
Encryption: $c_i = x_i \oplus k_i$
This mode is very fast because a key stream can be parallelised to any degree. Because of that this mode is used in network security applications.

- In 1993 M. J. Weiner suggested that one could design, using one million dollars, a computer capable to decrypt, using brute force, DES in 3.5 hours.

KILLERS and DEATH of DES

- In 1993 M. J. Weiner suggested that one could design, using one million dollars, a computer capable to decrypt, using brute force, DES in 3.5 hours.
- In 1998 group of P. Kocher designed, using a quarter million of dolars, a computer capable to decrypt DES in 56 hours.

KILLERS and DEATH of DES

- In 1993 M. J. Weiner suggested that one could design, using one million dollars, a computer capable to decrypt, using brute force, DES in 3.5 hours.
- In 1998 group of P. Kocher designed, using a quarter million of dolars, a computer capable to decrypt DES in 56 hours.
- In 1999 they did that in 24 hours.

KILLERS and DEATH of DES

- In 1993 M. J. Weiner suggested that one could design, using one million dollars, a computer capable to decrypt, using brute force, DES in 3.5 hours.
- In 1998 group of P. Kocher designed, using a quarter million of dolars, a computer capable to decrypt DES in 56 hours.
- In 1999 they did that in 24 hours.
- It started to be clear that a new cryptosystem with larger keys is badly needed.

PRODUCT and FEISTEL CRYPTOSYSTEM

Design of several important practical cryptosystems used the following three general design principles for cryptosystems.

PRODUCT and FEISTEL CRYPTOSYSTEM

Design of several important practical cryptosystems used the following three general design principles for cryptosystems.

A product cryptosystem combines two or more crypto-transformations in such a way that resulting cryptosystem is more secure than component transformations.

PRODUCT and FEISTEL CRYPTOSYSTEM

Design of several important practical cryptosystems used the following three general design principles for cryptosystems.

A product cryptosystem combines two or more crypto-transformations in such a way that resulting cryptosystem is more secure than component transformations.

An iterated block cryptosystem iteratively uses a round function (and it has as parameters number of rounds r, block bit-size n, subkeys bit-size k) of the input key K from which r subkeys K_i are derived.

PRODUCT and FEISTEL CRYPTOSYSTEM

Design of several important practical cryptosystems used the following three general design principles for cryptosystems.

A product cryptosystem combines two or more crypto-transformations in such a way that resulting cryptosystem is more secure than component transformations.

An iterated block cryptosystem iteratively uses a round function (and it has as parameters number of rounds r, block bit-size n, subkeys bit-size k) of the input key K from which r subkeys K_i are derived.

A Feistel cryptosystem is an iterated cryptosystem mapping 2t-bit plaintext (L_0, R_0) of t-bit blocks L_0 and R_0 to a 2t-bit cryptotext (L_r, R_r), through an r-round process, where $r > 0$.

PRODUCT and FEISTEL CRYPTOSYSTEM

Design of several important practical cryptosystems used the following three general design principles for cryptosystems.

A product cryptosystem combines two or more crypto-transformations in such a way that resulting cryptosystem is more secure than component transformations.

An iterated block cryptosystem iteratively uses a round function (and it has as parameters number of rounds r, block bit-size n, subkeys bit-size k) of the input key K from which r subkeys K_i are derived.

A Feistel cryptosystem is an iterated cryptosystem mapping 2t-bit plaintext (L_0, R_0) of t-bit blocks L_0 and R_0 to a 2t-bit cryptotext (L_r, R_r), through an r-round process, where $r > 0$.

For $0 < I < r + 1$, the round i maps (L_{i-1}, R_{i-1}) to (L_i, R_i) using a subkey K_i as follows

$$L_i = R_{i-1},\ R_i = K_{i-1} \oplus f(R_{i-1}, K_i),$$

where each subkey K_i is derived from the main key K.

BLOWFISH CRYPTOSYSTEM

- Blowfish is a Feistel type cryptosystem developed in 1993 by Bruce Schneier.
- **Blowfish is more secure and faster than DES.**
- It encrypts 8-bytes blocks into 8-bytes blocks.
- **Key length is a variable 32k, for k = 1, 2, . . . , 14.**
- For decryption, Blowfish does not reverse the order of encryption, but follows it.
- **S-boxes are of key dependence and they, as well as subkeys, are created by repeated execution of Blowfish enciphering transformation.**
- Blowfish has very strong avalanche effect.
- **A follower of Blowfish, Twofish, was one of 5 main candidates for AES.**
- Blowfish can be downloaded free from the B. Schneier web site.

FEISTEL ENCRYPTION/DECRYPTION SCHEME

FEISTEL ENCRYPTION/DECRYPTION SCHEME

This is a general scheme for design of cryptosystems that was used at the design of several important cryptosystems, such as Lucifer and DES.

FEISTEL ENCRYPTION/DECRYPTION SCHEME

This is a general scheme for design of cryptosystems that was used at the design of several important cryptosystems, such as Lucifer and DES.

Its main advantage is that encryption and decryption are very similar, and even identical in some cases, and then the same hardware can be used for both encryption and decryption.

FEISTEL ENCRYPTION/DECRYPTION SCHEME

This is a general scheme for design of cryptosystems that was used at the design of several important cryptosystems, such as Lucifer and DES.

Its main advantage is that encryption and decryption are very similar, and even identical in some cases, and then the same hardware can be used for both encryption and decryption.

Let F a be a so-called round function and $K_0, K_1, \ldots, K_n$ be sub-keys for rounds $0, 1, 2, \ldots, n$.

FEISTEL ENCRYPTION/DECRYPTION SCHEME

This is a general scheme for design of cryptosystems that was used at the design of several important cryptosystems, such as Lucifer and DES.

Its main advantage is that encryption and decryption are very similar, and even identical in some cases, and then the same hardware can be used for both encryption and decryption.

Let F a be a so-called round function and $K_0, K_1, \ldots, K_n$ be sub-keys for rounds $0, 1, 2, \ldots, n$.

Encryption is as follows:

- Split the plaintext into two equal size parts L_0, R_0.

FEISTEL ENCRYPTION/DECRYPTION SCHEME

This is a general scheme for design of cryptosystems that was used at the design of several important cryptosystems, such as Lucifer and DES.

Its main advantage is that encryption and decryption are very similar, and even identical in some cases, and then the same hardware can be used for both encryption and decryption.

