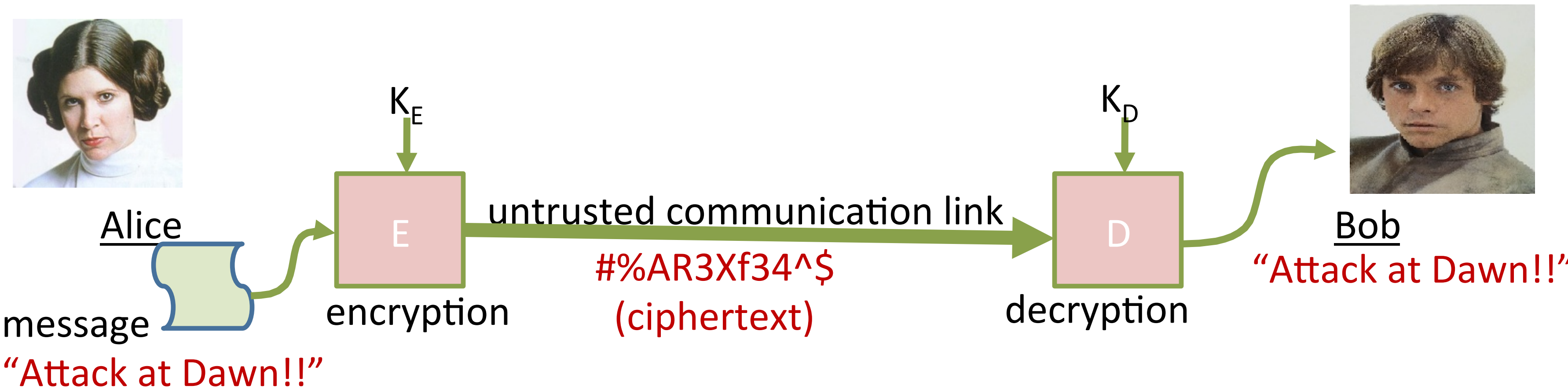
Block Ciphers

Chester Rebeiro

IIT Madras

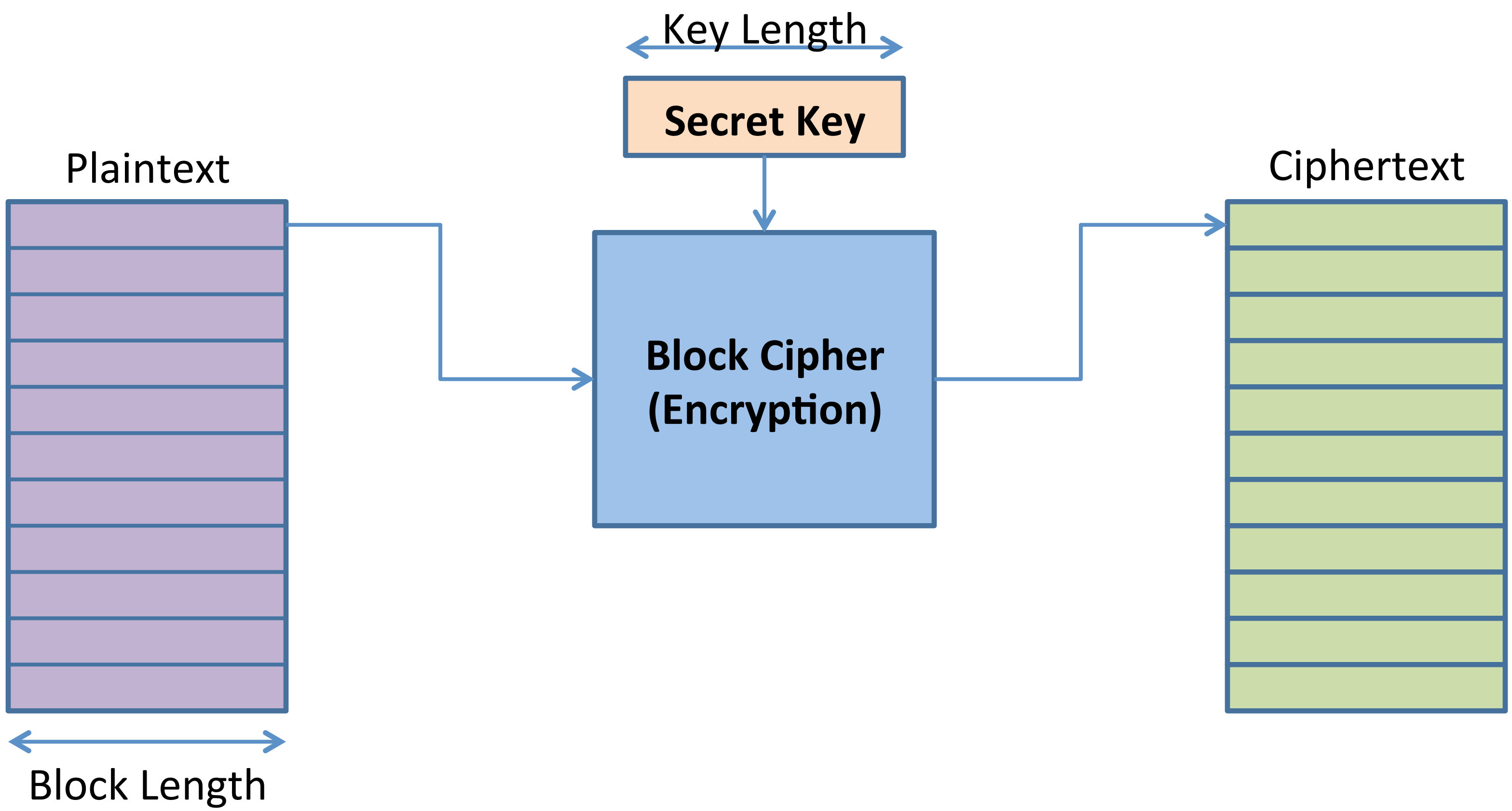
STINSON : chapters 3

Block Cipher



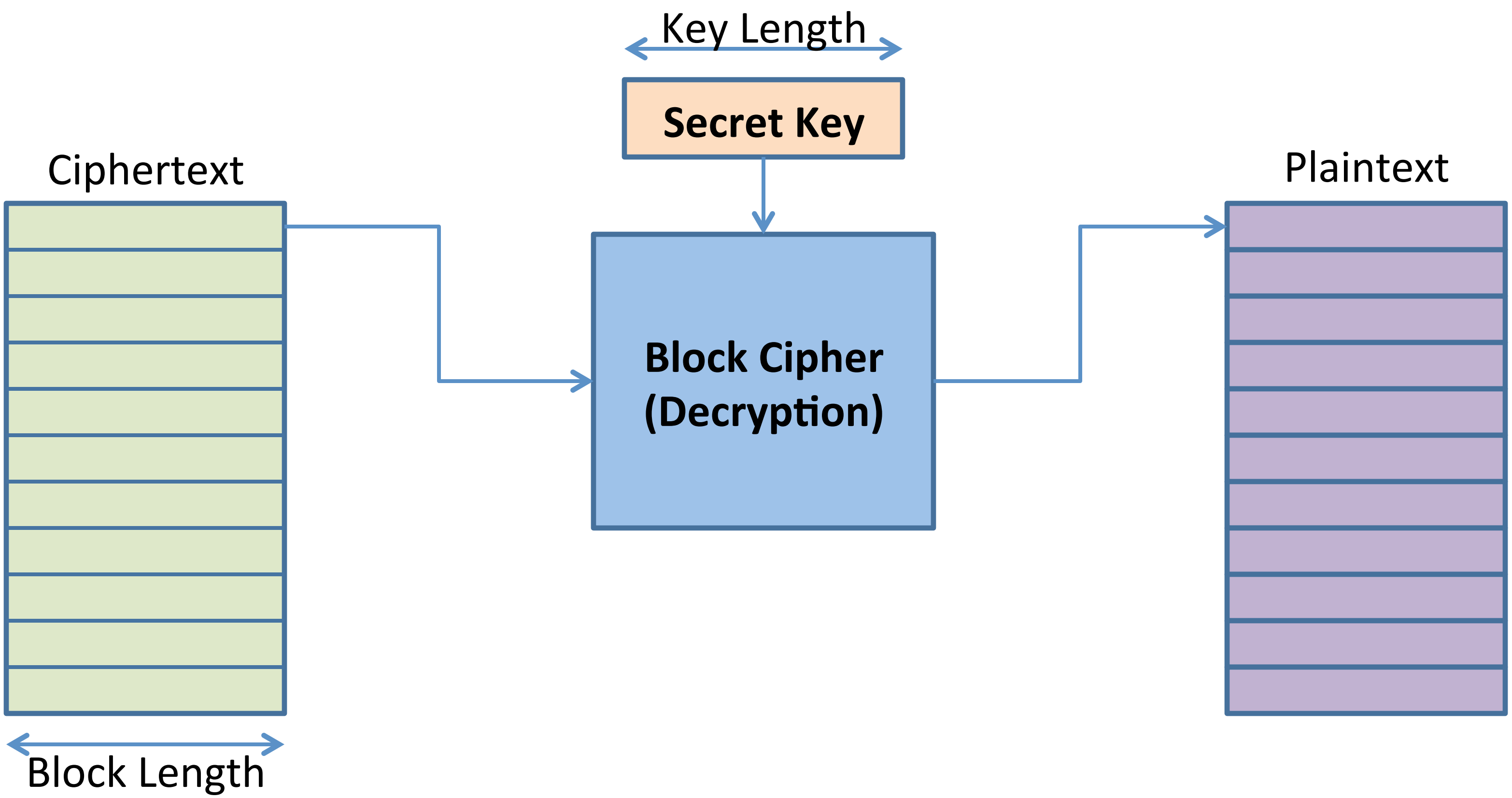
Encryption key is the same as the decryption key ($K_E = K_D$)

Block Cipher : Encryption



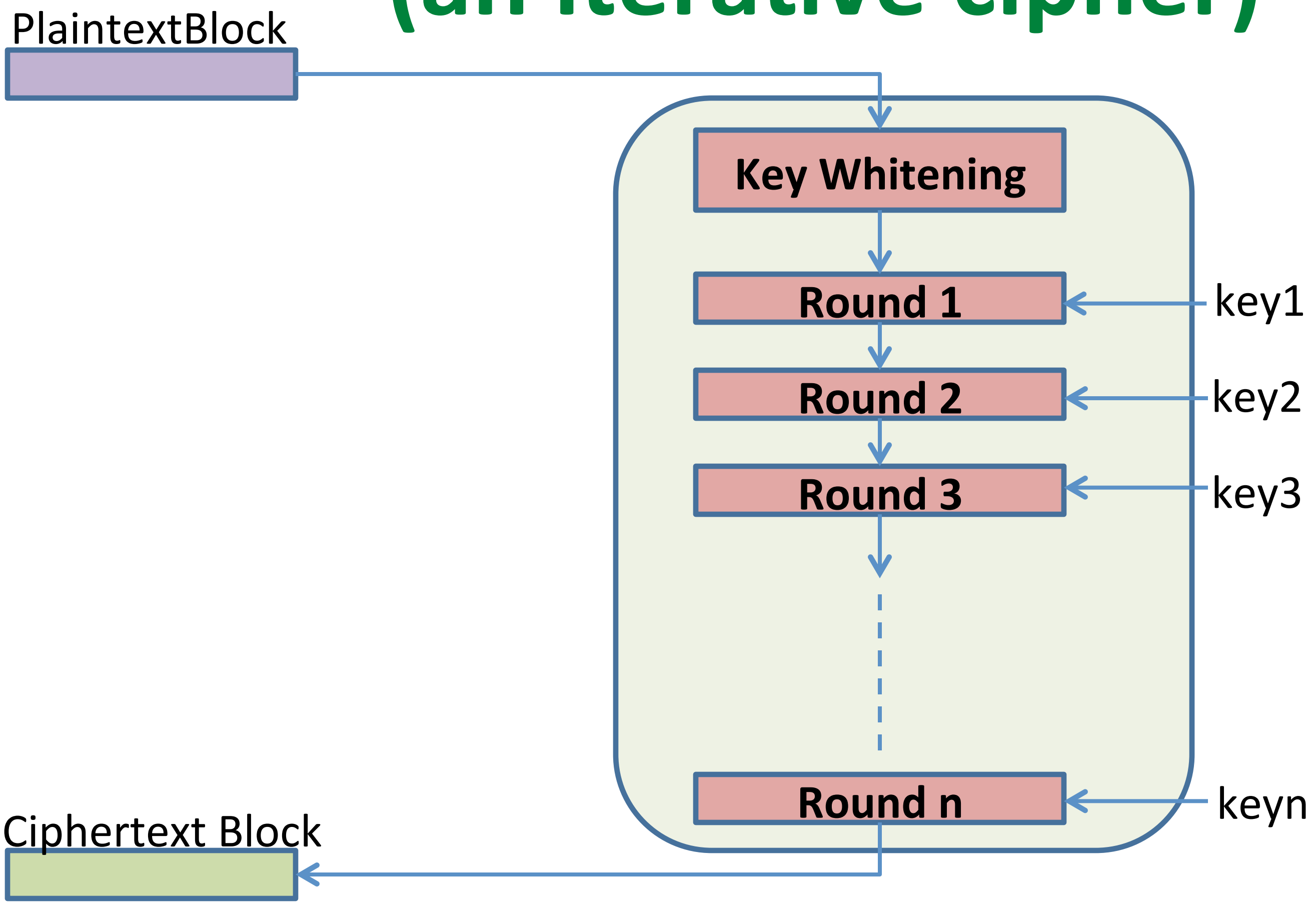
- A block cipher encryption algorithm encrypts n bits of plaintext at a time
- May need to pad the plaintext if necessary
- $y = e_k(x)$

Block Cipher : Decryption



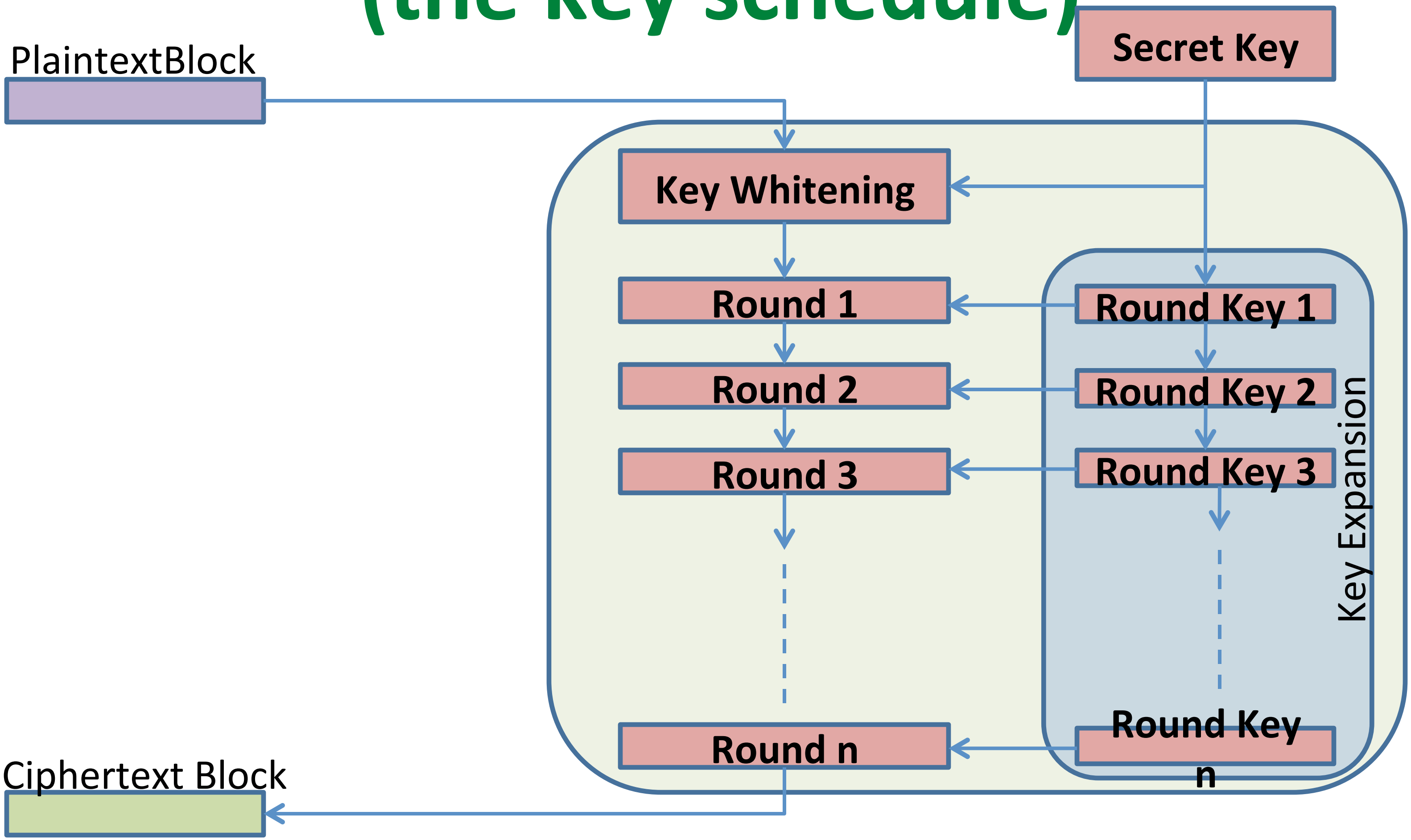
- A block cipher decryption algorithm recovers the plaintext from the ciphertext.
- $x = d_k(y)$

Inside the Block Cipher (an iterative cipher)



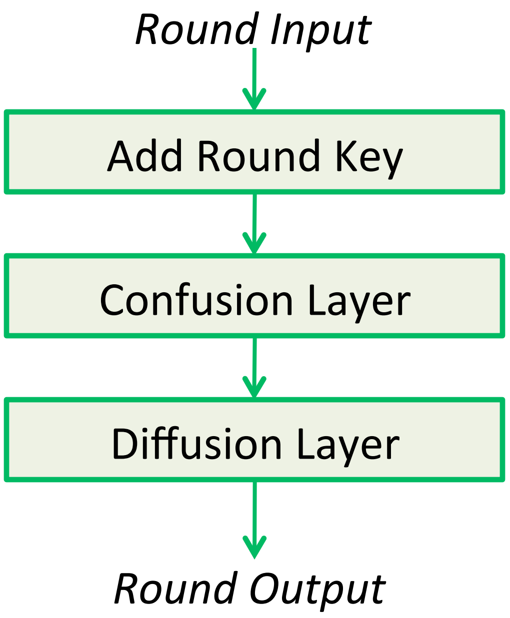
- Each round has the same endomorphic cryptosystem, which takes a key and produces an intermediate ouput
- Size of the key is huge… much larger than the block size.

Inside the Block Cipher (the key schedule)



- A single secret key of fixed size used to generate 'round keys' for each round

Inside the Round Function



- ***Add Round key :***

  Mixing operation between the round input and the round key.
  typically, an ex-or operation

- ***Confusion layer :***

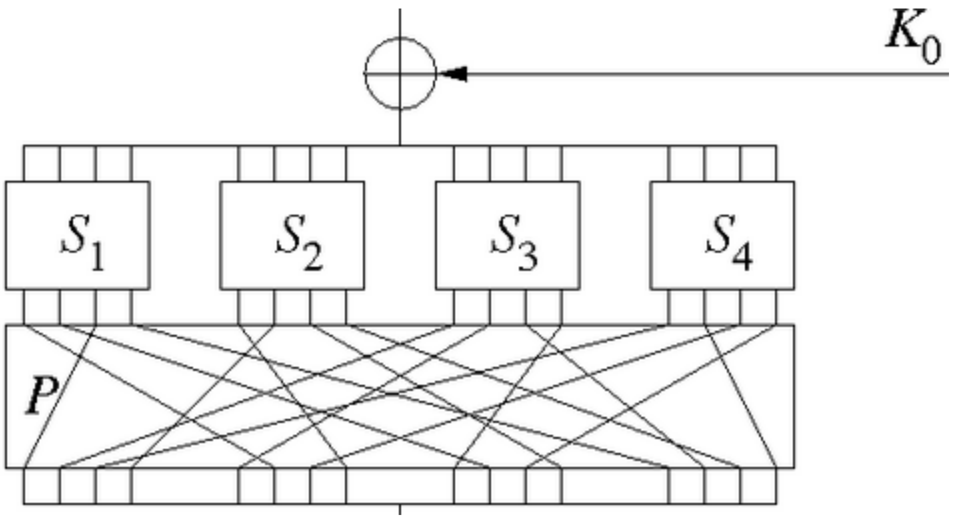
  Makes the relationship between round input and output complex.

- ***Diffusion layer :***
  dissipate the round input.
  Avalanche effect : A single bit change in the round input should cause huge changes in the output.

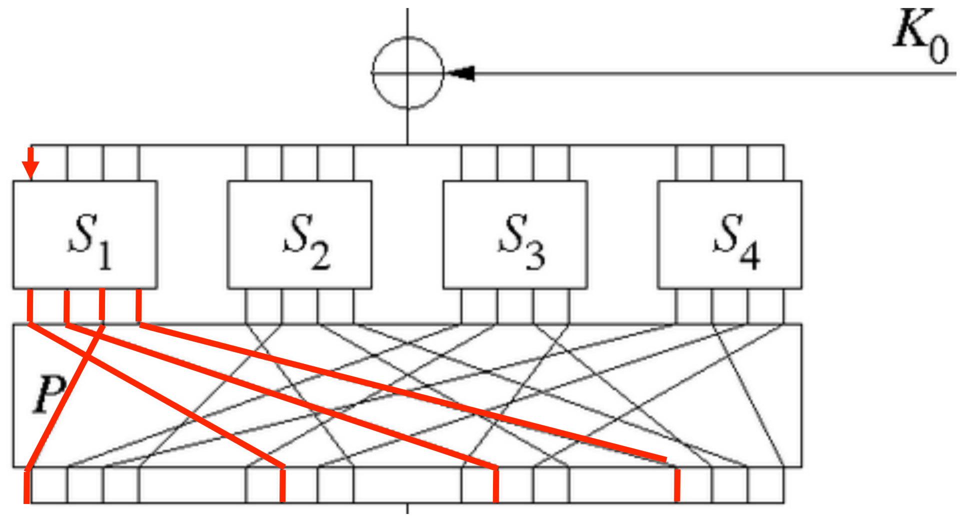
  Makes it difficult for the attacker to pick out some bits over the others (think Hill cipher)

Achieving Confusion and Diffusion
(Substitution-Permutation Networks)

- Confusion achieved by small substitution functions
- Diffusion achieved by diffusion functions
  - Permutations
  - Linear Transformations



Diffusion with Permutations



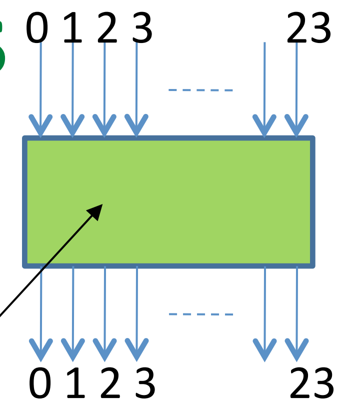
- Spreads the output of one s-box to other s-boxes
- Thus causing a diffusion.
  - A single bit change in one input (before S1 for instance) affects four inputs of the next round
- Bit wise permutations efficient in hardware but not in software implementations

Permutation Layer Types

- straight (24x24)

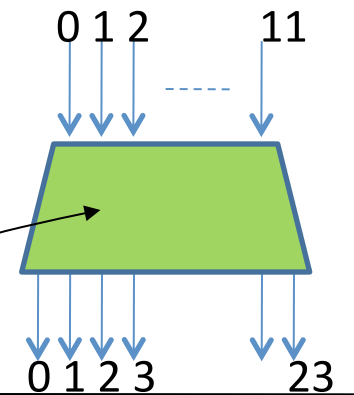
0^{th} bit of input goes to 1^{st} bit of output
1^{st} bit of input goes to 15^{th} bit of output

01	15	02	13	06	17	03	19	09	04	21	11
14	05	12	16	18	07	24	10	23	08	22	20



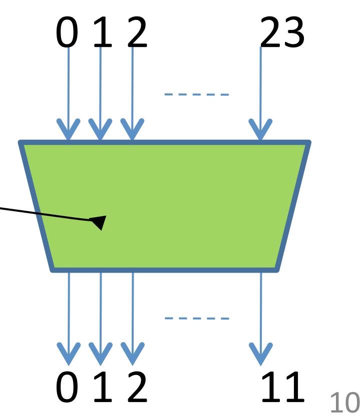
- expansion (12x24)

01	03	02	01	06	17	03	07	09	04	09	11
02	05	12	04	06	07	12	10	11	08	10	08



- compression (24x12)

01	15	02	13	06	17	03	19	09	04	21	11



Permutation Layer (more variants)

- Common permutation operations which are used in block ciphers
  - circular shift
    - Circular shift input N bits to right (or left)
  - swap
    - Special case of circular shift with shift = N/2

Diffusion with Linear Transformation

- Linear combination of the inputs (can be done byte wise; more software friendly, as no bit manipulations needed)

$$\begin{bmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{bmatrix} * \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix}$$

Example.
The AES mix column operation

- How to choose the linear transformation in the Permutation layer?
  - Need to have good diffusion properties
  - Should have Maximum Branch Number

$$BranchNumber = MIN_{(a \neq 0)}(W(a) + W(F(a)))$$

Branch Number

$$BranchNumber = MIN_{(a \neq 0)}(W(a) + W(F(a)))$$

- **Byte Vector :** Number of non-zero input bytes
- W(a) : Byte vector of input (i.e. non-zero bytes in a)
- W(F(a)) : Byte vector of output (i.e. non-zero bytes in the output)

$$\begin{bmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{bmatrix} * \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix}$$

Example.
The AES mix column operation

- **example:** AES mix column matrix has a branch number of 5
  - 1 non-zero byte in input causes all 4 bytes of output to change
  - 2 non-zero byte in input causes at-least 3 bytes of output to change (and so on…)

Substitution Layer (Sbox)

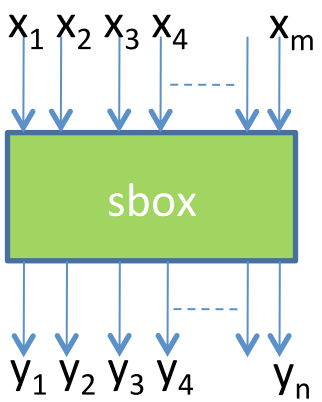
- A lot of the block cipher's security rests with this.
- Replaces its input with another

z	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
$\pi_S(z)$	E	4	D	1	2	F	B	8	3	A	6	C	5	9	0	7

- As with the permutation layer, can be
  straight sbox (mxm)
  expansion sbox (mxn, m<n)
  compression sbox (mxn, m>n)

Sboxes

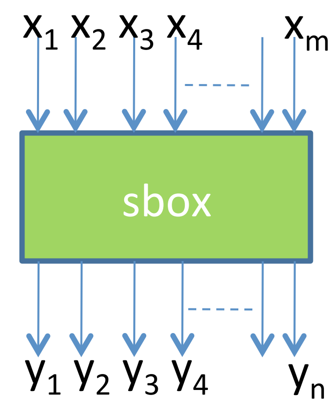
- In an s-box each output bit can be represented as a function of its input bits



$$
\begin{aligned}
y_1 &= f_1(x_1, x_2, x_3, \cdots, x_m) \\
y_2 &= f_2(x_1, x_2, x_3, \cdots, x_m) \\
y_3 &= f_3(x_1, x_2, x_3, \cdots, x_m) \\
\vdots \quad & \quad \vdots \quad\quad \vdots \quad\quad \vdots \quad\quad \vdots \\
y_n &= f_n(x_1, x_2, x_3, \cdots, x_m)
\end{aligned}
$$

The functions have to be non-linear.
Linear functions are easily reversed.

S-boxes are Non-linear transformations



$$y_1 = a_1x_1 \oplus a_2x_2 \oplus a_3x_3 \oplus \cdots \oplus a_mx_m$$
$$y_2 = a_1x_1 \oplus a_2x_2 \oplus a_3x_3 \oplus \cdots \oplus a_mx_m$$
$$\cdots = \cdots$$
$$y_n = a_1x_1 \oplus a_2x_2 \oplus a_3x_3 \oplus \cdots \oplus a_mx_m$$

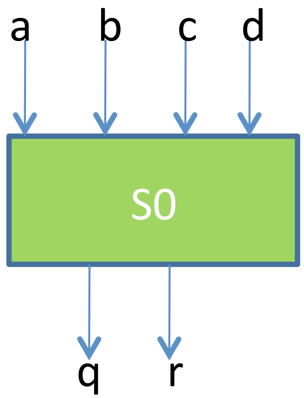
where $a_1, a_2, \cdots a_m \in \{0, 1\}$

- Non-linear s-boxes **do not** have equations like the above.
- Instead they non-linear equations as follows

$$y_1 = a_1x_1 \oplus a_2x_1x_2 \oplus a_3x_1x_5x_2 \cdots$$

CR

example : Simplified DES SBox



$$y = S0(x)$$
$$q \parallel r = S0[a \parallel d][b \parallel c]$$

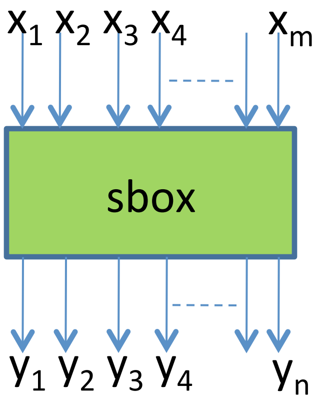
$$S0 = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \end{array} \begin{array}{c} \begin{array}{cccc} 0 & 1 & 2 & 3 \end{array} \\ \begin{bmatrix} 1 & 0 & 3 & 2 \\ 3 & 2 & 1 & 0 \\ 0 & 2 & 1 & 3 \\ 3 & 1 & 3 & 2 \end{bmatrix} \end{array}$$

Non-linear equations for S0

$$q = abcd + ab + ac + b + d$$
$$r = abcd + abd + ab + ac + ad + a + c + 1$$

Why Non-linearity?

