Symmetric Cryptography

Stanislav Palúch

Fakula riadenia a informatiky, Žilinská univerzita

25. októbra 2017

General Principle of Symmetric Cryptography

1. A and B make an agreement about cryptosystem
2. A and B make an agreement about key K
3. A (resp. B) encipheres a plaintext x as $y = E_K(x)$
4. B (resp. A) decipheres a ciphertext y as $x = D_K(y)$

Feistel ciphers

A Feistel cipher is a structure used in the construction of symmetric block ciphers, named after the German-born physicist and cryptographer Horst Feistel.
A large proportion of block ciphers use the Feistel scheme e.g. Ameican DES and Russian GOST.

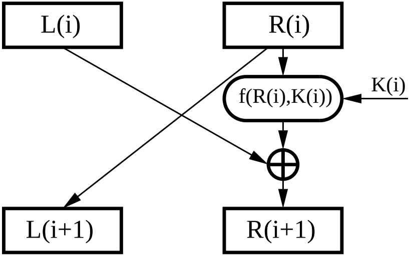
Feistel cipher enciphers a block of plaintext. A block should to have an even number of bits since it will be divided into two parts with the same number of bits.

A Feistel network is an iterated cipher with an internal function called a round function.

A round function processes input left and right part of enciphered text into new output left and right part which are used as input parts in subsequent round.

Round Function of Feistel Cipher

Block is divided ito two parts – left L_i and right R_i.
Every round makes use of its round key K_i, which enters along with i-th right part into a round function f.
Round function f is the same for all rounds

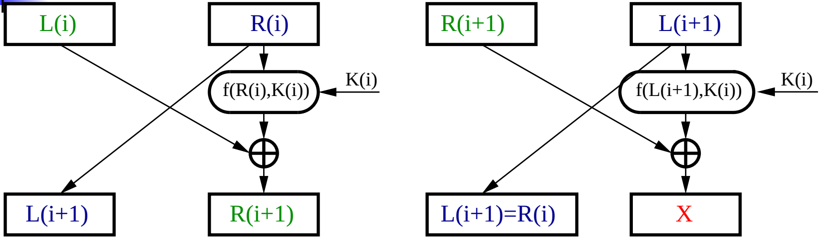


One round makes:

$$\begin{aligned} R_{i+1} &= L_i \oplus f(R_i, K_i) \\ L_{i+1} &= R_i \end{aligned}$$

Notice that output left part $L(i+1)$ of a round is a copy of input right part $R(i)$.

Deciphering

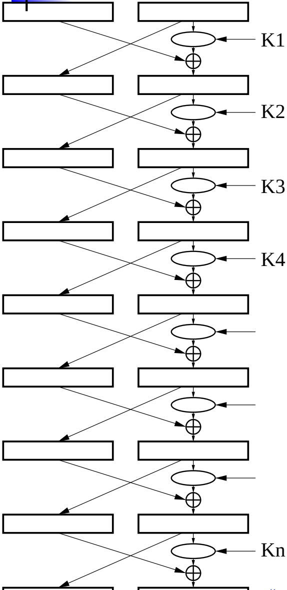


Let us calculate X.

$$X = \underbrace{R_{i+1}}_{=L_i \oplus f(R_i,K_i)} \oplus f(\underbrace{L_{i+1}}_{=R_i}, K_i) = L_i \oplus \underbrace{f(R_i,K_i) \oplus f(R_i,K_i)}_{=0} = L_i$$

Colorary: If a round alorithm uses round key K_i, and is applied with L_{i+1} on the right input and R_{i+1} on the left input, then we get on its left output an right output orinal L_i a R_i.
The same round algorithm with swapped left and right sides can be used as an inverse function.

Feistel Network



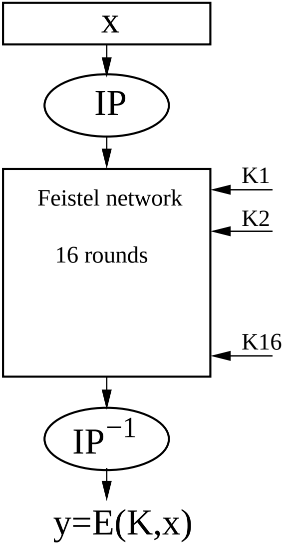
Feistel network is an iterated multifod repeating of round keys every one with another round key $K_1, K_2, \ldots, K_n$.

Deciphering is executed with the same network, applicated on ciphertext with swapped left and right part and inverse order of round keys $K_n, K_{n-1}, \ldots, K_1$.

Important: Just described inverse mechanism does not depend on the type of function $f(R_i, K_i)$.

However, function $f(R_i, K_i)$ significantly affects cryptographic properties of Feistel network.

DES – Data Encryption Standard



- Deigned in IBM, published in 1975
- Block cipher – uses 64-bit block of plaintext
- Uses 56-bit key
- Type – a Feistel network with 16 rounds and with input and output permutation
- IP – input permutation
- IP^{-1} – output permutation

Input and output permutation have no influence on security of cryptosystem.