Let F a be a so-called round function and $K_0, K_1, \ldots, K_n$ be sub-keys for rounds $0, 1, 2, \ldots, n$.

Encryption is as follows:

- Split the plaintext into two equal size parts L_0, R_0.
- For rounds $i \in \{0, 1, \ldots, n\}$ compute

$$L_{i+1} = R_i; \; R_{i+1} = L_i \oplus F(R_i, K_i)$$

776,13 36

FEISTEL ENCRYPTION/DECRYPTION SCHEME

This is a general scheme for design of cryptosystems that was used at the design of several important cryptosystems, such as Lucifer and DES.
Its main advantage is that encryption and decryption are very similar, and even identical in some cases, and then the same hardware can be used for both encryption and decryption.
Let F a be a so-called round function and $K_0, K_1, \ldots, K_n$ be sub-keys for rounds $0, 1, 2, \ldots, n$.
Encryption is as follows:

- Split the plaintext into two equal size parts L_0, R_0.
- For rounds $i \in \{0, 1, \ldots, n\}$ compute

$$L_{i+1} = R_i; \;\; R_{i+1} = L_i \oplus F(R_i, K_i)$$

776,13 36The ciphertext is then: (R_{n+1}, L_{n+1})

FEISTEL ENCRYPTION/DECRYPTION SCHEME

This is a general scheme for design of cryptosystems that was used at the design of several important cryptosystems, such as Lucifer and DES.

Its main advantage is that encryption and decryption are very similar, and even identical in some cases, and then the same hardware can be used for both encryption and decryption.

Let F a be a so-called round function and $K_0, K_1, \ldots, K_n$ be sub-keys for rounds $0, 1, 2, \ldots, n$.

Encryption is as follows:

- Split the plaintext into two equal size parts L_0, R_0.
- For rounds $i \in \{0, 1, \ldots, n\}$ compute

$$L_{i+1} = R_i;\;\; R_{i+1} = L_i \oplus F(R_i, K_i)$$

776,13 36The ciphertext is then: (R_{n+1}, L_{n+1})

Decryption of (R_{n+1}, L_{n+1}) is done by computing, for $i = n, n-1, \ldots, 0$

FEISTEL ENCRYPTION/DECRYPTION SCHEME

This is a general scheme for design of cryptosystems that was used at the design of several important cryptosystems, such as Lucifer and DES.
Its main advantage is that encryption and decryption are very similar, and even identical in some cases, and then the same hardware can be used for both encryption and decryption.
Let F a be a so-called round function and $K_0, K_1, \ldots, K_n$ be sub-keys for rounds $0, 1, 2, \ldots, n$.
Encryption is as follows:

- Split the plaintext into two equal size parts L_0, R_0.
- For rounds $i \in \{0, 1, \ldots, n\}$ compute

$$L_{i+1} = R_i;\ \ R_{i+1} = L_i \oplus F(R_i, K_i)$$

776,13 36The ciphertext is then: (R_{n+1}, L_{n+1})
Decryption of (R_{n+1}, L_{n+1}) is done by computing, for $i = n, n-1, \ldots, 0$

$$R_i = L_{i+1}, L_i = R_{i+1} \oplus F(L_{i+1}, K_i)$$

and (L_0, R_0) is the plaintext

FEISTEL ENCRYPTION/DECRYPTION SCHEME

This is a general scheme for design of cryptosystems that was used at the design of several important cryptosystems, such as Lucifer and DES.

Its main advantage is that encryption and decryption are very similar, and even identical in some cases, and then the same hardware can be used for both encryption and decryption.

Let F a be a so-called round function and $K_0, K_1, \ldots, K_n$ be sub-keys for rounds $0, 1, 2, \ldots, n$.

Encryption is as follows:

- Split the plaintext into two equal size parts L_0, R_0.
- For rounds $i \in \{0, 1, \ldots, n\}$ compute

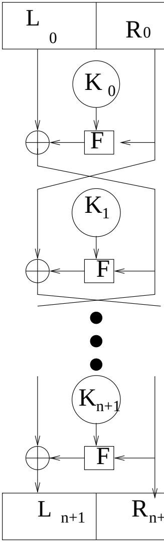
$$L_{i+1} = R_i;\;\; R_{i+1} = L_i \oplus F(R_i, K_i)$$

776,13 36The ciphertext is then: (R_{n+1}, L_{n+1})

Decryption of (R_{n+1}, L_{n+1}) is done by computing, for $i = n, n-1, \ldots, 0$

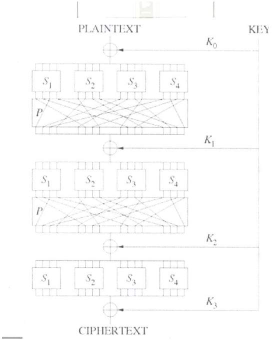
$$R_i = L_{i+1}, L_i = R_{i+1} \oplus F(L_{i+1}, K_i)$$

and (L_0, R_0) is the plaintext



SUBSTITUTION-PERMUTATION ENCRYPTION/DECRYPTION SCHEMES

This scheme, known also as **substitution-permutation network**, is an encryption/decryption method/network that performs a series of substitution-permutation layers of operations composed of S-boxes (substitution boxes) and P-boxes (permutation boxes) as shown in the picture - K_i are keys.



Encryption/decryption system AES discussed next is the most known example of such a system.

V. AES CRYPTOSYSTEM

AES CRYPTOSYSTEM - PREHISTORY

- NIST in USA decided in 1997 to create a new standard for encryption and decryption called **AES**.

AES CRYPTOSYSTEM - PREHISTORY

- NIST in USA decided in 1997 to create a new standard for encryption and decryption called **AES**.
- Anyone could make a proposal and 15 candidates were accepted in 1998.

AES CRYPTOSYSTEM - PREHISTORY

- NIST in USA decided in 1997 to create a new standard for encryption and decryption called **AES**.
- Anyone could make a proposal and 15 candidates were accepted in 1998.
- Based on public comments 5 candidates got into second round.

AES CRYPTOSYSTEM - PREHISTORY

- NIST in USA decided in 1997 to create a new standard for encryption and decryption called **AES**.
- Anyone could make a proposal and 15 candidates were accepted in 1998.
- Based on public comments 5 candidates got into second round.
- High security as well as fast speed and low memory requirements on a variety of computing systems were main criteria.