- We want to make it difficult for reversing an s-box: i.e. determine x from y



  - Solving linear equations can be done in polynomial time
  - Solving non-linear equation is NP hard

- Note the difference with the permutation layer, which is a linear layer. The main purpose of the permutation layer is to provide diffusion and not to confuse!

ex-or (An Important Operation)

- Used considerably for key addition

  - Ex-or ($\oplus$) is a binary operation, which results in 1 when both inputs have a different value. Otherwise 0
  - Application of Ex-or in ciphers
    - During Encryption : $p \oplus k = c$
    - During decryption : $c \oplus k = p$

  The same operation can be used during encryption as well as decryption

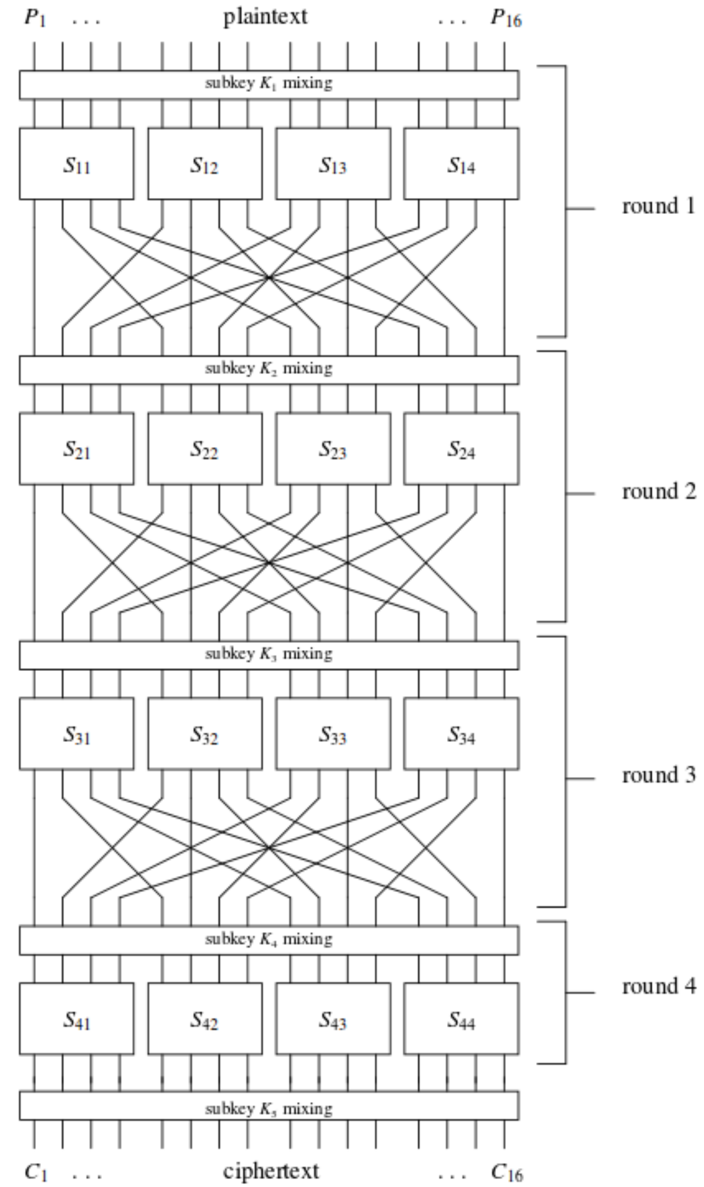
Block Cipher Design Techniques

- Substitution-Permutation Networks (SPN)
  - AES, PRESENT, SHARK
- Feistel Ciphers
  - DES, CLEFIA, SERPENT, RC5, … and many more

A Four Round SPN Block Cipher

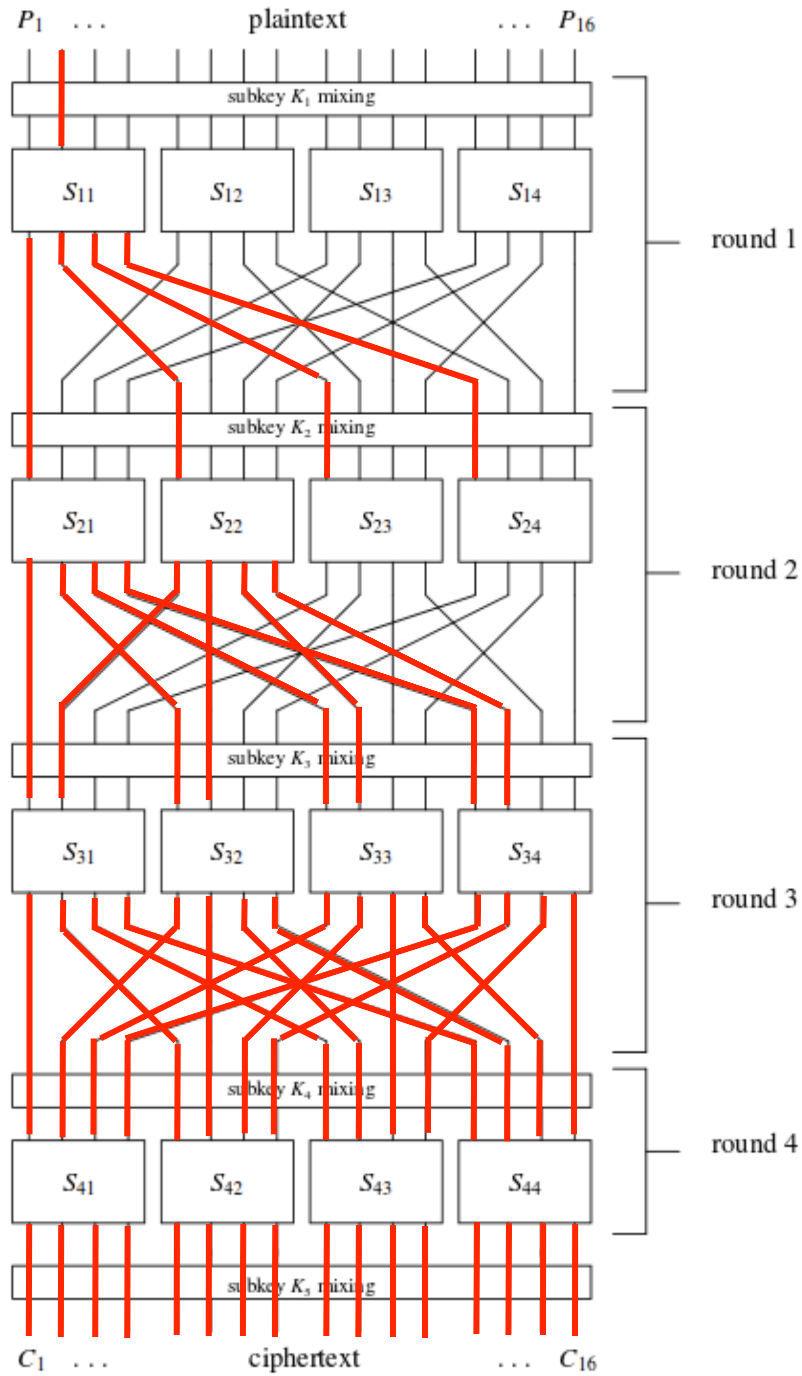
- An SPN block cipher contains repeating rounds of
  - Key addition
    - Add randomization
  - Substitution
    - A non-linear layer
  - Diffusion
    - A linear layer for spreading
- The repeating randomization, non-linear and linear layers makes it difficult to cryptanalyse
- Used in ciphers such as
  - AES (Advanced Encryption Standard)
  - PRESENT (The Light weight block cipher standard)

SPN: Substitution Permutation Network



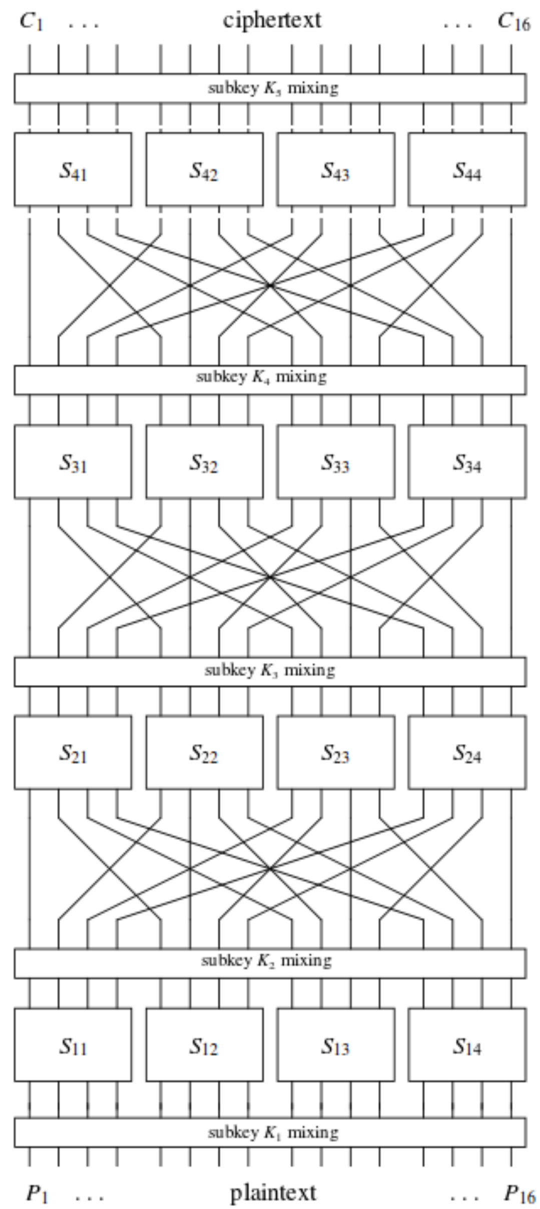
Diffusion in the SPN

- A single bit of plaintext gets diffused to all bits of the ciphertext.
- If a single bit in the plaintext is flipped
  - Each bit of the ciphertext will flip with probability 1/2
  - In other words, half the bits of the ciphertext will flip.
- If, even a single bit of the key is wrong, half the bits of the ciphertext is flipped



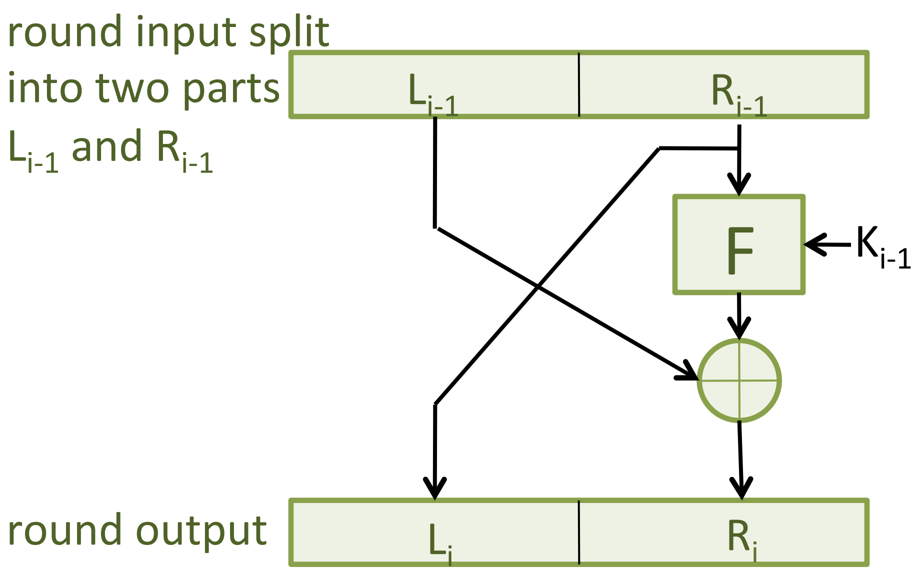
Decryption

- Is the reverse process
  - Start with the ciphertext and do all operations in the reverse order
  - The round keys are applied in the reverse order
  - Permutation layer should be inverse
  - Substitution (S-boxes) should be inverse
    - This also means that the inverse of the s-box should exist



Feistel Ciphers

- A popular technique for designing block ciphers
  - Examples: DES, RC5, CLEFIA,
- Does not require invertible substitution and permutation layers



Encryption

$$L_i = R_i$$

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

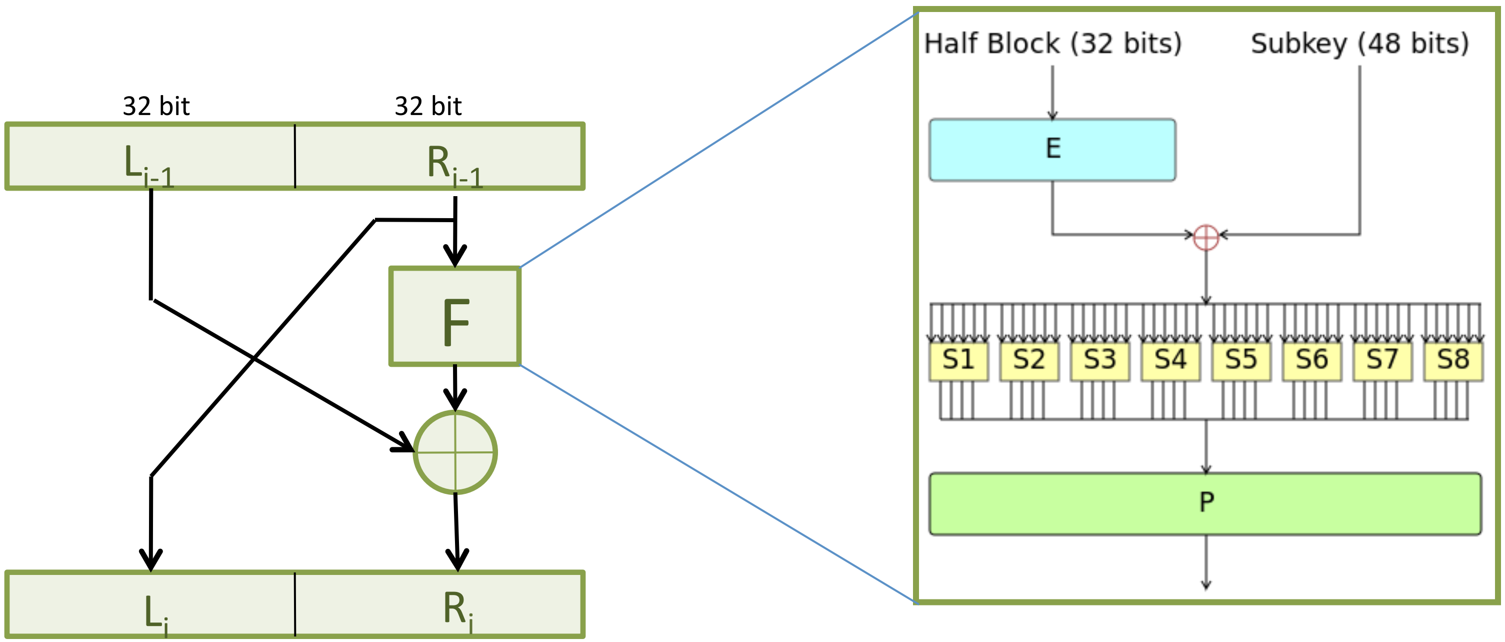
Decryption

$$R_{i-1} = L_i$$

$$L_{i-1} = R_i \oplus F(L_{i-1}, K_{i-1})$$

What does F contain?

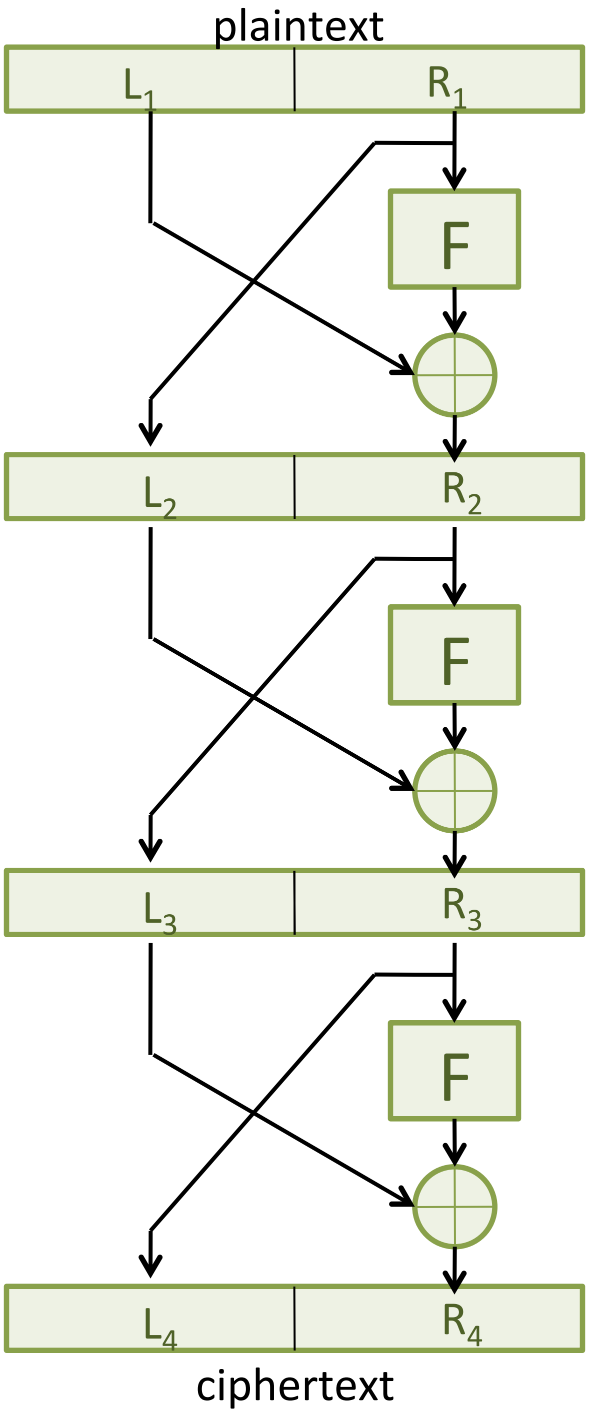
- contains : key mixing, substitution, permutation
- A single round of DES



the sboxes (S1 to S8) are 6x4… they are not invertible

3 round Fiestel cipher

- Iterative

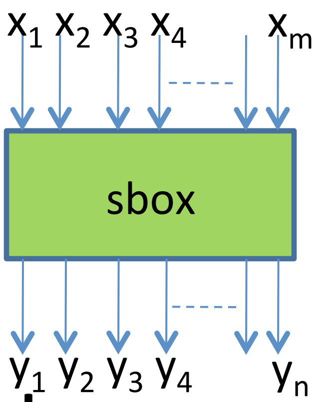


Linear Cryptanalysis

Non-linearity in S-boxes

- In the 1970s, cryptographers took a lot of care in designing s-boxes
  - each output bit of the s-box was the output of a complex non-linear function of the input bits. Like this

$$y_1 = a_1 x_1 \oplus a_2 x_1 x_2 \oplus a_3 x_1 x_5 x_2 \cdots$$



  - also, the value of each output bit was **un-biased**

    i.e. $\Pr[y_i = 0] = \Pr[y_i = 1] = \frac{1}{2} \qquad \text{for } 1 \le i \le n$

    This meant that it was difficult to infer anything about x from an output bit

However….

CR

Linear Approximations

- they overlooked about linear combinations of the s-box output which turned out to be **biased**...such as

$$\Pr[y_1 \oplus x_1 \oplus x_5 \oplus x_7 = 0] << \tfrac{1}{2} \quad or$$

$$\Pr[y_1 \oplus x_1 \oplus x_5 \oplus x_7 = 1] >> \tfrac{1}{2}$$

low probability of occurrence

high probability of occurrence

- This bias was exploited by Mitsuru Matsui in 1993 to attack DES. The attack was known as linear cryptanalysis
  - it is a known plaintext attack
  - required 2^{43} known plaintext-ciphertext pairs to break DES

background needed for the understanding the attack…

CR

Bias
(A measure of deviation from uniform randomness)

- Consider $X_1, X_2, \ldots$ discrete **independent** random variables over {0,1}

- Let $\mathbf{Pr}[X_i = 0] = p_i$, thus $\mathbf{Pr}[X_i = 1] = 1 - p_i$ for i=1,2,3,….

- Due to independence, the joint probability is obtained by simply multiplying. Thus for i ≠ j,

$$\mathbf{Pr}[X_i = 0, X_j = 0] = p_i p_j \qquad \mathbf{Pr}[X_i = 1, X_j = 0] = (1 - p_i)p_j,$$
$$\mathbf{Pr}[X_i = 0, X_j = 1] = p_i(1 - p_j) \qquad \mathbf{Pr}[X_i = 1, X_j = 1] = (1 - p_i)(1 - p_j).$$

- Consider discrete random variables $X_i \oplus X_j$ where i ≠ j

$$\mathbf{Pr}[X_i \oplus X_j = 0] = p_i p_j + (1 - p_i)(1 - p_j)$$
$$\mathbf{Pr}[X_i \oplus X_j = 1] = p_i(1 - p_j) + (1 - p_i)p_j.$$

Bias

- Define **bias** of X_i as $\epsilon_i = p_i - \frac{1}{2}.$

- Some properties of the bias

1 $-\frac{1}{2} \leq \epsilon_i \leq \frac{1}{2},$

2 $\mathbf{Pr}[\mathbf{X_i} = 0] = \frac{1}{2} + \epsilon_i,$

3 $\mathbf{Pr}[\mathbf{X_i} = 1] = \frac{1}{2} - \epsilon_i$

4
$$\Pr[X_i \oplus X_j = 0] = \Pr[X_i = 0]\Pr[X_j = 0] + \Pr[X_i = 1]\Pr[X_j = 1]$$
$$= \left(\frac{1}{2} + \varepsilon_i\right)\left(\frac{1}{2} + \varepsilon_j\right) + \left(\frac{1}{2} - \varepsilon_i\right)\left(\frac{1}{2} - \varepsilon_j\right) = \left(\frac{1}{2} + 2\varepsilon_i\varepsilon_j\right)$$

- If the bias is 0 then X_i can take values of 0 or 1 with equal probability
  The further the bias is from 0 (ie. close to ±1/2) then X_i takes 0 with higher (or lower) probability
- The bias is therefore a measure of the randomness

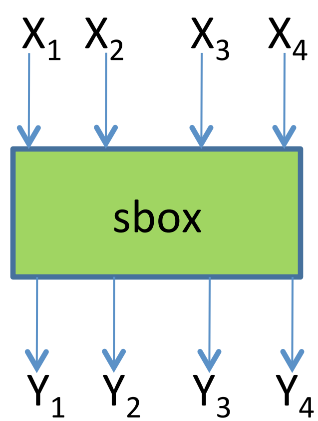
CR

Linear Approximations of an s-box

How to construct?

z	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
$\pi_S(z)$	E	4	D	1	2	F	B	8	3	A	6	C	5	9	0	7

X_1	X_2	X_3	X_4	Y_1	Y_2	Y_3	Y_4
0	0	0	0	1	1	1	0
0	0	0	1	0	1	0	0
0	0	1	0	1	1	0	1
0	0	1	1	0	0	0	1
0	1	0	0	0	0	1	0
0	1	0	1	1	1	1	1
0	1	1	0	1	0	1	1
0	1	1	1	1	0	0	0
1	0	0	0	0	0	1	1
1	0	0	1	1	0	1	0
1	0	1	0	0	1	1	0
1	0	1	1	1	1	0	0
1	1	0	0	0	1	0	1
1	1	0	1	1	0	0	1
1	1	1	0	0	0	0	0
1	1	1	1	0	1	1	1



Represent the s-box in binary as in the following table

Linear Approximations of an s-box

Consider a linear combination of inputs and ouputs

For example $X_1 \oplus X_4 \oplus Y_2$ and fill in the truth table

X_1	X_4	Y_2	
0	0	1	1
0	1	1	0
0	0	1	1
0	1	0	1
0	0	0	0
0	1	1	0
0	0	0	0
0	1	0	1
1	0	0	1
1	1	0	0
1	0	1	0
1	1	1	1
1	0	1	0
1	1	0	0
1	0	0	1
1	1	1	1

#1s = 8
#0s = 8

$$p = \Pr[X_1 \oplus X_4 \oplus Y_2 = 0] = 1/2$$

$$\varepsilon = p - \frac{1}{2} = 0$$

unbiased

Linear Approximations of an s-box

Consider a linear combination of inputs and ouputs
for example $X_1 \oplus X_2 \oplus X_3 \oplus Y_2$ and fill in the truth table

X_1	X_2	X_3	Y_2	
0	0	0	1	1
0	0	0	1	1
0	0	1	1	0
0	0	1	0	1
0	1	0	0	1
0	1	0	1	0
0	1	1	0	0
0	1	1	0	0
1	0	0	0	1
1	0	0	0	1
1	0	1	1	1
1	0	1	1	1
1	1	0	1	1
1	1	0	0	0
1	1	1	0	1
1	1	1	1	0

#1s = 10
#0s = 6

$$p = \Pr[X_1 \oplus X_2 \oplus X_3 \oplus Y_2 = 0] = 3/8$$

$$\varepsilon = p - \frac{1}{2} = -\frac{1}{8} = -0.125$$

biased

Linear Approximations of an s-box

Consider another example $X_3 \oplus X_4 \oplus Y_1 \oplus Y_4$ and fill in the truth table

	X_3	X_4	Y_1		Y_4	
	0	0	1		0	1
	0	1	0		0	1
	1	0	1		1	0
	1	1	0		1	1
	0	0	0		0	0
	0	1	1		1	1
	1	0	1		1	1
	1	1	1		0	1
	0	0	0		1	1
	0	1	1		0	1
	1	0	0		0	1
	1	1	1		0	1
	0	0	0		1	1
	0	1	1		1	1
	1	0	0		0	1
	1	1	0		1	1

#1s = 14
#0s = 2

$$p = \Pr[X_3 \oplus X_4 \oplus Y_1 \oplus Y_4 = 0] = 1/8$$

$$\varepsilon = p - \frac{1}{2} = -\frac{3}{8} = -.375$$

Highly biased

Linear Approximation Tables

$$\left(\bigoplus_{i=1}^{4} a_i \mathbf{X_i}\right) \oplus \left(\bigoplus_{i=1}^{4} b_i \mathbf{Y_i}\right)$$

where $a_i \in \{0,1\}$, $b_i \in \{0,1\}$, $i = 1, 2, 3, 4$.

$$\varepsilon(a,b) = \frac{NL(a,b) - 8}{16}$$

$X_3 \oplus X_4 \oplus Y_1 \oplus Y_4$ → $a = 3$, $b = 9$ (value 2)

$X_1 \oplus X_4 \oplus Y_2$ → $a = 9$, $b = 4$ (value 8)

$X_1 \oplus X_2 \oplus X_3 \oplus Y_2$ → $a = \text{E}$, $b = 4$ (value 6)

	b															
a	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	16	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8
1	8	8	6	6	8	8	6	14	10	10	8	8	10	10	8	8
2	8	8	6	6	8	8	6	6	8	8	10	10	8	8	2	10
3	8	8	8	8	8	8	8	8	10	2	6	6	10	10	6	6
4	8	10	8	6	6	4	6	8	8	6	8	10	10	4	10	8
5	8	6	6	8	6	8	12	10	6	8	4	10	8	6	6	8
6	8	10	6	12	10	8	8	10	8	6	10	12	6	8	8	6
7	8	6	8	10	10	4	10	8	6	8	10	8	12	10	8	10
8	8	8	8	8	8	8	8	8	6	10	10	6	10	6	6	2
9	8	8	6	6	8	8	6	6	4	8	6	10	8	12	10	6
A	8	12	6	10	4	8	10	6	10	10	8	8	10	10	8	8
B	8	12	8	4	12	8	12	8	8	8	8	8	8	8	8	8
C	8	6	12	6	6	8	10	8	10	8	10	12	8	10	8	6
D	8	10	10	8	6	12	8	10	4	6	10	8	10	8	8	10
E	8	10	10	8	6	4	8	10	6	8	8	6	4	10	6	8
F	8	6	4	6	6	8	10	8	8	6	12	6	6	8	10	8

Linear Approximation Table

(captures number of 0s in the truth table)

What does the linear approximations mean

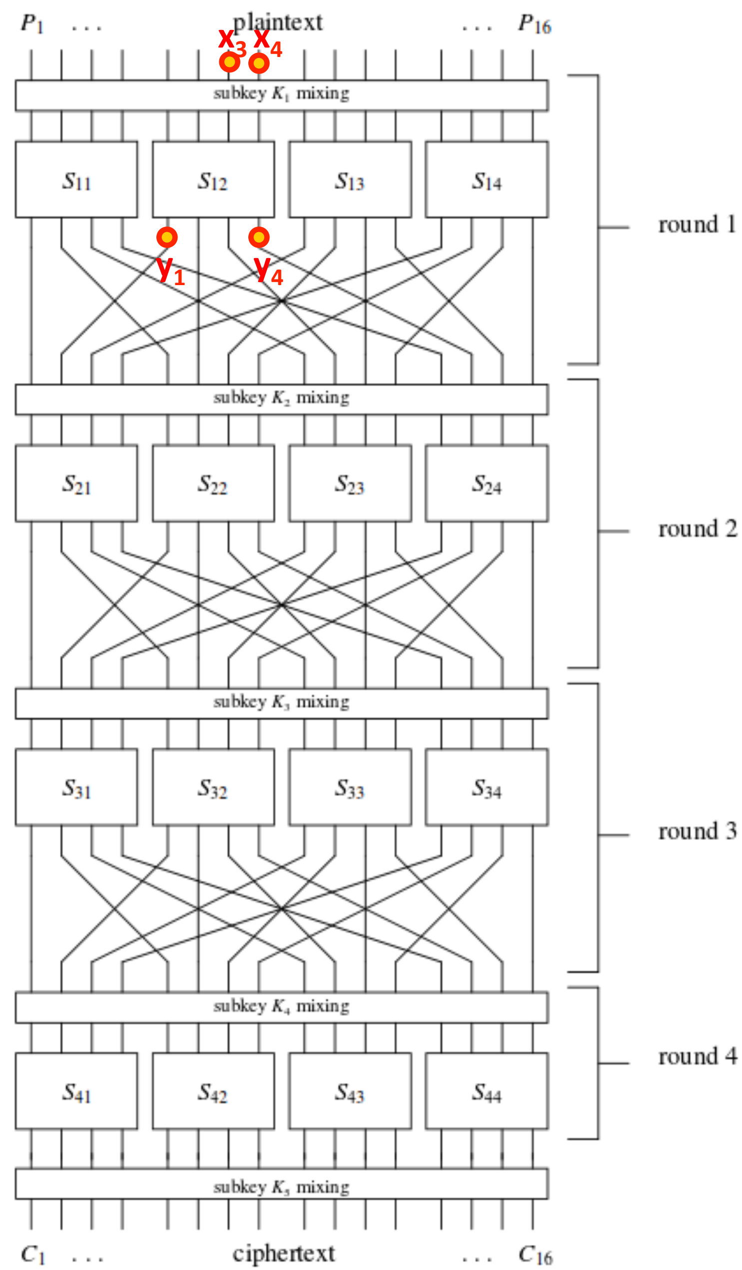
$$X_3 \oplus X_4 \oplus Y_1 \oplus Y_4$$

- If we do the following

```
while(large number of times){
    generate a random plaintext
    z = ex-or(x3,x4,y1,y4)
}
```

- The probability that z takes the value 0 is 1/8

How do we use this fact to attack the block cipher?



