Secret Key Systems (block encoding)

Encrypting a small block of text (say 64 bits)

General Considerations:

Secret Key Systems

Encrypting a small block of text (say 64 bits)

General Considerations:

1. Encrypted data should look random.

   As though someone flipped a fair coin 64 times and heads means 1 and tails 0.

   Any change in one bit of output corresponds to a huge change in the input (bits are uncorrelated).

   Should be about as many 1's as 0's usually.

Secret Key Systems

Encrypting a small block of text (say 64 bits)

General Considerations:

1. Encrypted data should look random.
   As though someone flipped a fair coin 64 times and heads means 1 and tails 0.
   Any change in one bit of output corresponds to a huge change in the input (bits are uncorrelated).
   Should be about as many 1's as 0's usually.
2. Crypto algorithms try to spread the influence of each input bit to all output bits and change in one input bit should have 50% chance of changing any of the output bits (hence many rounds).

Secret Key Systems

Encrypting a small block of text (say 64 bits)

General Considerations:

1. Encrypted data should look random.
   As though someone flipped a fair coin 64 times and heads means 1 and tails 0.
   Any change in one bit of output corresponds to a huge change in the input (bits are uncorrelated).
   Should be about as many 1's as 0's usually.
2. Crypto algorithms try to spread the influence of each input bit to all output bits and change in one input bit should have 50% chance of changing any of the output bits (hence many rounds).
3. Operations should be invertable – hence *xor* and table lookup. Use of one key for both encryption and decryption.

Secret Key Systems

Encrypting a small block of text (say 64 bits)

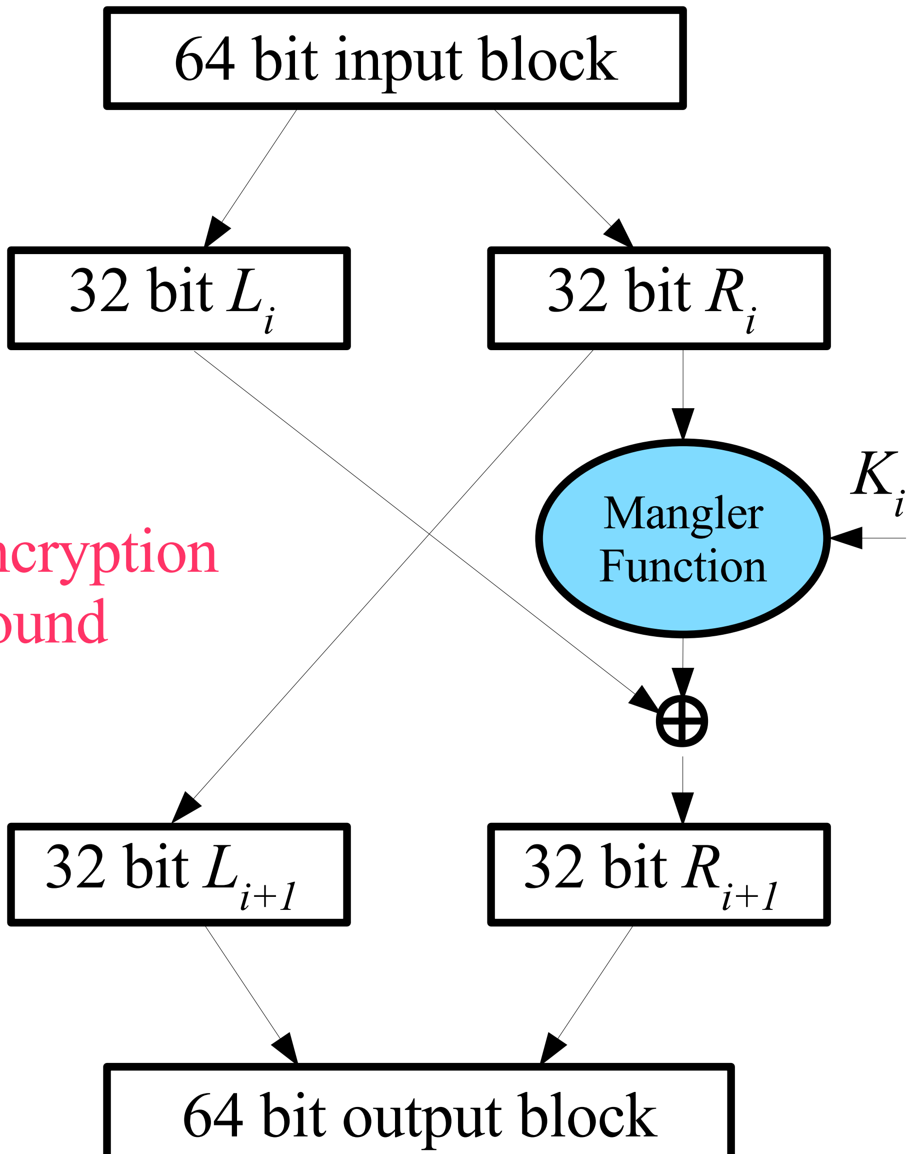
General Considerations:

1. Encrypted data should look random.
   As though someone flipped a fair coin 64 times and heads means 1 and tails 0.
   Any change in one bit of output corresponds to a huge change in the input (bits are uncorrelated).
   Should be about as many 1's as 0's usually.
2. Crypto algorithms try to spread the influence of each input bit to all output bits and change in one input bit should have 50% chance of changing any of the output bits (hence many rounds).
3. Operations should be invertable – hence xor and table lookup. Use of one key for both encryption and decryption.
4. Attacks may be mitigated if they rely on operations that are not efficiently implemented in hardware yet allow normal operation to complete efficiently, even in software (e.g. permute)

Secret Key Systems - DES

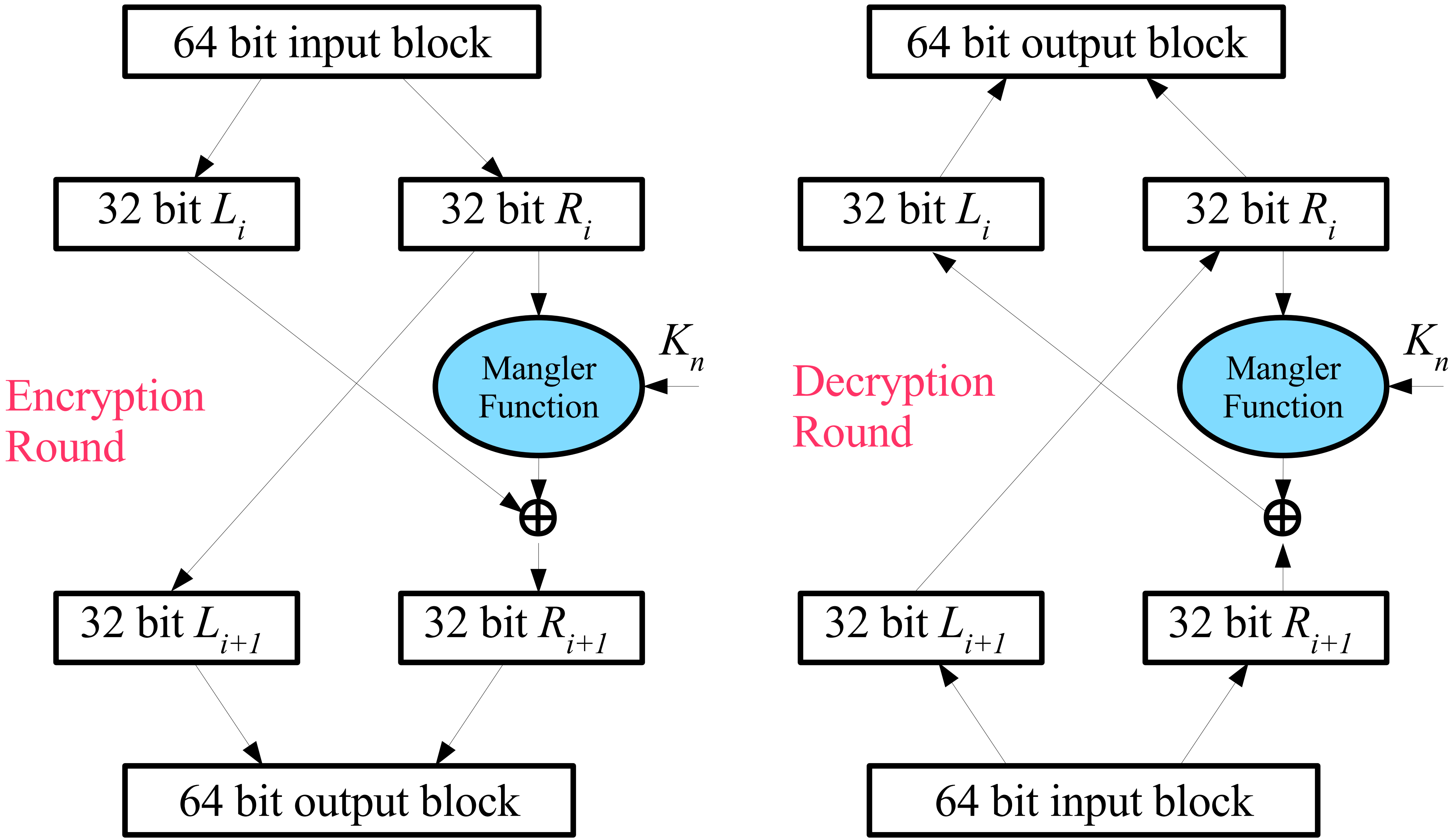
IBM/NSA 1977 - 64 bit blocks, 56 bit key, 8 bits parity

Encryption Round



Secret Key Systems - DES

IBM/NSA 1977 - 64 bit blocks, 56 bit key, 8 bits parity



Secret Key Systems - DES

Generating per round keys K_1 K_2 ... K_{16} from the 56 bit Key + 8 parity bits

Key bits:								
	1...8	9...16	17...24	25...32	33...40	41...48	49...56	57...64

Secret Key Systems - DES

Generating per round keys K_1 K_2 ... K_{16} from the 56 bit Key + 8 parity bits

Key bits:								
	1...8	9...16	17...24	25...32	33...40	41...48	49...56	57...64

C_0:

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

D_0:

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

Secret Key Systems - DES

Generating per round keys K_1 K_2 ... K_{16} from the 56 bit Key + 8 parity bits

Key bits:

1...8	9...16	17...24	25...32	33...40	41...48	49...56	57...64

C_0:

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

D_0:

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

Each round: K_i has 48 bits assembled in 2 halves permuted from 24 bits each of C_i and D_i, K_{i+1} is obtained by rotating C_i and D_i left to form C_{i+1} and D_{i+1} (rotation is 1 bit for rounds 1,2,9,16 and 2 bits for other rounds)

Secret Key Systems - DES

Generating per round keys K_1 K_2 ... K_{16} from the 56 bit Key + 8 parity bits

Key bits:

1...8	9...16	17...24	25...32	33...40	41...48	49...56	57...64

C_0:

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

D_0:

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

Each round: K_i has 48 bits assembled in 2 halves permuted from 24 bits each of C_i and D_i, K_{i+1} is obtained by rotating C_i and D_i left to form C_{i+1} and D_{i+1} (rotation is 1 bit for rounds 1,2,9,16 and 2 bits for other rounds)

Permutations:

Left half C_i: (9,18,22,25 are missing)