AES CRYPTOSYSTEM - HISTORY

On October 2, 2000, NIST selected, as new **A**dvanced **E**ncryption **S**tandard, the cryptosystem Rijndael, designed in 1998 by Joan Daemen and Vincent Rijmen.

AES CRYPTOSYSTEM - HISTORY

On October 2, 2000, NIST selected, as new **A**dvanced **E**ncryption **S**tandard, the cryptosystem Rijndael, designed in 1998 by Joan Daemen and Vincent Rijmen.

The main goal has been to develop a new cryptographic standard that could be used to encrypt sensitive governmental information securely, well into the next century.

AES CRYPTOSYSTEM - HISTORY

On October 2, 2000, NIST selected, as new **A**dvanced **E**ncryption **S**tandard, the cryptosystem Rijndael, designed in 1998 by Joan Daemen and Vincent Rijmen.

The main goal has been to develop a new cryptographic standard that could be used to encrypt sensitive governmental information securely, well into the next century.

AES was expected to be used obligatory by U.S. governmental institution and, naturally, voluntarily, but as a necessity, also by the private sector.

AES CRYPTOSYSTEM - HISTORY

On October 2, 2000, NIST selected, as new **A**dvanced **E**ncryption **S**tandard, the cryptosystem Rijndael, designed in 1998 by Joan Daemen and Vincent Rijmen.

The main goal has been to develop a new cryptographic standard that could be used to encrypt sensitive governmental information securely, well into the next century.

AES was expected to be used obligatory by U.S. governmental institution and, naturally, voluntarily, but as a necessity, also by the private sector.

AES is dedicated to 8-bit microprocessors to encrypt 128-bit messages using a key with 128, 192 or 256 bits. In addition, AES is to be used as a standard for authentication (MAC), hashing and pseudorandom numbers generation.

AES CRYPTOSYSTEM - HISTORY

On October 2, 2000, NIST selected, as new **A**dvanced **E**ncryption **S**tandard, the cryptosystem Rijndael, designed in 1998 by Joan Daemen and Vincent Rijmen.

The main goal has been to develop a new cryptographic standard that could be used to encrypt sensitive governmental information securely, well into the next century.

AES was expected to be used obligatory by U.S. governmental institution and, naturally, voluntarily, but as a necessity, also by the private sector.

AES is dedicated to 8-bit microprocessors to encrypt 128-bit messages using a key with 128, 192 or 256 bits. In addition, AES is to be used as a standard for authentication (MAC), hashing and pseudorandom numbers generation.

Motivations and advantages of AES:

AES CRYPTOSYSTEM - HISTORY

On October 2, 2000, NIST selected, as new **A**dvanced **E**ncryption **S**tandard, the cryptosystem Rijndael, designed in 1998 by Joan Daemen and Vincent Rijmen.

The main goal has been to develop a new cryptographic standard that could be used to encrypt sensitive governmental information securely, well into the next century.

AES was expected to be used obligatory by U.S. governmental institution and, naturally, voluntarily, but as a necessity, also by the private sector.

AES is dedicated to 8-bit microprocessors to encrypt 128-bit messages using a key with 128, 192 or 256 bits. In addition, AES is to be used as a standard for authentication (MAC), hashing and pseudorandom numbers generation.

Motivations and advantages of AES:

- Short code and fast and low memory implementations.
- Simplicity and transparency of the design.
- Variable key length.
- Resistance against all known attacks.

- Some operations in AES are define on **bytes**.

- Some operations in AES are define on **bytes**.
- Bytes will be seen as elements of the finite field $GF(2^8)$.

- Some operations in AES are define on **bytes**.
- Bytes will be seen as elements of the finite field $GF(2^8)$.
- Bytes will be represented either by binary 8-bit strings $b_7 b_6 b_5 b_4 b_3 b_2 b_1 b_0$

AES MATHEMATICS - I

- Some operations in AES are define on **bytes**.
- Bytes will be seen as elements of the finite field $GF(2^8)$.
- Bytes will be represented either by binary 8-bit strings $b_7b_6b_5b_4b_3b_2b_1b_0$ or by polynomials

$$b_7x^7 + b_6x^6 + b_5x^5 + b_4x^4 + b_3x^3 + b_2x^2 + b_1x + b_0$$

- Some operations in AES are define on **bytes**.
- Bytes will be seen as elements of the finite field $GF(2^8)$.
- Bytes will be represented either by binary 8-bit strings $b_7 b_6 b_5 b_4 b_3 b_2 b_1 b_0$ or by polynomials

$$b_7x^7 + b_6x^6 + b_5x^5 + b_4x^4 + b_3x^3 + b_2x^2 + b_1x + b_0$$

- Some operations in AES will be defined in terms of **4-bytes words** .

OPERATIONS in THE FIELD $GF(2^8)$

OPERATIONS in THE FIELD $GF(2^8)$

Addition

In polynomial representation, the sum of two bytes is the polynomial whose coeficiants are given by xor-ing coefficients of both bytes-polynomials.

$$(x^6 + x^4 + x^2 + x + 1) + (x^7 + x + 1) = x^7 + x^6 + x_4 + x^2$$

Multiplication

In polynomial representation of bytes, multiplication in $GF(2^8)$ corresponds with multiplication of polynomials modulo an irreducible polynomial

$$m(x) = x^8 + x^4 + x^3 + x + 1$$

Example

$$(x^6 + x^4 + x^2 + x + 1)(x^7 + x + 1) = x^{13} + x^{11} + x^9 + x^8 + x^5 + x^5 + x^4 + x^3 + 1$$

and

$$(x^{13} + x^{11} + x^9 + x^8 + x^5 + x^5 + x^4 + x^3 + 1) \bmod m(x) = x^7 + x_6 + 1$$

The set of 256 possible byte values with operations of addition and multiplication as defined above has the structure of the finite field $GF(2^8)$.

POLYNOMIALS WITH COEFFICIENTS in $GF(2^8)$

POLYNOMIALS WITH COEFFICIENTS in $GF(2^8)$

In polynomial representation, **addition of two polynomials** is given by xor-ng corresponding coefficients.

POLYNOMIALS WITH COEFFICIENTS in $GF(2^8)$

In polynomial representation, **addition of two polynomials** is given by xor-ng corresponding coefficients.