Piling-up Lemma

Consider two linear combinations of random variables

$$X_A = X_1 \oplus X_2 \oplus X_3 \qquad \text{having bias } \varepsilon_A$$

$$X_B = X_4 \oplus X_5 \oplus X_6 \qquad \text{having bias } \varepsilon_B$$

What is the bias of $X_A \oplus X_B$?

The resulta nt bias ε_{AB} can be computed by the Pilingup Lemma

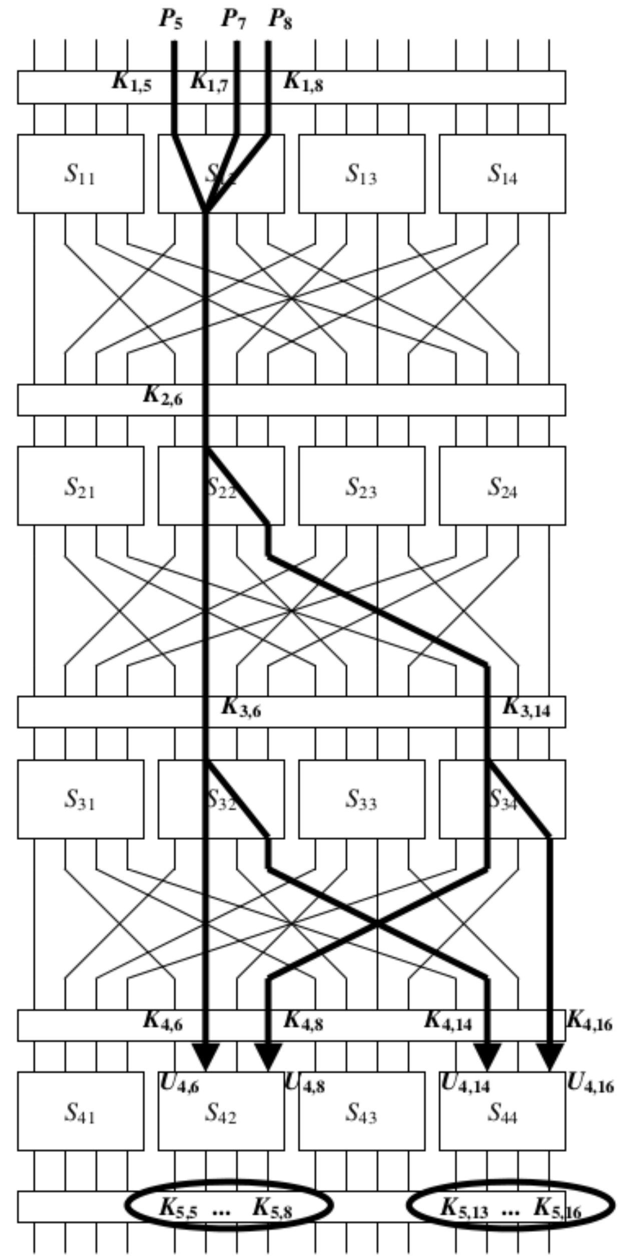
LEMMA **(Piling-up lemma)** *Let $\epsilon_{i_1,i_2,\ldots,i_k}$ denote the bias of the random variable $\mathbf{X_{i_1}} \oplus \cdots \oplus \mathbf{X_{i_k}}$. Then*

$$\epsilon_{i_1,i_2,\ldots,i_k} = 2^{k-1} \prod_{j=1}^{k} \epsilon_{i_j}.$$

Proof by Mathematical Induction

The General Attack Scheme

1. Use piling up lemma to identify linear trails in the cipher, which have high bias.
   – Compute the bias till the pen-ultimate round
2. To determine **k = (**$K_{5,5}$ --- $K_{5,8}$**)** do the following
   a. Guess the value of **k (16 possibilities)**
   b. Compute S^{-1}(**k** ^ c_i) for each ciphertext (we get a distribution)
   c. Determine if the bias matches the theoretical estimates.



Applying Piling-up Lemma for the cipher

Find paths which are highly biased

$\mathbf{T_1} = \mathbf{U^1_5} \oplus \mathbf{U^1_7} \oplus \mathbf{U^1_8} \oplus \mathbf{V^1_6}$ has bias $1/4$

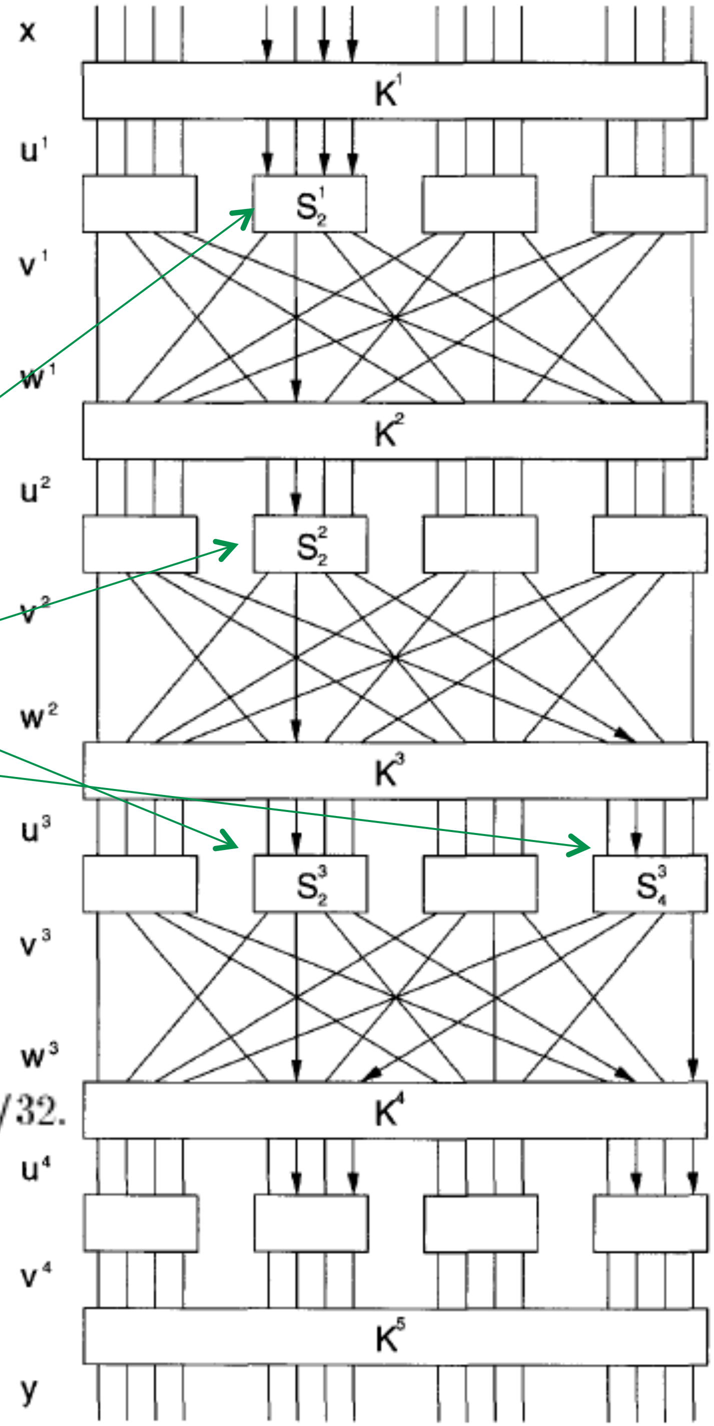
$\mathbf{T_2} = \mathbf{U^2_6} \oplus \mathbf{V^2_6} \oplus \mathbf{V^2_8}$ has bias $-1/4$

$\mathbf{T_3} = \mathbf{U^3_6} \oplus \mathbf{V^3_6} \oplus \mathbf{V^3_8}$ has bias $-1/4$

$\mathbf{T_4} = \mathbf{U^3_{14}} \oplus \mathbf{V^3_{14}} \oplus \mathbf{V^3_{16}}$ has bias $-1/4$

$$\mathbf{T_1} \oplus \mathbf{T_2} \oplus \mathbf{T_3} \oplus \mathbf{T_4}$$

has bias equal to $2^3(1/4)(-1/4)^3 = -1/32$.



$a = 1011,\ b = 0100, N_L = 12,$
$\varepsilon = 1/4$

$a = 0100,\ b = 0101, N_L = 4,$
$\varepsilon = -1/4$

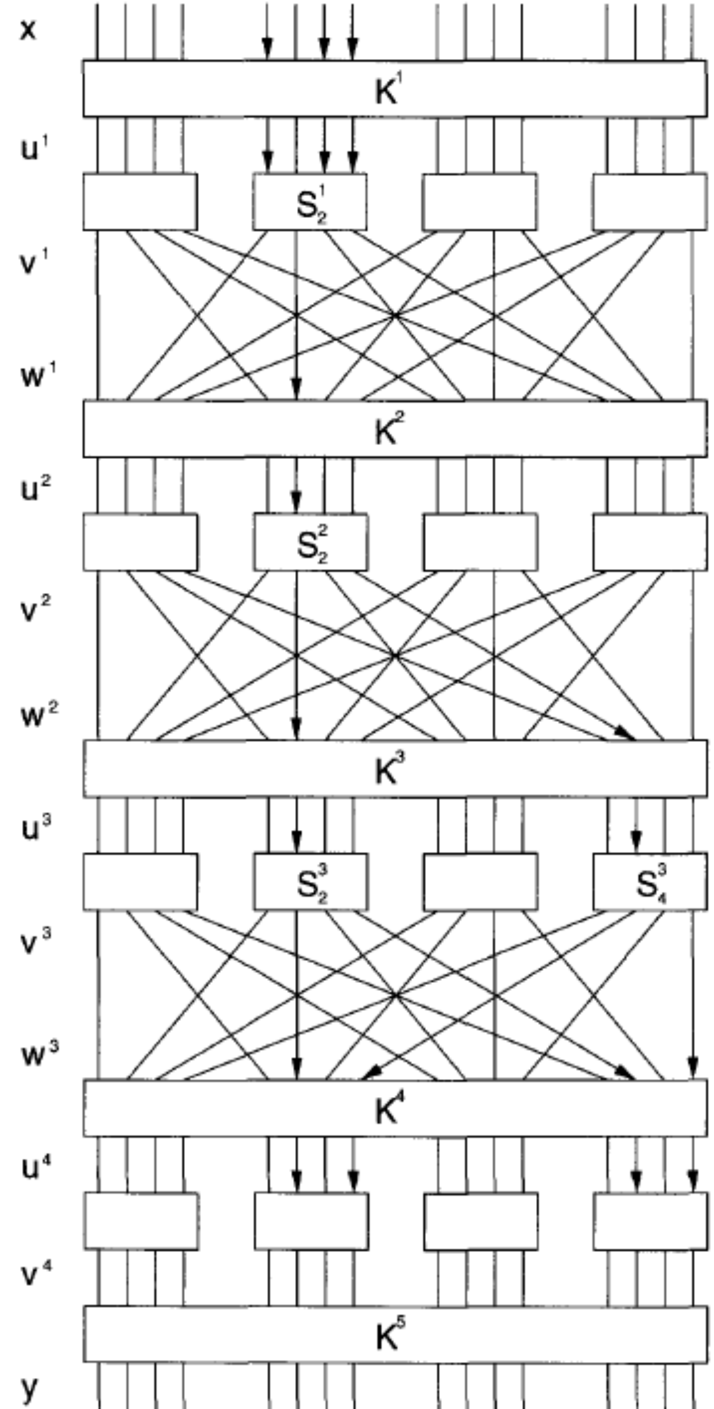
$a = 0100,\ b = 0101, N_L = 4,$
$\varepsilon = -1/4$

$$\begin{aligned}
T_1 &= U^1_5 \oplus U^1_7 \oplus U^1_8 \oplus V^1_6 = X_5 \oplus K^1_5 \oplus X_7 \oplus K^1_7 \oplus X_8 \oplus K^1_8 \oplus V^1_6\\
T_2 &= U^2_6 \oplus V^2_6 \oplus V^2_8 \qquad = V^1_6 \oplus K^2_6 \oplus V^2_6 \oplus V^2_8\\
T_3 &= U^3_6 \oplus V^3_6 \oplus V^3_8 \qquad = V^2_6 \oplus K^3_6 \oplus V^3_6 \oplus V^3_8\\
T_4 &= U^3_{14} \oplus V^3_{14} \oplus V^3_{16} \qquad = V^2_8 \oplus K^3_{14} \oplus V^3_{14} \oplus V^3_{16}.
\end{aligned}$$

$$T_1 \oplus T_2 \oplus T_3 \oplus T_4$$

$$X_5 \oplus X_7 \oplus X_8 \oplus V^3_6 \oplus V^3_8 \oplus V^3_{14} \oplus V^3_{16} \oplus K^1_5 \oplus K^1_7 \oplus K^1_8 \oplus K^2_6 \oplus K^3_6 \oplus K^3_{14}$$

has bias equal to $2^3(1/4)(-1/4)^3 = -1/32$.



$$X_5 \oplus X_7 \oplus X_8 \oplus V_6^3 \oplus V_8^3 \oplus V_{14}^3 \oplus V_{16}^3 \oplus K_5^1 \oplus K_7^1 \oplus K_8^1 \oplus K_6^2 \oplus K_6^3 \oplus K_{14}^3$$

From the cipher

$$V_6^3 = U_6^4 \oplus K_6^4$$
$$V_8^3 = U_{14}^4 \oplus K_{14}^4$$
$$V_{14}^3 = U_8^4 \oplus K_8^4$$
$$V_{16}^3 = U_{16}^4 \oplus K_{16}^4$$

Thus,

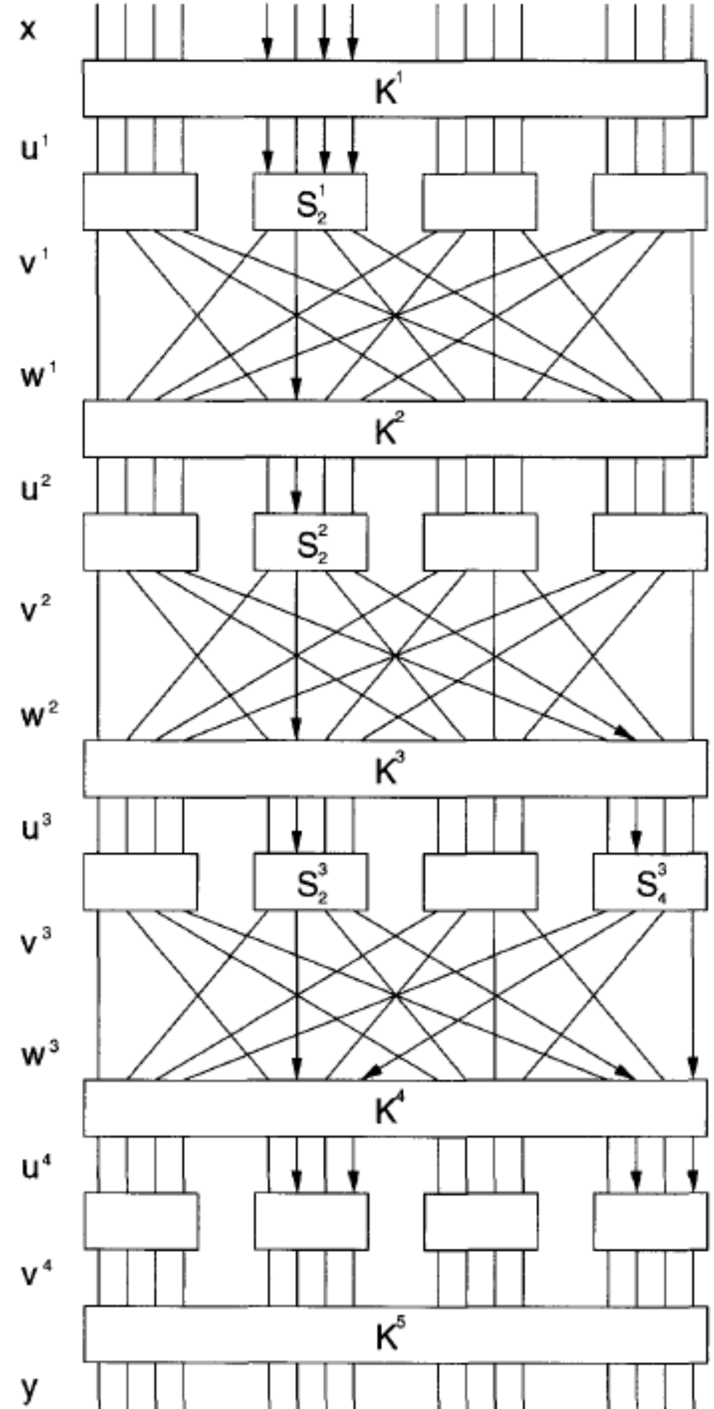
$$X_5 \oplus X_7 \oplus X_8 \oplus U_6^4 \oplus U_8^4 \oplus U_{14}^4 \oplus U_{16}^4 \oplus K_5^1 \oplus K_7^1 \oplus K_8^1 \oplus K_6^2 \oplus K_6^3 \oplus K_{14}^3 \oplus K_6^4 \oplus K_8^4 \oplus K_{14}^4 \oplus K_{16}^4$$

has bias equal to $2^3(1/4)(-1/4)^3 = -1/32$.

Now,, the key part is a constant (either 0 or 1)

$$K_5^1 \oplus K_7^1 \oplus K_8^1 \oplus K_6^2 \oplus K_6^3 \oplus K_{14}^3 \oplus K_6^4 \oplus K_8^4 \oplus K_{14}^4 \oplus K_{16}^4$$

Thus, bias of $X_5 \oplus X_7 \oplus X_8 \oplus U_6^4 \oplus U_8^4 \oplus U_{14}^4 \oplus U_{16}^4$
is either +1/32 or -1/32 depending on the key bits

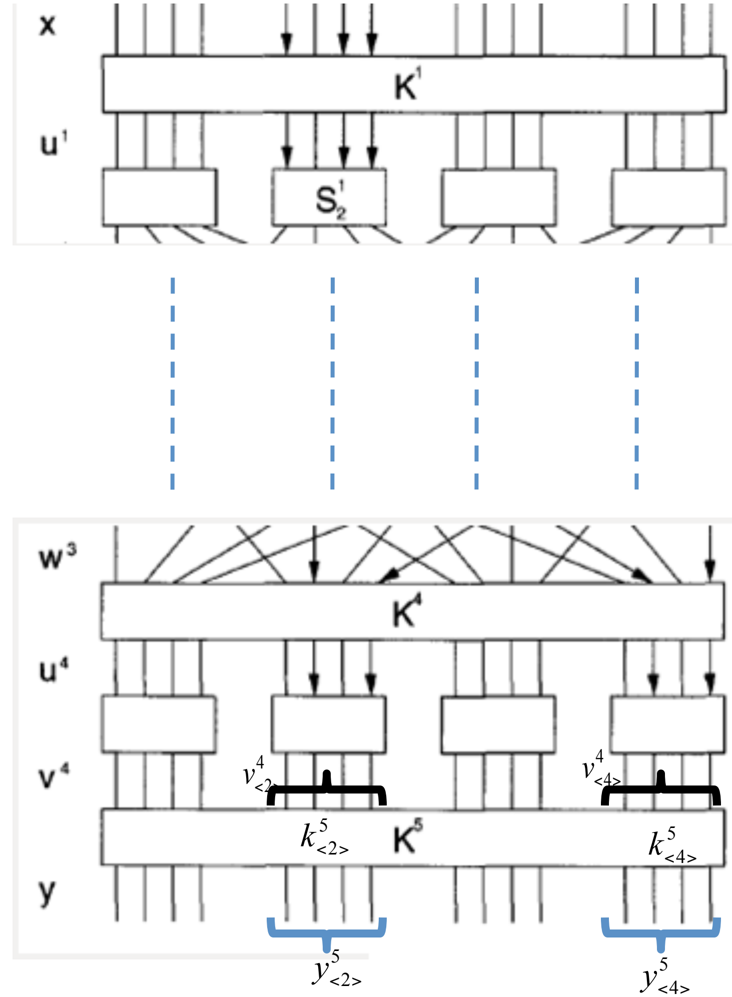


CR

The Linear Cryptanalysis Attack

- The attacker needs
  - A large number of plaintext-ciphertext pairs
    - We denote each pair by (x,y) – x: plaintext, y: ciphertext
    - For the Toy cipher above (approx 8000)
    - For a cipher like DES 2^{48}
  - all plaintexts are encrypted with the same key
- The attack
  1. Guess $k^5_{<2>}$ and $k^5_{<4>}$ (256 possibilities)
  2. For each $y^5_{<2>}$ and $y^5_{<4>}$ compute $v^4_{<2>}$ and $v^4_{<4>}$
  3. Then compute inv-sbox($v^4_{<2>}$) and inv-sbox($v^4_{<4>}$) to obtain $u^4_{<2>}$ and $u^4_{<4>}$
  4. Now compute

$$z \leftarrow x_5 \oplus x_7 \oplus x_8 \oplus u^4_6 \oplus u^4_8 \oplus u^4_{14} \oplus u^4_{16}$$



If the key guess is correct, the bias of z must be ± 1/32

(i.e. z must be 0 (or 1) with probability 1/2 ± 1/32)

If the key guess is wrong, the bias of z must be 0

(i.e. z must be 0 (or 1) with probability 1/2)

The Linear Cryptanalysis Attack

The plaintext-ciphertext pair array

Number of the ptext-ctext pairs

Inverse s-box

Algorithm $\textsc{LinearAttack}(\mathcal{T}, T, \pi_S^{-1})$

for $(L_1, L_2) \leftarrow (0,0)$ **to** (F, F)
  do $Count[L_1, L_2] \leftarrow 0$
for each $(x, y) \in \mathcal{T}$
  do {
    for $(L_1, L_2) \leftarrow (0,0)$ **to** (F, F)
      do {
        $v^4_{<2>} \leftarrow L_1 \oplus y_{<2>}$
        $v^4_{<4>} \leftarrow L_2 \oplus y_{<4>}$
        $u^4_{<2>} \leftarrow \pi_S^{-1}(v^4_{<2>})$
        $u^4_{<4>} \leftarrow \pi_S^{-1}(v^4_{<4>})$
        if $z = 0$
          then $Count[L_1, L_2] \leftarrow Count[L_1, L_2] + 1$ }}
$max \leftarrow -1$
for $(L_1, L_2) \leftarrow (0,0)$ **to** (F, F)
  do {
    $Count[L_1, L_2] \leftarrow |Count[L_1, L_2] - T/2|$
    if $Count[L_1, L_2] > max$
      then { $max \leftarrow Count[L_1, L_2]$
        $maxkey \leftarrow (L_1, L_2)$ }}
output $(maxkey)$

This is the guessed key which varies from 0 to 255.