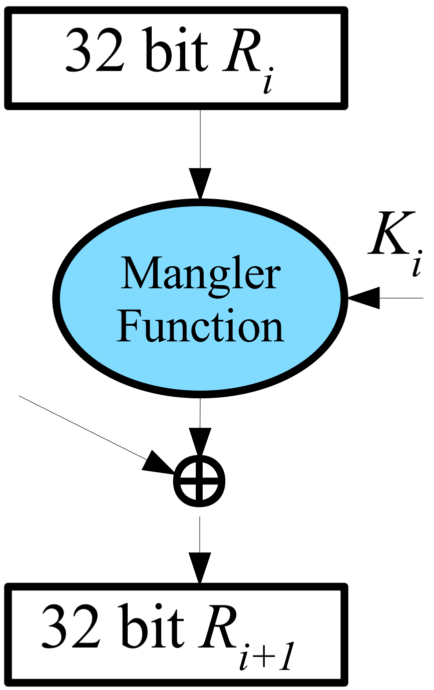
14 17 11 24 1 5 3 28 15 6 21 10 23 19 12 4 26 8 16 7 27 20 13 2

Right half D_i: (35,38,43,54 are missing)

41 52 31 37 47 55 30 40 51 45 33 48 44 49 39 56 34 53 46 42 50 36 29 32

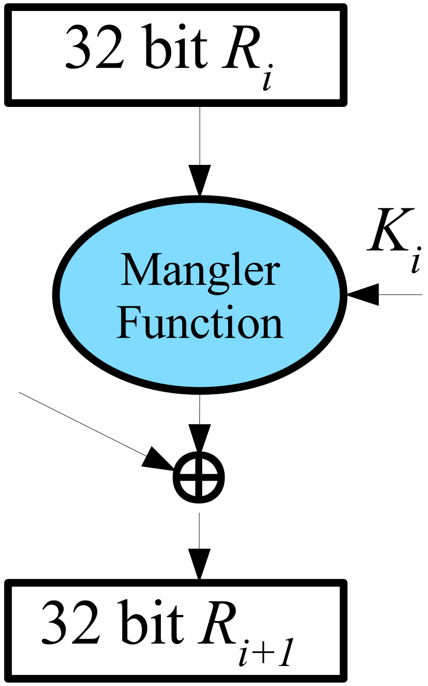
Secret Key Systems - DES

The Mangler function: mixes 32 bit input with 48 bit key to produce 32 bits



Secret Key Systems - DES

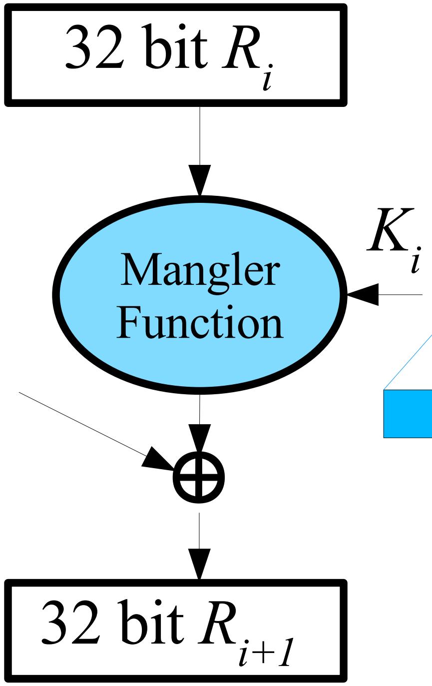
The Mangler function: mixes 32 bit input with 48 bit key to produce 32 bits



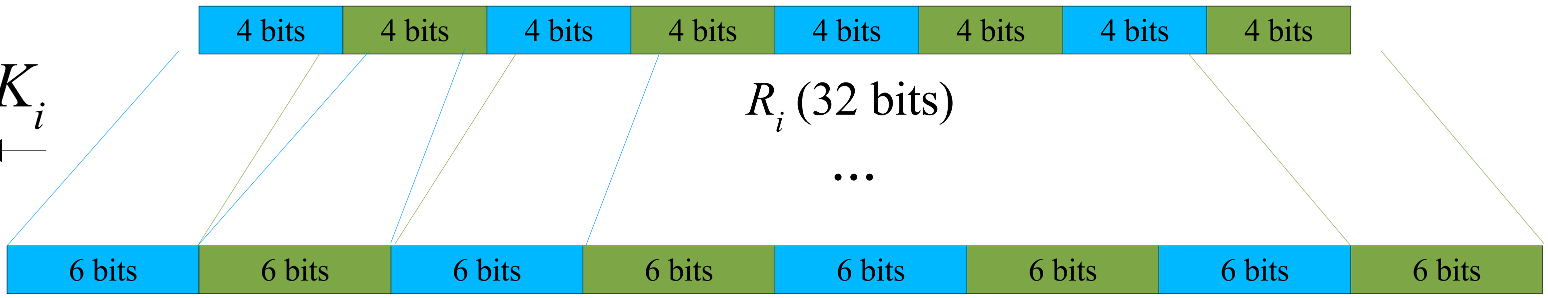
1. Expansion of input bits:

Secret Key Systems - DES

The Mangler function: mixes 32 bit input with 48 bit key to produce 32 bits

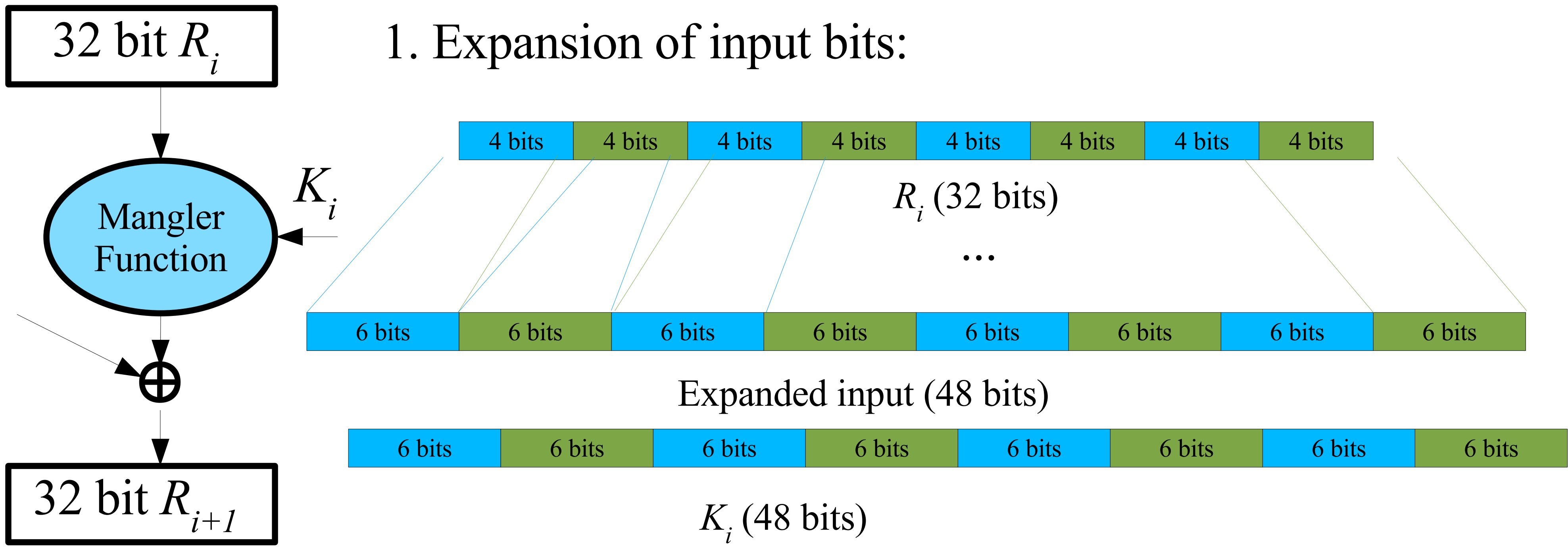


1. Expansion of input bits:



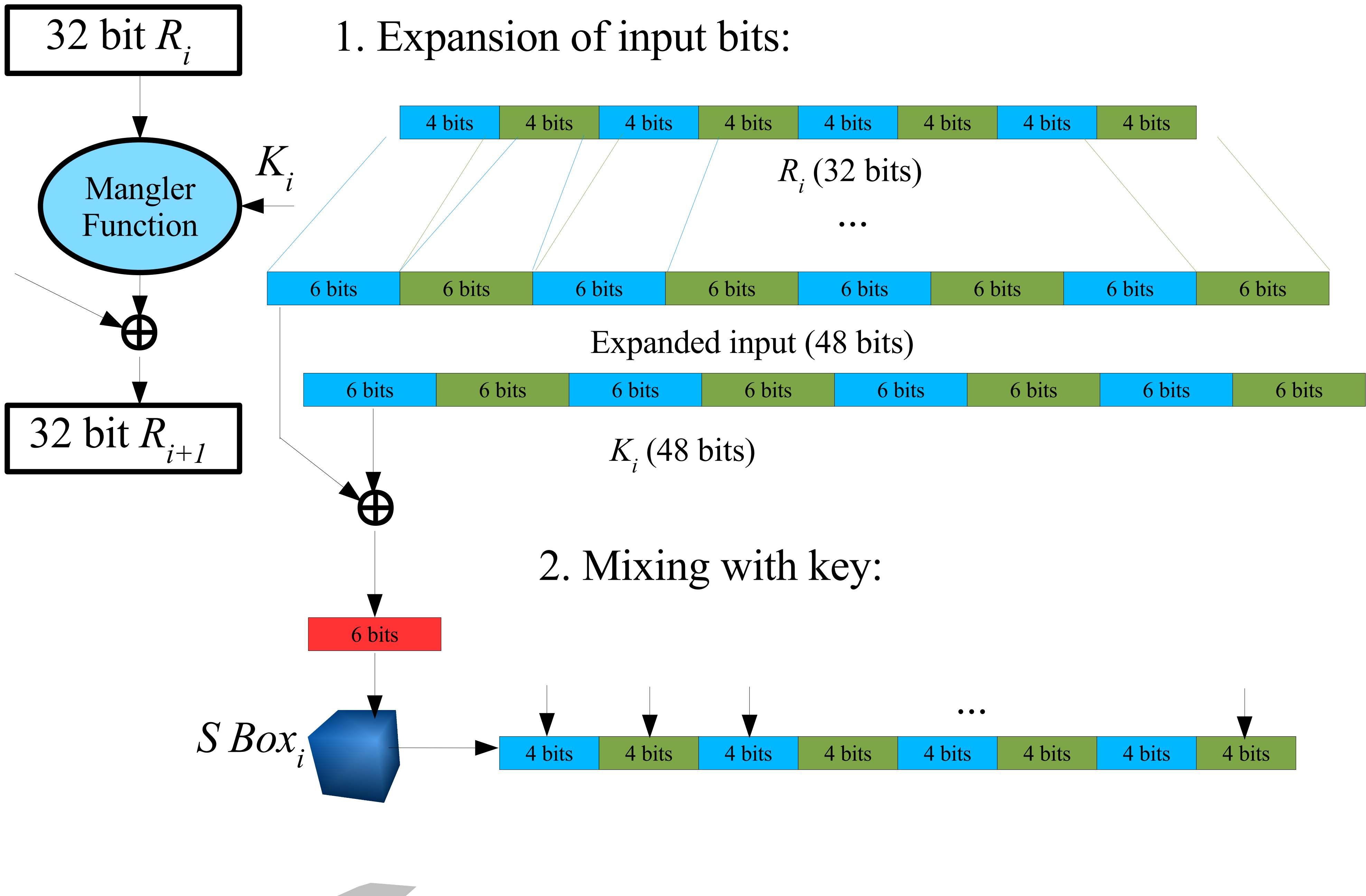
Secret Key Systems - DES

The Mangler function: mixes 32 bit input with 48 bit key to produce 32 bits



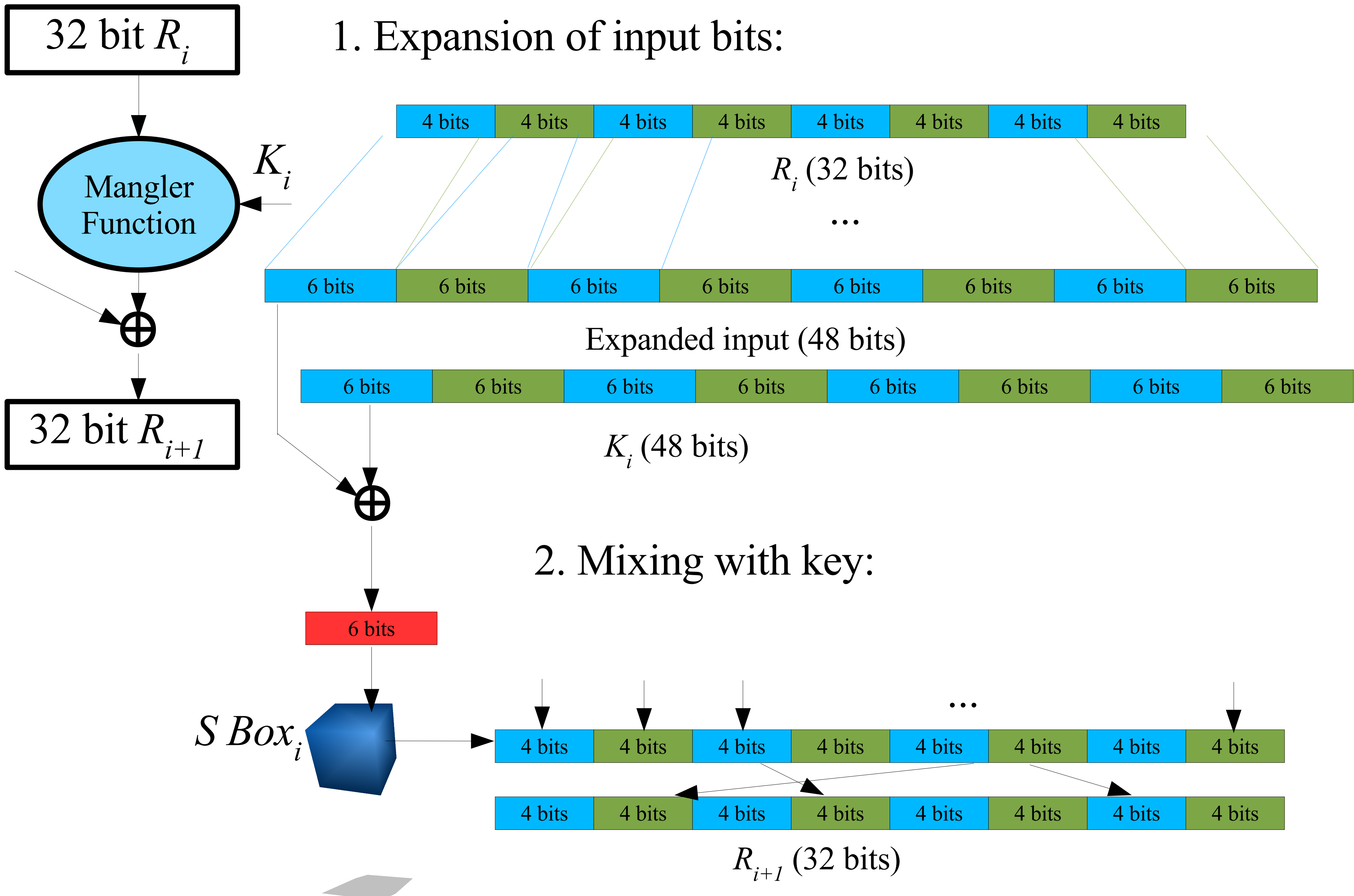
Secret Key Systems - DES

The Mangler function: mixes 32 bit input with 48 bit key to produce 32 bits



Secret Key Systems - DES

The Mangler function: mixes 32 bit input with 48 bit key to produce 32 bits



Secret Key Systems - DES

The *S Box*: maps 6 bit blocks to 4 bit sections

S Box$_1$ (first 6 bits):

Input bits 2,3,4,5

	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
00	1110	0100	1101	0001	0010	1111	1011	1000	0011	1010	0110	1100	0101	1001	0000	0111
01	0000	1111	0111	0100	1110	0010	1101	0001	1010	0110	1100	1011	1001	0101	0011	1000
10	0100	0001	1110	1000	1101	0110	0010	1011	1111	1100	1001	0111	0011	1010	0101	0000
11	1111	1100	1000	0010	0100	1001	0001	0111	0101	1011	0011	1110	1010	0000	0110	1101

Input bits 1 and 6

Final 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

Secret Key Systems - DES

Weak and semi-weak keys:

If key is such that C_0 *or* D_0 are: 1) all 0s; 2) all 1s; 3) alternating 1s and 0s, then attack is easy. There are 16 such keys. Keys for which C_0 *and* D_0 are both 0 or both 1 are called *weak* (encrypting with key gives same result as decrypting).

Secret Key Systems - DES

Weak and semi-weak keys:

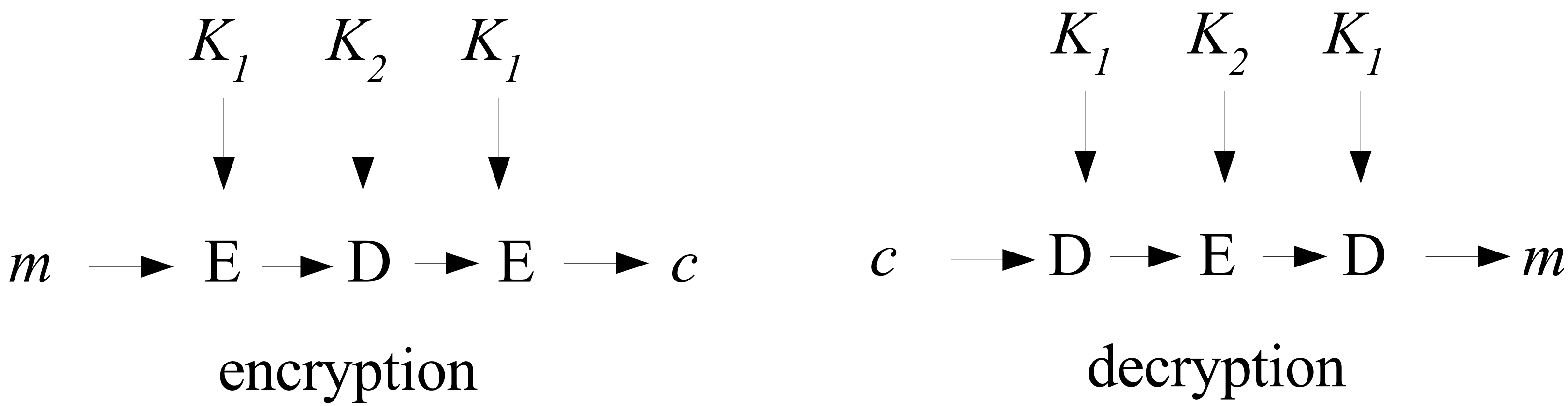
If key is such that C_0 *or* D_0 are: 1) all 0s; 2) all 1s; 3) alternating 1s and 0s, then attack is easy. There are 16 such keys. Keys for which C_0 *and* D_0 are both 0 or both 1 are called *weak* (encrypting with key gives same result as decrypting).

Discussion:

1. Not much known about the design – not made public
   Probably attempt to prevent known attacks
2. Changing S-Boxes has resulted in provably weaker system

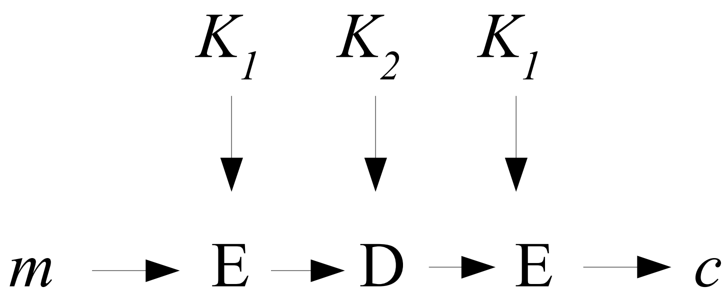
Secret Key Systems - 3DES

Two keys K_1 and K_2:

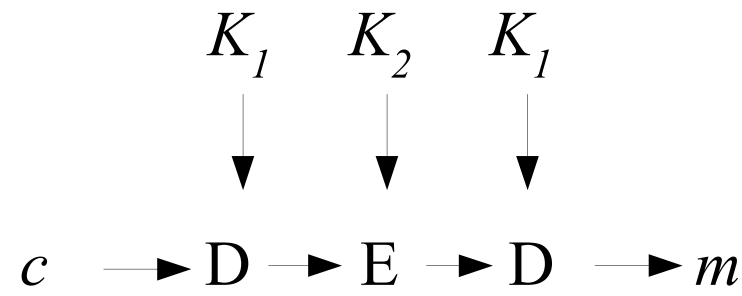


Secret Key Systems - 3DES

Two keys K_1 and K_2:



encryption



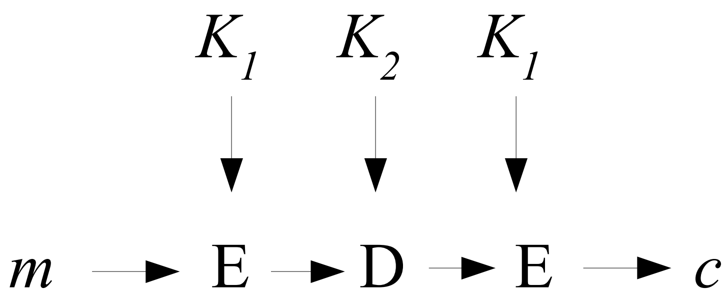
decryption

Why not 2DES

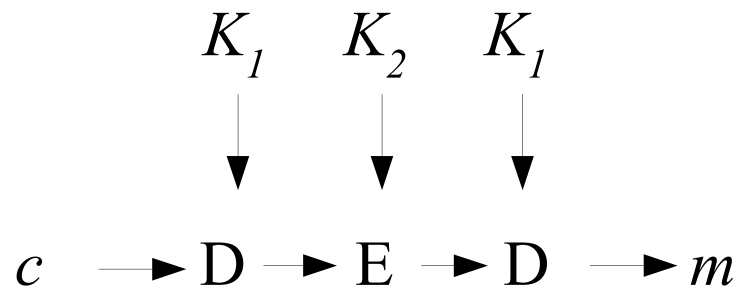
1. Double encryption with the same key still requires searching 2^{56} keys

