Network Security
Unit-II

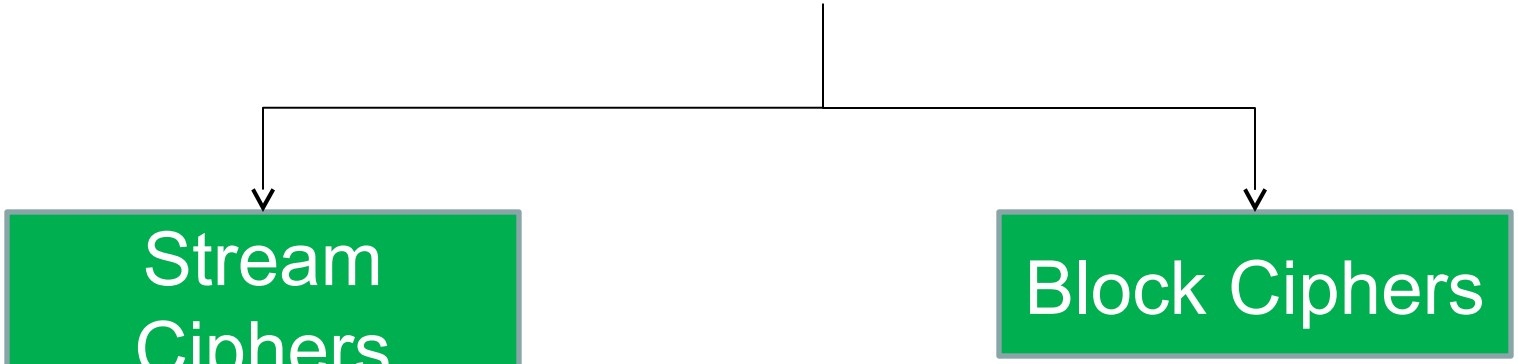
Symmetric Key Algorithms

Algorithm Types and modes

- An **algorithm type** defines what **size of plain text** should be encrypted in each step of the algorithm.

- The **algorithm mode** defines the **details of the cryptographic algorithm**, once the type is decided.