For a key guess, Count counts how often z=0. For the correct key guess, count should be highest

For each plaintext-ciphertext pair

Compute $u^4_{<2>}$ and $u^4_{<4>}$

Increment count if z=0

Determine most probable key byte of the256 possible keys
The correct key should have max count value
Wrong keys should have count value approximately T/2

Differential Cryptanalysis

Differential Cryptanalysis

- Attributed to Eli Biham and Adi Shamir in CRYPTO'90
  - Althought, the idea was known in the 1970s by IBM (and the NSA)
    - In IBM, this used to be known as T-attack or Tickle attack
- Differential cryptanalysis is a chosen plaintext attack
  - It requires 2^{47} chosen plaintexts to break DES

Differentials

- If we have two Boolean linear equations such as

$$A = a \oplus b \oplus k_1 \oplus k_2 \qquad B = c \oplus d \oplus k_1 \oplus k_2$$

- Then, the differential is their ex-or

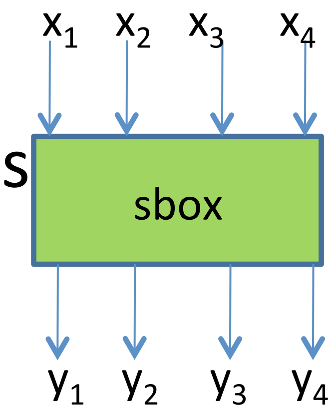
$$A \oplus B = a \oplus b \oplus c \oplus d$$

- Note that the common terms are cancelled out

Differentials of an s-box

z	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
$\pi_S(z)$	E	4	D	1	2	F	B	8	3	A	6	C	5	9	0	7

- Let x and x* be the inputs to an s-box
- Let y and y* be the corresponding outputs



$$\text{Differential Input} \quad : x' = x \oplus x^*$$

$$\text{Differential Output} : y' = y \oplus y^*$$

- If x' is $(1011)_2$:

x	x^*	y	y^*	y'
0000	1011	1110	1100	0010
0001	1010	0100	0110	0010
0010	1001	1101	1010	0111
0011	1000	0001	0011	0010
0100	1111	0010	0111	0101
0101	1110	1111	0000	1111
0110	1101	1011	1001	0010
0111	1100	1000	0101	1101
1000	0011	0011	0001	0010
1001	0010	1010	1101	0111
1010	0001	0110	0100	0010
1011	0000	1100	1110	0010
1100	0111	0101	1000	1101
1101	0110	1001	1011	0010
1110	0101	0000	1111	1111
1111	0100	0111	0010	0101

Differentials of an s-box

If x’ is $(1011)_2$:

x	x^*	y	y^*	y'
0000	1011	1110	1100	0010
0001	1010	0100	0110	0010
0010	1001	1101	1010	0111
0011	1000	0001	0011	0010
0100	1111	0010	0111	0101
0101	1110	1111	0000	1111
0110	1101	1011	1001	0010
0111	1100	1000	0101	1101
1000	0011	0011	0001	0010
1001	0010	1010	1101	0111
1010	0001	0110	0100	0010
1011	0000	1100	1110	0010
1100	0111	0101	1000	1101
1101	0110	1001	1011	0010
1110	0101	0000	1111	1111
1111	0100	0111	0010	0101

0000	0001	0010	0011	0100	0101	0110	0111
0	0	8	0	0	2	0	2

1000	1001	1010	1011	1100	1101	1110	1111
0	0	0	0	0	2	0	2

Note the non-uniformity….. This non-uniformity
Is used in differential cryptanalysis

Differential Distribution Table of the s-box

S-box output difference

S-box input difference

a'	b'															
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	2	0	0	0	2	0	2	4	0	4	2	0	0
2	0	0	0	2	0	6	2	2	0	2	0	0	0	0	2	0
3	0	0	2	0	2	0	0	0	0	4	2	0	2	0	0	4
4	0	0	0	2	0	0	6	0	0	2	0	4	2	0	0	0
5	0	4	0	0	0	2	2	0	0	0	4	0	2	0	0	2
6	0	0	0	4	0	4	0	0	0	0	0	0	2	2	2	2
7	0	0	2	2	2	0	2	0	0	2	2	0	0	0	0	4
8	0	0	0	0	0	0	2	2	0	0	0	4	0	4	2	2
9	0	2	0	0	2	0	0	4	2	0	2	2	2	0	0	0
A	0	2	2	0	0	0	0	0	6	0	0	2	0	0	4	0
B	0	0	8	0	0	2	0	2	0	0	0	0	0	2	0	2
C	0	2	0	0	2	2	2	0	0	0	0	2	0	6	0	0
D	0	4	0	0	0	0	0	4	2	0	2	0	2	0	2	0
E	0	0	2	4	2	0	0	0	6	0	0	0	0	0	2	0
F	0	2	0	0	6	0	0	0	0	4	0	2	0	0	2	0

$$R_p(a', b') = \frac{N_D(a', b')}{2^m}.$$

Probability that output difference Is b' given that input difference is a'

This is known as the ***Propagation Ratio***

$$N_D(x', y') = |\{(x, x^*) \in \Delta(x') : \pi_S(x) \oplus \pi_S(x^*) = y'\}|.$$

Counts the number of times input difference is x' and output difference of the s-box is y'

Differential trails in a cipher

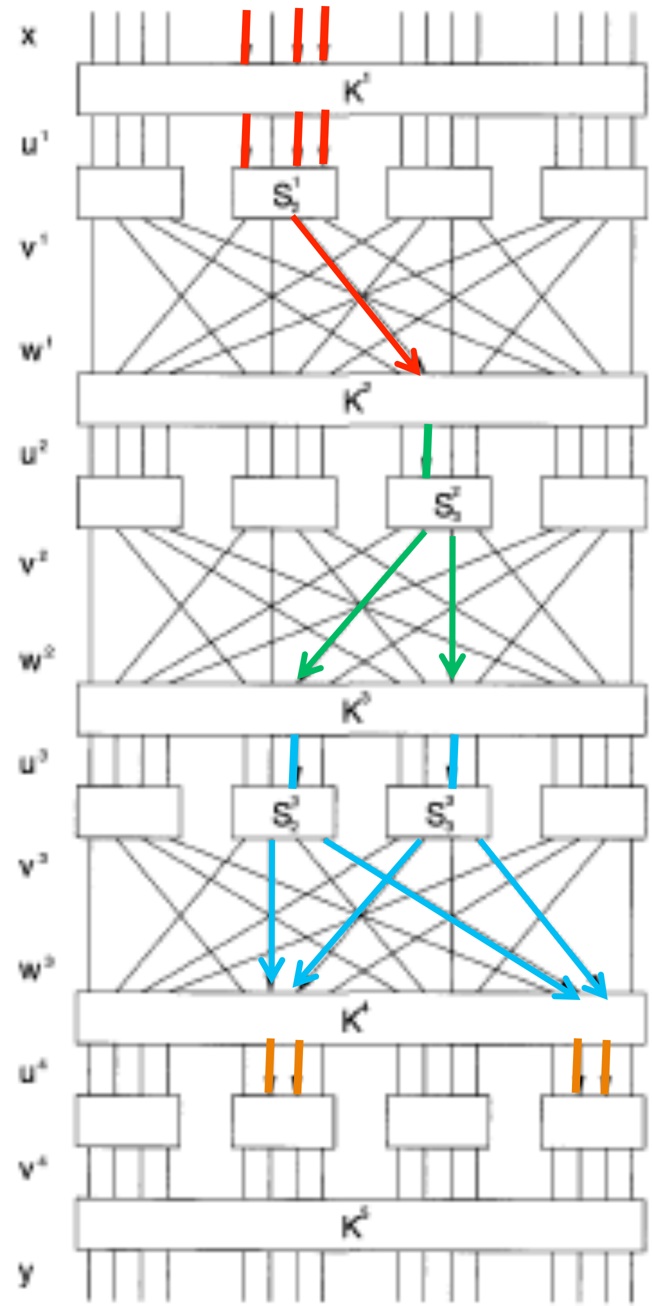
- First note that the differential output y' does not depend on the secret key
- Choose a set of consecutive s-boxes so that differences propagate with high propagation ratio. This is the differential trail.

> - In S_2^1, $R_p(1011, 0010) = 1/2$
> - In S_3^2, $R_p(0100, 0110) = 3/8$
> - In S_2^3, $R_p(0010, 0101) = 3/8$
> - In S_3^3, $R_p(0010, 0101) = 3/8$

- Assuming independence between the s-boxes in the trail, propagation ratio for the trail is the product of individual propagation ratios.

$$R_p(0000\ 1011\ 0000\ 0000, 0000\ 0101\ 0101\ 0000) = \frac{1}{2} \times \left(\frac{3}{8}\right)^3 = \frac{27}{1024}.$$

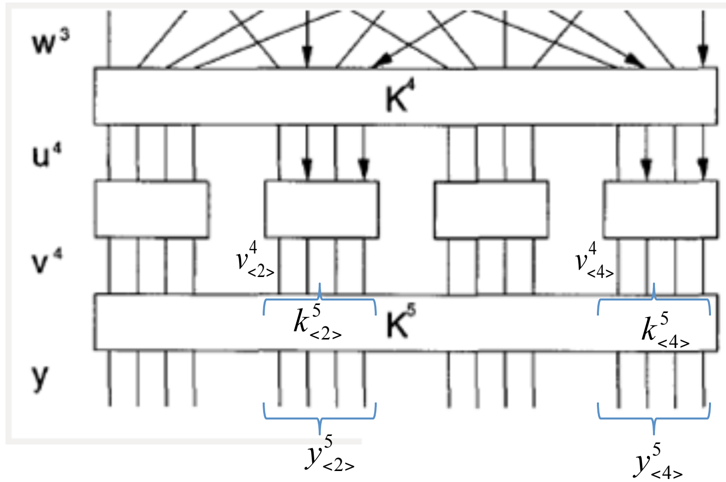
  – This means that, if the input difference is (0000 1011 0000 0000) then the probability that the output difference is (0000 0101 0101 0000) is 27/1024



The Differential Cryptanalysis Attack

$v^4_{<4>}$

- The attacker needs
  - A large number of chosen plaintext-ciphertext pairs encrypted with the same key
- The attack
  1. Guess $k^5_{<2>}$ and $k^5_{<4>}$ (256 possibilities)
  2. Compute $v^4_{<2>}$ and $v^4_{<4>}$ for each plaintext –ciphertext using the guessed key
  3. Compute the difference between the inv-sbox($v^4_{<2>}$) and inv-sbox($v^4_{<4>}$)
  4. Test if the required differential is obtained.



If the key guess is correct, the correct differential will be obtained with a probability of 27/1024

If the key guess is wrong, the differential will be obtained with a probability which is much lower (1/256)

The Differential Cryptanalysis Algorithm

Algorithm 3.3: DIFFERENTIALATTACK$(\mathcal{T}, T, {\pi_S}^{-1})$

```
for (L1, L2) ← (0,0) to (F, F)
  do Count[L1, L2] ← 0
for each (x, y, x*, y*) ∈ 𝒯
  do { if (y<1> = (y<1>)*) and (y<3> = (y<3>)*)
         then { for (L1, L2) ← (0,0) to (F, F)
                  do { v4<2> ← L1 ⊕ y<2>
                       v4<4> ← L2 ⊕ y<4>
                       u4<2> ← πS^-1(v4<2>)
                       u4<4> ← πS^-1(v4<4>)
                       (v4<2>)* ← L1 ⊕ (y<2>)*
                       (v4<4>)* ← L2 ⊕ (y<4>)*
                       (u4<2>)* ← πS^-1((v4<2>)*)
                       (u4<4>)* ← πS^-1((v4<4>)*)
                       (u4<2>)' ← u4<2> ⊕ (u4<2>)*
                       (u4<4>)' ← u4<4> ⊕ (u4<4>)*
                       if ((u4<2>)' = 0110) and ((u4<4>)' = 0110)
                         then Count[L1, L2] ← Count[L1, L2] + 1
max ← -1
for (L1, L2) ← (0,0) to (F, F)
  do { if Count[L1, L2] > max
         then { max ← Count[L1, L2]
                maxkey ← (L1, L2)
output (maxkey)
```

Function inputs are the plaintext-ciphertext Differentials, T is the number of them, and the Inverse of the targeted s-box

The guessed key (L1, L2) : is of 256 values

For each differential, do an initial filtering, and then compute $u^4_{<2>}$ and $u^4_{<4>}$. If these result in the targeted differential 0110, 0110, then increment The count for the corresponding key guess

The values of (L_1, L_2) which has the maximum count Implies, that it is the case where the targeted Differential appears most often. This (L_1, L_2) is the likely key.

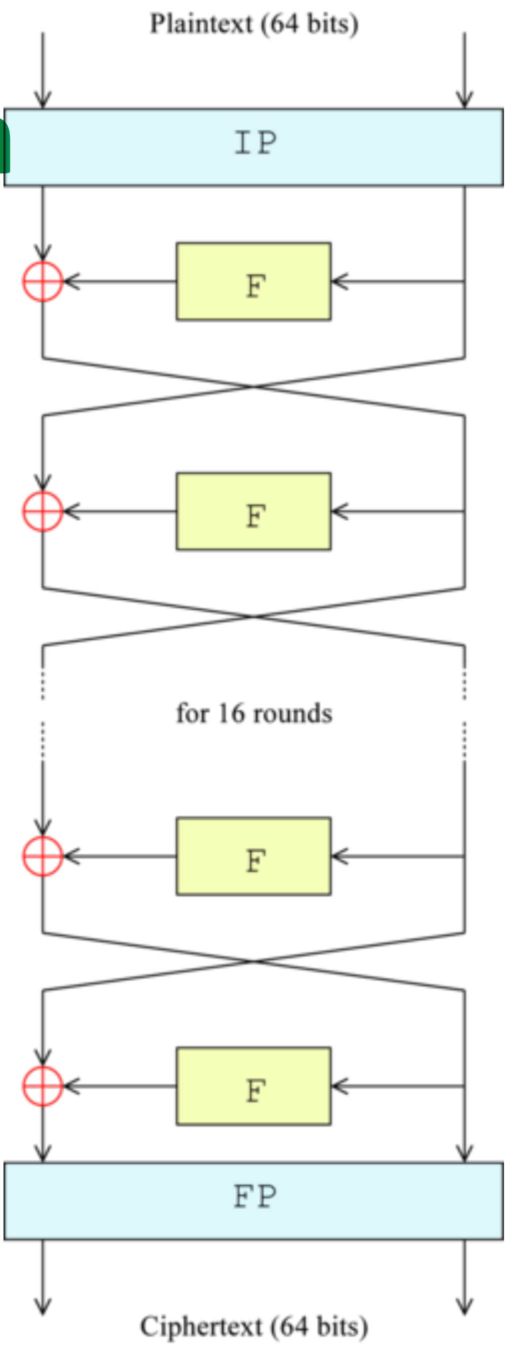
DES
(Data Encryption Standard)

History of DES

- Standardized in 1977 by FIPS , as the standard for data encryption
- Based on a Feistel cipher called Lucifer
  (Lucifer is a Feistel cipher developed by IBM in the early ‘70s)
- NSA made some minor (supposedly controversial) modifications to the Lucifer algorithm
  - Reduced the key size from 64 bits to 56 bits
  - Modifications to the s-boxes

DES Specification

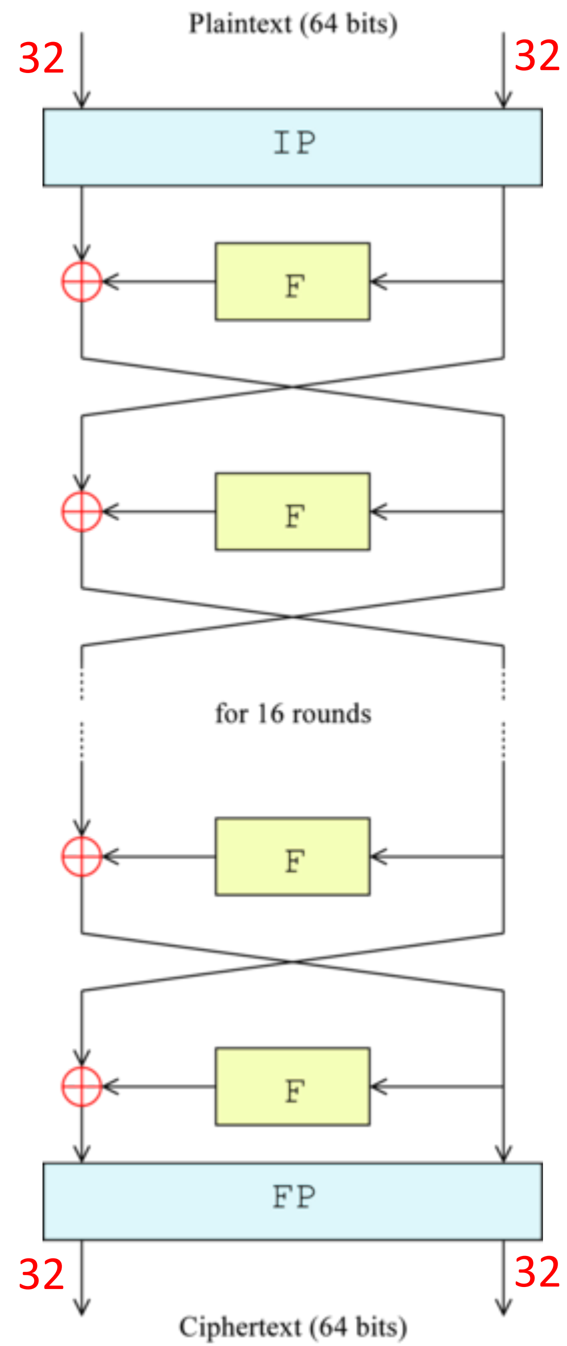
- Block Size : 64 bits
- Key size : 56 bits (+8 parity bits)
- Structure : Fiestel
- Rounds : 16
- Algorithm specifies :
  - encryption / decryption algorithm
  - key expansion algorithm



DES Initial and Final Permutation

• Plaintext subjected to an Initial permutation (IP) initially
• After 16 rounds, there is a final permutation (FP) before the ciphertext is generated

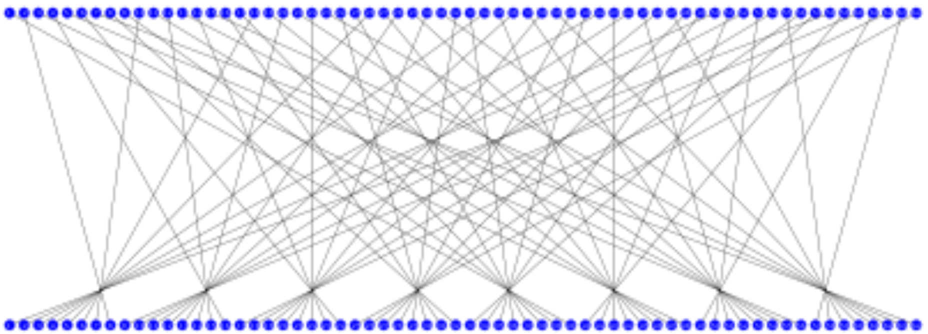
neither operation has any cryptographic significance. Used to facilitate loading of blocks in and out of 1970s eight bit computer



IP and FP

Initial Permutation (IP)

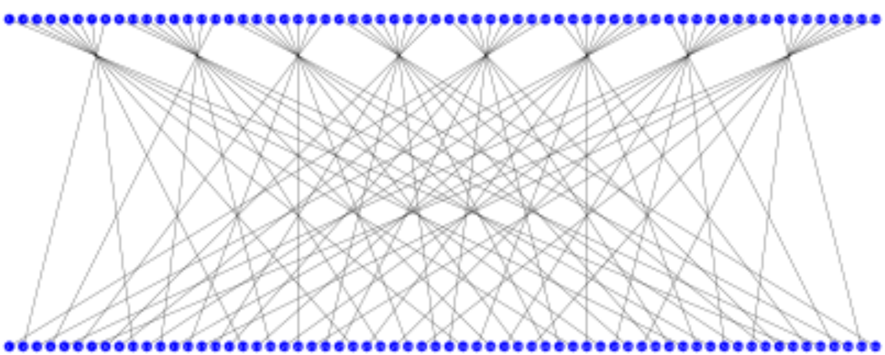
58	50	42	34	26	18	10	2
60	52	44	36	28	20	12	4
62	54	46	38	30	22	14	6
64	56	48	40	32	24	16	8
57	49	41	33	25	17	9	1
59	51	43	35	27	19	11	3
61	53	45	37	29	21	13	5
63	55	47	39	31	23	15	7



The first bit of the o/p is taken from the 58^{th} input bit

Final Permutation ($FP = IP^{-1}$)

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

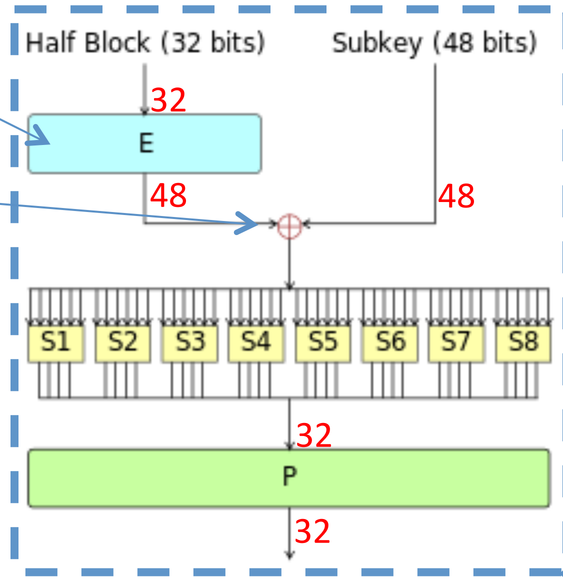


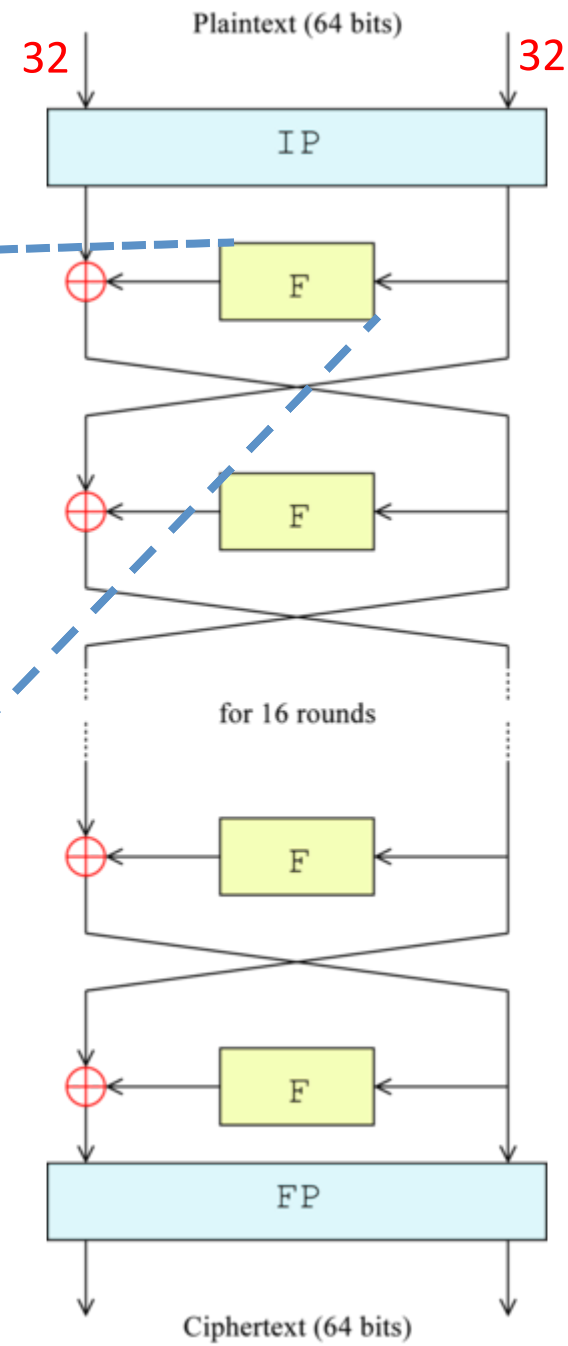
This is the inverse of IP

DES F Function (E and Key mixing)

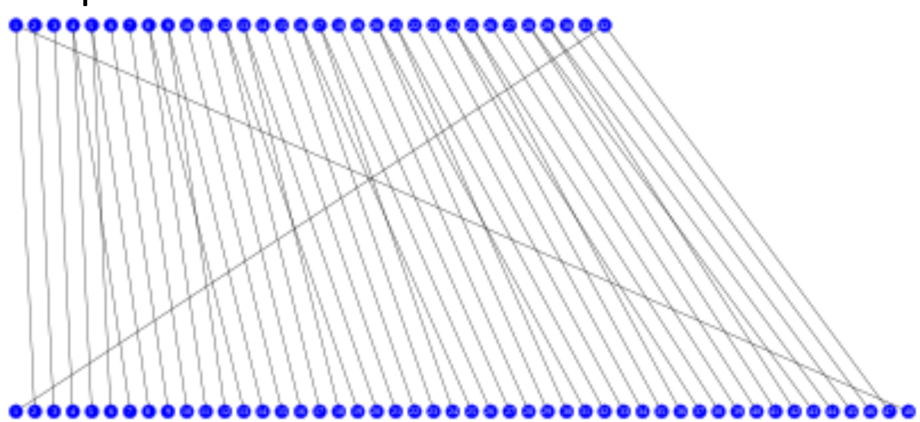
E is the expansion block. The 32 bit input is expanded to 48 bits by duplicating some of the bits

key mixing with subkey,

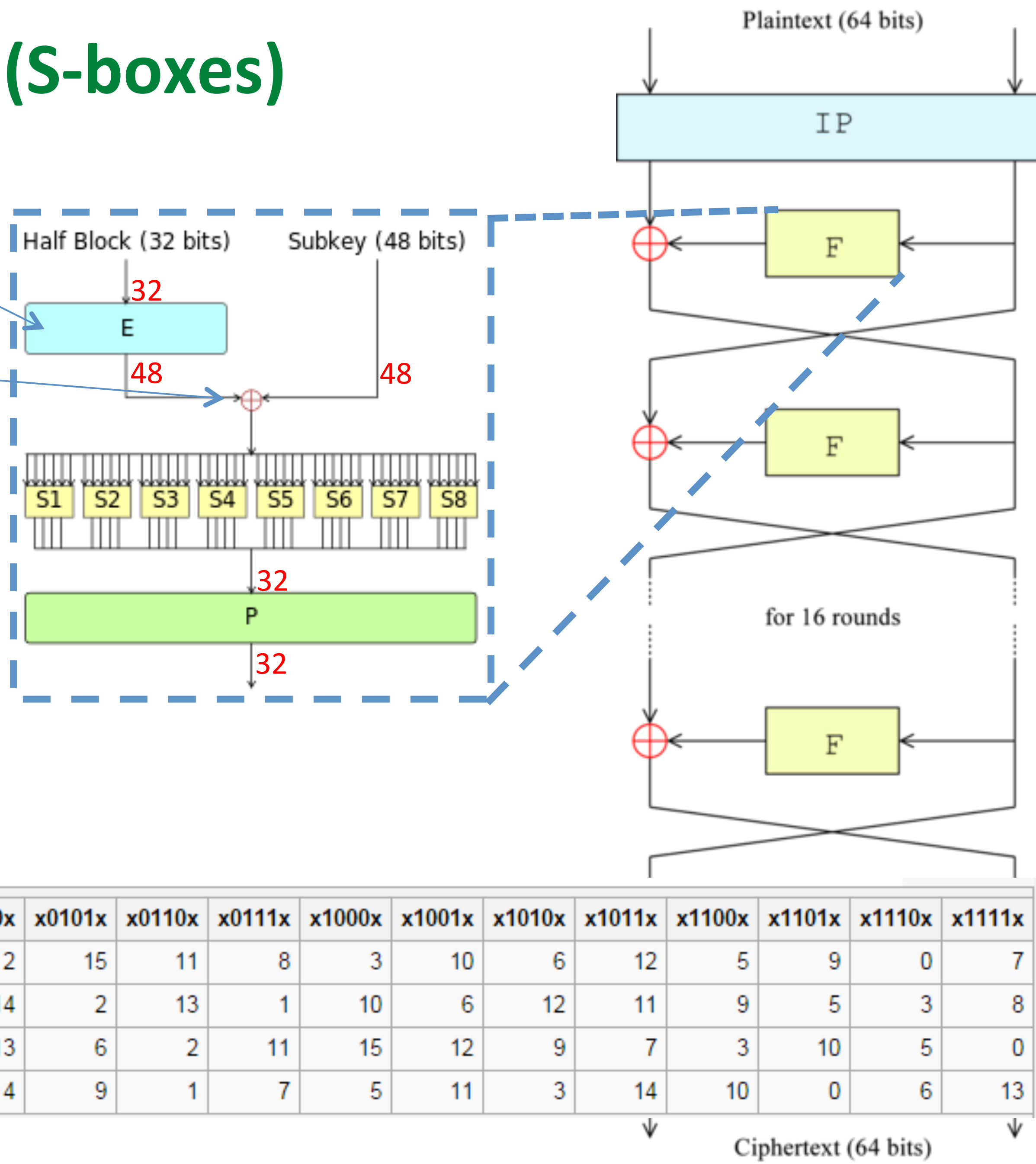




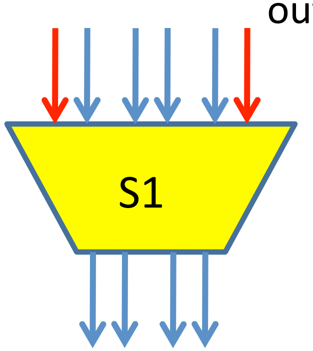
Expansion Function



DES F Function (S-boxes)

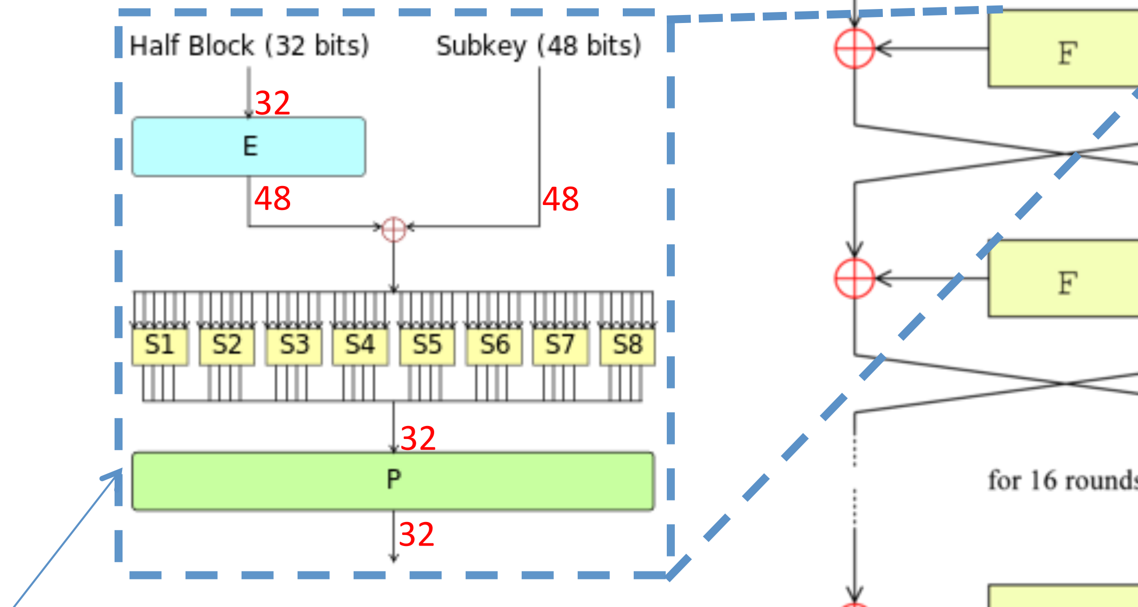


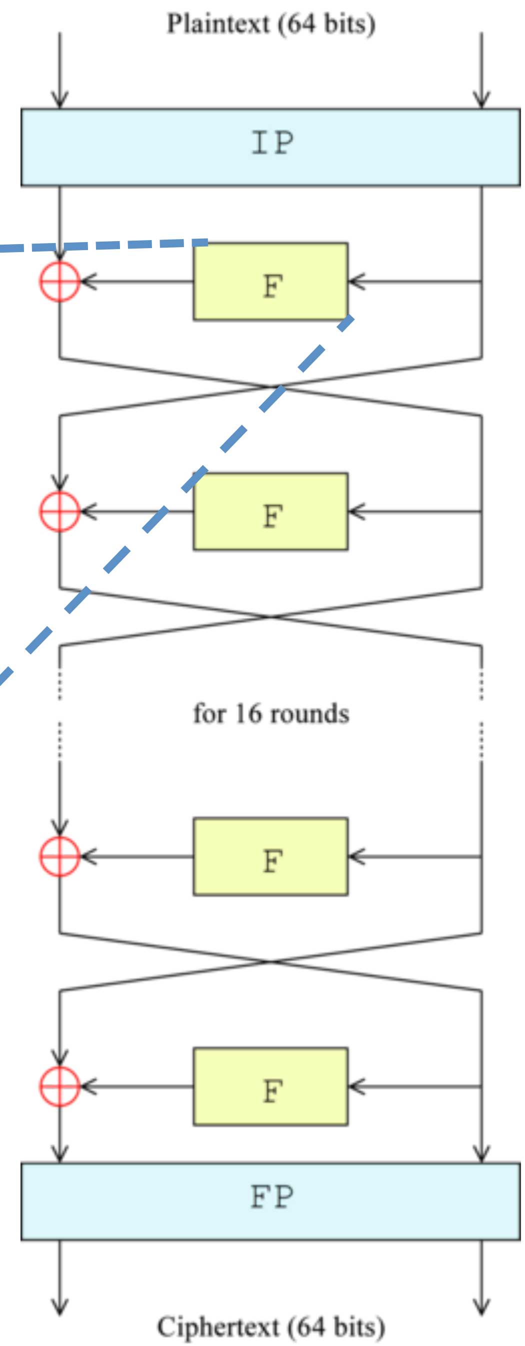
S1 to S8 are compression s-boxes. Each s-box takes 6 input bits and outputs 4 bits.