DES – Input and Output Permutation

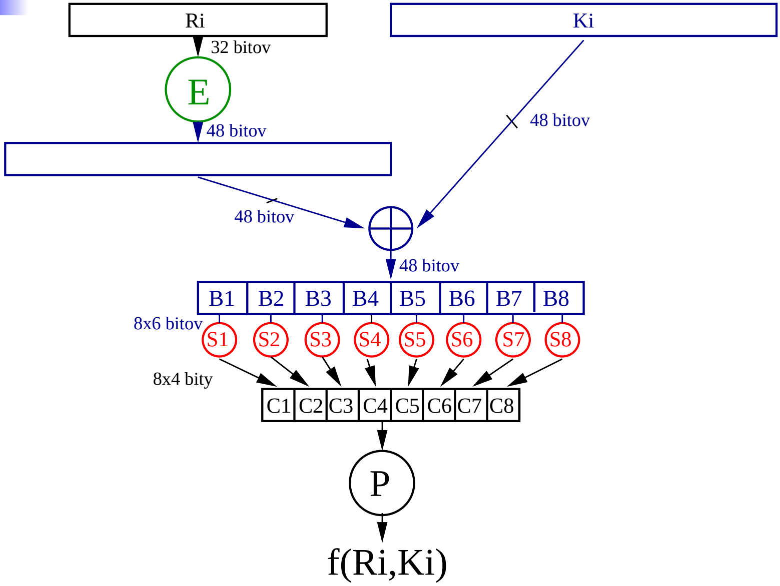
Table 12.1 Initial Permutation

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

Table 12.8 Final Permutation

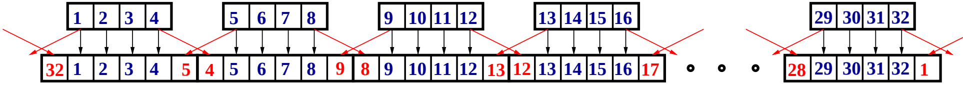
40	8	48	16	56	24	64	32	39	7	47	15	55	23	63	31
38	6	46	14	54	22	62	30	37	5	45	13	53	21	61	29
36	4	44	12	52	20	60	28	35	3	43	11	51	19	59	27
34	2	42	10	50	18	58	26	33	1	41	9	49	17	57	25

DES – Function f in DES

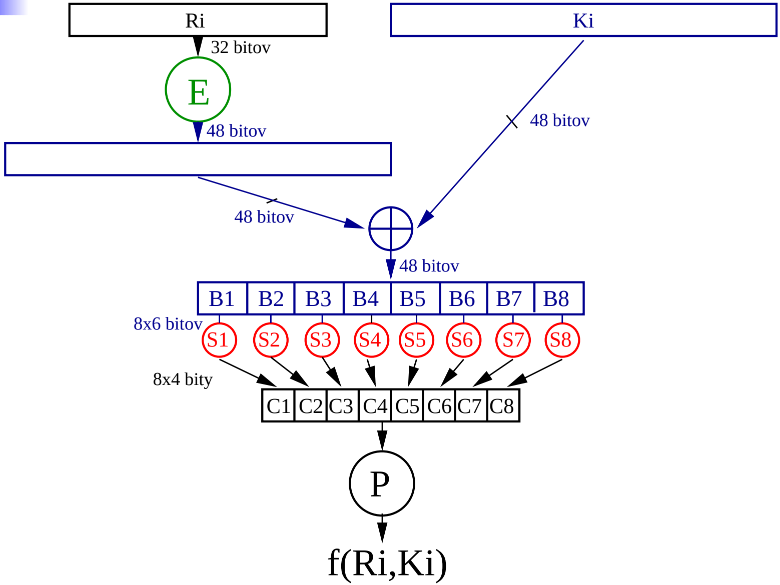


DES – Expansion Operation

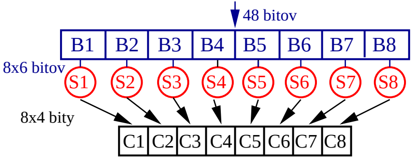
32	1	2	3	4	5
4	5	6	7	8	9
8	9	10	11	12	13
12	13	14	15	16	17
16	17	18	19	20	21
20	21	22	23	24	25
24	25	26	27	28	29
28	29	30	31	32	1



DES – Function f in DES



DES – Using S-boxes



- A S-box is a table with 4 rows and 16 columns.
- Rows are numbered by indices from 0 to 3, columns are numbered by numbers from 0 to 15.
- DES uses 8 S-boxes, S-box S_i is assigned to block B_i.
- Every B_i is a 6-bit number $b_1 b_2 b_3 b_4 b_5 b_6$ and represents an address of corresponding 4-bit number C_i in S-box S_i.

DES – Adressing in a S-box

Adress is calculated as follows:

Let $B_1 = b_1 b_2 b_3 b_4 b_5 b_6$.

$b_1 b_6$ is the number of row and $b_2 b_3 b_4 b_5$ is the number of column in corresponding S-box.
(Rows resp. columns are numbered from 0 to 3 resp. from 0 to 15.)