In polynomial representation **multiplication** of two polynomilas is done performing usual multiplicaion and then taking modulus a special polynomial in such a way that multiplication has inverse elements.

AES - BASIC IDEA

- AES is a substitution-permutation network.

AES - BASIC IDEA

- AES is a substitution-permutation network.
- Basic AES implementations operate on 4×4 matrices of bytes called **states**.

AES - BASIC IDEA

- AES is a substitution-permutation network.
- Basic AES implementations operate on 4×4 matrices of bytes called **states**. A 128-bit message is also written as a 4×4 matrix of bytes.

AES - BASIC IDEA

- AES is a substitution-permutation network.
- Basic AES implementations operate on 4×4 matrices of bytes called **states**. A 128-bit message is also written as a 4×4 matrix of bytes.
- Some AES implementations work with states with additional columns in the state matrices.
- Encryption is performed through 10, 12 or 14 rounds depending on whether the key size is 128, 196 or 256.

AES - BASIC IDEA

- AES is a substitution-permutation network.
- Basic AES implementations operate on 4×4 matrices of bytes called **states**. A 128-bit message is also written as a 4×4 matrix of bytes.
- Some AES implementations work with states with additional columns in the state matrices.
- Encryption is performed through 10, 12 or 14 rounds depending on whether the key size is 128, 196 or 256.
- Each round (but the final one) consists of four simple transformations:

AES - BASIC IDEA

- AES is a substitution-permutation network.
- Basic AES implementations operate on 4×4 matrices of bytes called **states**. A 128-bit message is also written as a 4×4 matrix of bytes.
- Some AES implementations work with states with additional columns in the state matrices.
- Encryption is performed through 10, 12 or 14 rounds depending on whether the key size is 128, 196 or 256.
- Each round (but the final one) consists of four simple transformations:
  1. **SubBytes** - byte-wise substitution defined by a special table of 256 bytes.

AES - BASIC IDEA

- AES is a substitution-permutation network.
- Basic AES implementations operate on 4×4 matrices of bytes called **states**. A 128-bit message is also written as a 4×4 matrix of bytes.
- Some AES implementations work with states with additional columns in the state matrices.
- Encryption is performed through 10, 12 or 14 rounds depending on whether the key size is 128, 196 or 256.
- Each round (but the final one) consists of four simple transformations:
  1. **SubBytes** - byte-wise substitution defined by a special table of 256 bytes.
  2. **ShiftRows** - a circular shift of i-th row of the matrix by i positions to the left.

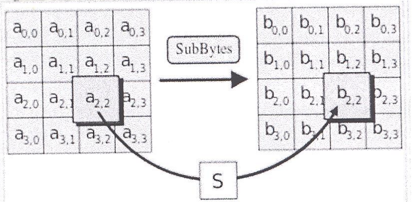
AES - BASIC IDEA

- AES is a substitution-permutation network.
- Basic AES implementations operate on 4×4 matrices of bytes called **states**. A 128-bit message is also written as a 4×4 matrix of bytes.
- Some AES implementations work with states with additional columns in the state matrices.
- Encryption is performed through 10, 12 or 14 rounds depending on whether the key size is 128, 196 or 256.
- Each round (but the final one) consists of four simple transformations:
  1. **SubBytes** - byte-wise substitution defined by a special table of 256 bytes.
  2. **ShiftRows** - a circular shift of i-th row of the matrix by i positions to the left.
  3. **MixColumns** - a linear transformation on each column defined by a 4×4 matrix of bytes.

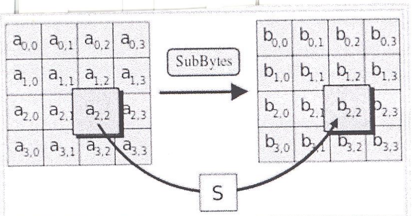
AES - BASIC IDEA

- AES is a substitution-permutation network.
- Basic AES implementations operate on 4×4 matrices of bytes called **states**. A 128-bit message is also written as a 4×4 matrix of bytes.
- Some AES implementations work with states with additional columns in the state matrices.
- Encryption is performed through 10, 12 or 14 rounds depending on whether the key size is 128, 196 or 256.
- Each round (but the final one) consists of four simple transformations:
  1. **SubBytes** - byte-wise substitution defined by a special table of 256 bytes.
  2. **ShiftRows** - a circular shift of i-th row of the matrix by i positions to the left.
  3. **MixColumns** - a linear transformation on each column defined by a 4×4 matrix of bytes.
  4. **AddRoundKey** - bit-wise XOR with a round key defined by another matrix.

THE SubBytes STEP



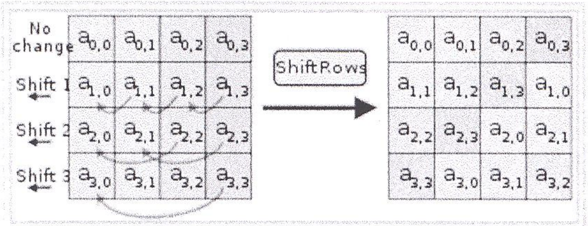
In the SubBytes step, each byte in the state is replaced with its entry in a fixed 8-bit lookup table, S; $b_{ij} = S(a_{ij})$.



In the SubBytes step, each byte in the state is replaced with its entry in a fixed 8-bit lookup table, S; $b_{ij} = S(a_{ij})$.

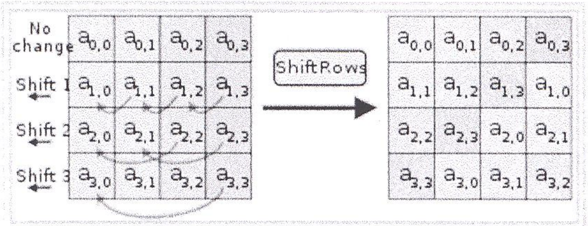
- In this step, each byte in the state is replaced with its entry in a fixed 8-bit lookup table.
- The operation introduces non-linearity into encryption.
- At decryption, an **Inverse SubBytes** step is used.

THE ShiftRows STEP



In the `ShiftRows` step, bytes in each row of the state are shifted cyclically to the left. The number of places each byte is shifted differs for each row.