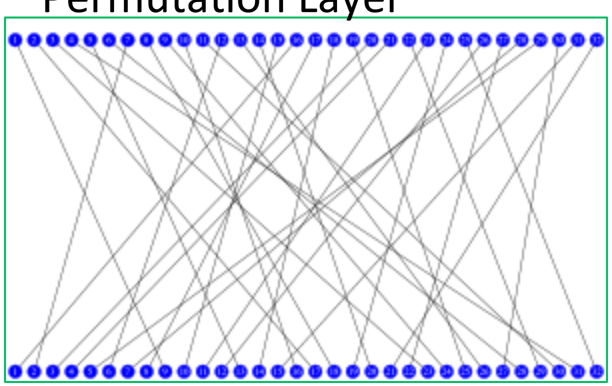
S_1	x0000x	x0001x	x0010x	x0011x	x0100x	x0101x	x0110x	x0111x	x1000x	x1001x	x1010x	x1011x	x1100x	x1101x	x1110x	x1111x
0yyyy0	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
0yyyy1	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
1yyyy0	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
1yyyy1	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13

DES F Function (Permutation)





Permutation Layer



DES Key Expansion

- 64 bits input
  - Of which 8 are discarded (or used for parity)
- No non-linear components

PC1

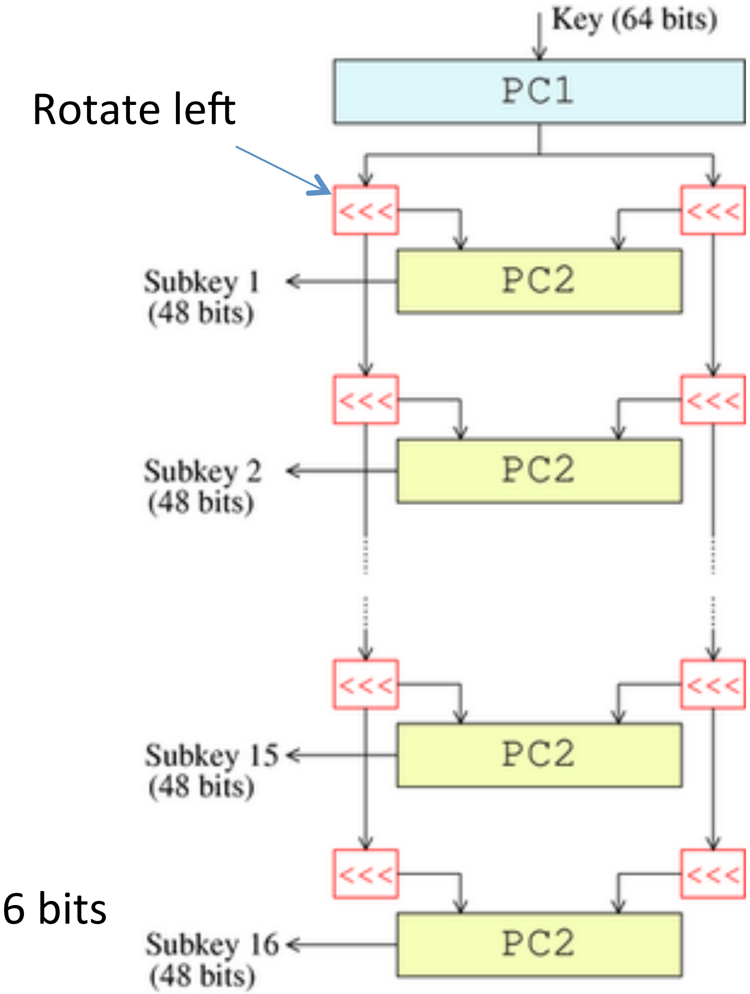
Left							*Right*						
57	49	41	33	25	17	9	63	55	47	39	31	23	15
1	58	50	42	34	26	18	7	62	54	46	38	30	22
10	2	59	51	43	35	27	14	6	61	53	45	37	29
19	11	3	60	52	44	36	21	13	5	28	20	12	4

PC2

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

Select 48 out of the 56 bits



DES Decryption

- Same as encryption algorithm, with subkeys applied in reverse order

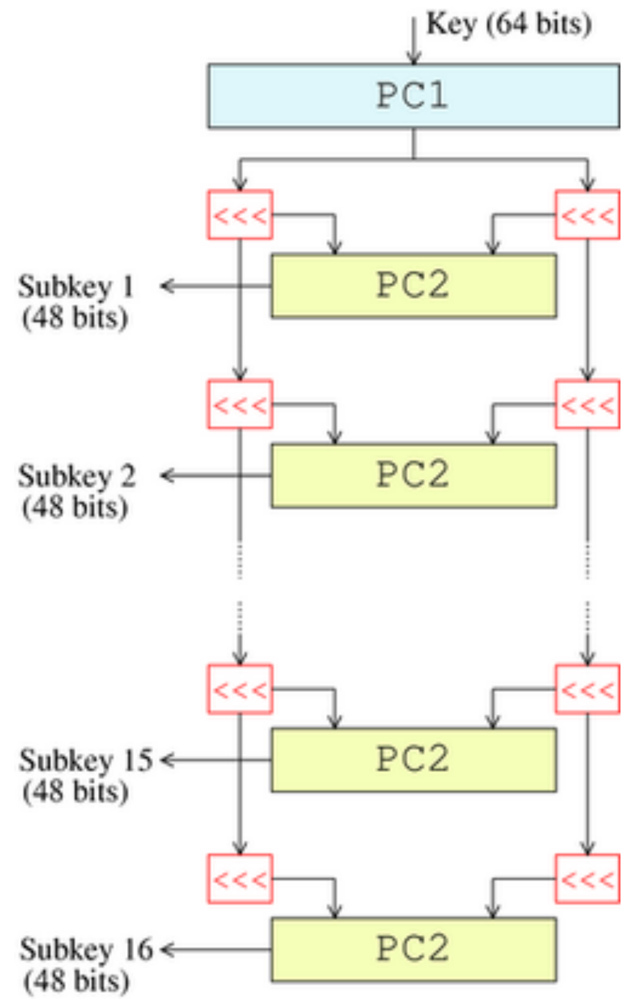
DES Weak Keys

- In a DES weak key, all the subkeys are the same

  Thus $DES_{WK}(DES_{WK}(x)) = x$

  (WK is a weak key)
- DES weak keys are as follows

56 bit DES weak keys
0000000 0000000
FFFFFFF FFFFFFF
0000000 FFFFFFF
FFFFFFF 0000000



DES Semi weak keys

- Semi-weak keys have the following properties
  - They appear in pairs: (SK1 and SK1')
  - $DES_{SK1}(DES_{SK1'}(x)) = x$
  - Each semi-weak key has only two sub keys.

	SK1	SK1'
1	9153E54319BD	6EAC1ABCE642
2	6EAC1ABCE642	9153E54319BD
3	6EAC1ABCE642	9153E54319BD
4	6EAC1ABCE642	9153E54319BD
5	6EAC1ABCE642	9153E54319BD
6	6EAC1ABCE642	9153E54319BD
7	6EAC1ABCE642	9153E54319BD
8	6EAC1ABCE642	9153E54319BD
9	9153E54319BD	6EAC1ABCE642
10	9153E54319BD	6EAC1ABCE642
11	9153E54319BD	6EAC1ABCE642
12	9153E54319BD	6EAC1ABCE642
13	9153E54319BD	6EAC1ABCE642
14	9153E54319BD	6EAC1ABCE642
15	9153E54319BD	6EAC1ABCE642
16	6EAC1ABCE642	9153E54319BD

DES Semi weak key pairs

First key in the pair	Second key in the pair
01FE 01FE 01FE 01FE	FE01 FE01 FE01 FE01
1FE0 1FE0 0EF1 0EF1	E01F E01F F10E F10E
01E0 01E0 01F1 01F1	E001 E001 F101 F101
1FFE 1FFE 0EFE 0EFE	FE1F FE1F FE0E FE0E
011F 011F 010E 010E	1F01 1F01 0E01 0E01
E0FE E0FE F1FE F1FE	FEE0 FEE0 FEF1 FEF1

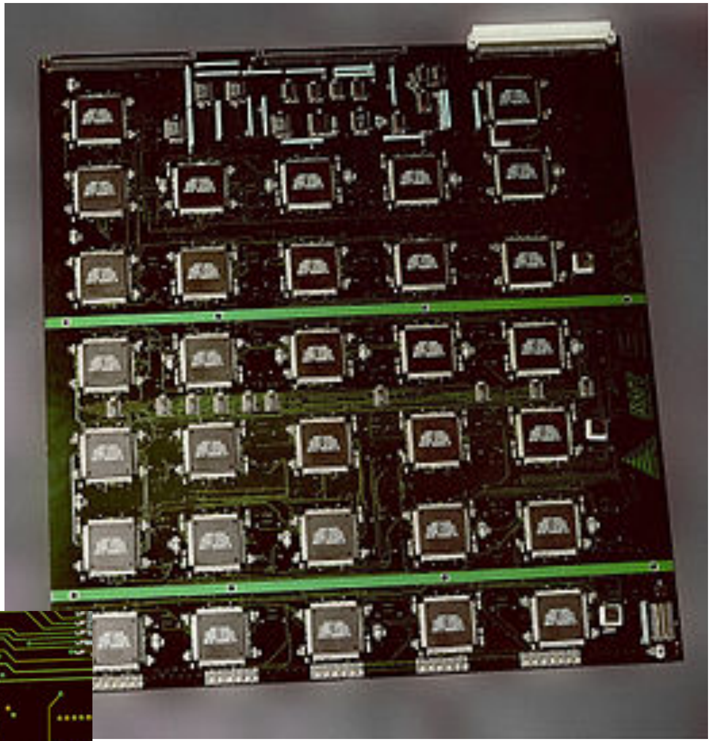
Objections to DES

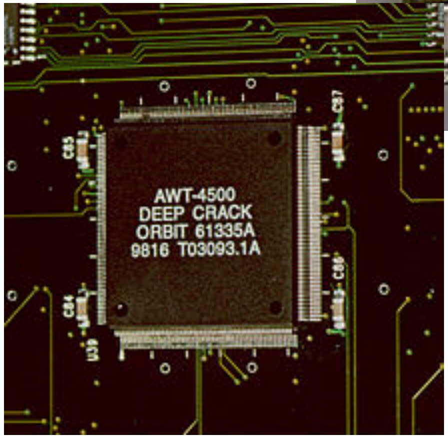
- Key size matters
  - Brute Force Attacks due to the small key size
- S-box secrecy
  - During the initial years, the rationale for the DES s-box was kept secret (… to increase security).
- Mathematical attacks :
  - Differential Cryptanalysis
  - Linear Cryptanalysis

DES Cracker

- Specialized ASICs for DES bruteforce
- Could determine the secret key in less than a day

.... Need to increase key length!!

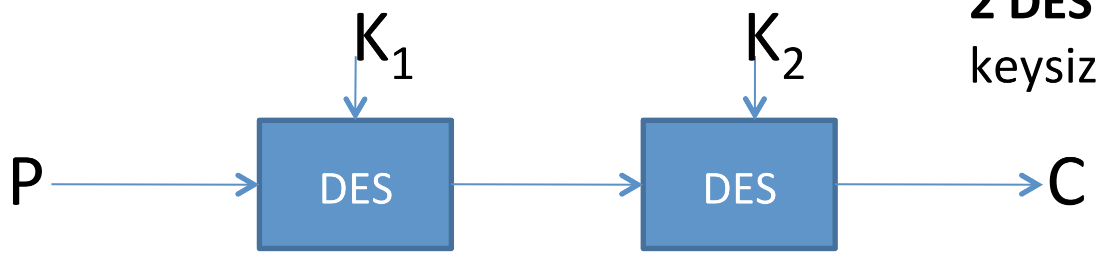




DES Composition

- Key size can be increased by composition

  $C = DES_{K1}(DES_{K2}(P))$

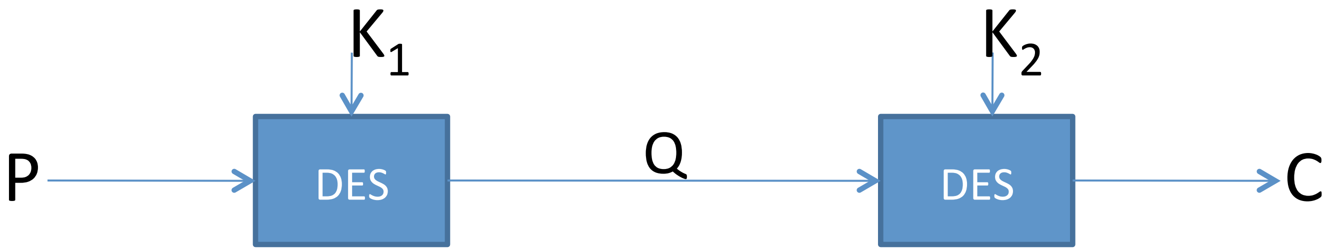


2 DES
keysize = 2*56=112 bits

- DES does not form a group under composition.
  i.e. It is not possible to obtain

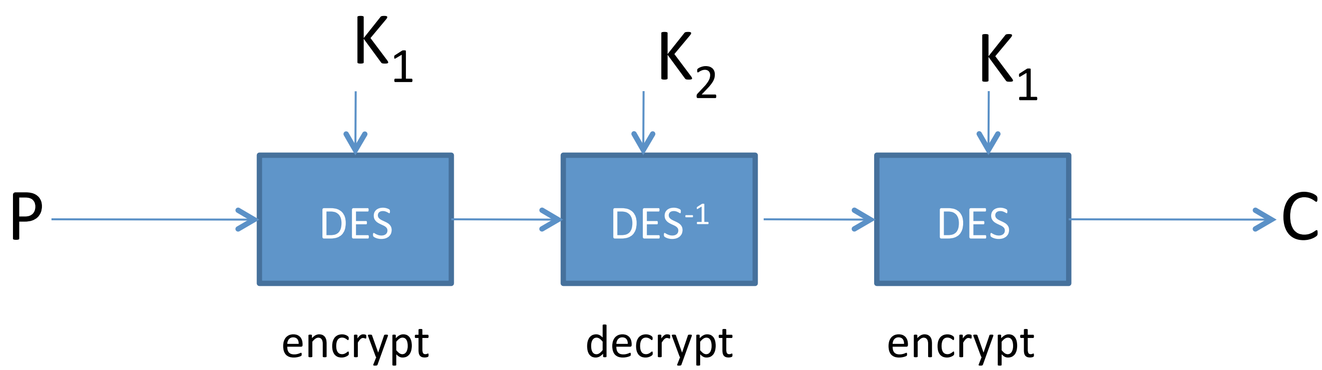
  $DES_{K1}(DES_{K2}(P)) = DES_{K3}(P)$ for some key K3

Meet in the Middle Attack against 2-DES



- Attacker collects a pair of (P,C)
  1. For P, compute $Q_{K1*} = DES_{K1*}(P)$ for every possible value of K1*. Record the corresponding Q_{K1*}
  2. For C, compute $Q_{K2*} = DES^{-1}_{K2*}(C)$ for every possible value of K2*. Record the corresponding Q_{K2*}
  3. Find all K1* and K2* such that $Q_{K1*} = Q_{K2*}$
  4. If Multiple such K1* and K2* are found, then repeat with another pair of (P,C)
- Complexity of this attack is $2^{56}+2^{56} = 2^{57}$

3-DES



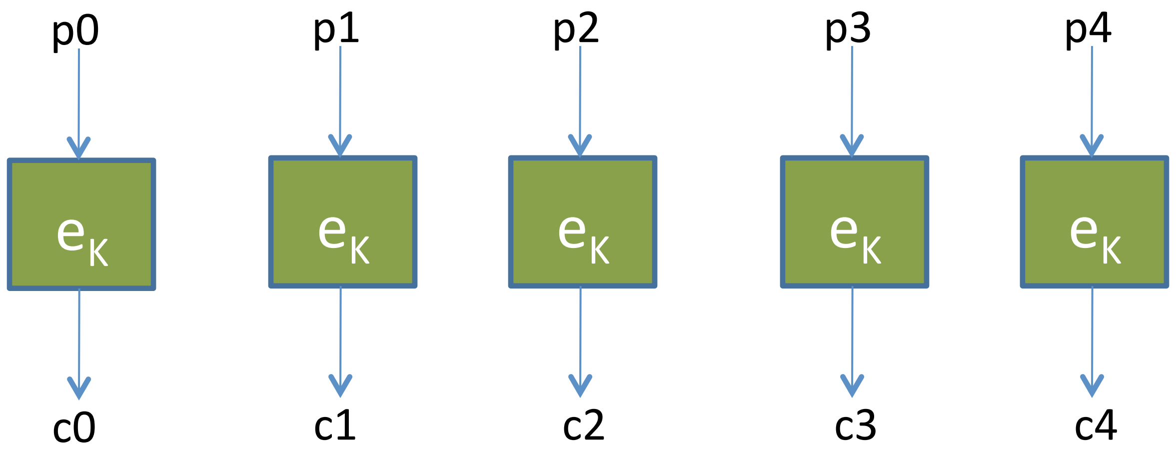
- 112 bit security as in 2-DES
- Encrypt → Decrypt → Encrypt
- K1 → K2 → K1 (two 56 bit keys)
- Why EDE and not EEE?
  - Compatibility with the classical DES if $K_1 = K_2$
- Used extensively as a stopgap arrangement until a new cipher standard (AES) was established
- Drawbacks of 3-DES:
  - Sluggish in software
  - Could only encrypt 64 bit blocks at a time

Modes of Operation

What are Modes of Operation?

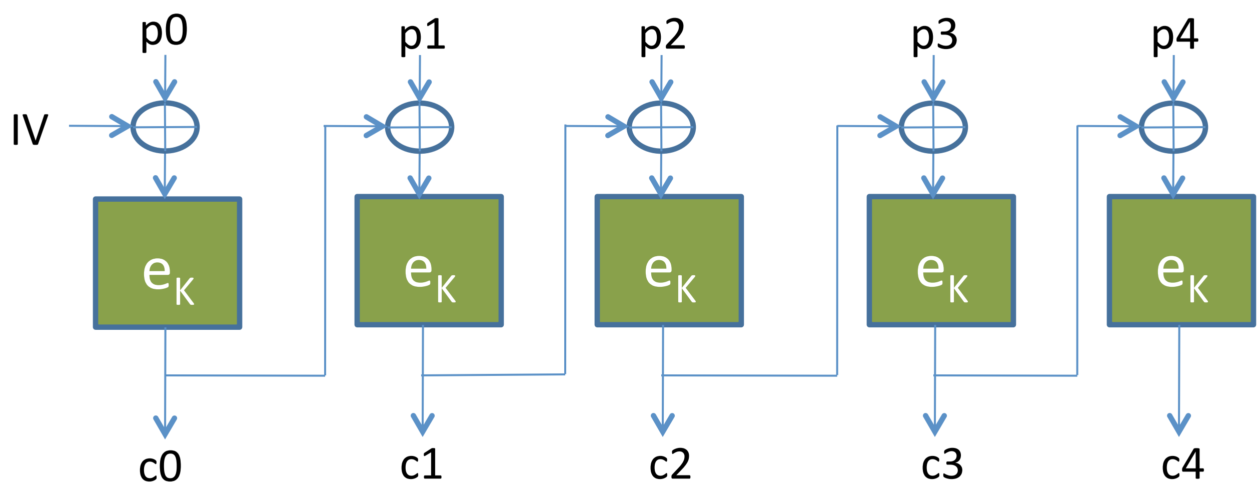
- Block cipher algorithms only encrypt a single block of message
- A mode of operation describes how to repeatedly apply a cipher's single-block operation to securely transform amounts of data larger than a block
- Modes of Operation
  - Electronic code book mode (ECB Mode)
  - Cipher feedback mode (CFB Mode)
  - Cipher block chaining mode (CBC mode)
  - Output feedback mode (OFB mode)
  - Counter mode

ECB Mode



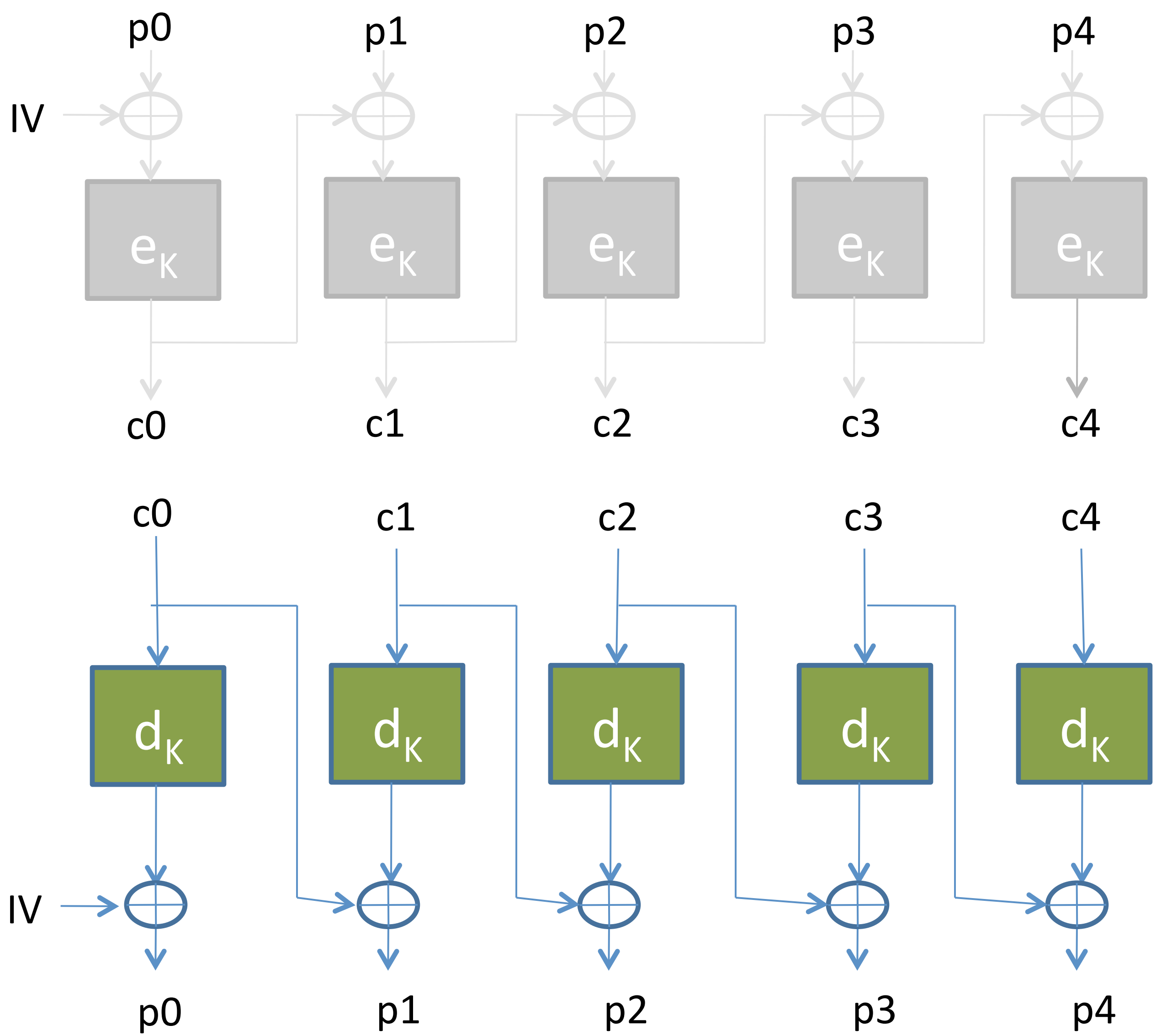
- Every block in the message is encrypted independently with the same key
- Drawback 1 : If $p_i = p_j$ $(i \neq j)$ then $c_i = c_j$
  - Encryption should protect against known plaintext attacks (since the attacker could guess parts of the message….. Like stereotype beginnings)
- Drawback 2 : An interceptor may alter the order of the blocks during transmission
- Not recommended for encryption of more than one block

CBC Mode



- Cipher Block Chaining
- Advantage 1 : Encryption dependent on a the ciphertext of a previous block, therefore
  - $c_i \neq c_j$ (i ≠ j) even if $p_i = p_j$
- Advantage 2: Intruder cannot alter the order of the blocks during transmission
- If an error is present in one received block (say c_i)
  - Then c_i and c_{i+1} will not be decrypted correctly
  - All remaining blocks will be correctly decrypted

CBC Mode Decryption

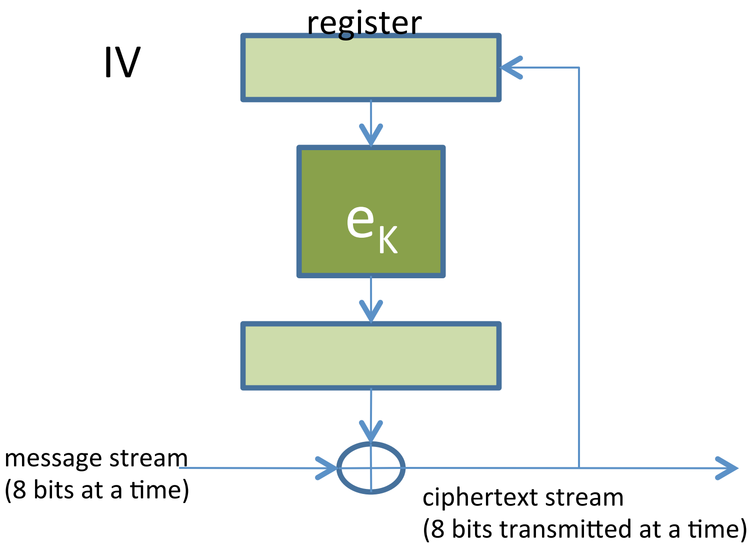


CFB (Cipher feedback Mode)

Can transform a block cipher into a stream cipher.