Algorithm types



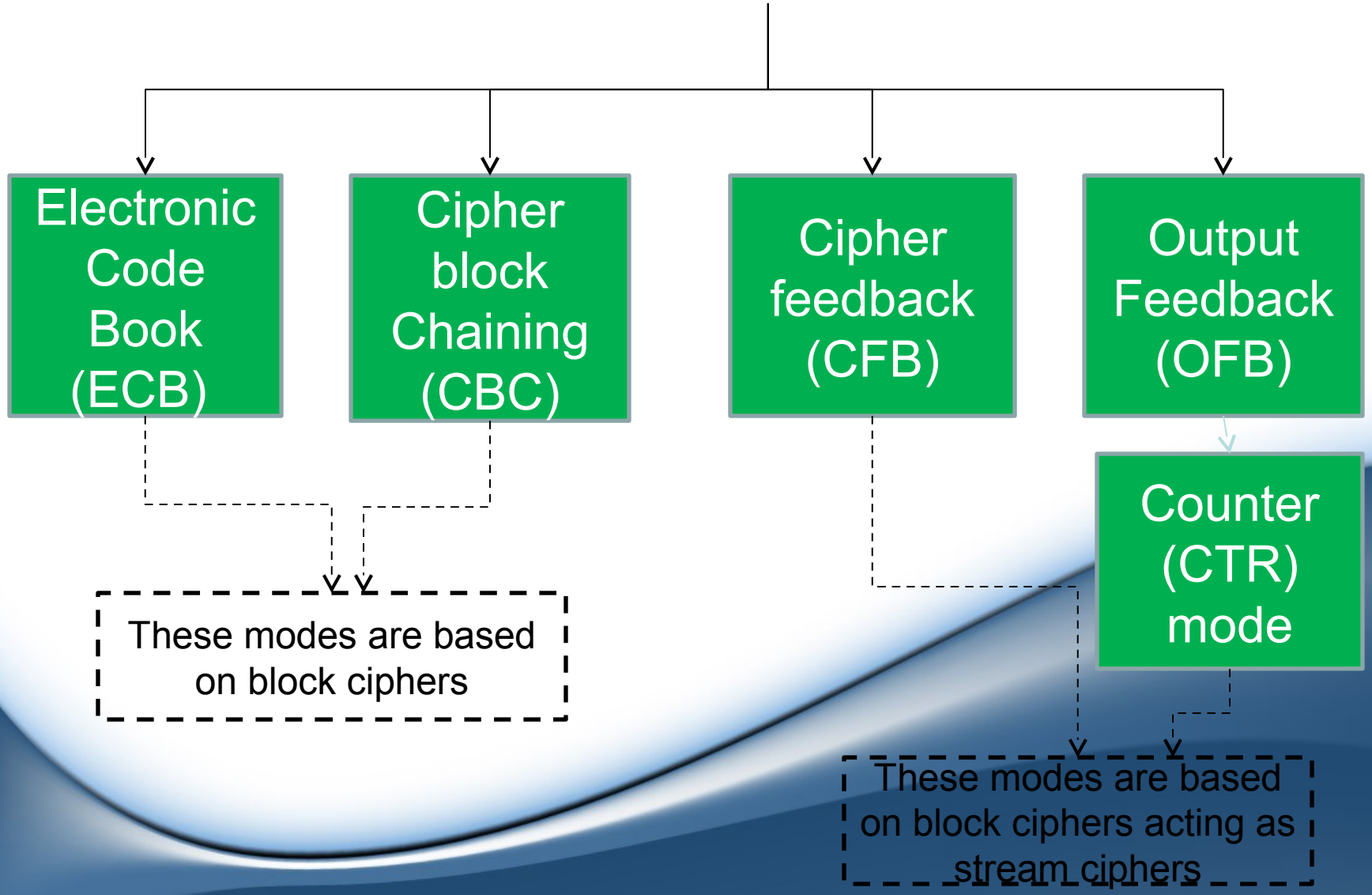
Stream Ciphers

- This technique involves **bit by bit** encryption of plain text.
- The decryption also happens one bit at a time

Block Ciphers

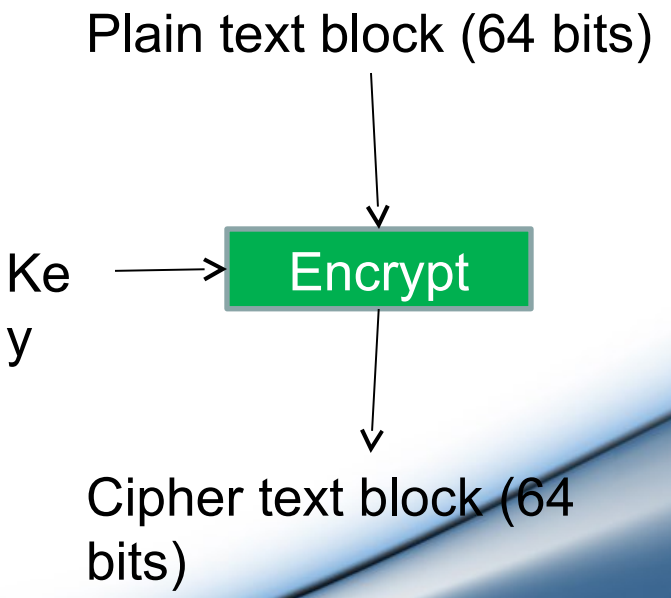
- This technique involves **block by block** of plain text.
- Decryption is done in same way