Secret Key Systems - 3DES

Two keys K_1 and K_2:



encryption



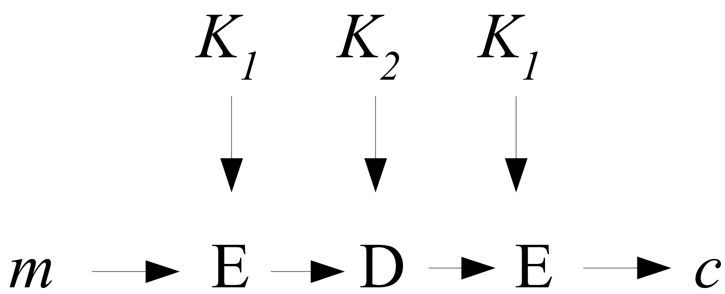
decryption

Why not 2DES

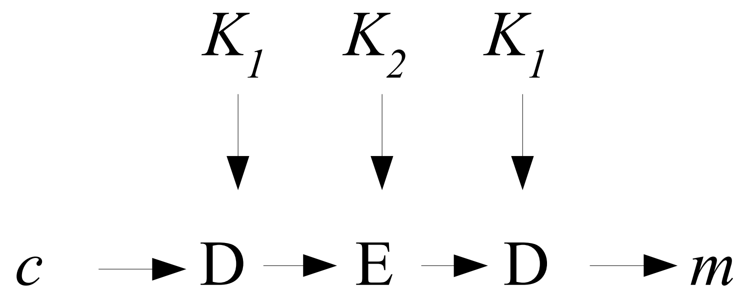
1. Double encryption with the same key still requires searching 2^{56} keys
2. Double encryption with two different keys is just as vulnerable as DES due to the following, assuming some <*m,c*> pairs are known:

Secret Key Systems - 3DES

Two keys K_1 and K_2:



encryption



decryption

Why not 2DES

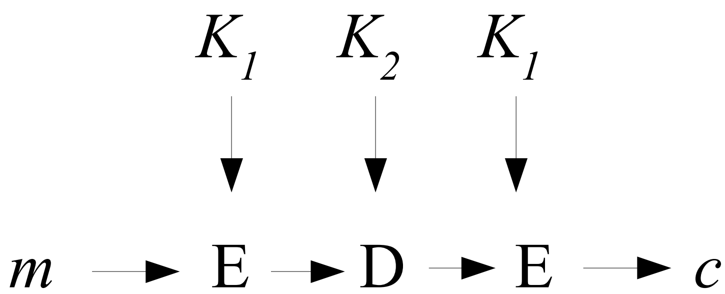
1. Double encryption with the same key still requires searching 2^{56} keys
2. Double encryption with two different keys is just as vulnerable as DES due to the following, assuming some <*m,c*> pairs are known:

m:

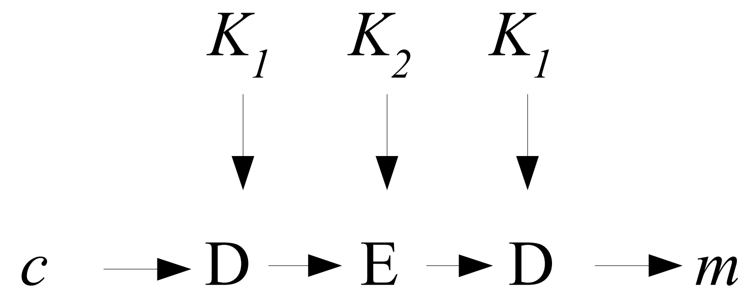
	K_1	$E(K_1, m)$
2^{56}	101010	1000011
	...	...
	100010	0101111
	001011	0001101

Secret Key Systems - 3DES

Two keys K_1 and K_2:



encryption



decryption

Why not 2DES

1. Double encryption with the same key still requires searching 2^{56} keys
2. Double encryption with two different keys is just as vulnerable as DES due to the following, assuming some $<m,c>$ pairs are known:

m: (2^{56} entries)

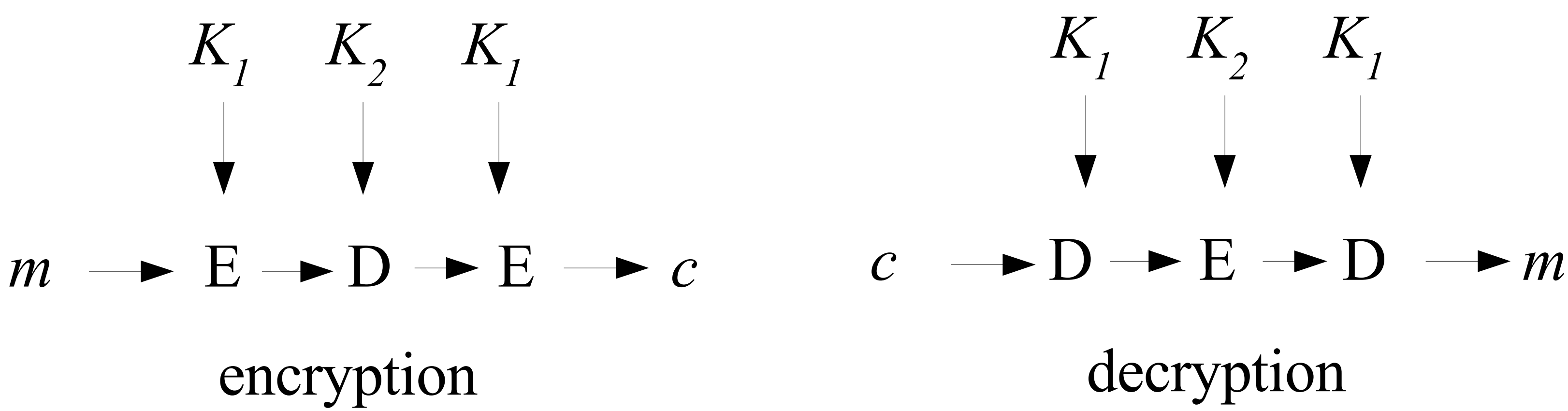
K_1	$E(K_1, m)$
101010	1000011
...	...
100010	0101111
001011	0001101

c: (2^{56} entries)

K_2	$D(K_2, c)$
101110	0001101
...	...
001110	1000011
001011	0001101

Secret Key Systems - 3DES

Two keys K_1 and K_2:



Why not 2DES

1. Double encryption with the same key still requires searching 2^{56} keys
2. Double encryption with two different keys is just as vulnerable as DES due to the following, assuming some <*m,c*> pairs are known:

m: (2^{56} rows)

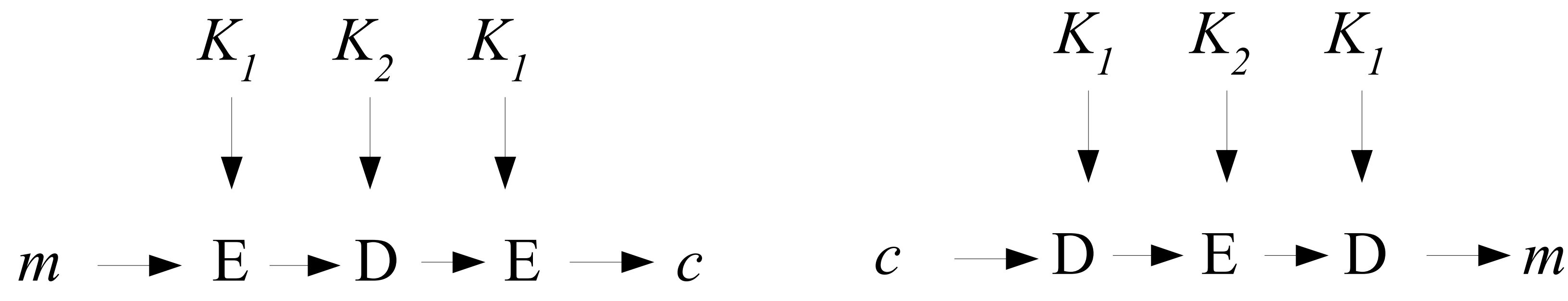
K_1	$E(K_1, m)$
101010	1000011
...	...
100010	0101111
001011	0001101

c: (2^{56} rows)

K_2	$D(K_2, c)$
101110	0001101
...	...
001110	1000011
001011	0001101

Secret Key Systems - 3DES

Two keys K_1 and K_2:



encryption　　decryption

Why not 2DES

1. Double encryption with the same key still requires searching 2^{56} keys
2. Double encryption with two different keys is just as vulnerable as DES due to the following, assuming some <m,c> pairs are known:

2^{56} entries, for m:

K_1	$E(K_1, m)$
101010	1000011
...	...
100010	0101111
001011	0001101

2^{56} entries, for c:

K_2	$D(K_2, c)$
101110	0001101
...	...
001110	1000011
001011	0001101

Test matches on other <m,c> pairs

$m \rightarrow E(K_1,m)$

$D(K_2,c) \leftarrow c$

Secret Key Systems - 3DES

Why not 2DES

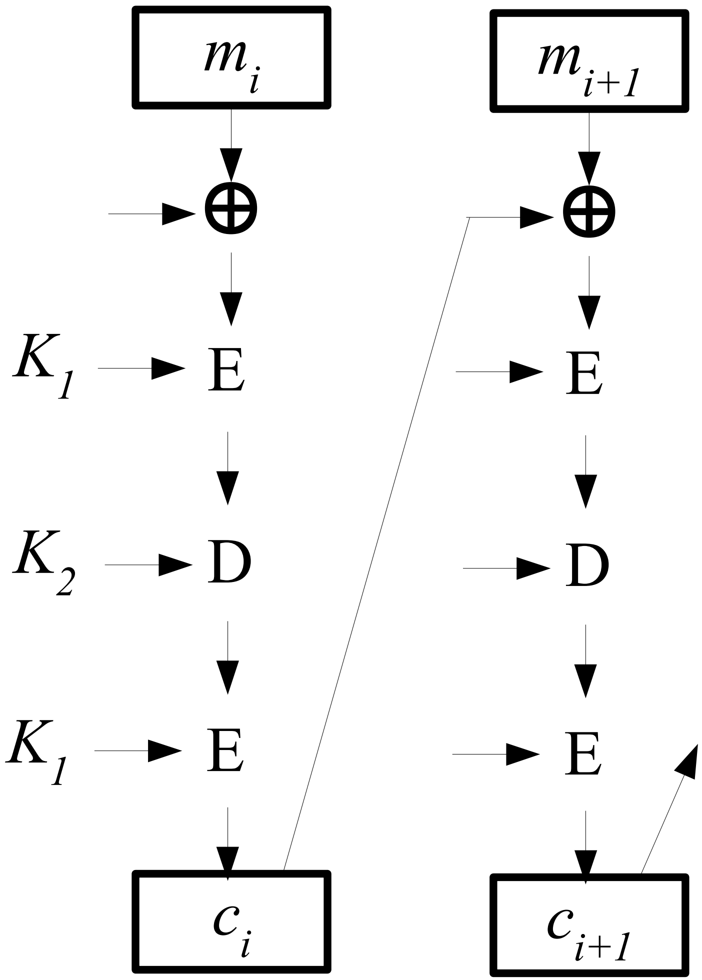
2. Double encryption with two different keys is just as vulnerable as DES due to the following, assuming some <*m,c*> pairs are known: How many <*m,c*> pairs do you need?