– i.e. Each block encrypted with a different key

Uses a shift register that is initialized with an IV

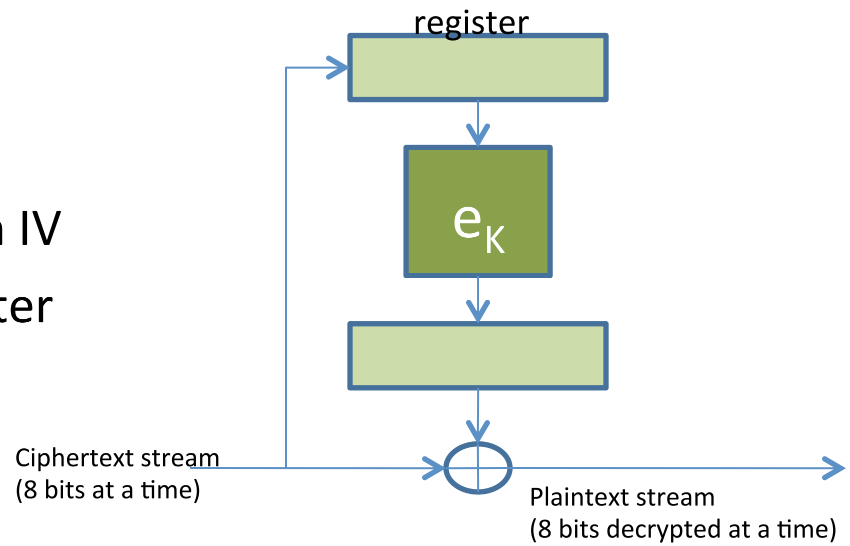


Encryption Scheme

CFB - Error Propagation

Uses a shift register that is initialized with an IV

Previous ciphertext block fed into shift register

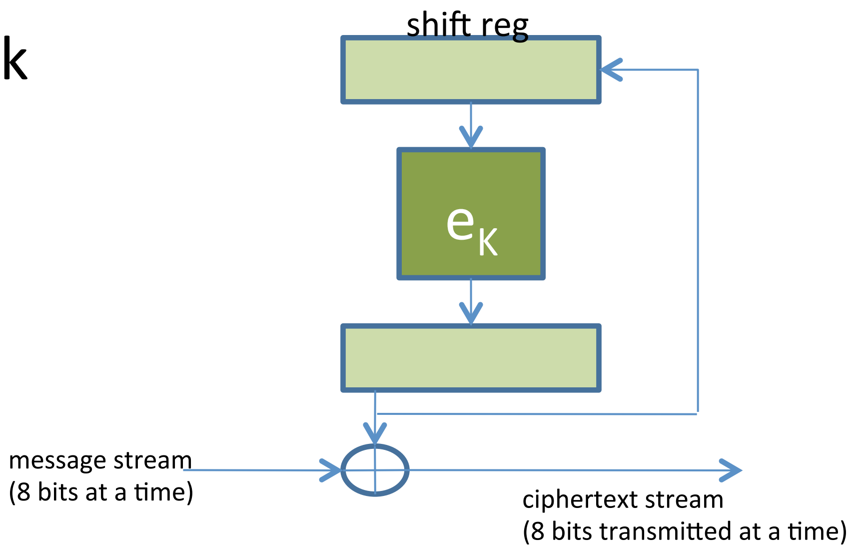


Decryption Scheme

Output Feedback Mode (OFB)

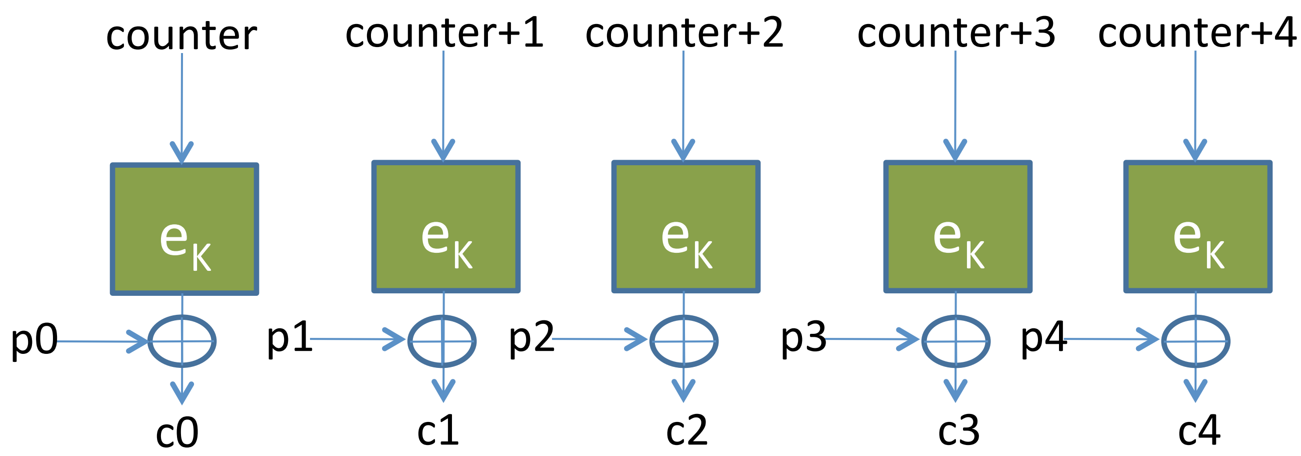
- Very similar to CFB but feedback taken from output of e_k

- An error in one byte of the ciphertexts affects only one decryption



Encryption Scheme
(Decryption scheme is similar)

Counter Mode



- A randomly initialized counter is incremented with every encryption
- Can be parallelized
  - Ie. Multiple encryption engines can simultaneously run
- As with OFB, an error in a single ciphertext block affects only one decrypted plaintext

How to choose a good s-box?

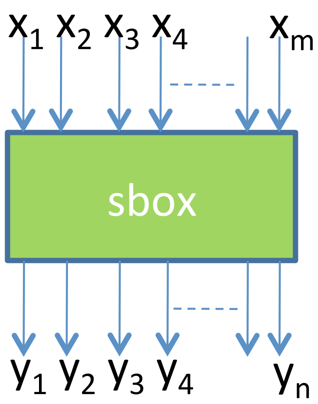
Mod-01, Lec-07, Overview of S-box Principles, by Debdeep Mukhopadhyay
https://www.youtube.com/watch?v=cJ7hmwHVwtc&list=PL71FE85723FD414D7&index=17

Criteria for a good s-box

- Completeness
- Balance
- Non-linearity
- Propagation criteria
- Good XOR profile
- High Algebraic Degree

Sboxes

- In an s-box each output bit can be represented as a **Boolean function** of its input bits



$$
\begin{aligned}
y_1 &= f_1(x_1, x_2, x_3, \cdots, x_m) \\
y_2 &= f_2(x_1, x_2, x_3, \cdots, x_m) \\
y_3 &= f_3(x_1, x_2, x_3, \cdots, x_m) \\
\vdots & \quad \vdots \quad \vdots \quad \vdots \quad \vdots \\
y_n &= f_n(x_1, x_2, x_3, \cdots, x_m)
\end{aligned}
$$

The functions have to be non-linear.
Linear functions are easily reversed.

Boolean Functions

- A Boolean function is a mapping from $\{0,1\}^m \rightarrow \{0,1\}$
- **Algebraic Normal Form representation of a Boolean function**
  - A Boolean function on m-inputs can be represented with sum (XOR +) of products (AND .) form:

$$y = a_0 \oplus a_1 x_1 \oplus a_2 x_2 \oplus a_3 x_1 x_2$$

where a_i is either 0 or 1.

- Affine Form: if all the terms have coefficients 0 (a_3=0 in the above example)
- Linear form : Affine form and a_0 = 0

CR

Truth Tables

$$f : y = x_1 \oplus x_2 \oplus x_1 x_2$$

- Consider a Boolean function $f:\{0,1\}^m \rightarrow \{0,1\}$
- The following Binary sequence is the truth table of f

$$\left(f(\alpha_0), f(\alpha_1), f(\alpha_2), \cdots, f(\alpha_{2^m-1}) \right)$$

$$where\ \ \alpha_i\ are\ m\ bit\ numbers\ and\ \alpha_i \neq \alpha_j\ unless\ i = j$$

X1	X2	Y
0	0	0
0	1	1
1	0	1
1	1	1

  - The truth table is therefore (0,1,1,1)
  - Sequence is (1,-1,-1,-1)

Balanced Boolean Functions

- A Boolean function is said to be balanced if its truth table has equal number of 0s and 1s.
- S-box equations should be balanced (i.e. 0 and 1 have an equal probability of occurrence)

$$f : y = x_1 \oplus x_2 \oplus x_1 x_2$$

Unbalanced function

X1	X2	Y
0	0	0
0	1	1
1	0	1
1	1	1

$$g : y = x_1 \oplus x_2$$

Balanced Function

X1	X2	Y
0	0	0
0	1	1
1	0	1
1	1	0

Distance Between functions

Let f and g be two Boolean functions

Let η be the truth table for f and ε the truth table for g

$HD(\eta, \varepsilon)$ is the Hamming distance between the two sequences

X1	X2	Y1	Y2
0	0	0	0
0	1	1	1
1	0	1	1
1	1	1	0

$$f : y_1 = x_1 \oplus x_2 \oplus x_1 x_2$$

$$g : y_2 = x_1 \oplus x_2$$

$$HD(\eta, \varepsilon) = 1$$

CR

Nonlinearity of a Boolean Function

- The non-linearity of a Boolean function is **the minimum distance between the function and the set of all linear functions.**
  - **Strengthens against linear cryptanalysis**

$$y_1 = x_1 \oplus x_2 \oplus x_1 x_2$$
$$y_2 = 0$$
$$y_3 = x_1$$
$$y_4 = x_2$$
$$y_5 = x_1 \oplus x_2$$

X1	X2	Y1	Y2	Y3	Y4	Y5
0	0	**0**	0	0	0	0
0	1	**1**	0	0	1	1
1	0	**1**	0	1	0	1
1	1	**1**	0	1	1	0

3

1

1

1

$$Nonlinearity:\ N_f = MIN_{g \varepsilon linear}\left(HD(f,g)\right)$$

$$Nonlinearity\ of\ y_1:\ N_{y_1} = 1$$

Walsh Hadamand Matrix

- A compact combinatorial representation of all affine functions
- Each row of the WH matrix forms the truth table of all affine functions with N variables can be represented by the matrix

$$H(2^N) = \begin{bmatrix} H(2^{N-1}) & H(2^{N-1}) \\ H(2^{N-1}) & complement(H(2^{N-1})) \end{bmatrix}$$

$$H(2^1) = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} \begin{matrix} \rightarrow 0 \\ \rightarrow x_1 \end{matrix}$$

$$H(2^2) = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix} \begin{matrix} \rightarrow 0 \\ \rightarrow x_2 \\ \rightarrow x_1 \\ \rightarrow x_2 \wedge x_1 \end{matrix}$$

On the Non-linearity of Boolean Functions

- HD of any two linear functions is always 2^{n-1}
- HD between two non-linear functions is $< 2^{n-1}$

Scalar product

$$Let\langle \xi,\eta \rangle = \#(f = g) - \#(f \neq g)$$

$$= 2^n - \#(f \neq g) - \#(f \neq g)$$

$$= 2^n - 2\#(f \neq g)$$

$$HD(f,g) = \#(f \neq g) = 2^{n-1} - \frac{1}{2}\langle \xi,\eta \rangle$$

Bent Functions

- Bent functions are non-linear Boolean functions which have maximum non-linearity
- The non-linearity of a Bent function is $2^{n-1} - 2^{\frac{n}{2}-1}$
- They satisfy SAC but are **not balanced**

- Example : $f(x) = x_1x_2 + x_3x_4$

Affine Transformations and Non-linearity

- If a Boolean function is **balanced**, then an affine transformation does not affect its non-linearity

$f(x)$ is a balanced Boolean function, then $f(xB \oplus A)$ is also balanced

$x = (x_1, x_2, x_3, \ldots, x_n)$

B is a $n \times n$ binary invertible matrix

A is an n bit vector

The nonlinearity of $f(x)$ = nonlinearity of $f(xB \oplus A)$

Strict Avalanche Criteria (SAC)

- For a function (f) to satisfy SAC,

$$f(x) \oplus f(x \oplus \alpha) \text{ must be balanced, for any } \alpha \text{ with } HW(\alpha) = 1$$

- Also called *propagation criteria of order 1*
- Higher order SAC,
  - Propagation criteria of order > 1
  - When input changes in more than 1 bit

- Show that

$$y = x_1 x_2 \oplus x_3 \quad \text{does not satisfy } SAC$$

$$z = x_1 x_2 \oplus x_3 x_4 \text{ satisfies } SAC$$

Note that z is a Bent function

How to make a Boolean function satisfy SAC

- Let $f(x)$ be a Boolean function of order n
- Let A be an nxn non-singular Boolean matrix
- If r is a row in the matrix A and $f(x) \oplus f(x \oplus r)$ is balanced then $g(x) = f(xA)$ satisfies SAC

Example :

$$f = x_1 x_2 \oplus x_3$$

$$A = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

$$then \quad g(x) = f(xA) \; satisfies \;\; SAC$$

verify this?

Completeness

- More a criteria for the complete cipher (SP)
- Given s-boxes with a fixed mapping,
  - P-layer needs to be fixed and rounds need to be fixed such that ciphertext is a complex function of every plaintext input

XOR Profile

- The difference distribution table of the s-box must contain small variations

The Advanced Encryption Standard (AES)

Advanced Encryption Standard (AES)

- NIST's standard for block cipher since October 2000.

	Key Length	No. of rounds
AES-128	16 bytes	10
AES-192	24bytes	12
AES-256	32bytes	14

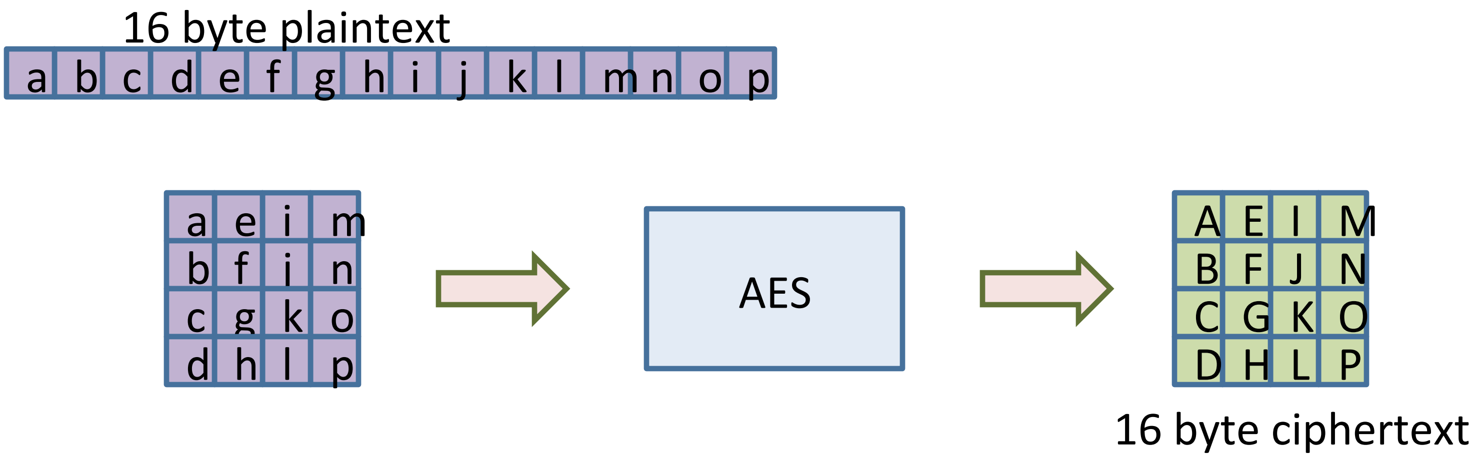
- SPN network with each round having
  - Randomness Layer: *Round key addition*
  - Confusion Layer : *Byte Substitution*
  - Diffusion Layer : *Shift row* and *Mix column*

  (the last round does not have mix column step)

Mathematical Background

Finite Fields

The AES State Representation

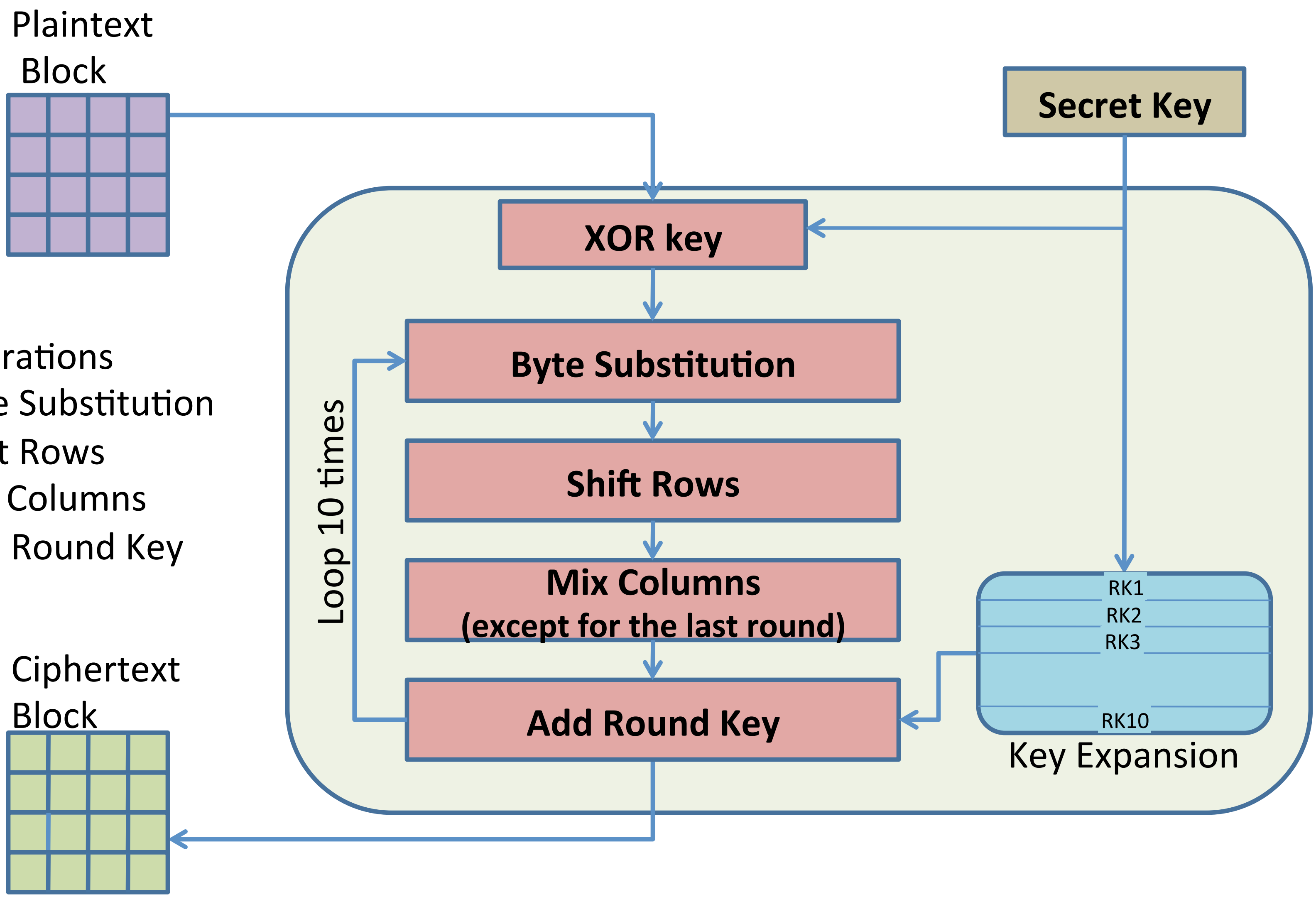


- 16 bytes arranged in a 4x4 matrix of bytes

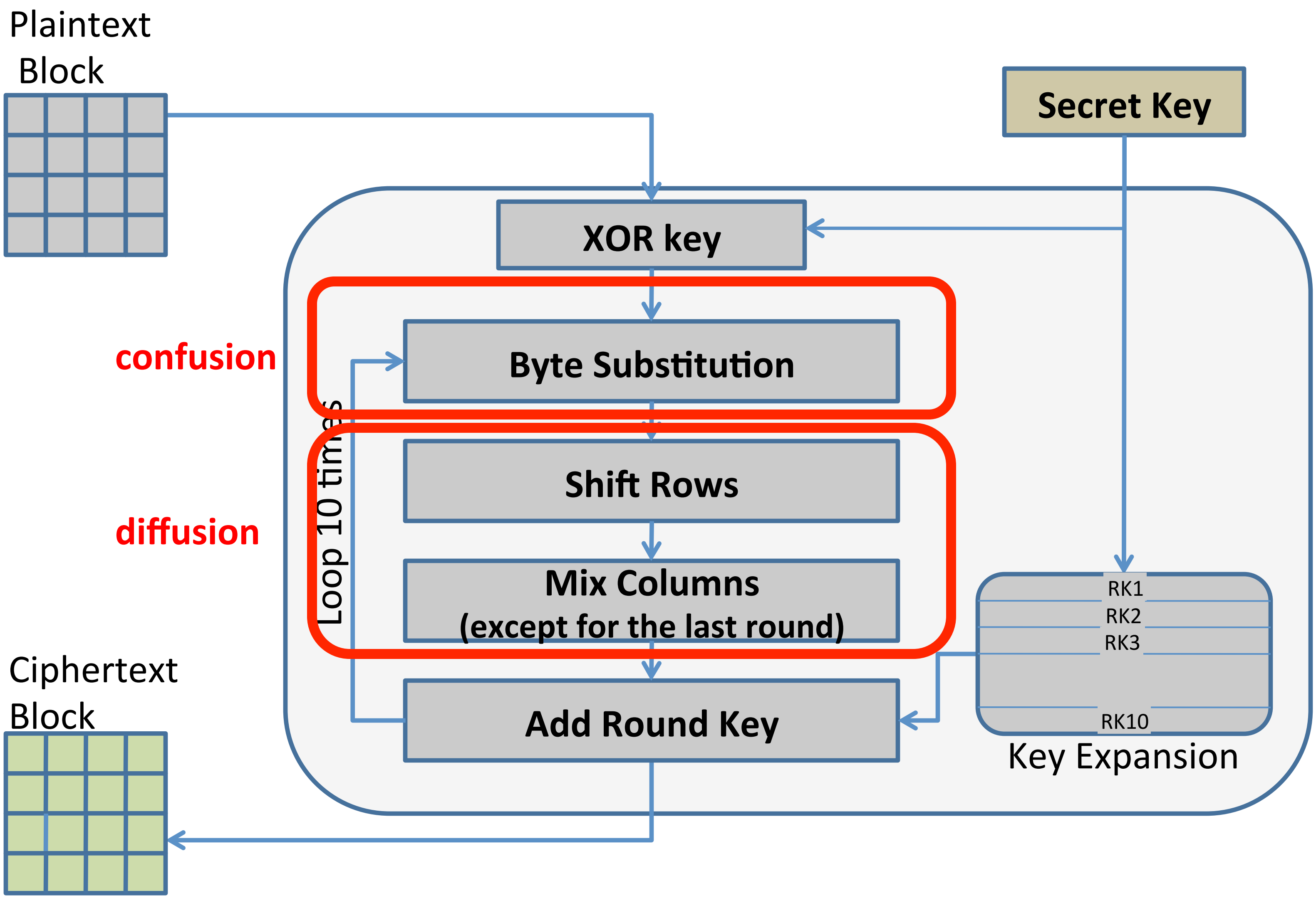
AES-128 Encryption

4 Operations

- Byte Substitution
- Shift Rows
- Mix Columns
- Add Round Key



AES-128 Encryption



AES Operations

- All AES operations are performed in the field $GF(2^8)$.

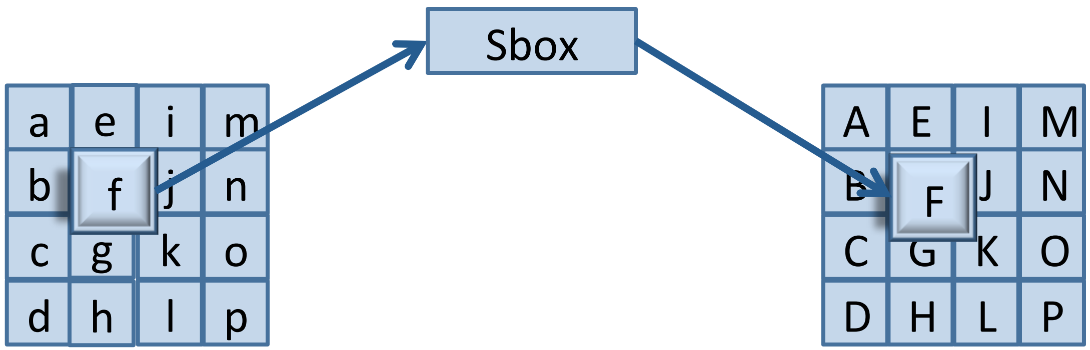
- The field's irreducible polynomial is

$$x^8 + x^4 + x^3 + x + 1$$

in binary notation $(1\ 0001\ 1011)_2$
in hex notation $(11B)_{16}$

Byte Substitution

- Makes a non-linear substitution for every byte in the 4x4 matrix



$$Sbox(A) = \begin{cases} Affine(A^{-1}) & \text{if } A(\theta) \neq 0 \\ Affine(0) & \text{if } A(\theta) = 0 \end{cases}$$

Affine Transformation

$$= \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} a_7 \\ a_6 \\ a_5 \\ a_4 \\ a_3 \\ a_2 \\ a_1 \\ a_0 \end{bmatrix} \oplus \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \end{bmatrix}$$

AES S-box Design Rationale

$$Sbox(A) = \begin{cases} Affine(A^{-1}) & \text{if } A(\theta) \neq 0 \\ Affine(0) & \text{if } A(\theta) = 0 \end{cases}$$

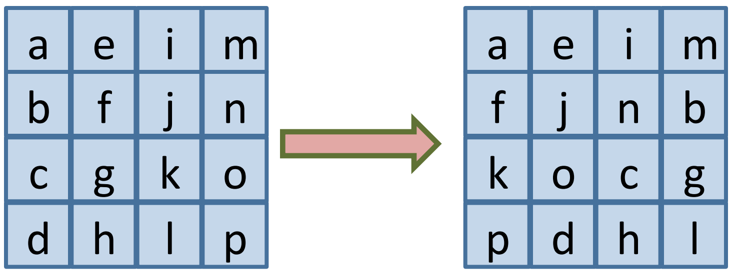
- This s-box construction was proposed by Kaiser Nyberg in 1993
- Steps:
  1. Inverse in GF(2^8)
     - Provides high degrees of non-linearity
     - Known to have good resistance against differential and linear cryptanalysis
  2. Affine transformation
     - ensures no fixed points : i.e. Fixed points : S(x) = x
     - Complicates Algebraic attacks

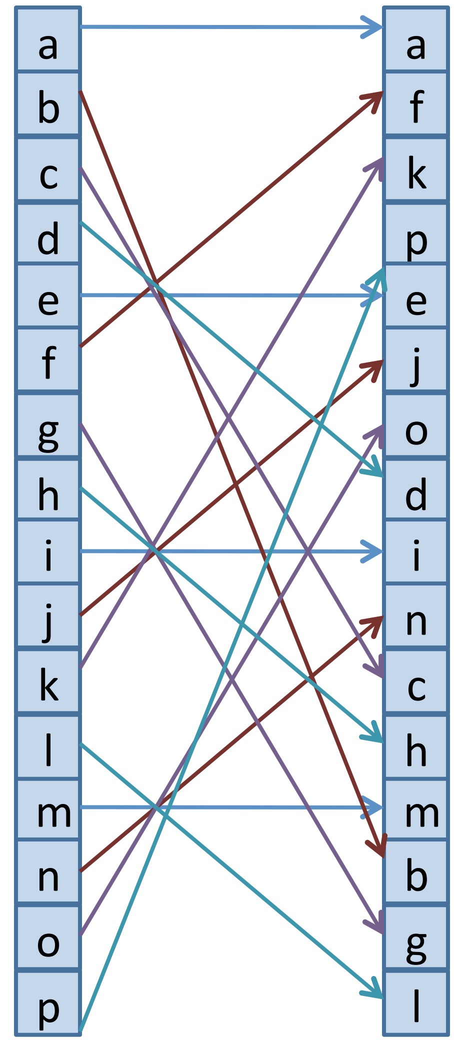
S-box Encryption Table

- Use a table to do the byte substitution
- eg. **Sbox[42] = 2c**

x \ y	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
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

Shift Rows





- ***ShiftRows***
  - Leave the First row untouched
  - Left Rotate (2nd Row by 8 bits)
  - Left Rotate (3rd Row by 16 bits)
  - Left Rotate (4th Row by 24 bits)
- Along with MixColumns provides high diffusion
  - Bits flip in at-least 25 s-boxes after 4 rounds

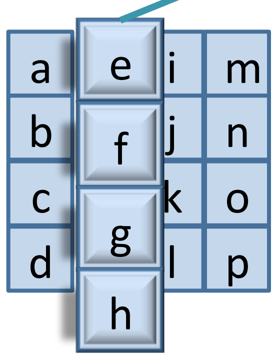
Mix Columns

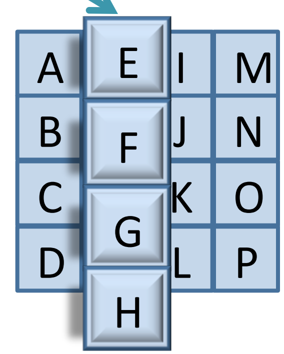
The 4x4 matrix is multiplied with the matrix

$$\begin{bmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{bmatrix} \times \begin{bmatrix} a & e & i & m \\ b & f & j & n \\ c & g & k & o \\ d & h & l & p \end{bmatrix}$$

Note that multiplications are in GF(2^8) field

$$\begin{aligned} \boldsymbol{E} &= 2e + 3f + g + h \\ \boldsymbol{F} &= e + 2f + 3g + h \\ \boldsymbol{G} &= e + f + 2g + 3h \\ \boldsymbol{H} &= 3e + f + g + 2h \end{aligned}$$





Mix Columns Rationale

Why use this matrix?