Algorithm Modes



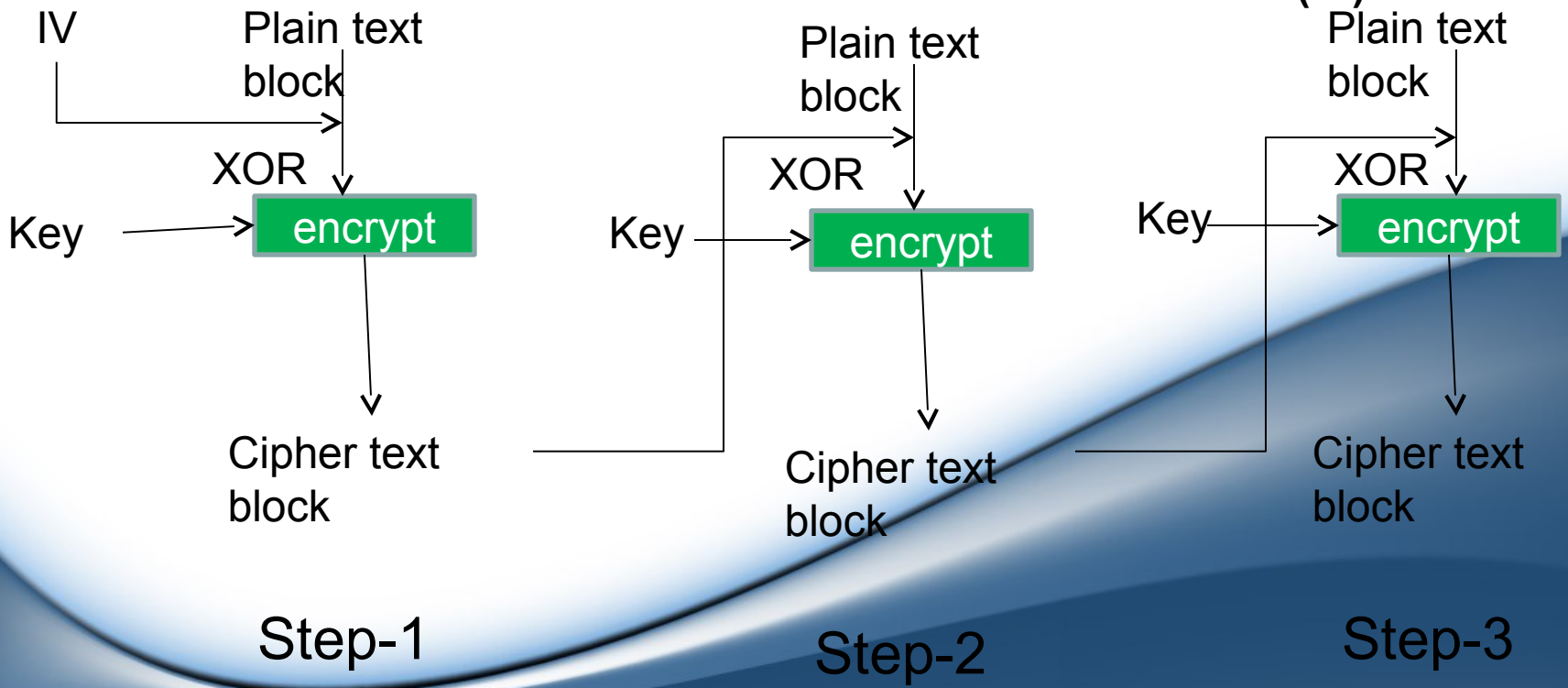
Electronic Code Book (ECB)

- The plain text is divided into block of 64 bits.
- Each such block is then encrypted with a key.
- All blocks use the same key.