S-box 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

Example:
$B_1 = 101011$. $b_1 b_6 = (11)_2 = 3$, $b_2 b_3 b_4 b_5 = (0101)_2 = 5$.
S-box S_1 contains in row 3 and column 5 number 9 (attention, rows and columns are numbered from 0). Binary equivalent of 9 is 1001.
Therfore

$$S_1(B_1) = S_1(101011) = 1001 = C_1.$$

DES – S-boxes 2, 3, 4

S-box 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

S-box 3:

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

S-box 4:

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

DES – S-boxes 5, 6, 7, 8

S-box 5:

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

S-box 6:

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

S-box 7:

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

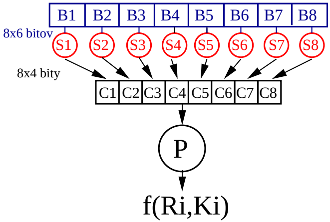
S-box 8:

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

DES – Final Permutation of Round Function

Table 12.7 P-Box Permutation

16	7	20	21
29	12	28	17
1	15	23	26
5	18	31	10
2	8	24	14
32	27	**3**	9
19	13	30	6
22	11	**4**	25

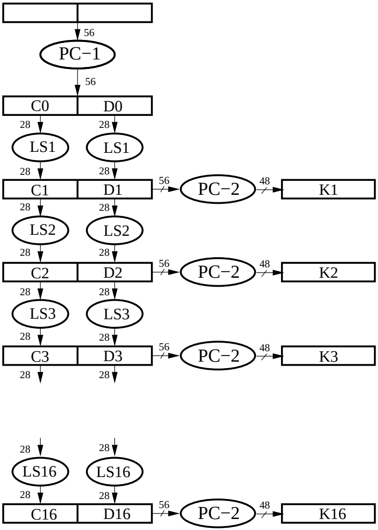


16	**7**	20	21
29	12	28	17
1	15	23	26
5	18	31	10
2	**8**	24	14
32	27	3	9
19	13	30	**6**
22	11	4	25

16	7	20	21
29	**12**	28	17
1	15	23	26
5	18	31	**10**
2	8	24	14
32	27	3	**9**
19	13	30	6
22	**11**	4	25

16	7	20	21
29	12	28	17
1	**15**	23	26
5	18	31	10
2	8	24	**14**
32	27	3	9
19	**13**	30	6
22	11	4	25

DES – Generation of Round Keys



Key for system DES is 56-bits long. Key is saved as 64 bits arranged in 8 bytes, every byte contains 7 bits of key and one parity bit completing number of ones to even number.

Round key generation procedure:

56 bits of key are gained after removing parity bits.

1. Order of those bits will be chained by permutation PC-1.

2. Then 56 bits of key are divided into two 28-bit parts C_0, D_0.

3. Round key K_i is computed as follows: **3a.** Apply left circular shift LS_i on C_{i-1} and on D_{i-1} with result C_i, D_i.

LS_i is left circular shif by one digit for $i = 1, 2, 9, 16$ otherwise by two digits.

3b. Apply operation PC-2 on 56-bit word C_iD_i. Operation PC-2 chooses and permutates 48 bits from C_iD_i with result used as round key K_i.

DES – Permutation PC-1 and Mapping PC-2

Permutation PC-1

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

Mapping PC-2

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

DES – Design Criteria for S- boxes

The only nonlinearity fo cipher DES is contained in S-boxes. Security of Des depend only on proper design of S-boxex.

1. Everey row is a permutation of numbers $0-15$.
2. No S-box is a linear or affine function of its inputs
3. Changing of one input bit of S-boxu causes the change at least two bits of output.
4. x
   $S(x)$ and $S(x \oplus 001100)$ differ at least at two bits fro every S-box a for every 6-bit x.
5. It holds $S(x) \neq S(x \oplus 11rs00)$ for every S-box, every 6-bit x and arbitrary bits $r, s \in \{0, 1\}$.
6. If we fix one output bit, then the number of input values, with this input is equal to 0 (or equal to 1), falls between 13 and 19.

Attack against DES

Brute force attack – ciphertext only attack.
The number of keys 2^{56} shows to be small in present days.
RSA announced a public challenge to crack the DES encryption algorithm in January 1997 with 10 thousands dollars prize.
Four months later, the DES encryption key was found by exhousted search using the collective resources and computing power of literally thousands of computers.

Differential attack.
This is an instance of "chosen plaintext attack".
Couples of plaintexts P_1, P_2 with certain difference $P_1 \oplus P_2$ are enciphered and some information about key is deduced from the differences $C_1 \oplus C_2$ of corresponding ciphertexts.

Linear Cryptoanalysis.
If it holds for plaintext $x_1x_2\dots x_{64}$, key $k_1k_2\dots k_{56}$ and corresponding ciphertext $y_1y_2\dots y_{64}$:

$$\bigoplus_{i=1}^{64} a_ix_i \oplus \bigoplus_{i=1}^{64} b_iy_i = \bigoplus_{i=1}^{56} c_ik_i$$

with probability different from $\frac{1}{2}$, this fact can be explited for cryptanalysis.
It hold for DES:

$$x_{17} \oplus y_3 \oplus y_8 \oplus y_{14} \oplus y_{25} = K_{i,26}$$

with probability $\dfrac{1}{2} - \dfrac{5}{16} = \dfrac{3}{16}$.
A chosen plaintext attack against DES was designed on the basis of this fact. This attack analyses on averige 2^{43} known plaintexts, and succeeded to reveal key in 50 days of work of 12 computers HP9735 in 1994.

Attampts to Lengsten the Key

The simplest way how to enlarge the key is to use double enciphering first with key K_1 and the with key K_2 instead of encipherig with a single key.

$$\text{šifrujeme: } y = E_{K_2}\left[E_{K_1}(x)\right] \quad \text{dešifrujeme: } x = D_{K_1}\left[D_{K_2}(y)\right]$$

However, if enciphering and deciphering operation would create a group then there would exist a key K_3 for every K_1, K_2 such that $E_{K_2}\left[E_{K_1}\right] = E_{K_3}$. In this case a double enciphering would have no sense.

Here are several examples of ciphers that are groups:

- Ceasar cipher
- Affine cipher
- General monoalphabetic cipher
- Hill cipher

However, there are several conjectures that DES is not a group.