   2^{64} possible blocks
   2^{56} table entries
   each block has probability $2^{-8} = 1/256$ of showing in a table
   probability a block is in both tables is 2^{-16}
   average number of matches is 2^{48}
   average number of matches for two <*m,c*> pairs is about 2^{32}
   for three <*m,c*> pairs is about 2^{16}
   for four <*m,c*> pairs is about 2^{0}

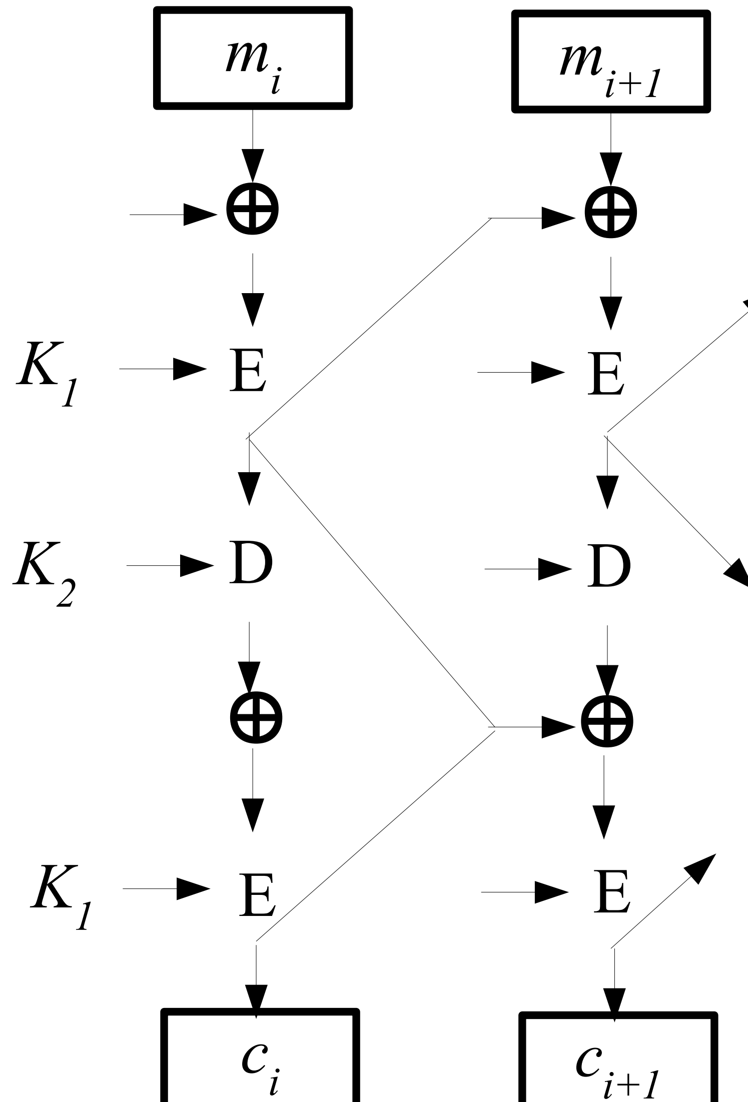
3. Triple encryption with two different keys
   - 112 bits of key is considered enough
   - straightforward to find a triple of keys that maps a given plaintext to a given ciphertext (no known attack with 2 keys)
   - EDE: use same keys to get DES, EEE: effect of permutations lost

Secret Key Systems - 3DES

CBC with 3DES:



outside



inside

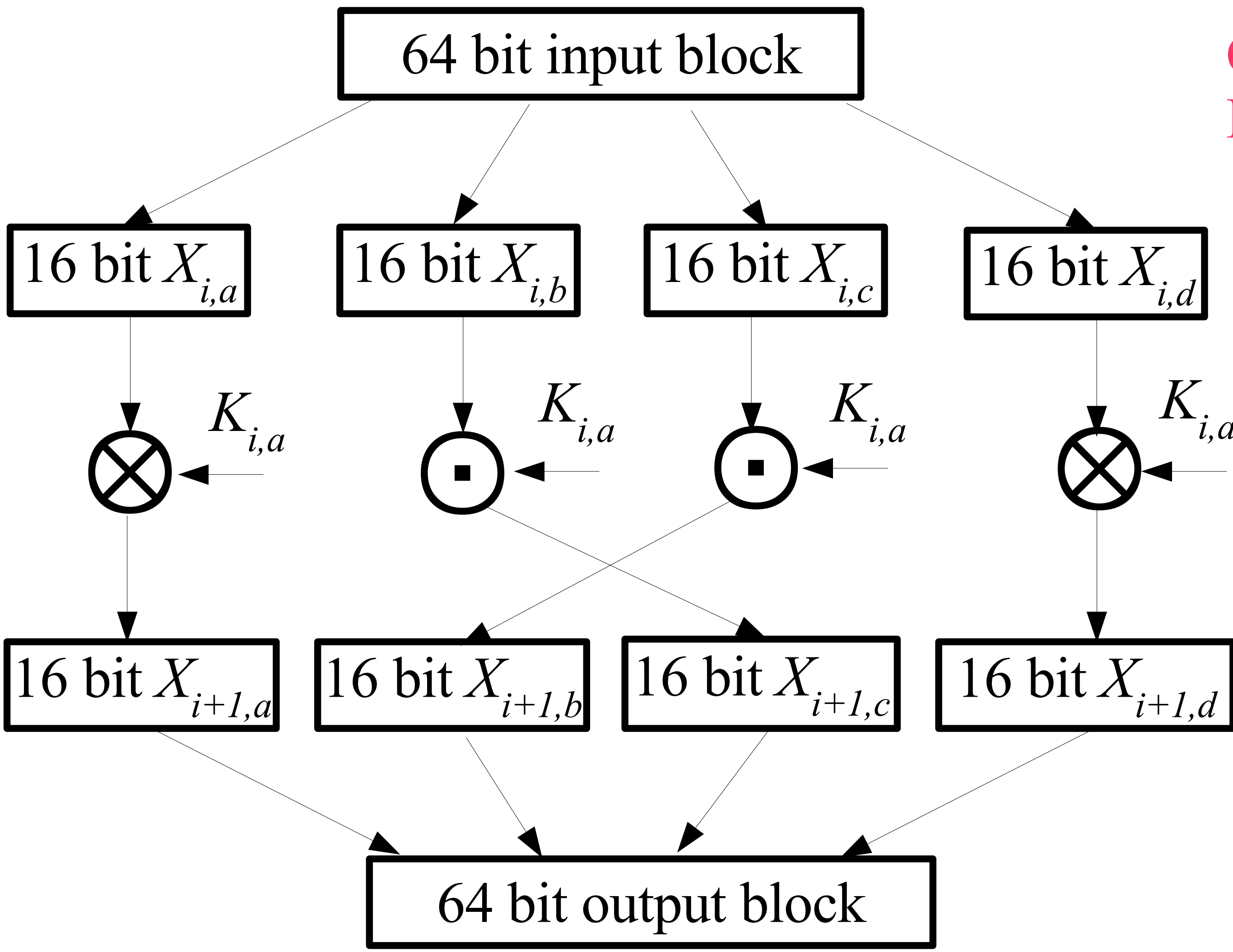
Secret Key Systems - 3DES

CBC with 3DES:

1. On the outside – same attack as with CBC – change a block with side effect of garbling another
2. On the inside – attempt at changing a block results in all block garbled to the end of the message.
3. On the inside – use three times as much hardware to pipeline encryptions resulting in DES speeds.
4. On the outside – EDE simply is a drop-in replacement for what might have been there before.

Secret Key Systems - IDEA

ETH Zuria 1991 - 64 bit blocks, 128 bit key, 0 bits parity:



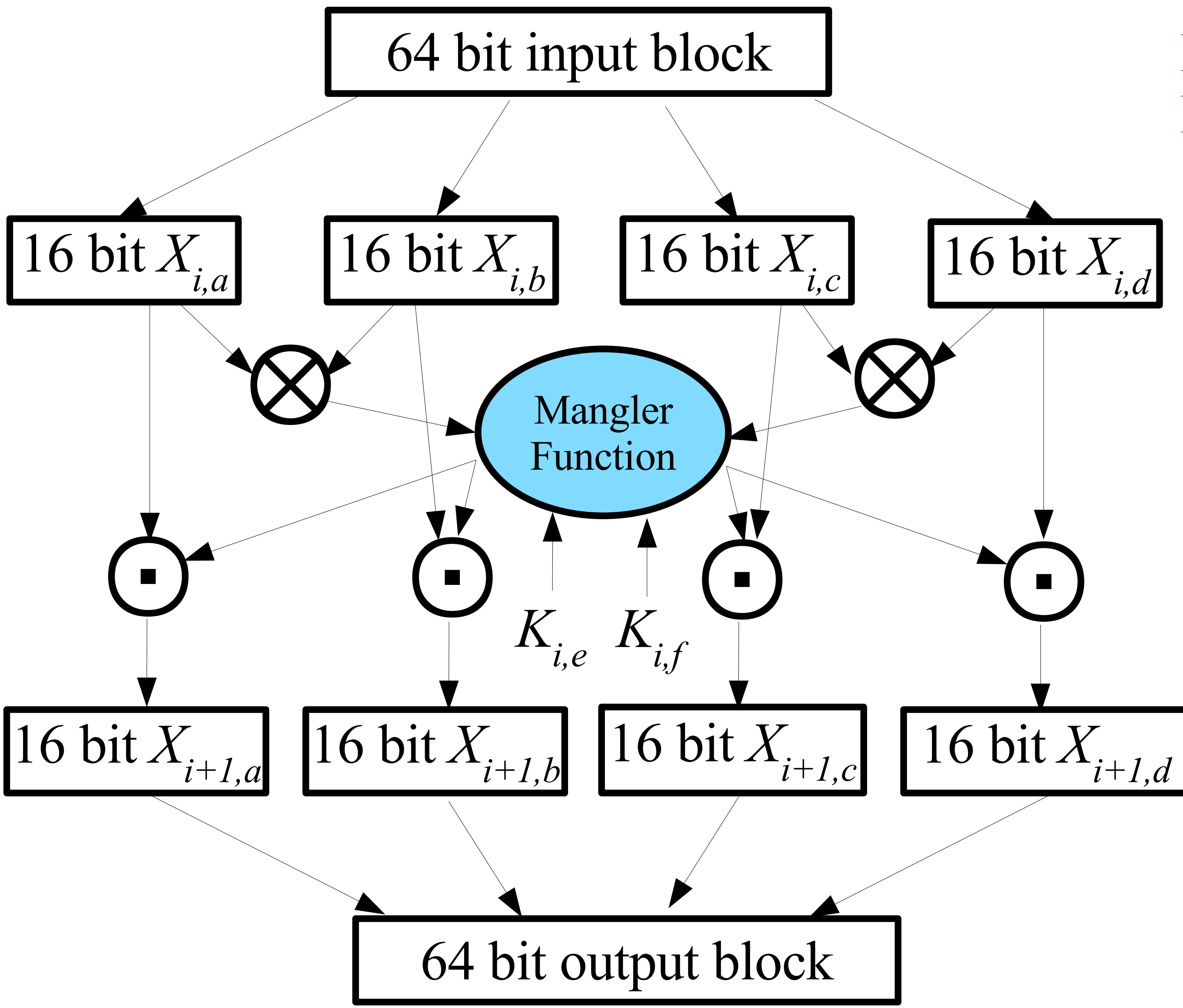
Odd Encryption Round

$\otimes$ Multiplication mod $2^{16}+1$. 0 is treated as 2^{16} (invertible)

$\odot$ Addition but throw away carries

Secret Key Systems - IDEA

ETH Zuria 1991 - 64 bit blocks, 128 bit key, 0 bits parity:



Even Encryption Round

$$Y_{in} = X_{i,a} \otimes X_{i,b}$$

$$Z_{in} = X_{i,c} \otimes X_{i,d}$$

$$Y_{out} = ((K_{i,e} \otimes Y_{in}) \odot Z_{in} \otimes K_{i,f}$$

$$Z_{out} = (K_{i,e} \otimes Y_{in}) \odot Y_{out}$$

$$X_{i+1,a} = X_{i,a} \odot Y_{out}$$

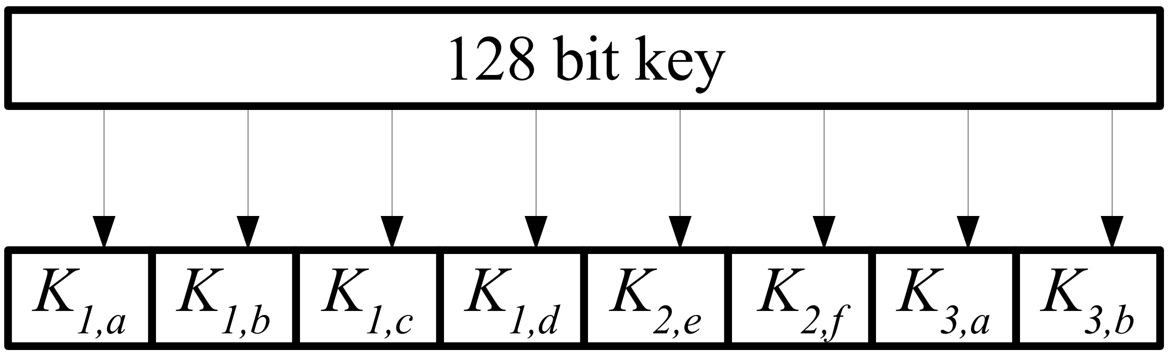
$$X_{i+1,b} = X_{i,b} \odot Y_{out}$$

$$X_{i+1,c} = X_{i,c} \odot Z_{out}$$

$$X_{i+1,d} = X_{i,d} \odot Z_{out}$$

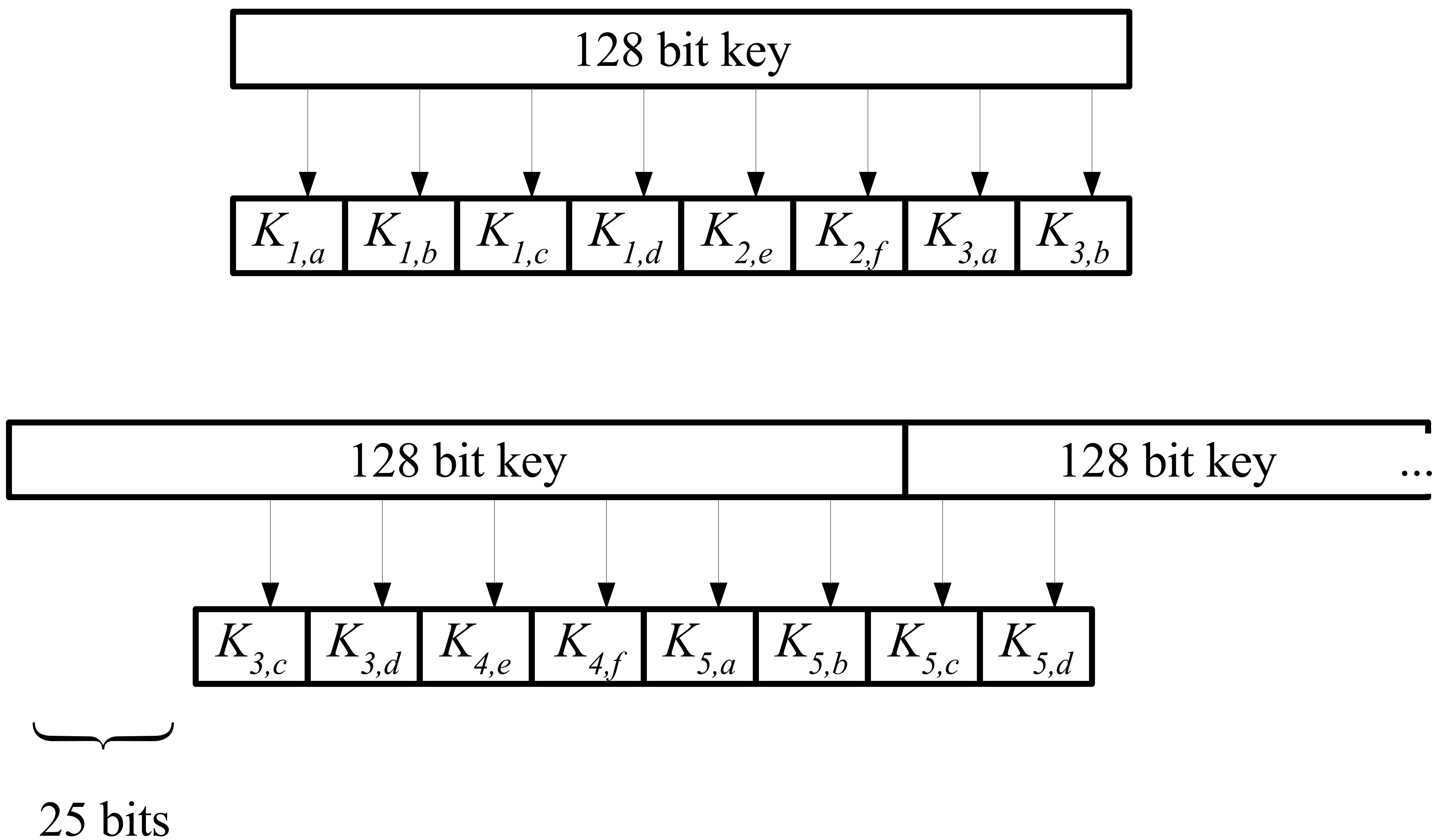
Secret Key Systems - IDEA

Key generation   52 16 bit keys needed (2 per even round 4 per odd):



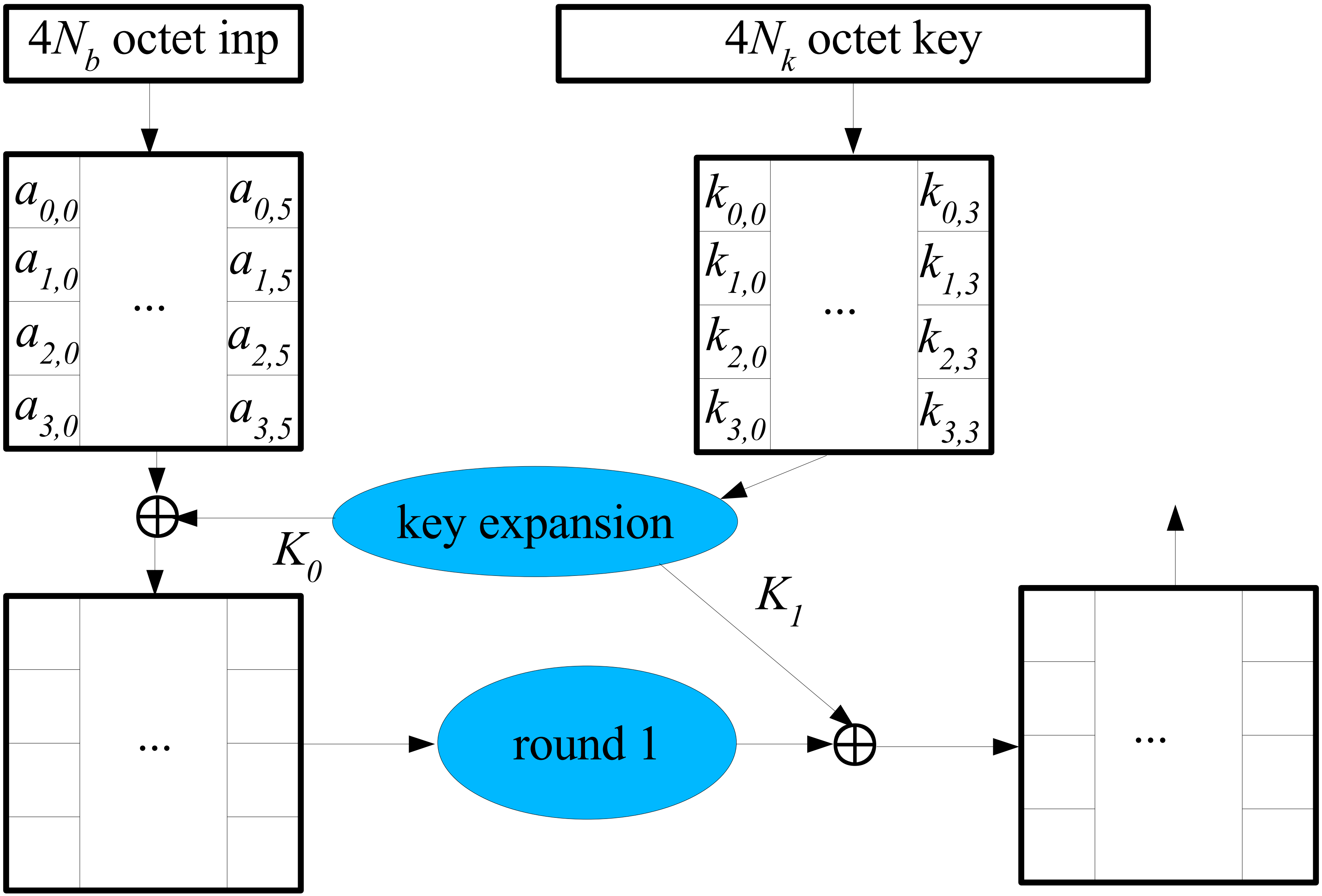
Secret Key Systems - IDEA

Key generation   52 16 bit keys needed (2 per even round 4 per odd):



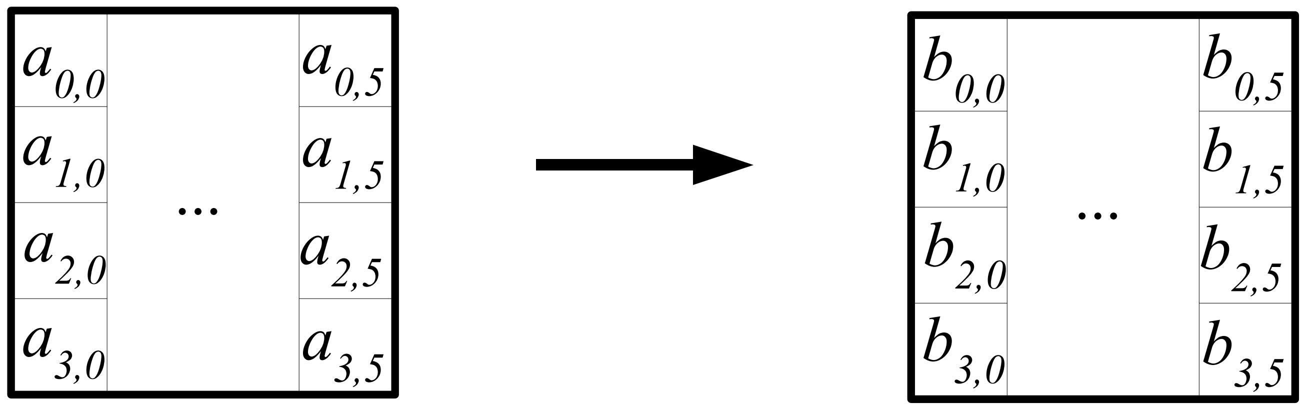
Secret Key Systems - AES

NIST (2001) parameterized key size (128 bits to 256 bits)