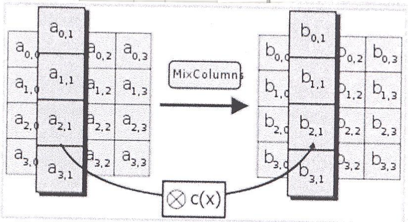
THE ShiftRows STEP



In the ShiftRows step, bytes in each row of the state are shifted cyclically to the left. The number of places each byte is shifted differs for each row.

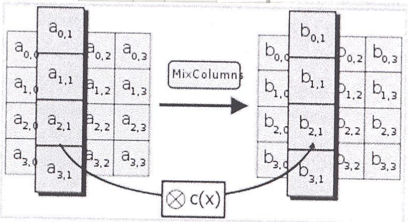
- At this step each row of the state is cyclically shifted by a certain offset.
- This step is done to avoid that columns of states are linearly dependent.

THE MixColumns STEP



In the `MixColumns` step, each column of the state is multiplied with a fixed polynomial $c(x)$.

THE MixColumns STEP

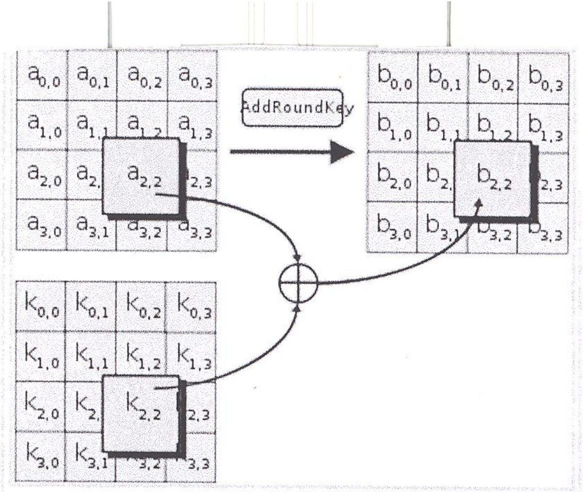


In the `MixColumns` step, each column of the state is multiplied with a fixed polynomial $c(x)$.

During this step, each column is multiplied by the matrix

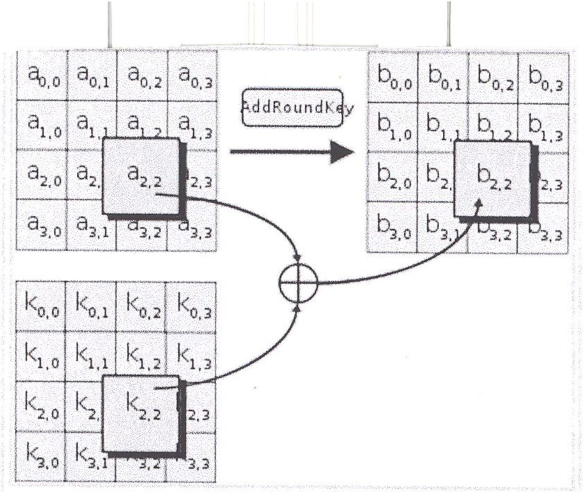
$$\begin{bmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{bmatrix}$$

THE AddRoundKey STEP



In the AddRoundKey step, each byte of the state is combined with a byte of the round subkey using the XOR operation ($\oplus$).

THE AddRoundKey STEP



In the AddRoundKey step, each byte of the state is combined with a byte of the round subkey using the XOR operation ($\oplus$).

PERFORMANCE

AES performs well on a variety of hardware - from 8-bit smart cards to supercomputers.

AES performs well on a variety of hardware - from 8-bit smart cards to supercomputers.

On a Pentium Pro throughput is about 11 MB/s for a 200 MHz processor. On a 1.7 GHz Pentium M, throughput is about 60 MB/s.

PERFORMANCE

AES performs well on a variety of hardware - from 8-bit smart cards to supercomputers.

On a Pentium Pro throughput is about 11 MB/s for a 200 MHz processor. On a 1.7 GHz Pentium M, throughput is about 60 MB/s.

On Intel Core i3/i5/i7 CPUs supporting AES-NI instruction set extensions, throughput can be over 700 MB/s.

BYTE-WISE SUBSTITUTION

Byte-wise substitution $\mathrm{b} = \mathrm{SubByte(a)}$ is defined by the following matrix operations

$$\begin{pmatrix} b_7 \\ b_6 \\ b_5 \\ b_4 \\ b_3 \\ b_2 \\ b_1 \\ b_0 \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \end{pmatrix} \times \begin{pmatrix} (a^{-1})_7 \\ (a^{-1})_6 \\ (a^{-1})_5 \\ (a^{-1})_4 \\ (a^{-1})_3 \\ (a^{-1})_2 \\ (a^{-1})_1 \\ (a^{-1})_0 \end{pmatrix} + \begin{pmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \end{pmatrix}$$

This operation is computationally heavy and it is assumed that it will be implemented by a pre-computed substitution table.

ENCRYPTION in AES

Encryption and decryption are done using state matrices

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

elements of which are bytes.

A byte-matrix with 4 rows and k = 4, 6 or 8 columns is also used to write down a key with D_k = 128, 192 or 256 bits.

ENCRYPTION in AES

Encryption and decryption are done using state matrices

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

elements of which are bytes.

A byte-matrix with 4 rows and k = 4, 6 or 8 columns is also used to write down a key with D_k = 128, 192 or 256 bits.

ENCRYPTION ALGORITHM

1. KeyExpansion

ENCRYPTION in AES

Encryption and decryption are done using state matrices

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

elements of which are bytes.

A byte-matrix with 4 rows and k = 4, 6 or 8 columns is also used to write down a key with D_k = 128, 192 or 256 bits.

ENCRYPTION ALGORITHM

1. KeyExpansion
2. AddRoundKey

ENCRYPTION in AES

Encryption and decryption are done using state matrices

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

elements of which are bytes.

A byte-matrix with 4 rows and k = 4, 6 or 8 columns is also used to write down a key with $D_k = 128$, 192 or 256 bits.

ENCRYPTION ALGORITHM

1. KeyExpansion
2. AddRoundKey
3. **do** (k + 5)-times:
    a) SubByte
    b) ShiftRow
    c) MixColumn
    d) AddRoundKey

ENCRYPTION in AES

Encryption and decryption are done using state matrices

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

elements of which are bytes.

A byte-matrix with 4 rows and k = 4, 6 or 8 columns is also used to write down a key with $D_k = 128$, 192 or 256 bits.