Suppose that we know a couple x, y of a plaintext and ciphertext enciphered by pair of keys K_1, K_2, i.e.
$y = E_{K_2}\left[E_{K_1}(x)\right]$. Then
$D_{K_2}(y) = D_{K_2}\left\{E_{K_2}\left[E_{K_1}(x)\right]\right\} = E_{K_1}(x)$
We are searching for a pair of keys K_1, K_2, such that

$$D_{K_2}(y) = E_{K_1}(x).$$

We create two tables –
Table 1. containing dependace $E_{K_1}(x)$ ona K_1 and
Table 2. containing dependace $D_{K_2}(y)$ on K_2.
If we find such entry in second colmumn of Table 1. which equals to some entry of second column of Table 2. then keys in correspnding rows are candidates on keys K_1, K_2.

K_1	$E_{K_1}(x)$
0	
1	
2	
L_1	z
$2^{56}-1$	

K_2	$D_{K_2}(y)$
0	
1	
2	
L_2	z
$2^{56}-1$	

Complexity of Meet-in-the-Middle

Just proposed procedure can be made simpler in such a way, that we will first create and store only Table 1. Then we will gererate $D_{K_2}(y)$ for $K_2 = 0, 1, \ldots$ and search its occurence in the second column of Table 1.

Memory requirements: 2^n ($= 2^{56}$) rows of Table 1.

Time requirements:
2×2^n $(= 2 \times 2^{56})$ encodings plus
$2^n . \log_2 2^n = n.2^n$ $(= 56.2^{56})$ steps to sort Table 1. by second column
and at most $2^n . \log_2 2^n = n.2^n$ $(= 56.2^{56})$ steps for searching in Table 1.
Together: $2.2^n + n.2^n + n.2^n = (2 + 2n)2^n = (1 + n).2^{n+1}$ $(=57.2^{57})$.

There are even more effective attacks.

Exhausted search for revealing combination of two keys K_1, K_2 requires in worst case 2^{2n} $(= 2^{112})$ encodings.

Colorary: Double enciphering does not awaited strengthening of cipher.

3DES

Enciphering: $y = E_{K_3}\left\{D_{K_2}\left[E_{K_1}(x)\right]\right\}$ Deciphering: $y = D_{K_1}\left\{E_{K_2}\left[D_{K_3}(x)\right]\right\}$

or

Enciphering: $y = E_{K_1}\left\{D_{K_2}\left[E_{K_1}(x)\right]\right\}$ Deciphering: $y = D_{K_1}\left\{E_{K_2}\left[D_{K_1}(x)\right]\right\}$

GOST

The GOST block cipher is a Soviet and Russian government standard symmetric key block cipher with a block size of 64 bits.

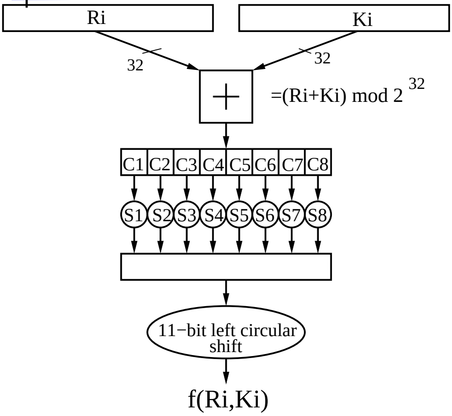
The new standard also specifies a new 128-bit block cipher called Kuznyechik.

GOST was developed in the 1970s. The standard had been marked Top Secret.

Shortly after the dissolution of the USSR, it was declassified and it was released to the public in 1994.

GOST was a Soviet alternative to the United States standard algorithm DES.

GOST



Soviet and Rusian cryptosystem used in period of cold war.

Block cipher.

64-bit block, 256-bit key.

Feistel network with 32 rounds.

S-boxes are one row tables containing permutations of numbers $0, 1, \ldots, 15$.

S-boxes of GOST

S-box 1:

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

S-box 2:

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

S-box 3:

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

S-box 4:

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

S-box 5:

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

S-box 6:

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

S-boxy kryptosystému GOST

S-box 7:

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

S-box 8:

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

Round Keys Generation

GOST uses 256-bit key. It can be devided into eight 32-bit keys $K_1, K_2, \ldots, K_8$.

K1	K2	K3	K4	K5	K6	K7	K8

Those are used in the following order:

$K_1, K_2, \ldots, K_8, K_1, K_2, \ldots, K_8, K_1, K_2, \ldots, K_8, K_8, K_7, \ldots K_1$

IDEA

IDEA – International Data Encryption Algorithm (Xueija Lai and James Massey) – 1992.
IDEA is patented, US patent expired 7.1.2012.

Block cipher – 64-bit blok
Key 128-bit.

64- bit block is divided into 4 16-bit parts x_1, x_2, x_3, x_4, which will be processed in 8 rounds of algorithm plus final half round.