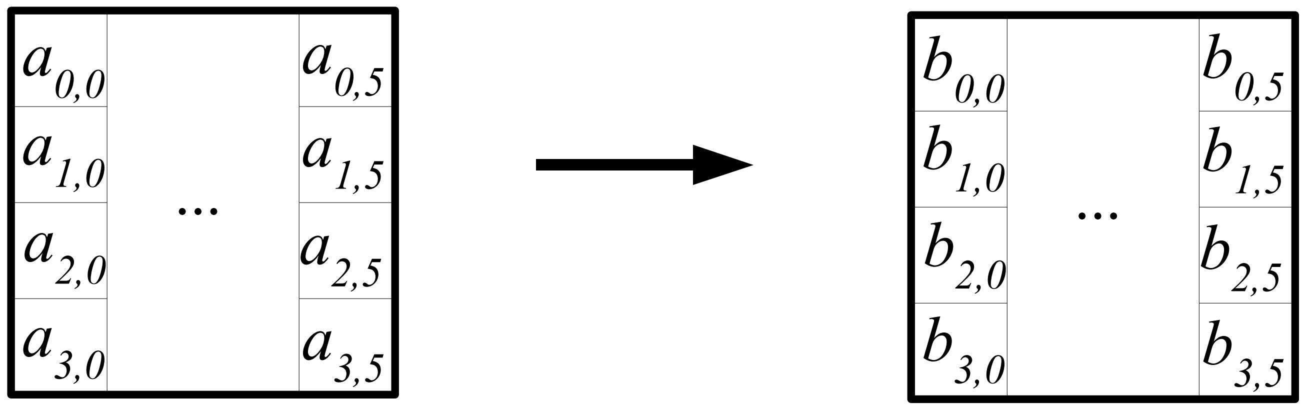
Secret Key Systems - AES

1. Byte Substitution via S-box - the transformation is invertable



Secret Key Systems - AES

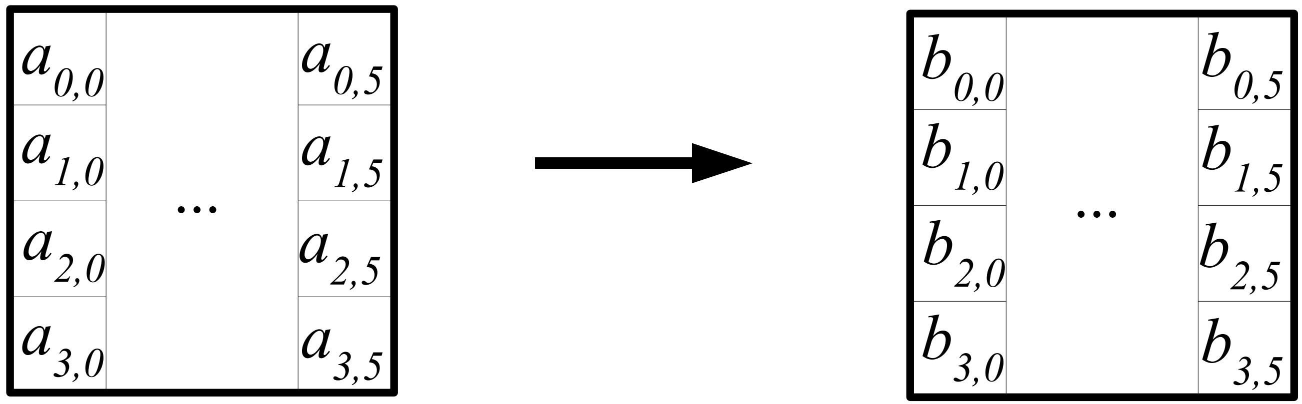
1. Byte Substitution via S-box - the transformation is invertable



A byte as a polynomial: $b_7x^7+b_6x^6+b_5x^5+b_4x^4+b_3x^3+b_2x^2+b_1x^1+b_0$, $b_i \in \{0,1\}$

Secret Key Systems - AES

1. Byte Substitution via S-box - the transformation is invertable

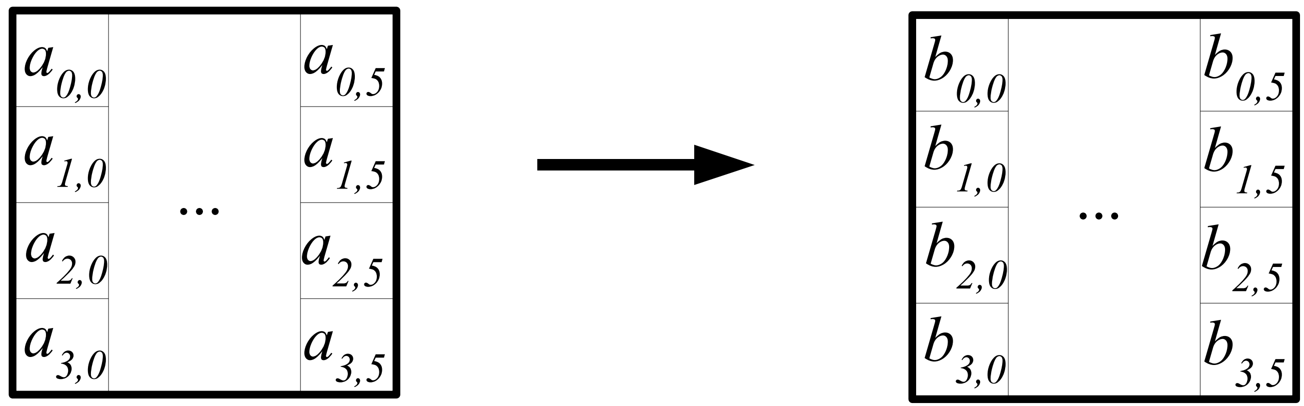


A byte as a polynomial: $b_7x^7+b_6x^6+b_5x^5+b_4x^4+b_3x^3+b_2x^2+b_1x^1+b_0$, $b_i \in \{0,1\}$

Addition: $(x^6+x^4+x^2+x^1+1) + (x^7+x^1+1) = x^7+x^6+x^4+x^2$ (exclusive or)

Secret Key Systems - AES

1. Byte Substitution via S-box - the transformation is invertable



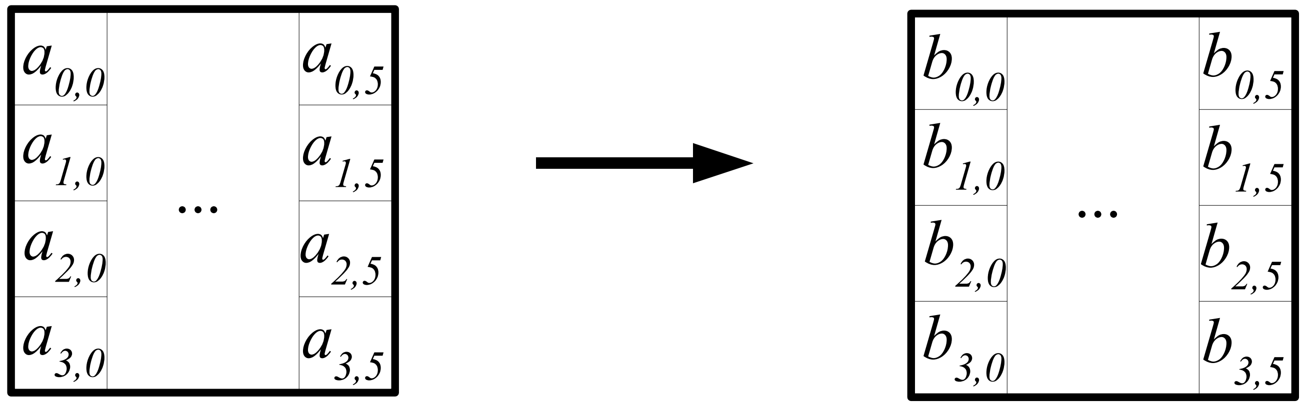
A byte as a polynomial: $b_7x^7+b_6x^6+b_5x^5+b_4x^4+b_3x^3+b_2x^2+b_1x^1+b_0$, $b_i \in \{0,1\}$

Addition: $(x^6+x^4+x^2+x^1+1) + (x^7+x^1+1) = x^7+x^6+x^4+x^2$ (exclusive or)

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

Secret Key Systems - AES

1. Byte Substitution via S-box - the transformation is invertable



A byte as a polynomial: $b_7x^7+b_6x^6+b_5x^5+b_4x^4+b_3x^3+b_2x^2+b_1x^1+b_0$, $b_i \in \{0,1\}$

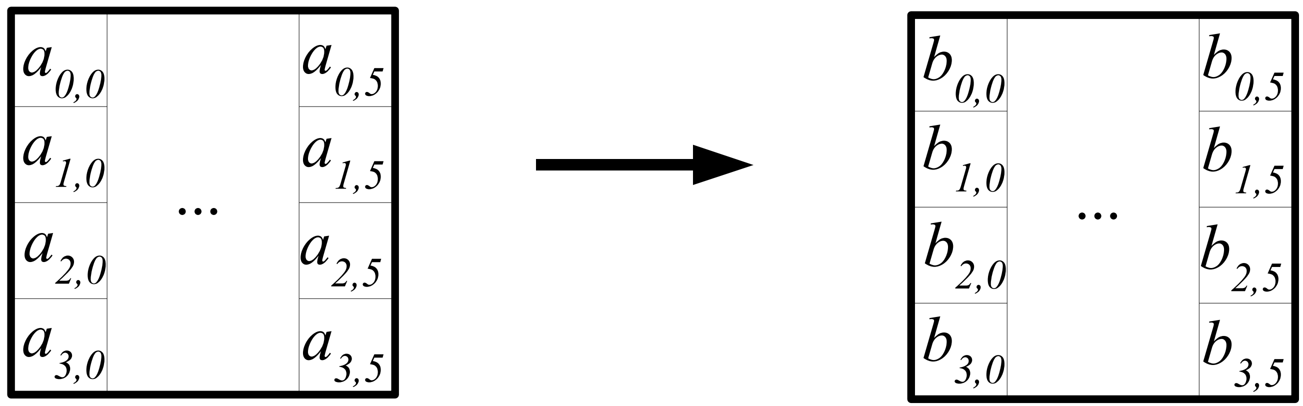
Addition: $(x^6+x^4+x^2+x^1+1) + (x^7+x^1+1) = x^7+x^6+x^4+x^2$ (exclusive or)

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

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

Secret Key Systems - AES

1. Byte Substitution via S-box - the transformation is invertable



A byte as a polynomial: $b_7x^7+b_6x^6+b_5x^5+b_4x^4+b_3x^3+b_2x^2+b_1x^1+b_0,\ b_i \in \{0,1\}$

Addition: $(x^6+x^4+x^2+x^1+1) + (x^7+x^1+1) = x^7+x^6+x^4+x^2$ (exclusive or)

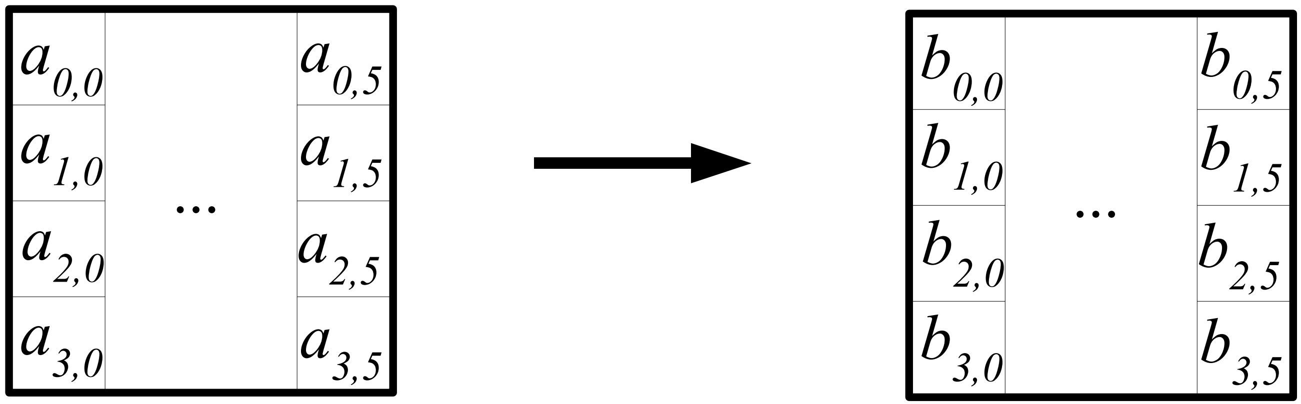
Multiplication: $(x^6+x^4+x^2+x^1+1)*(x^7+x^1+1) = x^{13}+x^{11}+x^9+x^8+x^6+x^5+x^4+x^3+x^1+1$