ENCRYPTION ALGORITHM

1. KeyExpansion
2. AddRoundKey
3. **do** (k + 5)-times:
   a) SubByte
   b) ShiftRow
   c) MixColumn
   d) AddRoundKey
4. Final round
   a SubByte
   b ShiftRow
   c AddRoundKey

ENCRYPTION in AES

Encryption and decryption are done using state matrices

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

elements of which are bytes.

A byte-matrix with 4 rows and k = 4, 6 or 8 columns is also used to write down a key with D_k = 128, 192 or 256 bits.

ENCRYPTION ALGORITHM

1. KeyExpansion
2. AddRoundKey
3. **do** (k + 5)-times:
   a) SubByte
   b) ShiftRow
   c) MixColumn
   d) AddRoundKey
4. Final round
   a SubByte
   b ShiftRow
   c AddRoundKey

The final round does not contain MixColumn procedure. The reason being is to be able to use the same hardware for encryption and decryption.

KEY EXPANSION

The basic key is written into the state matrix with 4, 6 or 8 columns.

KEY EXPANSION

The basic key is written into the state matrix with 4, 6 or 8 columns.
The goal of the key expansion procedure is to extend the number of keys in such a way that each time a key is used actually a new key is used.

KEY EXPANSION

The basic key is written into the state matrix with 4, 6 or 8 columns.
The goal of the key expansion procedure is to extend the number of keys in such a way that each time a key is used actually a new key is used.

The key extension algorithm generates new columns W_i of the state matrix from the columns W_{i-1} and W_{i-k} using the following rule

$$W_i = W_{i-k} \oplus V,$$

where

$$\mathrm{V} = \begin{cases} F(W_{i-1}), & \text{if i mod k} = 0 \\ G(W_{i-1}), & \text{if i mod k} = 4 \text{ and } D_k = 256 \text{ bits,} \\ W_{i-1} & \text{otherwise} \end{cases}$$

and where the function G performs only the byte-substitution of the corresponding bytes. **F**unction F is defined in a quite a complicated way.

DECRYPTION in AES

Steps of the encryption algorithm map an input state matrix into an output matrix. All encryption operations have inverse operations. Decryption algorithm applies, in the opposite order as at the encryption, the inverse versions of the encryption operations.

DECRYPTION

1. Key Expansion

DECRYPTION in AES

Steps of the encryption algorithm map an input state matrix into an output matrix. All encryption operations have inverse operations. Decryption algorithm applies, in the opposite order as at the encryption, the inverse versions of the encryption operations.

DECRYPTION

1. Key Expansion
2. AddRoundKey

DECRYPTION in AES

Steps of the encryption algorithm map an input state matrix into an output matrix. All encryption operations have inverse operations. Decryption algorithm applies, in the opposite order as at the encryption, the inverse versions of the encryption operations.

DECRYPTION

1. Key Expansion
2. AddRoundKey
3. **do** k+5 - times:
   a) InvSubByte
   b) InvShiftRow
   c) InvMixColumn
   d) AddInvRoundKey

DECRYPTION in AES

Steps of the encryption algorithm map an input state matrix into an output matrix. All encryption operations have inverse operations. Decryption algorithm applies, in the opposite order as at the encryption, the inverse versions of the encryption operations.

DECRYPTION

1. Key Expansion
2. AddRoundKey
3. **do** k+5 - times:
   a) InvSubByte
   b) InvShiftRow
   c) InvMixColumn
   d) AddInvRoundKey
4. Final round
   a) InvSubByte
   b) InvShiftRow
   c) AddInvRoundKey

SECURITY GOALS

The goal of the authors was that Rijndael (AES) is K-secure and hermetic in the following sense:

Definition A cryptosystem is K-secure if all possible attack strategies for it have the same expected work factor and storage requirements as for the majority of possible cryptosystems with the same security.

Definition A block cryptosystem is hermetic if it does not have weaknesses that are not present for the majority of cryptosystems with the same block and key length.

MISCELLANEOUS

Pronunciation of the name **Rijndael** is as “Reign Dahl’‘ or “rain Doll” or “Rhine Dahl”.

AES attacks

AES attacks

Best known is so called **Biclique attack**.

Best known is so called **Biclique attack**.

Complexity of the biclique attack

- AES-128 - $2^{126.1}$ - brute force (2^{128}).
- AES-192 - $2^{189.7}$ - brute force (2^{192})
- AES-256 - $2^{254.4}$ - brute force (2^{256})

Best known is so called **Biclique attack**.

Complexity of the biclique attack

- AES-128 - $2^{126.1}$ - brute force (2^{128}).
- AES-192 - $2^{189.7}$ - brute force (2^{192})
- AES-256 - $2^{254.4}$ - brute force (2^{256})

Best known is so called **Biclique attack**.

Complexity of the biclique attack

- AES-128 - $2^{126.1}$ - brute force (2^{128}).
- AES-192 - $2^{189.7}$ - brute force (2^{192})
- AES-256 - $2^{254.4}$ - brute force (2^{256})

Comment 1: Biclique is a complete bipartite graph - all nodes of which are connected to all potential neighbours.

Comment 2: For cryptographers, a cryptographic "break" is anything faster than a brute force.

VI. PKC versus SKC – comparisons

VI. PKC versus SKC – comparisons

Security: If PKC is used, only one party needs to keep secret its (single) key.

VI. PKC versus SKC – comparisons

Security: If PKC is used, only one party needs to keep secret its (single) key. If SKC is used, both party needs to keep secret one key.

VI. PKC versus SKC – comparisons

Security: If PKC is used, only one party needs to keep secret its (single) key. If SKC is used, both party needs to keep secret one key. No PKC has been shown to be perfectly secure.

VI. PKC versus SKC – comparisons

Security: If PKC is used, only one party needs to keep secret its (single) key. If SKC is used, both party needs to keep secret one key. No PKC has been shown to be perfectly secure. Perfect secrecy has been shown for One-time Pad and for quantum generation of classical keys.

VI. PKC versus SKC – comparisons

Security: If PKC is used, only one party needs to keep secret its (single) key. If SKC is used, both party needs to keep secret one key. No PKC has been shown to be perfectly secure. Perfect secrecy has been shown for One-time Pad and for quantum generation of classical keys.

Longevity: With PKC, keys may need to be kept secure for (very) long time; with SKC a change of keys for each session is recommended.