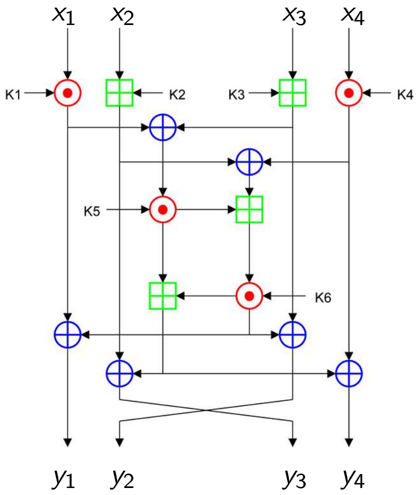
Rounds use the following operations:

$\oplus$ – bitwise XOR

$\boxplus$ – adding mod 2^{16}

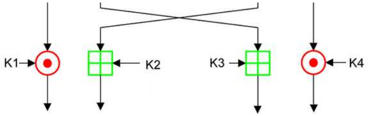
$\odot$ – multiplication mod $(2^{16}+1)$ while 16-bit word consisting of all 0 is taken as reprezentation of the number 2^{16}.

One Round of Algorithm IDEA



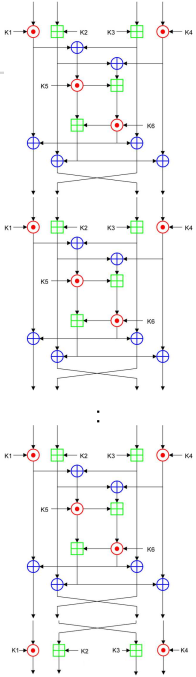
IDEA – Generation of Round Keys

Final Half Round



Generation of Round Keys

Every round needs 6 keys and the final half round needs 4 keys, i.e. together $6 * 8 + 4 = 52$ 16-bit keys. 128 bit key will first divided into first 8 16-bit round keys.
Then left circular shift by 25 bits is applied to 128 bits of key and further 8 16-bit round keys are gained. Key is again rotated by circular shif by 25 bits and next 8 round keys are generated. Etc.



Deciphering

The same algorithm is used also for deciphering with the only difference that instead of the sequence of round keys $K_1, K_2, \dots K_{52}$ the sequence of inverse values resp. opposite values of keys $K_{52}, K_{51}, \dots, K_1$ is used.

Opetional Modes of Block Ciphers

Let us have a block cipher with enciphering function $y = E_K(x)$ and deciphering function $x = D_K(y)$.
We have a plaintext represented as a sequence of blocks:

$$x_1, x_2, \dots, x_n$$

There are several ways how to create corresponding sequence of blocks of ciphertext

$$y_1, y_2, \dots, y_n$$

using enciphering function $E_K(x)$ in such a way, that it is possible to reconstruct original plaintext

$$x_1, x_2, \dots, x_n$$

using deciphering mapping $D_K(y)$.
Those ways are called operational modes of block ciphers.

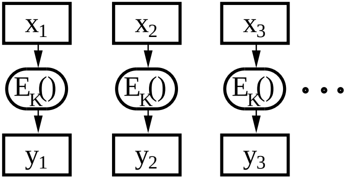
ECB – Electronic Code Book mód

ECB mode is the simplest way where a plaintext is enciphered by formula

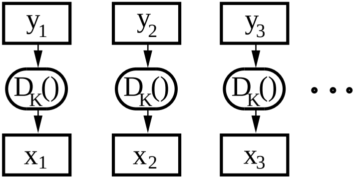
$$y_i = E_K(x_i)$$

and deciphered as

$$x_i = D_K(y_i)$$



Enciphering in ECB mode



Deciphering in ECB mode

Disadvantage of ECB mode:
The same block x_i of plaintext is enciphered every time into the same block of ciphertext what makes some attacks easier.

OFB – Output Feedback Mode

OFB – Output Feedback Mode

This mode requires first to choose a random initial block IV called also initial vector, set $y_0 = IV$.
Then z_1 is calculated as $z_1 = E_K(y_0)$, and recurently $z_{i+1} = E_K(z_i)$.

Enciphering procedure is

$$y_i = z_i \oplus x_i$$

Enciphered message is the sequence $\color{red}{y_0}, y_1, y_2, \ldots, y_n$ (it is one block longer then the original message).
Deciphering procedure is

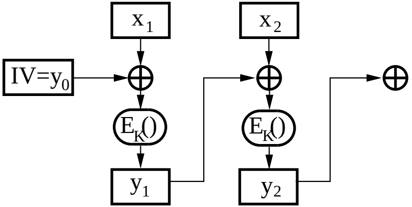
$$x_i = z_i \oplus y_i.$$

This mode is in fact a stream cipher with key stream $z_1, z_2, \ldots, z_n$, therefore it is necessary to use every time another initial vector.

Cipher Block Chaining Mode

Enciphering procedure is

$$y_i = E_K(x_i \oplus y_{i-1})$$



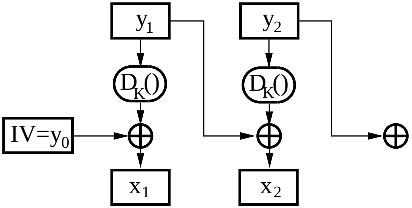
Eciphered message is the sequence

$$\color{red}{y_0}, y_1, y_2, \ldots, y_n$$

(it is one block longer than the original message).

Deciphering procedure is

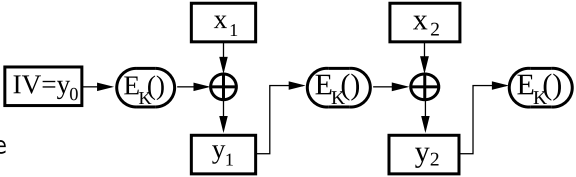
$$x_i = y_{i-1} \oplus D_K(y_i).$$



Cipher Feedback Mode

Enciphering procedure is

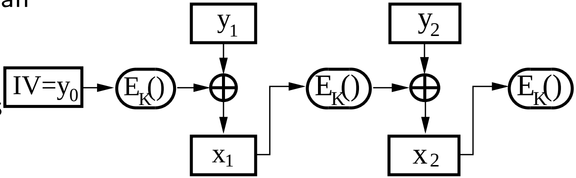
$$y_i = E_K(y_{i-1}) \oplus x_i$$



Eciphered message is the sequence

$$\color{red}{y_0}, y_1, y_2, \dots, y_n$$

(it is one block longer than the original message).