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

- It is an MDS matrix (Maximum Distance Separable codes)
  - If the input of a column changes then all outputs change
  - This maximizes the branch number
  - For AES, the branch number is 5
- Values [2,3,1,1], are the smallest which result in MDS matrix that is also circulant
- Has an inverse in the AES field

AES Operations (Add Round Key)

a	e	i	m
b	f	j	n
c	g	k	o
d	h	l	p

$\oplus$

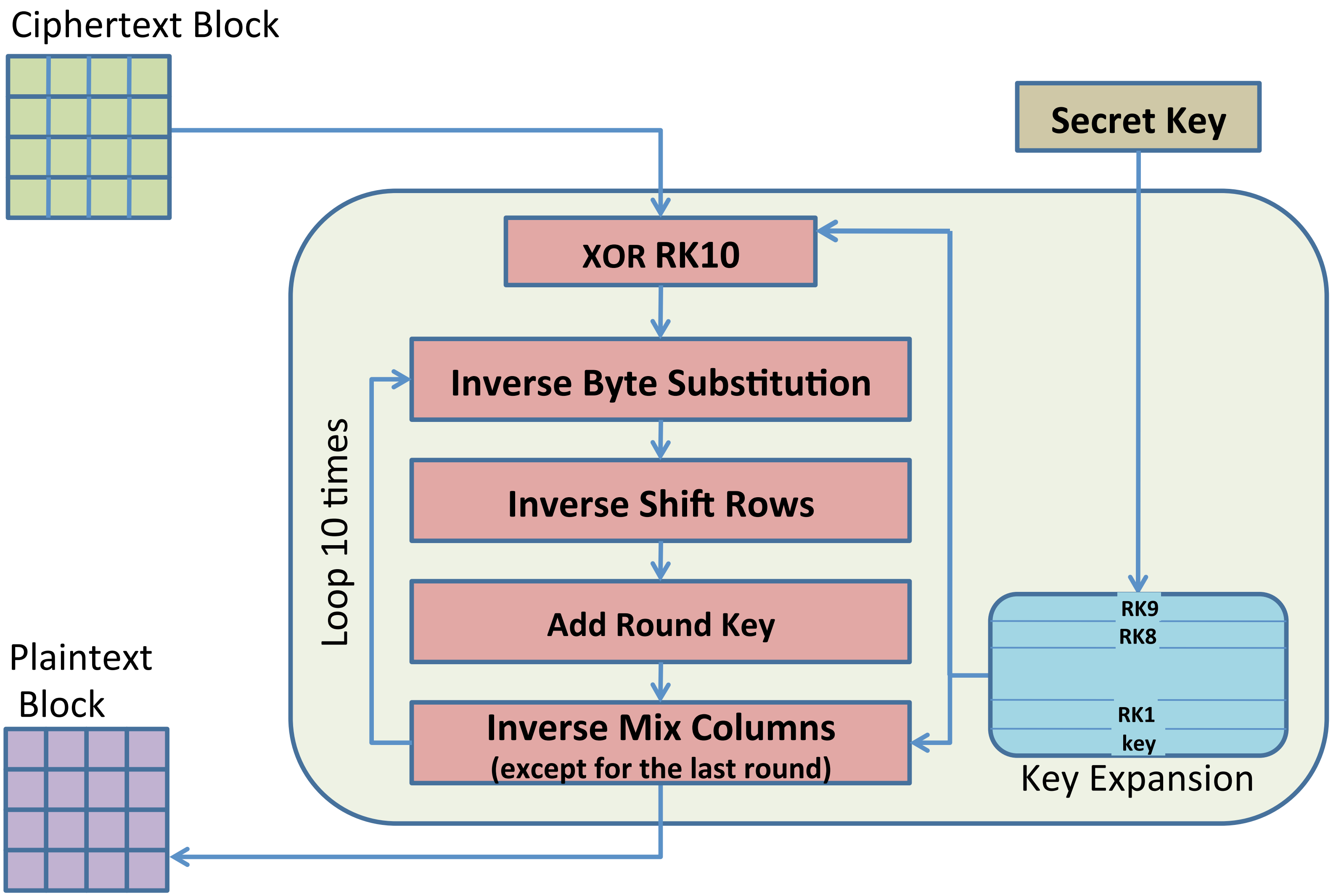
k_0	k_4	k_8	k_{12}
k_1	k_5	k_9	k_{13}
k_2	k_6	k_{10}	k_{14}
k_3	k_7	k_{11}	k_{15}

$\Rightarrow$

$a + k_0$	$e+k_4$	$i+k_8$	$m+k_{12}$
$b+k_1$	$f+k_5$	$j+k_9$	$n+k_{13}$
$c+k_2$	$g+k_6$	$k+k_{10}$	$o+k_{14}$
$d+k_3$	$h+k_7$	$l+k_{11}$	$p+k_{15}$

Addition here is addition in GF(2^8), which is the ex-or operation

AES-128 Decryption

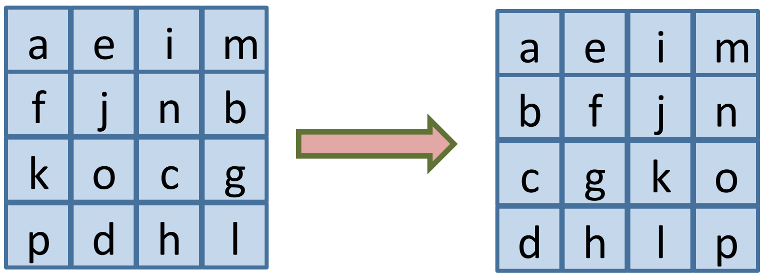


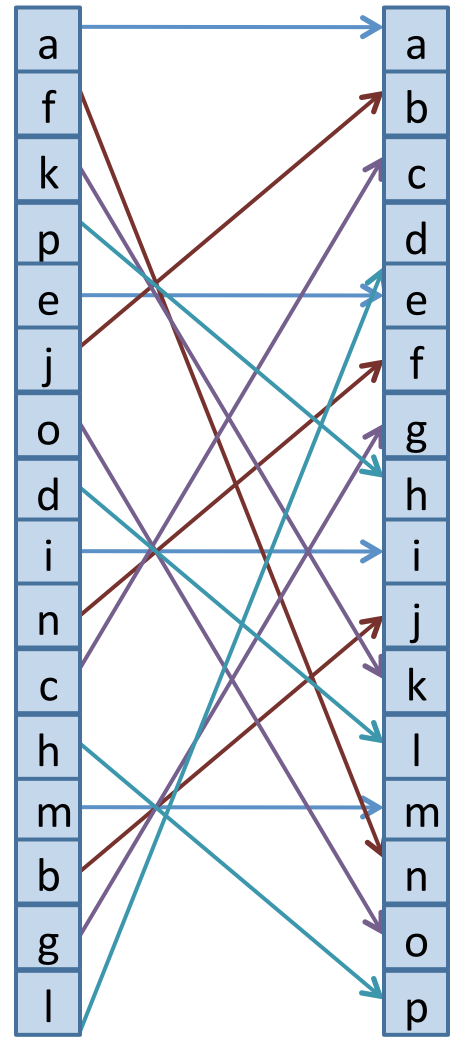
Inverse S-box

- Simply the AES s-box run in reverse
- As with the s-box operation, a lookup table can be used

	x0	x1	x2	x3	x4	x5	x6	x7	x8	x9	xa	xb	xc	xd	xe	xf
0x	52	09	6a	d5	30	36	a5	38	bf	40	a3	9e	81	f3	d7	fb
1x	7c	e3	39	82	9b	2f	ff	87	34	8e	43	44	c4	de	e9	cb
2x	54	7b	94	32	a6	c2	23	3d	ee	4c	95	0b	42	fa	c3	4e
3x	08	2e	a1	66	28	d9	24	b2	76	5b	a2	49	6d	8b	d1	25
4x	72	f8	f6	64	86	68	98	16	d4	a4	5c	cc	5d	65	b6	92
5x	6c	70	48	50	fd	ed	b9	da	5e	15	46	57	a7	8d	9d	84
6x	90	d8	ab	00	8c	bc	d3	0a	f7	e4	58	05	b8	b3	45	06
7x	d0	2c	1e	8f	ca	3f	0f	02	c1	af	bd	03	01	13	8a	6b
8x	3a	91	11	41	4f	67	dc	ea	97	f2	cf	ce	f0	b4	e6	73
9x	96	ac	74	22	e7	ad	35	85	e2	f9	37	e8	1c	75	df	6e
ax	47	f1	1a	71	1d	29	c5	89	6f	b7	62	0e	aa	18	be	1b
bx	fc	56	3e	4b	c6	d2	79	20	9a	db	c0	fe	78	cd	5a	f4
cx	1f	dd	a8	33	88	07	c7	31	b1	12	10	59	27	80	ec	5f
dx	60	51	7f	a9	19	b5	4a	0d	2d	e5	7a	9f	93	c9	9c	ef
ex	a0	e0	3b	4d	ae	2a	f5	b0	c8	eb	bb	3c	83	53	99	61
fx	17	2b	04	7e	ba	77	d6	26	e1	69	14	63	55	21	0c	7d

Inverse Shift Rows

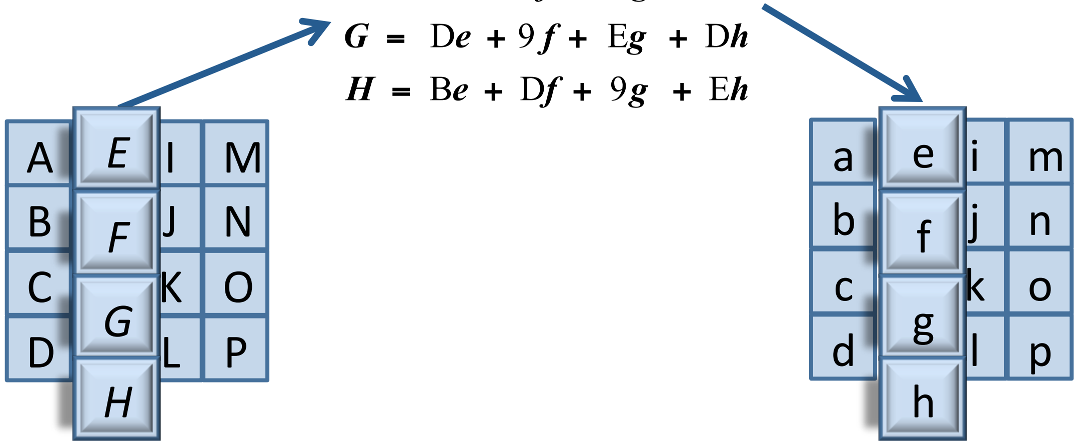




- ***ShiftRows***
  - Leave the First row untouched
  - Right Rotate (2nd Row by 8 bits)
  - Right Rotate (3rd Row by 16 bits)
  - Right Rotate (4th Row by 24 bits)

Inverse Mix Column

$$
\begin{aligned}
\boldsymbol{E} &= \mathrm{E}\boldsymbol{e} + \mathrm{B}\boldsymbol{f} + \mathrm{D}\boldsymbol{g} + 9\boldsymbol{h} \\
\boldsymbol{F} &= 9\boldsymbol{e} + \mathrm{E}\boldsymbol{f} + \mathrm{B}\boldsymbol{g} + \mathrm{D}\boldsymbol{h} \\
\boldsymbol{G} &= \mathrm{D}\boldsymbol{e} + 9\boldsymbol{f} + \mathrm{E}\boldsymbol{g} + \mathrm{D}\boldsymbol{h} \\
\boldsymbol{H} &= \mathrm{B}\boldsymbol{e} + \mathrm{D}\boldsymbol{f} + 9\boldsymbol{g} + \mathrm{E}\boldsymbol{h}
\end{aligned}
$$



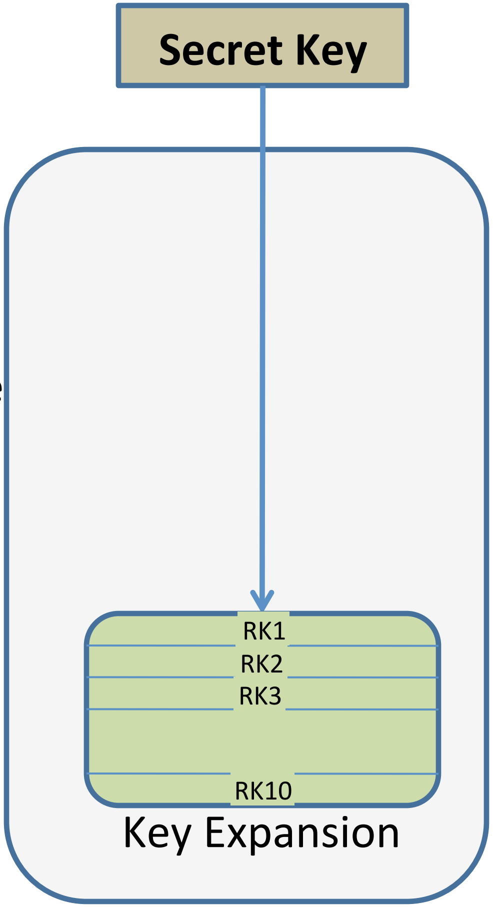
- The 4x4 matrix is multiplied with the matrix

$$
\begin{bmatrix} \mathrm{E} & \mathrm{B} & \mathrm{D} & 9 \\ 9 & \mathrm{E} & \mathrm{B} & \mathrm{D} \\ \mathrm{D} & 9 & \mathrm{E} & \mathrm{B} \\ \mathrm{B} & \mathrm{D} & 9 & \mathrm{E} \end{bmatrix}
$$

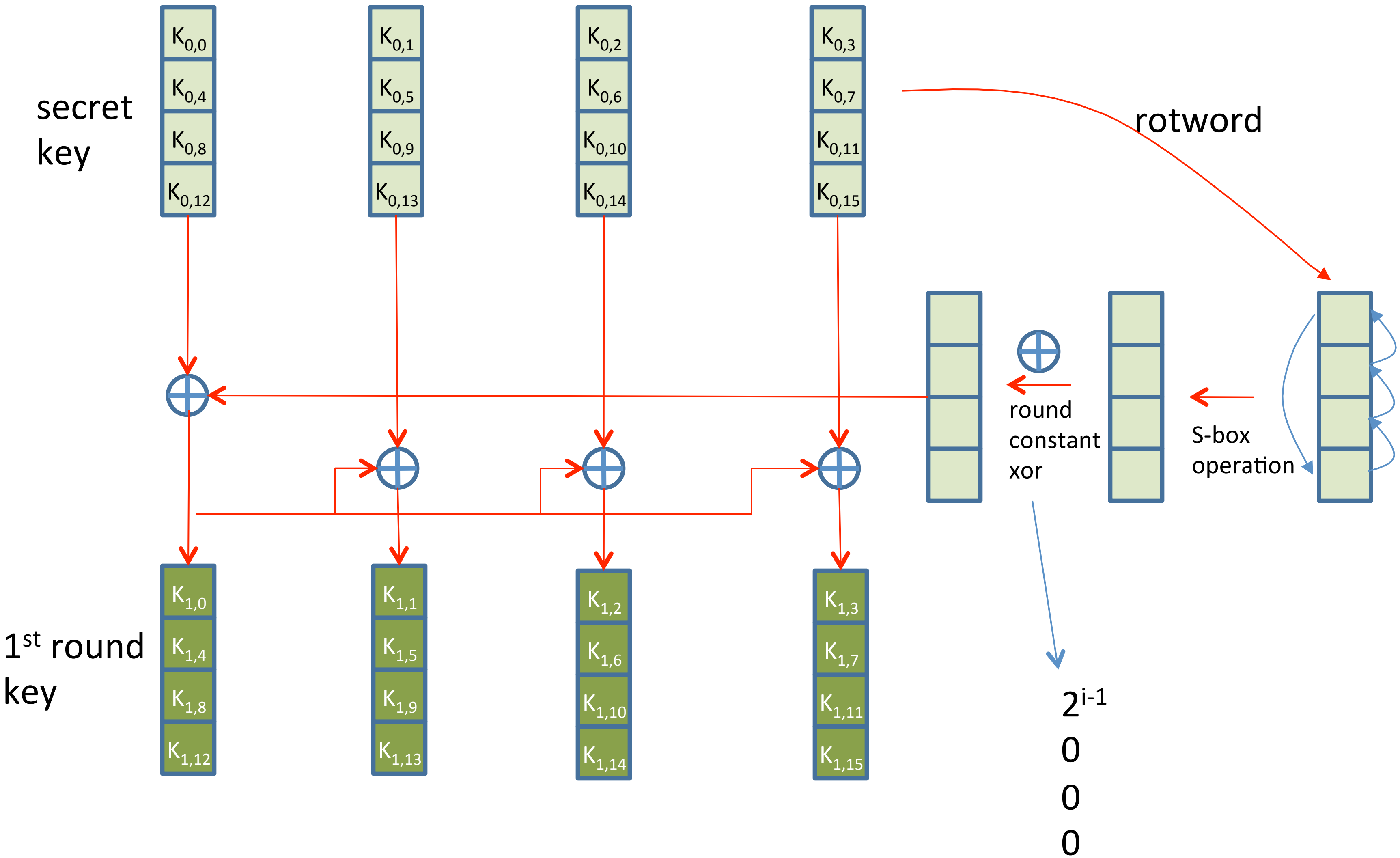
- The hardware implementation can be done in a similar way as mix columns

AES Key Schedule

- How to expand the secret key
- Design Criteria
  - Efficient
  - Non-symmetric : Ensured by round constants
  - Efficient diffusion properties of secret key into round keys
  - It should exhibit enough non-linearity to prohibit the full determination of differences in the expanded key from ciphe key differences only .



AES Key Schedule



Implementation Aspects of AES

Software Implementations of AES Encryption

- S-box implemented as a lookup-table (256 bytes)
- Shift rows combined with Mix columns
- Multiplication with MDS matrix easily achieved
  - x2, done by left shift. If there is an overflow an ex-or with 0x1B is needed
  - x3 = x2 + x

AES on 32 bit Systems

Byte Substitution

$$b_{i,j} = S(a_{i,j}) \;\; for \;\; i,j \in \{0,1,2,3\}$$

AES state

$$\begin{bmatrix} a_{0,0} & a_{0,1} & a_{0,2} & a_{0,3} \\ a_{1,0} & a_{1,1} & a_{1,2} & a_{1,3} \\ a_{2,0} & a_{2,1} & a_{2,2} & a_{2,3} \\ a_{3,0} & a_{3,1} & a_{3,2} & a_{3,3} \end{bmatrix}$$

Shift Rows
(c1 = c2 = c3 = 1 are cyclic shifts)

$$\begin{bmatrix} c_{0,j} \\ c_{1,j} \\ c_{2,j} \\ c_{3,j} \end{bmatrix} = \begin{bmatrix} b_{0,j} \\ b_{1,C1-j} \\ b_{2,C2-j} \\ b_{3,C3-j} \end{bmatrix}$$

Mix Columns

$$\begin{bmatrix} d_{0,j} \\ d_{1,j} \\ d_{2,j} \\ d_{3,j} \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} c_{0,j} \\ c_{1,j} \\ c_{2,j} \\ c_{3,j} \end{bmatrix}.$$

Combining Operations

$$\begin{bmatrix} e_{0,j} \\ e_{1,j} \\ e_{2,j} \\ e_{3,j} \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} S[a_{0,j}] \\ S[a_{1,j-C1}] \\ S[a_{2,j-C2}] \\ S[a_{3,j-C3}] \end{bmatrix} \oplus \begin{bmatrix} k_{0,j} \\ k_{1,j} \\ k_{2,j} \\ k_{3,j} \end{bmatrix}.$$

Add Round Key

$$e_{i,j} = d_{i,j} \oplus k_{i,j} \;\; for \;\; i,j \in \{0,1,2,3\}$$

T Tables

Combining Operations

$$\begin{bmatrix} e_{0,j} \\ e_{1,j} \\ e_{2,j} \\ e_{3,j} \end{bmatrix} = \mathrm{S}\left[a_{0,j}\right]\begin{bmatrix} 02 \\ 01 \\ 01 \\ 03 \end{bmatrix} \oplus \mathrm{S}\left[a_{1,j-C1}\right]\begin{bmatrix} 03 \\ 02 \\ 01 \\ 01 \end{bmatrix} \oplus \mathrm{S}\left[a_{2,j-C2}\right]\begin{bmatrix} 01 \\ 03 \\ 02 \\ 01 \end{bmatrix} \oplus \mathrm{S}\left[a_{3,j-C3}\right]\begin{bmatrix} 01 \\ 01 \\ 03 \\ 02 \end{bmatrix} \oplus \begin{bmatrix} k_{0,j} \\ k_{1,j} \\ k_{2,j} \\ k_{3,j} \end{bmatrix}.$$

Define 4 T-Tables

$$\mathrm{T}_0[a] = \begin{bmatrix} \mathrm{S}[a] \bullet 02 \\ \mathrm{S}[a] \\ \mathrm{S}[a] \\ \mathrm{S}[a] \bullet 03 \end{bmatrix} \quad \mathrm{T}_1[a] = \begin{bmatrix} \mathrm{S}[a] \bullet 03 \\ \mathrm{S}[a] \bullet 02 \\ \mathrm{S}[a] \\ \mathrm{S}[a] \end{bmatrix} \quad T_2[a] = \begin{bmatrix} \mathrm{S}[a] \\ \mathrm{S}[a] \bullet 03 \\ \mathrm{S}[a] \bullet 02 \\ \mathrm{S}[a] \end{bmatrix} \quad \mathrm{T}_3[a] = \begin{bmatrix} \mathrm{S}[a] \\ \mathrm{S}[a] \\ \mathrm{S}[a] \bullet 03 \\ \mathrm{S}[a] \bullet 02 \end{bmatrix}.$$

One Round of AES using T-Tables

$$e_j = \mathrm{T}_0\left[a_{0,j}\right] \oplus \mathrm{T}_1\left[a_{1,j-C1}\right] \oplus \mathrm{T}_2\left[a_{2,j-C2}\right] \oplus \mathrm{T}_3\left[a_{3,j-C3}\right] \oplus k_j \,.$$

OpenSSL Implementation of AES (with T-tables)

```
static const u32 Te0[256] = {
    0xc66363a5U, 0xf87c7c84U, 0xee777799U, 0xf67b7b8dU,
    0xfff2f20dU, 0xd66b6bbdU, 0xde6f6fb1U, 0x91c5c554U,
    0x60303050U, 0x02010103U, 0xce6767a9U, 0x562b2b7dU,

static const u32 Te1[256] = {
    0xa5c66363U, 0x84f87c7cU, 0x99ee7777U, 0x8df67b7bU,
    0x0dfff2f2U, 0xbdd66b6bU, 0xb1de6f6fU, 0x5491c5c5U,

static const u32 Te2[256] = {
    0x63a5c663U, 0x7c84f87cU, 0x7799ee77U, 0x7b8df67bU,
    0xf20dfff2U, 0x6bbdd66bU, 0x6fb1de6fU, 0xc55491c5U,

static const u32 Te3[256] = {
    0x6363a5c6U, 0x7c7c84f8U, 0x777799eeU, 0x7b7b8df6U,
    0xf2f20dffU, 0x6b6bbdd6U, 0x6f6fb1deU, 0xc5c55491U,
    0x30305060U, 0x01010302U, 0x6767a9ceU, 0x2b2b7d56U,
```

```
s0 = GETU32(in     ) ^ rk[0];
s1 = GETU32(in +  4) ^ rk[1];
s2 = GETU32(in +  8) ^ rk[2];
s3 = GETU32(in + 12) ^ rk[3];

/* round 1: */
t0 = Te0[s0 >> 24] ^ Te1[(s1 >> 16) & 0xff] ^ Te2[(s2 >>  8) & 0xff] ^ Te3[s3 & 0xff] ^ rk[ 4];
t1 = Te0[s1 >> 24] ^ Te1[(s2 >> 16) & 0xff] ^ Te2[(s3 >>  8) & 0xff] ^ Te3[s0 & 0xff] ^ rk[ 5];
t2 = Te0[s2 >> 24] ^ Te1[(s3 >> 16) & 0xff] ^ Te2[(s0 >>  8) & 0xff] ^ Te3[s1 & 0xff] ^ rk[ 6];
t3 = Te0[s3 >> 24] ^ Te1[(s0 >> 16) & 0xff] ^ Te2[(s1 >>  8) & 0xff] ^ Te3[s2 & 0xff] ^ rk[ 7];

/* round 2: */
s0 = Te0[t0 >> 24] ^ Te1[(t1 >> 16) & 0xff] ^ Te2[(t2 >>  8) & 0xff] ^ Te3[t3 & 0xff] ^ rk[ 8];
s1 = Te0[t1 >> 24] ^ Te1[(t2 >> 16) & 0xff] ^ Te2[(t3 >>  8) & 0xff] ^ Te3[t0 & 0xff] ^ rk[ 9];
s2 = Te0[t2 >> 24] ^ Te1[(t3 >> 16) & 0xff] ^ Te2[(t0 >>  8) & 0xff] ^ Te3[t1 & 0xff] ^ rk[10];
s3 = Te0[t3 >> 24] ^ Te1[(t0 >> 16) & 0xff] ^ Te2[(t1 >>  8) & 0xff] ^ Te3[t2 & 0xff] ^ rk[11];
```

Last Round of AES

- Uses a different table (Te4)

```
s0 =
        (Te4[(t0 >> 24)       ] & 0xff000000) ^
        (Te4[(t1 >> 16) & 0xff] & 0x00ff0000) ^
        (Te4[(t2 >>  8) & 0xff] & 0x0000ff00) ^
        (Te4[(t3      ) & 0xff] & 0x000000ff) ^
        rk[0];
PUTU32(out     , s0);
s1 =
        (Te4[(t1 >> 24)       ] & 0xff000000) ^
        (Te4[(t2 >> 16) & 0xff] & 0x00ff0000) ^
        (Te4[(t3 >>  8) & 0xff] & 0x0000ff00) ^
        (Te4[(t0      ) & 0xff] & 0x000000ff) ^
        rk[1];
PUTU32(out +  4, s1);
s2 =
        (Te4[(t2 >> 24)       ] & 0xff000000) ^
        (Te4[(t3 >> 16) & 0xff] & 0x00ff0000) ^
        (Te4[(t0 >>  8) & 0xff] & 0x0000ff00) ^
        (Te4[(t1      ) & 0xff] & 0x000000ff) ^
        rk[2];
PUTU32(out +  8, s2);

s3 =
        (Te4[(t3 >> 24)       ] & 0xff000000) ^
        (Te4[(t0 >> 16) & 0xff] & 0x00ff0000) ^
        (Te4[(t1 >>  8) & 0xff] & 0x0000ff00) ^
        (Te4[(t2      ) & 0xff] & 0x000000ff) ^
        rk[3];
PUTU32(out + 12, s3);
```

AES NI

- Accelerating AES on modern Intel and AMD processors with dedicated instructions

Instruction	Description[2]
AESENC	Perform one round of an AES encryption flow
AESENCLAST	Perform the last round of an AES encryption flow
AESDEC	Perform one round of an AES decryption flow
AESDECLAST	Perform the last round of an AES decryption flow
AESKEYGENASSIST	Assist in AES round key generation
AESIMC	Assist in AES Inverse Mix Columns
PCLMULQDQ	Carryless multiply (CLMUL).[3]

Compact Implementations of AES

- How should the S-box be implemented?
  - Look up table (256 bytes)
    - This may be too large for some devices
  - Finding the inverse (using Itoh-Tsujii or the extended Euclidean algorithm) and then affine transformation
    - Again expensive (too big!!!)
  - Third alternative
    - Use composite fields

Composite Fields (refer Math. Background)

Composite Fields for AES

- The AES Field is $GF(2^8)/x^8+x^4+x^3+x+1$
  - Has order 256
- Many composite fields for AES exists
  - $GF(2^4)^2$
    - Requires two irreducible polynomials

      One has the form $x^4 + ...$, where coefficients are in GF(2)

      The second has the form $x^2 + ax + b$, where a, b are in $GF(2^4)$
  - $GF((2^2)^2)^2$
    - Requires three irreducible polynomials

      First of the form $x^2 + a_1x + b_1$, where a_1, b_1 in GF(2)

      Second has the form $x^2 + a_2x + b_2$, where a_2, b_2 in $GF(2^2)$

      Third has the form $x^2 + a_3x + b_3$, where a_3, b_3 in $GF(2^2)^2$

Mapping between GF(2^8) and Composite Fields

FindMap(){

Initilize $MAP[0] = 0$ and $REVMAP[0] = 0$

Find α a primitive root of field $GF(2^8)$

Find β a primitive root of field $GF(2^4)^2$

$\alpha' = 1; \beta' = 1$

For $i = 1$ *to* 255

$\alpha' = \alpha \cdot \alpha'$ (Multiplication in the field GF(2^8))