VI. PKC versus SKC – comparisons

Security: If PKC is used, only one party needs to keep secret its (single) key. If SKC is used, both party needs to keep secret one key. No PKC has been shown to be perfectly secure. Perfect secrecy has been shown for One-time Pad and for quantum generation of classical keys.

Longevity: With PKC, keys may need to be kept secure for (very) long time; with SKC a change of keys for each session is recommended.

Key management: If a multiuser network is used, then fewer private keys are required with PKC than with SKC.

VI. PKC versus SKC – comparisons

Security: If PKC is used, only one party needs to keep secret its (single) key. If SKC is used, both party needs to keep secret one key. No PKC has been shown to be perfectly secure. Perfect secrecy has been shown for One-time Pad and for quantum generation of classical keys.

Longevity: With PKC, keys may need to be kept secure for (very) long time; with SKC a change of keys for each session is recommended.

Key management: If a multiuser network is used, then fewer private keys are required with PKC than with SKC.

Key exchange: With PKC no key exchange between communicating parties is needed; with SKC a hard-to-implement secret key exchange is needed.

VI. PKC versus SKC – comparisons

Security: If PKC is used, only one party needs to keep secret its (single) key. If SKC is used, both party needs to keep secret one key. No PKC has been shown to be perfectly secure. Perfect secrecy has been shown for One-time Pad and for quantum generation of classical keys.

Longevity: With PKC, keys may need to be kept secure for (very) long time; with SKC a change of keys for each session is recommended.

Key management: If a multiuser network is used, then fewer private keys are required with PKC than with SKC.

Key exchange: With PKC no key exchange between communicating parties is needed; with SKC a hard-to-implement secret key exchange is needed.

Digital signatures: Only PKC are usable for digital signatures.

VI. PKC versus SKC – comparisons

Security: If PKC is used, only one party needs to keep secret its (single) key. If SKC is used, both party needs to keep secret one key. No PKC has been shown to be perfectly secure. Perfect secrecy has been shown for One-time Pad and for quantum generation of classical keys.

Longevity: With PKC, keys may need to be kept secure for (very) long time; with SKC a change of keys for each session is recommended.

Key management: If a multiuser network is used, then fewer private keys are required with PKC than with SKC.

Key exchange: With PKC no key exchange between communicating parties is needed; with SKC a hard-to-implement secret key exchange is needed.

Digital signatures: Only PKC are usable for digital signatures.

Efficiency: PKC is much slower than SKC (10 times when software implementations of RSA and DES are compared).

VI. PKC versus SKC – comparisons

Security: If PKC is used, only one party needs to keep secret its (single) key. If SKC is used, both party needs to keep secret one key. No PKC has been shown to be perfectly secure. Perfect secrecy has been shown for One-time Pad and for quantum generation of classical keys.

Longevity: With PKC, keys may need to be kept secure for (very) long time; with SKC a change of keys for each session is recommended.

Key management: If a multiuser network is used, then fewer private keys are required with PKC than with SKC.

Key exchange: With PKC no key exchange between communicating parties is needed; with SKC a hard-to-implement secret key exchange is needed.

Digital signatures: Only PKC are usable for digital signatures.

Efficiency: PKC is much slower than SKC (10 times when software implementations of RSA and DES are compared).

Key sizes: Keys for PKC (2048 bits for RSA) are significantly larger than for SCK (128 bits for AES).

VI. PKC versus SKC – comparisons

Security: If PKC is used, only one party needs to keep secret its (single) key. If SKC is used, both party needs to keep secret one key. No PKC has been shown to be perfectly secure. Perfect secrecy has been shown for One-time Pad and for quantum generation of classical keys.

Longevity: With PKC, keys may need to be kept secure for (very) long time; with SKC a change of keys for each session is recommended.

Key management: If a multiuser network is used, then fewer private keys are required with PKC than with SKC.

Key exchange: With PKC no key exchange between communicating parties is needed; with SKC a hard-to-implement secret key exchange is needed.

Digital signatures: Only PKC are usable for digital signatures.

Efficiency: PKC is much slower than SKC (10 times when software implementations of RSA and DES are compared).

Key sizes: Keys for PKC (2048 bits for RSA) are significantly larger than for SCK (128 bits for AES).

Non-repudiation: With PKC we can ensure, using digital signatures, non-repudiation, but not with SKC.

DIGITAL ENVELOPES

DIGITAL ENVELOPES

Modern cryptography uses both **SKC** and **PKC**, in so-called **hybrid cryptosystems** or in **digital envelopes**.

DIGITAL ENVELOPES

Modern cryptography uses both **SKC** and **PKC**, in so-called **hybrid cryptosystems** or in **digital envelopes**. To send a message **m**,

DIGITAL ENVELOPES

Modern cryptography uses both **SKC** and **PKC**, in so-called **hybrid cryptosystems** or in **digital envelopes**. To send a message **m**, using a secret key **k** chosen by the sender, using the public encryption exponent **e** of the receiver, and using the secret decryption exponent **d** of the receiver, the following steps have to be made:

DIGITAL ENVELOPES

Modern cryptography uses both **SKC** and **PKC**, in so-called **hybrid cryptosystems** or in **digital envelopes**. To send a message **m**, using a secret key **k** chosen by the sender, using the public encryption exponent **e** of the receiver, and using the secret decryption exponent **d** of the receiver, the following steps have to be made:

1. Key **k** is encrypted using **e** and sent as $E_e(k)$

DIGITAL ENVELOPES

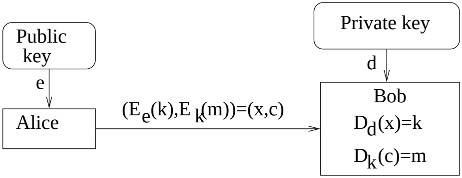
Modern cryptography uses both **SKC** and **PKC**, in so-called **hybrid cryptosystems** or in **digital envelopes**. To send a message **m**, using a secret key **k** chosen by the sender, using the public encryption exponent **e** of the receiver, and using the secret decryption exponent **d** of the receiver, the following steps have to be made:

1. Key **k** is encrypted using **e** and sent as $E_e(k)$
2. Secret description exponent **d** is used to get $k = D_d(E_e(k))$

DIGITAL ENVELOPES

Modern cryptography uses both **SKC** and **PKC**, in so-called **hybrid cryptosystems** or in **digital envelopes**. To send a message **m**, using a secret key **k** chosen by the sender, using the public encryption exponent **e** of the receiver, and using the secret decryption exponent **d** of the receiver, the following steps have to be made:

1. Key **k** is encrypted using **e** and sent as $E_e(k)$
2. Secret description exponent **d** is used to get $k = D_d(E_e(k))$
3. SKC with **k** is then used to encrypt a message



A variety of brute force and also sophisticated methods of attacks have been developed to deal with (also "complex" and without simple mathematical structures) encryption systems.