Deciphering procedure is

$$x_i = y_i \oplus E_K(y_{i-1}).$$

Galois field $GF(2^8)$

Évariste Galois (25.10. – 31.5.1832) was a French mathematician. His work laid the foundations for Galois theory and group theory, two major branches of abstract algebra. He died at age 20 from wounds suffered in a duel.

Elements of $GF(2^8)$ are polynomials of the type

$$b_7x^7 + b_6x^6 + b_5x^5 + b_4x^4 + b_3x^3 + b_2x^2 + b_1x^1 + b_0$$

in coefficients in $\mathbb{Z}_2$.

Such polynomial models a byte $b_7b_6b_5b_4b_3b_2b_1b_0$. For example $\{0\ 1\ 0\ 1\ 0\ 1\ 1\ 1\}$ corresponds to polynomial $x^6 + x^4 + x^2 + x + 1$.

Addition in $GF(2^8)$ is addition of polynomials over $\mathbb{Z}_2$.

$(x^6 + x^4 + x^2 + x + 1) + (x^7 + x^6 + x^4 + x^2) = (x^7 + x + 1)$
$\{0\ 1\ 0\ 1\ 0\ 1\ 1\ 1\} \oplus \{1\ 1\ 0\ 1\ 0\ 1\ 0\ 0\} = \{1\ 0\ 0\ 0\ 0\ 1\ 1\}$
In hexadecimal notatione $(57)_H \oplus (D4)_H = (83)_H$.

Byte addition $\oplus$ corresponds to computer operation bitwise XOR.

Multiplication in $GF(2^8)$ is defined as

$$p(x) \otimes q(x) = p(x).q(x) \mod m(x),$$

where $m(x)$ je irreducible polynomial of degree 8 over $GF(2^8)$.

AES uses this irreducible polynomial

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

Example.

$$\Big(\underbrace{(x^6 + x^4 + x^2 + x + 1)}_{57_H=\{01010111\}}.\underbrace{(x^7 + x + 1)}_{83_H=\{10000011\}}\Big) \mod \underbrace{(x^8 + x^4 + x^3 + x + 1)}_{=m(x)} =$$

$$(x^{13} + x^{11} + x^9 + x^8 + x^7 + x^6 + x^5 + x^4 + x^3 + 1) \mod m(x) =$$
$$= \underbrace{(x^7 + x^6 + 1)}_{C1_H=\{11000001\}}$$

Therefore it holds in $GF(2^8)$:

$$\{01010111\} \otimes \{10000011\} = \{11000001\}$$

$$57_H \otimes 83_H = C1_H$$

AES – Multiplication by Number $2 \equiv \{00000010\} \equiv x$

The following text is devoted to efficient computer implementation of multiplication in alois Field $GF(2^8)$ where its elements are represented by bytes.
Polynomial x corresponds to byte $\{00000010\}$, i.e. to the number $2 = (02)_H$. Let us examine $\{00000010\} \otimes b$.

Let
$b(x) = b_7x^7 + b_6x^6 + b_5x^5 + b_4x^4 + b_3x^3 + b_2x^2 + b_1x^1 + b_0$.
Then
$x.b(x) = b_7x^8 + b_6x^7 + b_5x^6 + b_4x^5 + b_3x^4 + b_2x^3 + b_1x^2 + b_0x$

If $b_7 = 0$, then $x.b(x) \mod m(x) = x.b(x)$,
where $m(x) = x^8 + x^4 + x^3 + x + 1$.
This operation is left shift of the byte b by 1 bit.

AES – Násobenie $\mathbf{a} \otimes \mathbf{b}$

If $b_7 = 1$, then

$$x.b(x) \mod m(x) = x.b(x) \ominus m(x) = x.b(x) \oplus m(x).$$

This operation can be executed by left shift of the byte b by 1 bit followed by bitwise XOR with byte $\{00011011\}$ (hexadecimal $(1B)_H$). Following function executes multiplication of **b** by 2:

```
xtime(b)
        1. if (b[7] == 1) t=00011011 else t=00000000;
        2. for(i=7 to 1) b[i]=b[i-1];
        3. b = b ⊕ t;
        4. return b;
```

Multiplication $\mathbf{a} \otimes \mathbf{b} = \mathbf{c}$ is realized as follows:

```
        1. c=00000000;
            p = a;
        2. for(i=0 to 7);
               if(b[i] == 1) c = c ⊕ p;
               p=xtime(p);
        3. return c;
```

AES – Computation of Inverse of b^{-1}

$GF(2^8)$ together with operations $\oplus$, $\otimes$ creates a finite field in which

- nulll element is 0 — polynomial – 00000000
- unit element is 1 – $00000001 \equiv 0x^7 + 0x^6 + \cdots + 0x + 1$
- for every element b the exists an opposite eldment – it is $b$by himself,
- for every element $b \neq 0$ there exists an inverse element b^{-1}.