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

Multiplication mod m(x): $(x^6+x^4+x^2+x^1+1)*(x^7+x^1+1) \bmod m(x) = x^7+x^6+1$

Secret Key Systems - AES

1. Byte Substitution via S-box - the transformation is invertable



A byte as a polynomial: $b_7x^7+b_6x^6+b_5x^5+b_4x^4+b_3x^3+b_2x^2+b_1x^1+b_0$, $b_i \in \{0,1\}$

Addition: $(x^6+x^4+x^2+x^1+1) + (x^7+x^1+1) = x^7+x^6+x^4+x^2$ (exclusive or)

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

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

Multiplication mod m(x): $(x^6+x^4+x^2+x^1+1)*(x^7+x^1+1) \bmod m(x) = x^7+x^6+1$

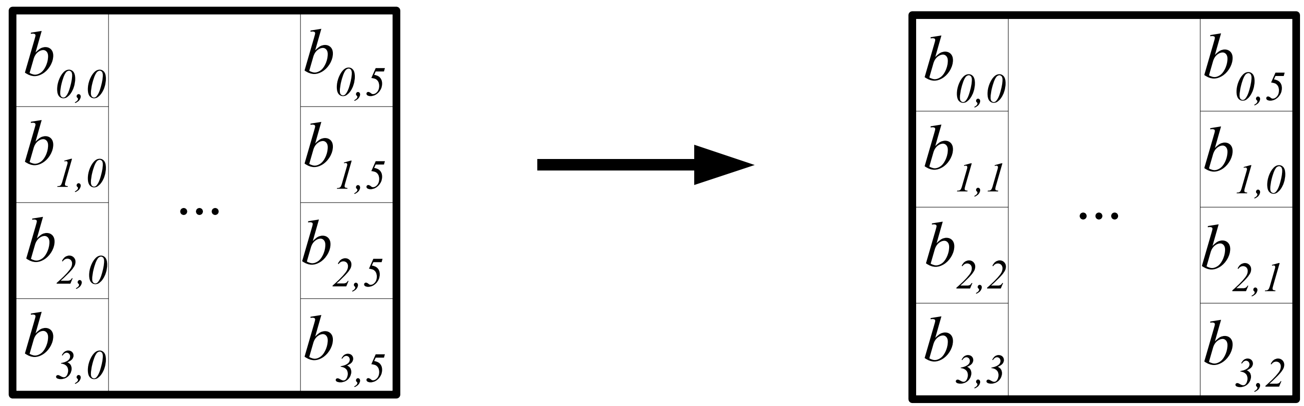
Use Euclid's algorithm to compute, for any b(x): $b(x)*a(x) + c(x)*m(x) = 1$

That is, b(x) and a(x) are inverses modulo m(x): $b^{-1}(x) = a(x) \bmod m(x)$

This defines the S-box.

Secret Key Systems - AES

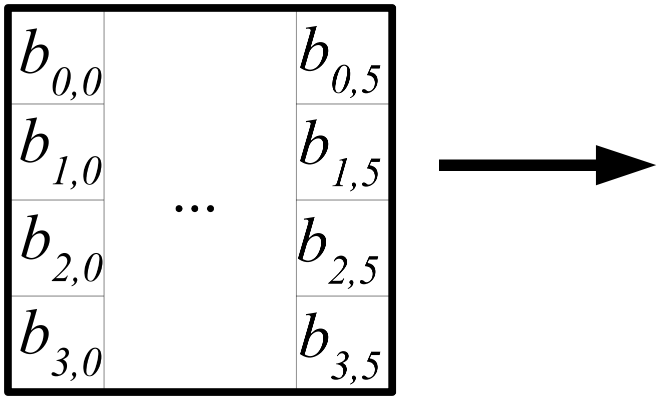
2. Row shift (cycle, left)



N_b \ *Row*	1	2	3
4	1	2	3
6	1	2	3
8	1	3	4

Secret Key Systems - AES

3. Mixed Column Transformation – constants are 8 bits now



E:

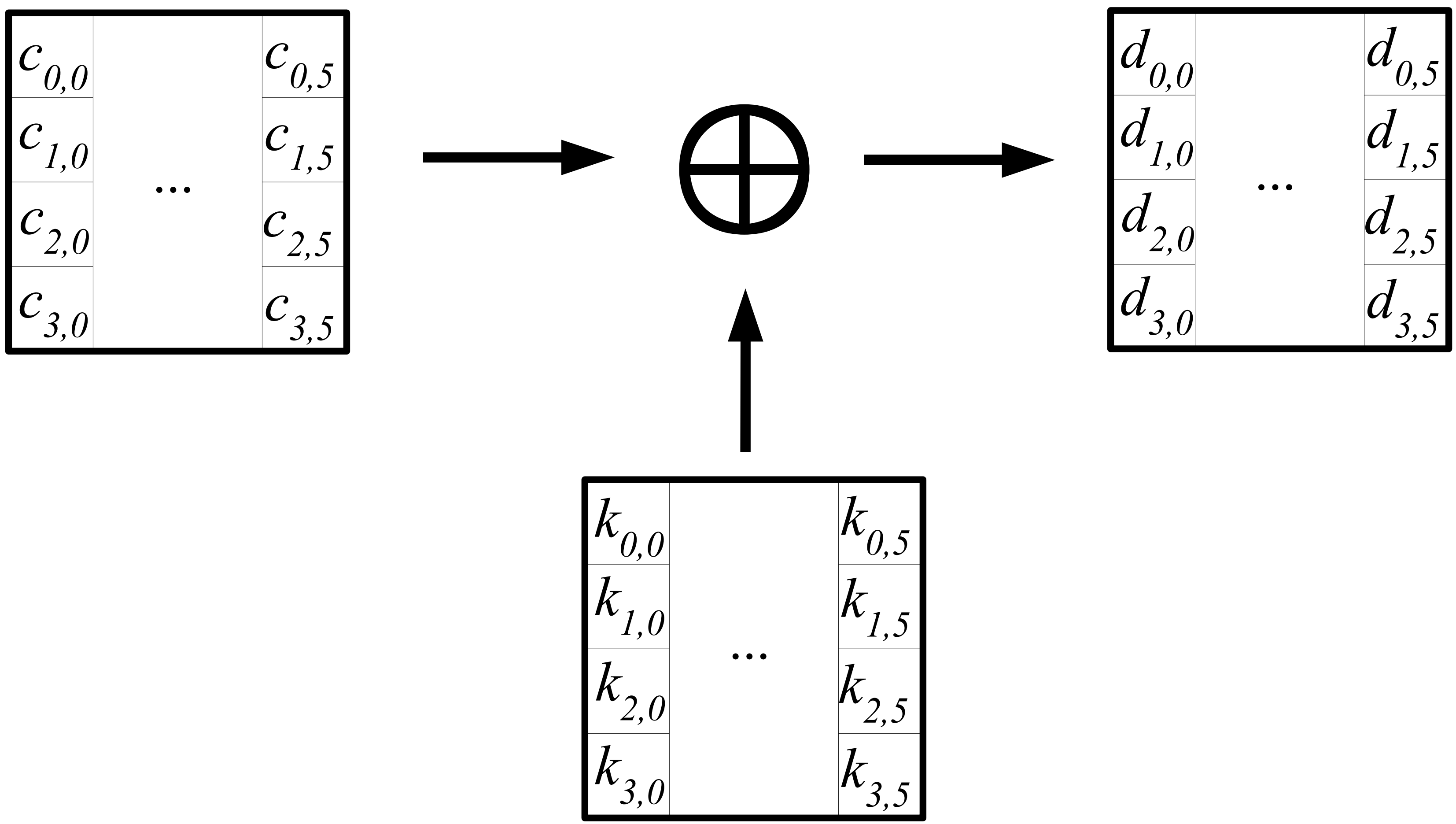
02	03	01	01
01	02	03	01
01	01	02	03
03	01	01	02

Let $e(x) = e_3x^3+e_2x^2+e_1x^1+e_0$, where e_3=0x03, e_2=e_1=0x01, e_0=0x02

Then $c(x) = e(x)*b(x) \bmod x^4+1$, multiplication obtained via E*b

Secret Key Systems - AES

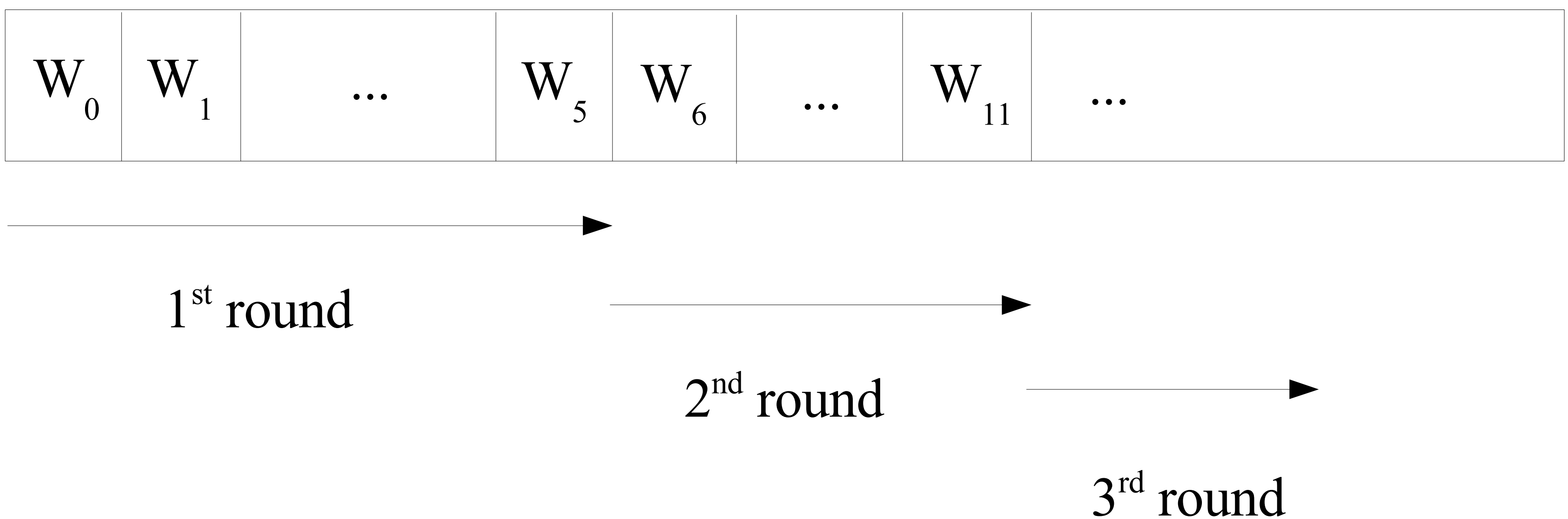
4. Round Key Addition (that is, exclusive or)



Number of round key bits = blck_lngth*(rnds+1) (e.g. 128bit, 10rnds = 1408)
Taken from expansion of cipher key
Let's forget about the expansion right now

Secret Key Systems - AES

Key Schedule example for N_b=6



Secret Key Systems - AES

Notes:

1. Many operations are table look ups so they are fast
2. Parallelism can be exploited
3. Key expansion only needs to be done one time until the key is changed
4. The S-box minimizes the correlation between input and output bits

Secret Key Systems - AES

Number of rounds

N_k \ N_b	4	6	8
4	10	12	14
6	12	12	14
8	14	14	14