VIII. CRYPTANALYSIS

A variety of brute force and also sophisticated methods of attacks have been developed to deal with (also "complex" and without simple mathematical structures) encryption systems.

In case of brute force attacks, an important issue is their performance in case they are applied to any encryption algorithm considered as a black box.

A variety of brute force and also sophisticated methods of attacks have been developed to deal with (also "complex" and without simple mathematical structures) encryption systems.

In case of brute force attacks, an important issue is their performance in case they are applied to any encryption algorithm considered as a black box.

Security is measured in such cases in terms of such encryption parameters as the length of the key and the size of message space.

ATTACKS - BRUTE FORCE METHODS

We will discuss several types of brute force attacks that can be applied to any symmetric cryptosystem C_k considered as an oracle that for each given key as input replies whether it is a correct key.

ATTACKS - BRUTE FORCE METHODS

We will discuss several types of brute force attacks that can be applied to any symmetric cryptosystem C_k considered as an oracle that for each given key as input replies whether it is a correct key.

Exhaustive search

ATTACKS - BRUTE FORCE METHODS

We will discuss several types of brute force attacks that can be applied to any symmetric cryptosystem C_k considered as an oracle that for each given key as input replies whether it is a correct key.

Exhaustive search

This method consists of trying all possible keys exhaustively until the correct key is found.

ATTACKS - BRUTE FORCE METHODS

We will discuss several types of brute force attacks that can be applied to any symmetric cryptosystem C_k considered as an oracle that for each given key as input replies whether it is a correct key.

Exhaustive search

This method consists of trying all possible keys exhaustively until the correct key is found.

Exhaustive search can be made more efficient if a probability distribution on keys can be guessed or keys are known to satisfy some relations.

ATTACKS - BRUTE FORCE METHODS

We will discuss several types of brute force attacks that can be applied to any symmetric cryptosystem C_k considered as an oracle that for each given key as input replies whether it is a correct key.

Exhaustive search

This method consists of trying all possible keys exhaustively until the correct key is found.

Exhaustive search can be made more efficient if a probability distribution on keys can be guessed or keys are known to satisfy some relations.

Dictionary attack

Creation of dictionary: For a fixed x and many k, values $C_k(x)$ are computed and pairs $(C_k(x), k)$ are inserted into a dictionary that is ordered according to the first item of each pair.

Search If we obtain a $C_k(x)$ value (by a chosen plaintext attack), dictionary gives us a list of potential keys.

ATTACKS - BRUTE FORCE METHODS

We will discuss several types of brute force attacks that can be applied to any symmetric cryptosystem C_k considered as an oracle that for each given key as input replies whether it is a correct key.

Exhaustive search

This method consists of trying all possible keys exhaustively until the correct key is found.

Exhaustive search can be made more efficient if a probability distribution on keys can be guessed or keys are known to satisfy some relations.

Dictionary attack

Creation of dictionary: For a fixed x and many k, values $C_k(x)$ are computed and pairs $(C_k(x), k)$ are inserted into a dictionary that is ordered according to the first item of each pair.

Search If we obtain a $C_k(x)$ value (by a chosen plaintext attack), dictionary gives us a list of potential keys.

A generalization for searching for several keys having several values $C_k(x)$ is easy.

SOPHISTICATED ATTACK METHODS

Two main attack methods for general cryptographic algorithms are **differential cryptanalysis** and **linear cryptanalysis**

SOPHISTICATED ATTACK METHODS

Two main attack methods for general cryptographic algorithms are **differential cryptanalysis** and **linear cryptanalysis**

Differential cryptanalysis: It is assumed that adversary can use the encryption devise as a black box, submitting chosen plaintexts and getting corresponding cryptotext.

SOPHISTICATED ATTACK METHODS

Two main attack methods for general cryptographic algorithms are **differential cryptanalysis** and **linear cryptanalysis**

Differential cryptanalysis: It is assumed that adversary can use the encryption devise as a black box, submitting chosen plaintexts and getting corresponding cryptotext.
The aim of the attack is to get the key.

SOPHISTICATED ATTACK METHODS

Two main attack methods for general cryptographic algorithms are **differential cryptanalysis** and **linear cryptanalysis**

Differential cryptanalysis: It is assumed that adversary can use the encryption devise as a black box, submitting chosen plaintexts and getting corresponding cryptotext.
The aim of the attack is to get the key.

The basic idea: pairs of random plaintexts are submitted the difference of which is a fixed value a until differences of corresponding cryptotext are at most a fixed value b.

SOPHISTICATED ATTACK METHODS

Two main attack methods for general cryptographic algorithms are **differential cryptanalysis** and **linear cryptanalysis**

Differential cryptanalysis: It is assumed that adversary can use the encryption devise as a black box, submitting chosen plaintexts and getting corresponding cryptotext.
The aim of the attack is to get the key.

The basic idea: pairs of random plaintexts are submitted the difference of which is a fixed value a until differences of corresponding cryptotext are at most a fixed value b.

Linear analysis: This is a dual method to differential cryptanalysis invented after discovering anomalies in S-boxes in DES. The idea is not to keep track of difference propagation by the chosen plaintext attack, but to keep track of Boolean information which is linearly obtained by a known plaintext attack: if one gets $(x, c(x))$ pair a statistical analysis of the special Boolean information $L(x, c(x))$ is made and some information on the key is deduced.

ANALYSIS of ATTACKS on AES

ANALYSIS of ATTACKS on AES

So far there have appeared several attacks on AES that are faster than brute force, but only by a minor factor and none of them is feasible.

So far there have appeared several attacks on AES that are faster than brute force, but only by a minor factor and none of them is feasible.

For AES-128 (AES-192) [AES-256] the key can be recovered with a computational complexity $2^{126.1}$ ($2^{189.7}$) [$2^{254.4}$].

APPENDIX