Inverse element can be calculated by extended Euclidean algorithm. However, for usage in AES it suffices to calculate table of binary operation $\otimes$ (it has dimensions 256×256) and to find that c, for every $b = 1, 2, \ldots, 255$ for which it hodls $b \otimes c = 1$, and the to set $b^{-1} = c$.

If we create an array INVERSE[0..255] with 256 entries of the type

0	1	2^{-1}	3^{-1}	...	...	255^{-1}

then we obtain the inverse element b^{-1} to element b as INVERSE[b] – element of array INVERSE[ ] with index b.

AES – Advanced Encryption Standard – History

- 1997 – initialisation of the process of choosing a new cryptographic algorithm – NIST
(National Institute of Standards and Technology - USA)
- 15 algorithms were taking part in competition
- Vincent Rijmen (1970) a Joan Daemen (1965) (Belgicko) published algorithm Rijndael in 1998
- Rijndael – later named as AES – became effective as a federal government standard on May 26, 2002, after five-year standardization process and after approval by the Secretary of Commerce. [1], NSA[2]
- AES is the only public enciphering algorithm approved by NSA for top secret informations.

[1] FIPS – Federal Information Processing Standard)
[2] NSA – National Security Agency

AES – Advanced Encryption Standard – Advantages

Advantages of AES:

- – High effectivity and speed both in hardware and software implementation
- – Low memory requirements
- – Possibility of protections against attack throgh side chanals

AES - Advanced Encryption Standard – Špecifikácia

- Symmetric block cipher
- Block lengthh: 128 bits
- Key length: optional 128, 192 or 256 bits

128-bit block of plaintext is considered as a 16-membered sequence of 8-bit bytes:

$$a_{00}\,a_{10}\,a_{20}\,a_{30}\,a_{01}\,a_{11}\,a_{21}\,a_{31}\,a_{02}\,a_{12}\,a_{22}\,a_{32}\,a_{03}\,a_{13}\,a_{23}\,a_{33}$$

which are arranged into tables called a **state**.

a_{00}	a_{01}	a_{02}	a_{03}
a_{10}	a_{11}	a_{12}	a_{13}
a_{20}	a_{21}	a_{22}	a_{23}
a_{30}	a_{31}	a_{32}	a_{33}

State

k_{00}	k_{01}	k_{02}	k_{03}
k_{10}	k_{11}	k_{12}	k_{13}
k_{20}	k_{21}	k_{22}	k_{23}
k_{30}	k_{31}	k_{32}	k_{33}

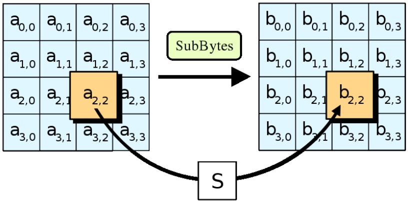
Round key

This state is processed by several rounds of operations. Some of them are dependant on round key which is also represented as a matrix of bytes.

AES - Operation SubBytes

Two operations are executed with every byte a of matrix State

1. First an inverse element $x = a^{-1}$ to a in $GF(2^8)$ is found if $a \neq 0$. If $a = 0$, then $x = 0$.
2. Then byte $b = b_0, b_1, b_2, b_3, b_4, b_5, b_6, b_7$ is calculated as follows:



$$
\begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \end{bmatrix} =
\begin{bmatrix}
1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\
1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\
1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\
1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\
1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\
0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\
0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\
0 & 0 & 0 & 1 & 1 & 1 & 1 & 1
\end{bmatrix} \cdot
\begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{bmatrix} +
\begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}
$$

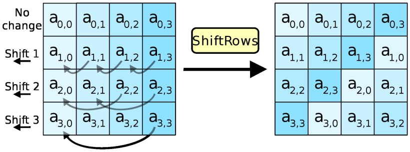
AES – Table of Function SubByte

		y															
		0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
x	0	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
	1	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
	2	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
	3	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
	4	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
	5	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
	6	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
	7	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
	8	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
	9	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
	a	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
	b	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
	c	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
	d	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
	e	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
	f	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

AES - Operation ShiftRows

Following left circular shift ar apllied on rows of State

1. 1. row remines unchanged
2. 2. row - shift by 1 byte - i.e. 8 bits
3. 3. row - shift by 2 bytes - i.e. 16 bits
4. 4. row - shift by 3 bytes - i.e. 24 bits



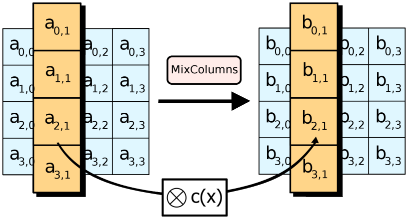
AES- Operation MixColumns

This operation consideres table State as a matrix of elements of field $GF(2^8)$. Every column of matrix State will be changed as follows:

$\mathbf{a}_i = \begin{bmatrix} a_{0i} & a_{1i} & a_{2i} & a_{3i} \end{bmatrix}^T$ vykonáme

$$\underbrace{\begin{bmatrix} b_{0i} \\ b_{1i} \\ b_{2i} \\ b_{3i} \end{bmatrix}}_{\mathbf{b}_i} = \underbrace{\begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix}}_{\mathbf{M}} \underset{GF(2^8)}{\bigotimes} \underbrace{\begin{bmatrix} a_{0i} \\ a_{1i} \\ a_{2i} \\ a_{3i} \end{bmatrix}}_{\mathbf{a}_i} \qquad \text{t. j.} \quad \mathbf{b}_i = \mathbf{M} \otimes \mathbf{a}_i$$