$\beta' = \beta \cdot \beta'$ (Multiplication in the field GF$(2^4)^2$

$MAP[\alpha'] = \beta'$

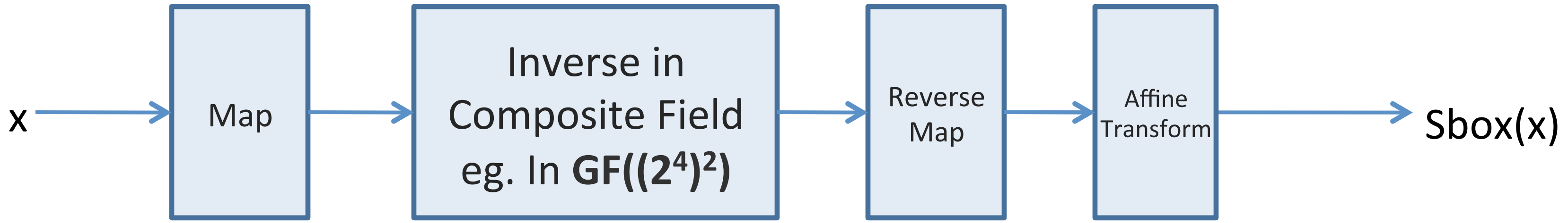
$REVMAP[\beta'] = \alpha'$

return MAP and $REVMAP$

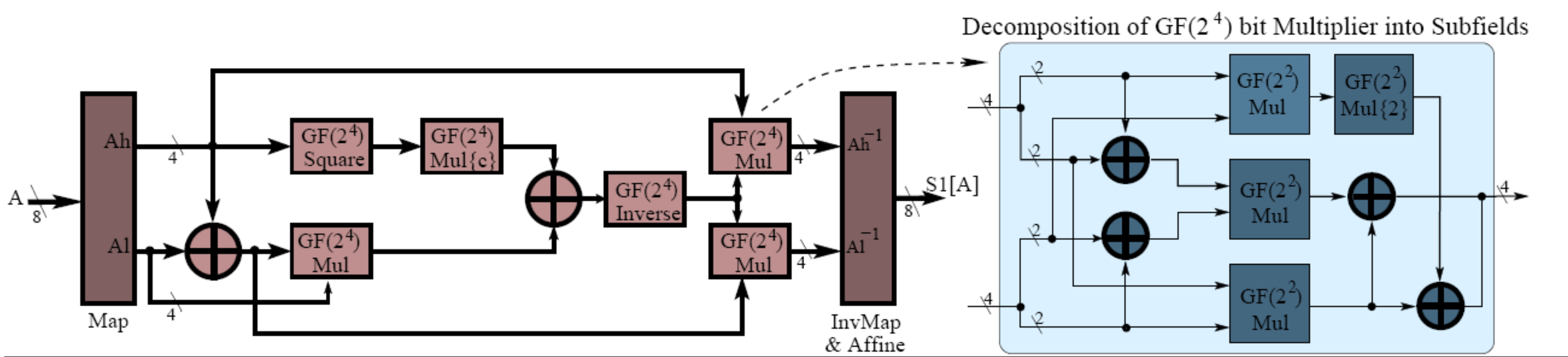
}

https://drive.google.com/file/d/0BwxUBZXYoUKCTEJmNUozMl9xM3M/edit?usp=sharing

Implementing the AES S-box in Composite Fields



S-box Based on Composite Fields



Gate Count for composite Sbox#

XOR	NAND	NOR	Total Gates in terms of NAND (using std cell lib)
80	34	6	180

Performance of S-boxes on FPGA*

S-box Approach	No. of Slices	Critical Path	Gate Count
Lookup table based	64	11.9ns	1128
Composite Field based	30	18.3ns	312

D. Canright, *A Very Compact S-box for AES*, CHES-2005

* Simulation Results using Xilinx ISE

Overhead of Composite Field s-boxes

- Composite field s-boxes require mapping and reverse mapping to and from the composite fields in each round
- An alternate approach is to convert all other round operations into composite field operations.
  - This would require just one mapping and one reverse mapping for the entire encryption
  - Operations Add Round Key and Shift Rows are not altered.
  - Mix Columns will need to be re-implemented

Attacks on AES

Differential and Linear Properties of AES

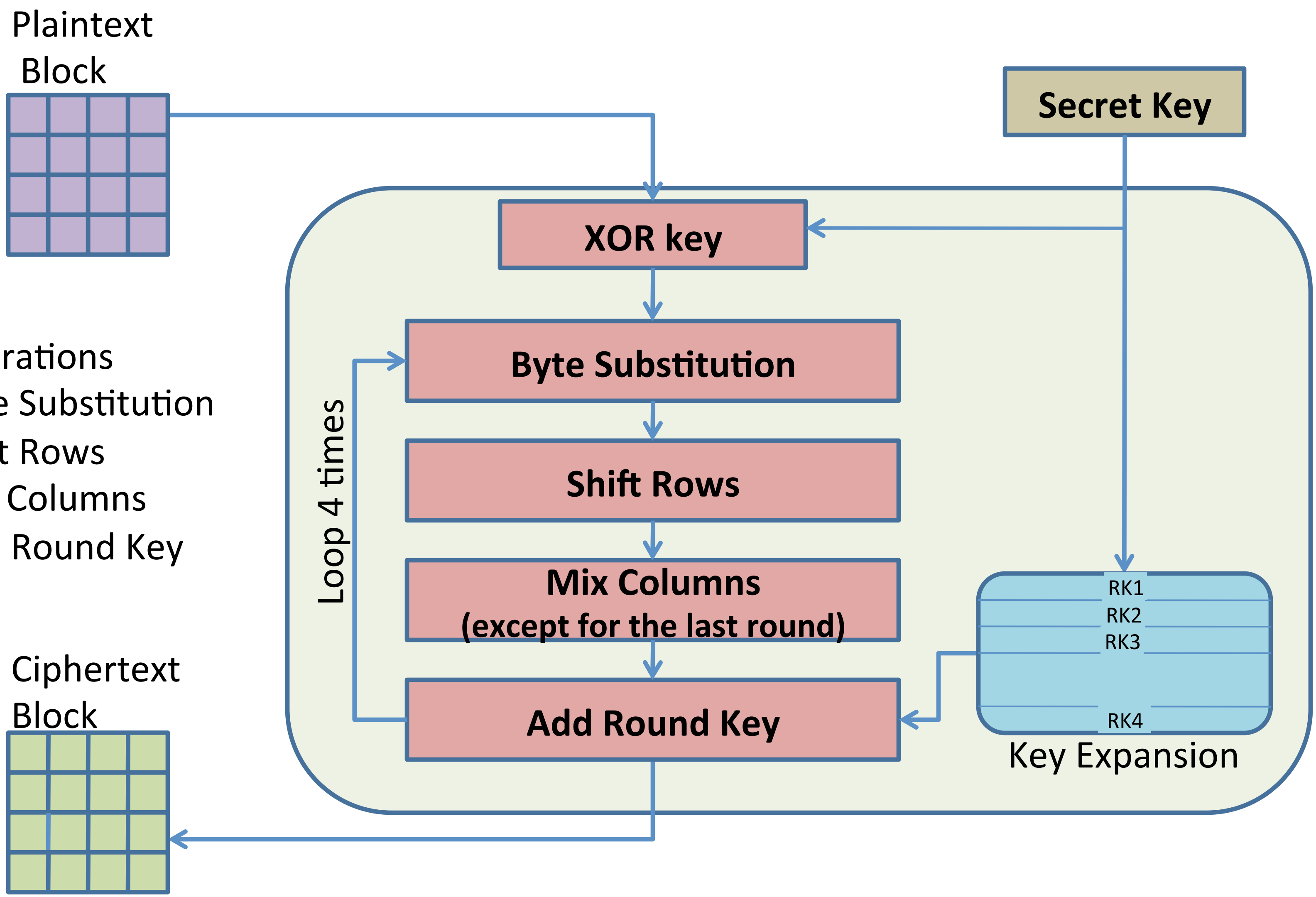
- Differential Cryptanalysis
  - No 4 round differential trail > $1/2^{150}$ and no 8 round differential trail > $1/2^{300}$ exists.
- Linear Cryptanalysis
  - No 4 round bias > $1/2^{75}$ and no 8 round bias > $1/2^{150}$ exists

AES can easily resist differential and linear cryptanalysis

Attack on 4 Rounds of AES

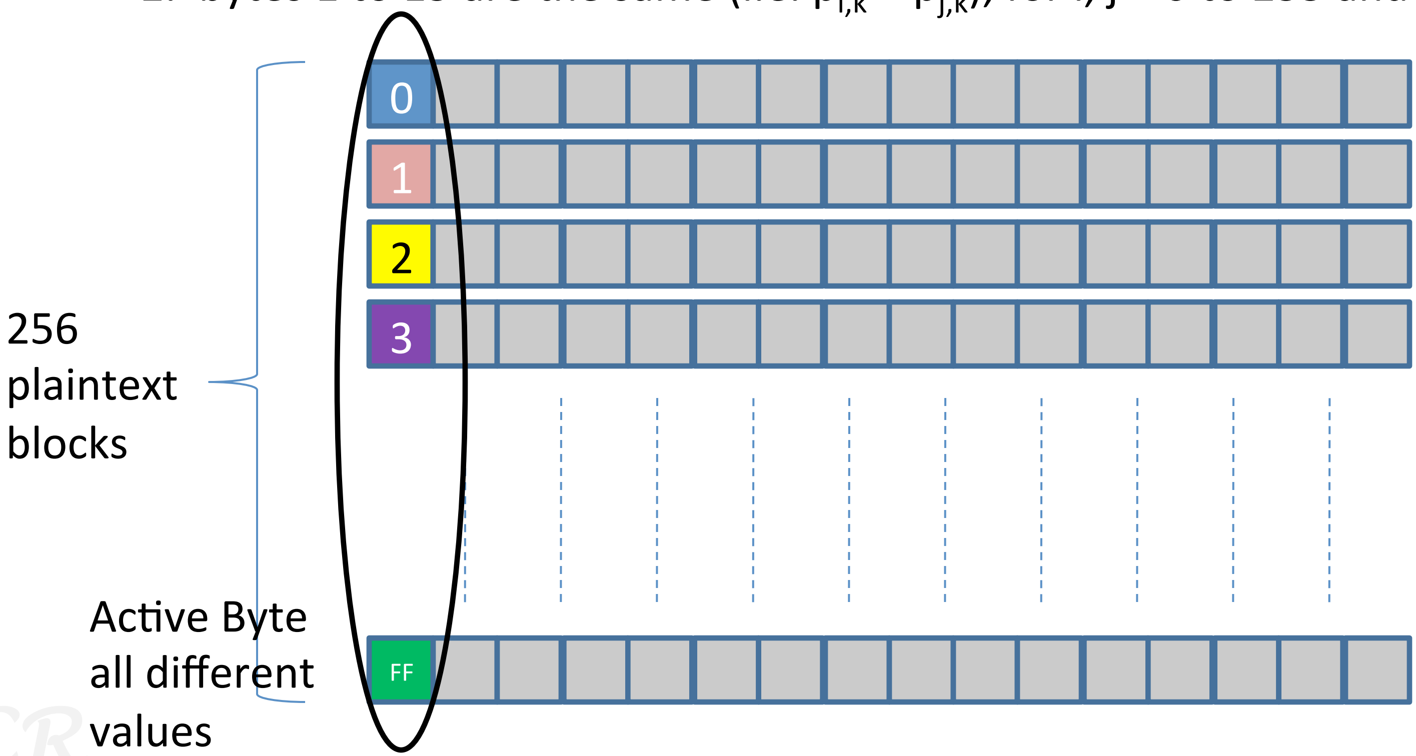
4 Operations

- Byte Substitution
- Shift Rows
- Mix Columns
- Add Round Key



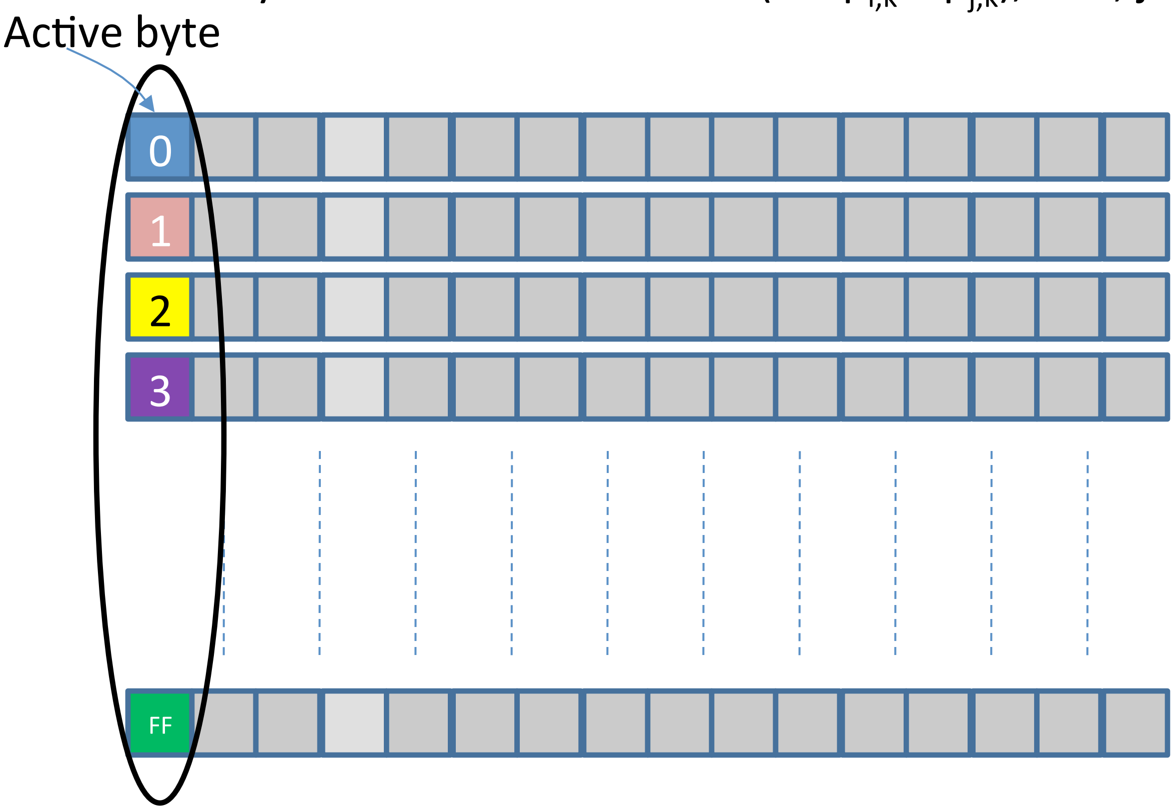
Square Attack (known by the AES designers)

- Works for 4 round of AES
- Can be extended up to 6 rounds
- Consider 256 plaintext blocks having the following properties
  1. byte 0 is different for in all cases (i.e. $p_{i,0} \neq p_{j,0}$), for i, j = 0 to 255 and i ≠ j
  2. bytes 1 to 15 are the same (i.e. $p_{i,k} = p_{j,k}$), for i, j = 0 to 255 and 1 ≤ k ≤ 15



Square Attack

- Consider 256 plaintext blocks having the following properties
  1. byte 0 is different in all cases (i.e. $p_{i,0} \neq p_{j,0}$), for i, j = 0 to 255 and i ≠ j
  2. bytes 1 to 15 are the same (i.e. $p_{i,k} = p_{j,k}$), for i, j = 0 to 255 and 1 ≤ k ≤ 15



Two properties

$$\bigoplus_{i=0}^{255} p_{i,k} = 0$$

For some k; 1 ≤ k ≤ 15

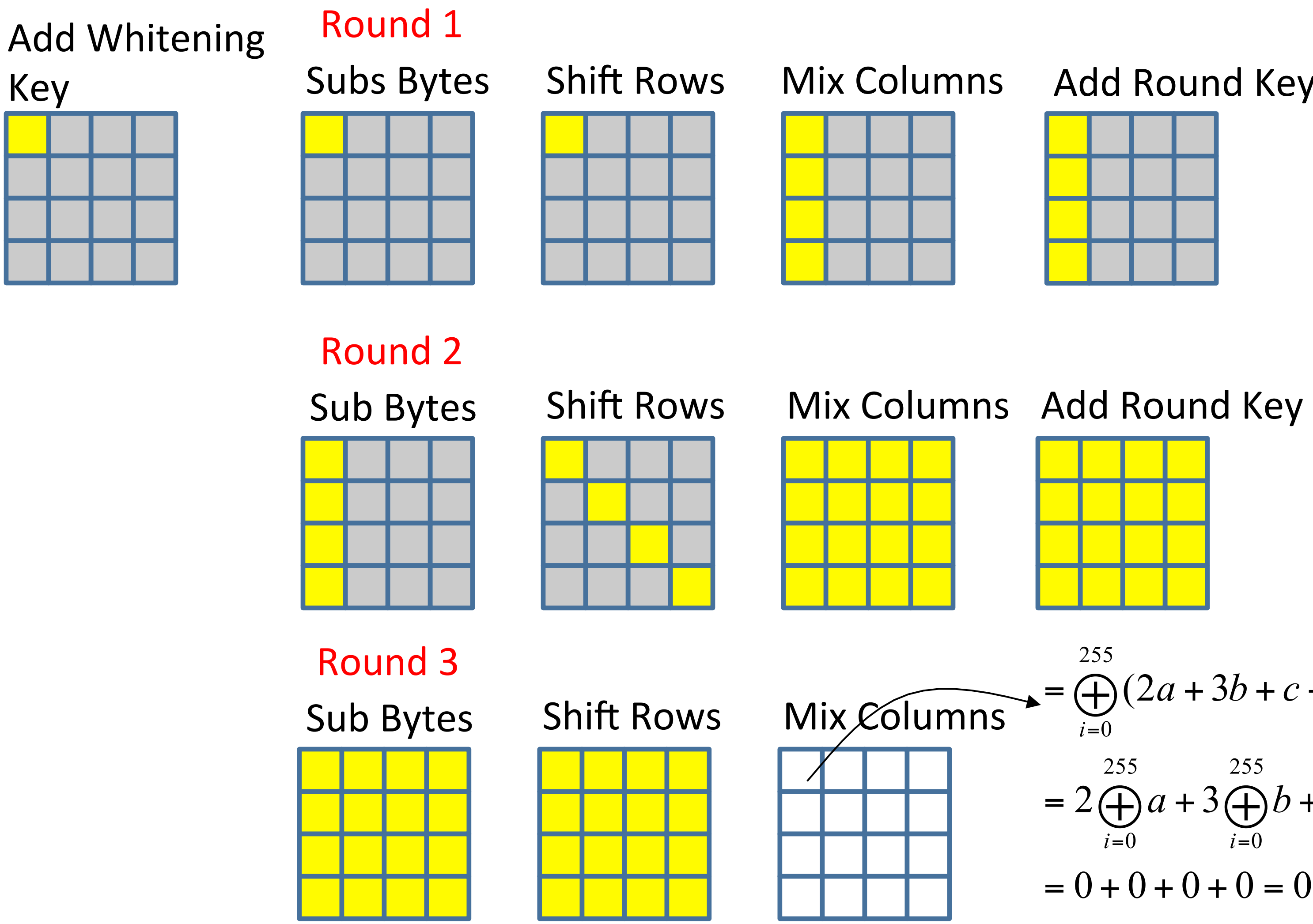
The state is balanced

$$\bigoplus_{i=0}^{255} p_{i,0} = 0$$

Square Attack
(Propagation in 3 rounds)

Active byte property

$$\bigoplus_{i=0}^{255} p_{i,0} = 0$$



$$= \bigoplus_{i=0}^{255} (2a + 3b + c + d)$$

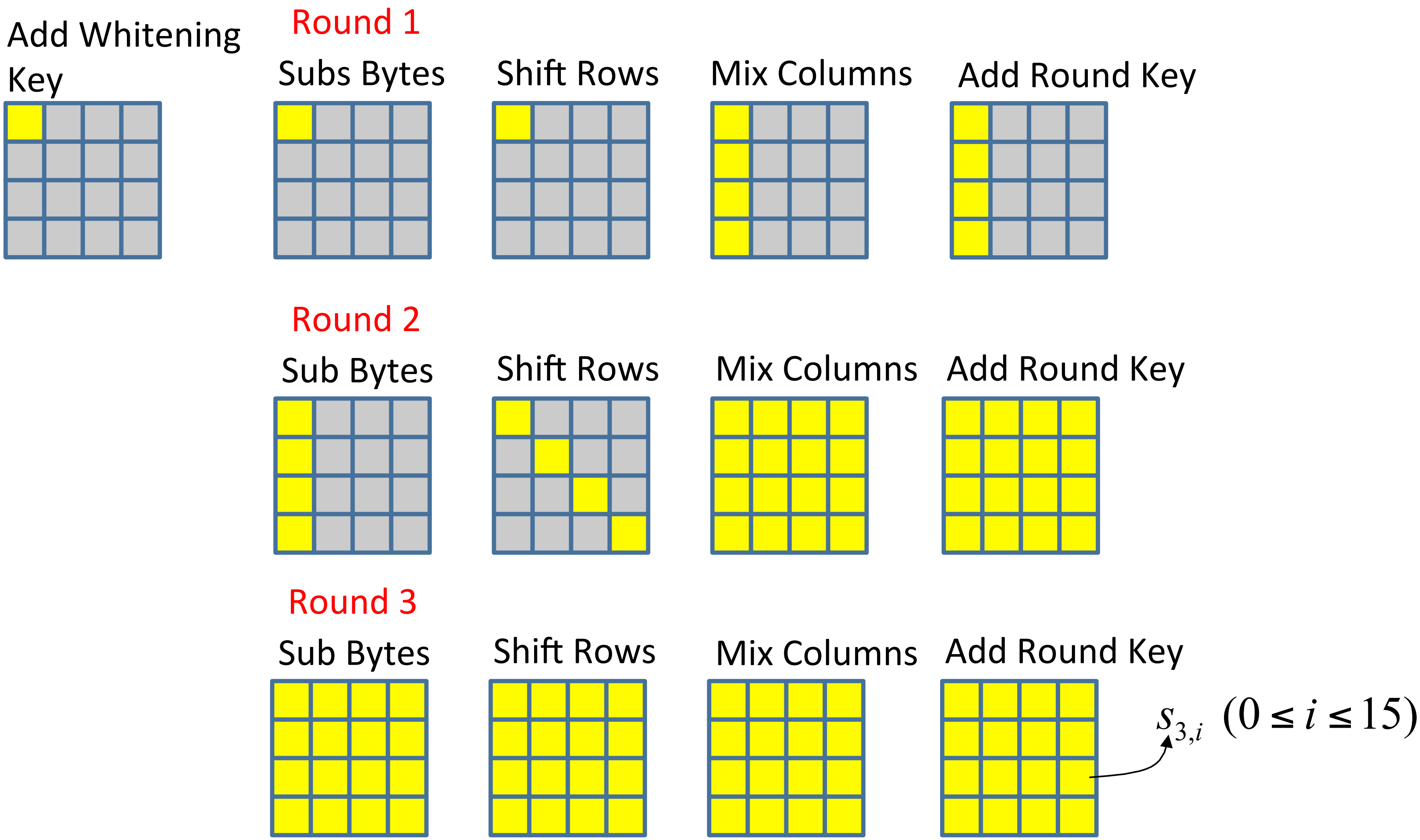
$$= 2\bigoplus_{i=0}^{255} a + 3\bigoplus_{i=0}^{255} b + \bigoplus_{i=0}^{255} c + \bigoplus_{i=0}^{255} d$$

$$= 0 + 0 + 0 + 0 = 0$$

Balanced retained

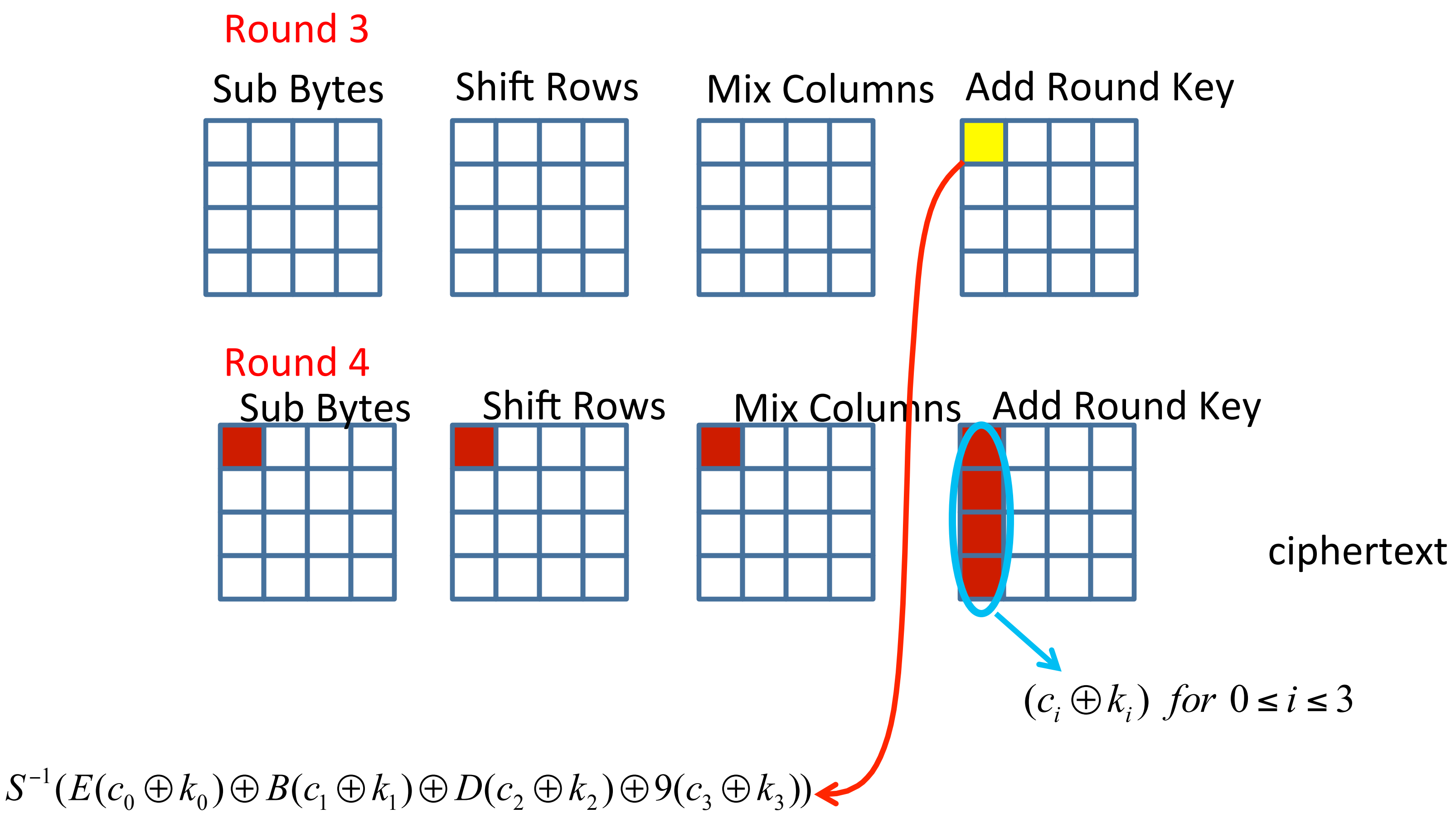
Square Attack
(Propagation in 3 rounds)

Active byte property: $\bigoplus_{i=0}^{255} p_{i,0} = 0$



This property does not hold after Sub Bytes in the 4th Round

A 4 round square attack



4 round square attack (A chosen plaintext attack)

1. Choose 256 plaintexts with one active byte
2. Perform 4 round encryption for each plaintext
3. For each potential key $(k_0 \| k_1 \| k_2 \| k_3)$ do the following,
   a. Compute $s_{3,0}$ corresponding to each c_i (there are 256 such c_i)
      call them $s_{3,0}^{(0)}, s_{3,0}^{(1)}, s_{3,0}^{(2)}, \cdots s_{3,0}^{(255)}$
   b. compute $\bigoplus_{i=0}^{255} s_{3,0}^{(i)}$
      If this is 0, then guessed $(k_0 \| k_1 \| k_2 \| k_3)$ **may** be correct
      If not, guessed key is incorrect

CR

Why square attack may lead to an incorrect key

- If the key guess is wrong, $\bigoplus_{i=0}^{255} s_{3,0}^{(i)}$ may still be 0.
- This is because $\bigoplus_{i=0}^{255} s_{3,0}^{(i)}$ evaluated to one of {0, 1, 2, 3, …., 255} with equal probability

- Thus with probability 2^{-8}, we may get $\bigoplus_{i=0}^{255} s_{3,0}^{(i)} = 0$ for the wrong key.

CR

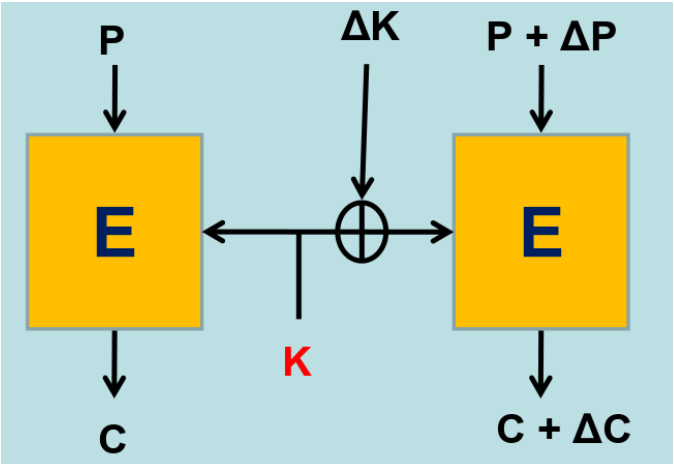
Extending beyond 4 rounds

Read how the square attack can be extended to 5 rounds and 6 rounds.

math.boisestate.edu/~liljanab/Math509Spring10/AES-security.pdf

Related Key Attacks on AES (theoretical attacks on full AES)

- By Alex Biryukov and Dmitry Khovratovich (2009)
- Strong assumption : the attacker forces the victim to choose keys of particular form.
- Determine how key differences affect the cipher text difference



Tracing key differences