This operation can be executed as single matrix operation: $\mathbf{B} = \mathbf{M} \otimes \mathbf{A}$



$$\mathbf{M}^{-1} = \begin{bmatrix} 0e & 0b & 0d & 09 \\ 09 & 0e & 0b & 0d \\ 0d & 09 & 0e & 0b \\ 0b & 0d & 09 & 0e \end{bmatrix}$$

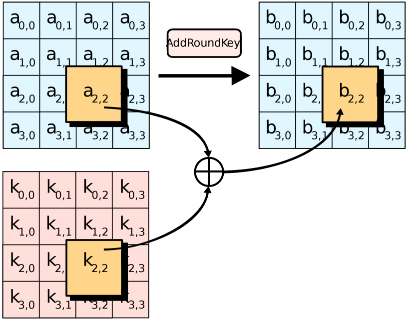
AES – FunkctionAddRoundKey

This operations XORs every a_{ij} element of State with entry k_{ij} of round key matrix $\mathbf{K}$ with the same indices

$$b_{ij} = a_{ij} \oplus k_{ij},$$

In matrix notation:

$$\mathbf{B} = \mathbf{A} \oplus \mathbf{K}.$$



AES – Enciphering Algorithm

1. Initial round
   1.1 AddRoundKey
2. for $Round = 1$ to $N_r - 1$
   2.1 SubBytes
   2.2 ShiftRows
   2.3 MixColumns
   2.4 AddRoundKey
3. Final round (without MixColumns)
   3.1 SubBytes
   3.2 ShiftRows
   3.3 AddRoundKey

Key length	128	192	256
Number of rounds N_r	10	12	14

AES – Deciphering

It should to be:

1. Initial round
   - *1.1* AddRoundKey
   - *1.2* InvShiftRows
   - *1.3* InvSubBytes
2. for $Round = 1$ to $N_r - 1$
   - *2.1* AddRoundKey
   - *2.2* InvMixColumns
   - *2.3* InvShiftRows
   - *2.4* InvSubBytes
3. Final round
   - *3.3* AddRoundKey

It is:

1. Initial round
   - *1.1* AddRoundKey
2. for $Round = 1$ to $N_r - 1$
   - *2.1* InvSubBytes
   - *2.2* InvShiftRows
   - *2.3* InvMixColumns
   - *2.4* AddRoundKey
3. Final round
   - *3.1* InvSubBytes
   - *3.2* InvShiftRows
   - *3.3* AddRoundKey

The order of operations InvShiftRows and InvSubBytes can be changed.

$$AddRoundKey(InvMixcolumns(\mathbf{B})) = \mathbf{K} \oplus \mathbf{M}^{-1}.\mathbf{B}.$$
$$InvMixcolumns(AddRoundKey(\mathbf{B})) = \mathbf{M}^{-1}.(\mathbf{K} \oplus \mathbf{B}) = \mathbf{M}^{-1}\mathbf{K} \oplus \mathbf{M}^{-1}\mathbf{B}.$$

AES – Round Key Expansion Funkction

Example for 128 bit key

$\mathbf{W}_0$	$\mathbf{W}_1$	$\mathbf{W}_2$	$\mathbf{W}_3$	$\mathbf{W}_4$	$\mathbf{W}_5$	$\mathbf{W}_6$	$\mathbf{W}_7$	$\mathbf{W}_8$	$\mathbf{W}_9$	$\mathbf{W}_{10}$	$\mathbf{W}_{11}$
k_{00}	k_{01}	k_{02}	k_{03}								
k_{10}	k_{11}	k_{12}	k_{13}								
k_{20}	k_{21}	k_{22}	k_{23}								
k_{30}	k_{31}	k_{32}	k_{33}								
1. Round Key				2. Round Key				3. Round Key			

$$\mathbf{W}_i = \begin{cases} \mathbf{W}_{i-4} \oplus W_{i-1} & \text{ak } i \text{ nie je delitené 4} \\ \mathbf{W}_{i-4} \oplus SubByte(RotByte(\mathbf{W}_{i-1})) \oplus Rcon(i/4) & \text{ak } i \text{ je delitené 4} \end{cases}$$

$$\begin{aligned} Rcon(i) &= [\{x^{i-1}\}\{00\}\{00\}\{00\}] \\ RotByte[w_1, w_2, w_3, w_4] &= [w_2, w_3, w_4, w_1] \end{aligned}$$

AES – Round Key Expansion Funkction

```
KeyExpansion(byte key[4*Nk], word w[Nb*(Nr+1)], Nk)
begin
  word temp
  i = 0
  while (i < Nk)
    w[i] = word(key[4*i], key[4*i+1], key[4*i+2], key[4*i+3])
    i = i+1
  end while
  i = Nk
  while (i < Nb * (Nr+1)]
    temp = w[i-1]
    if (i mod Nk = 0)
        temp = SubWord(RotWord(temp)) xor Rcon[i/Nk]
    else if (Nk > 6 and i mod Nk = 4)
        temp = SubWord(temp)
    end if
    w[i] = w[i-Nk] xor temp
    i = i + 1
  end while
end
```

Nb – $=4$ – the number of columns of matrix State
Nk – $=4$, 6 resp. 8 for 128-, 192- resp. 256-bit key
(the number of 32-bit words of key $=$ the number of columns of key matrix)

Nr – $=10$, 12, resp. 16 for 128-, 192- resp. 256-bit key – the number of rounds