Cipher Block Chaining (CBC)

- If a block of plain text is repeated, then it will generate same cipher block. Thus giving clues to the cryptanalyst.
- CBC makes sure that a same plain text block yields a different cipher block.
- For this a feedback mechanism is used. This is done through ***chaining*** the blocks.
- It also uses a random set of bits called as **Initialization Vector (IV).**



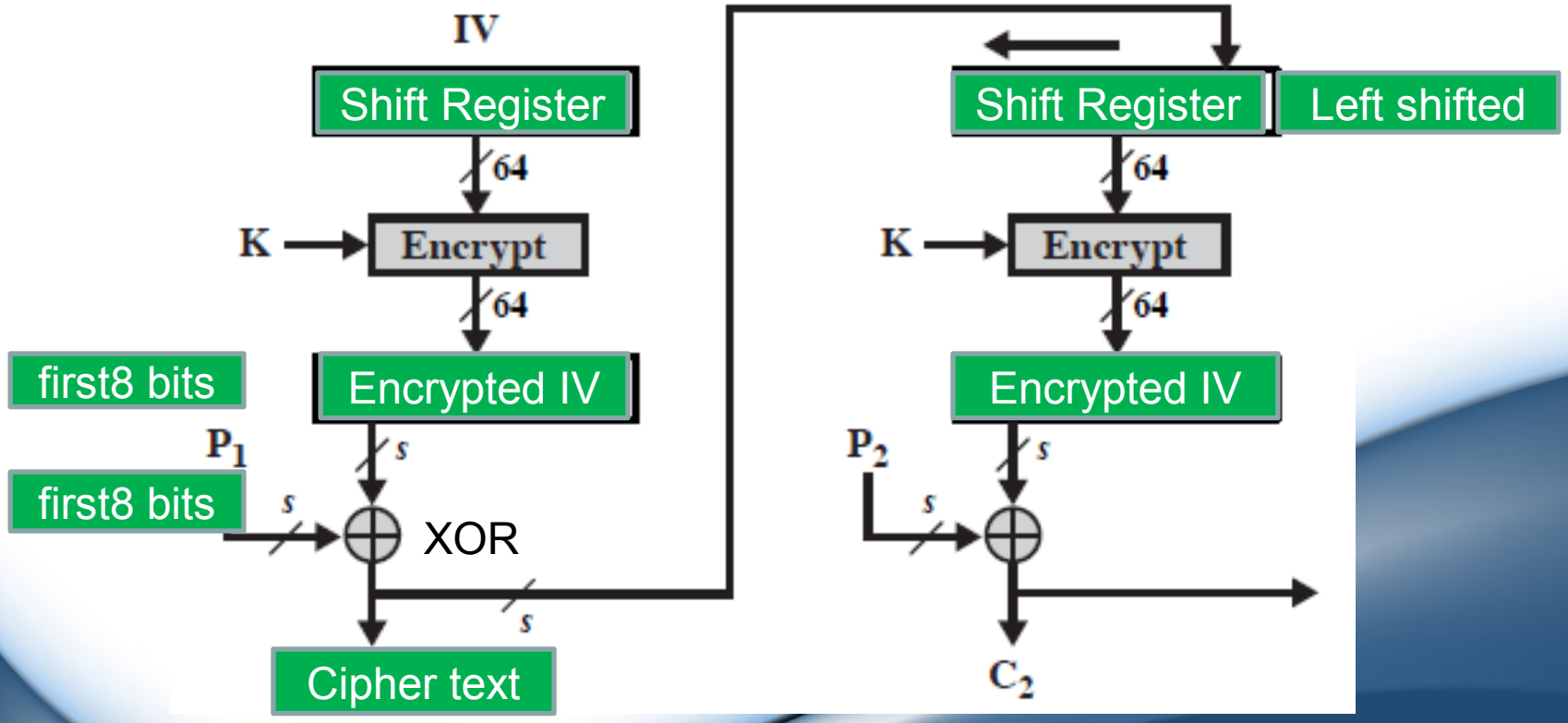
Cipher Feedback (CFB)

- Security is also required in applications that are character oriented.
- For example: typing keystrokes which are immediately transmitted across the network link.
- Thus CFB is useful, where data is encrypted in units that are smaller (i.e. 8 bits --. size of 1 character)

Step-1 : A 64-bit Initialization Vector **(IV) is kept in shift register and is encrypted using a key** and produced a 64-bit Encrypted IV.

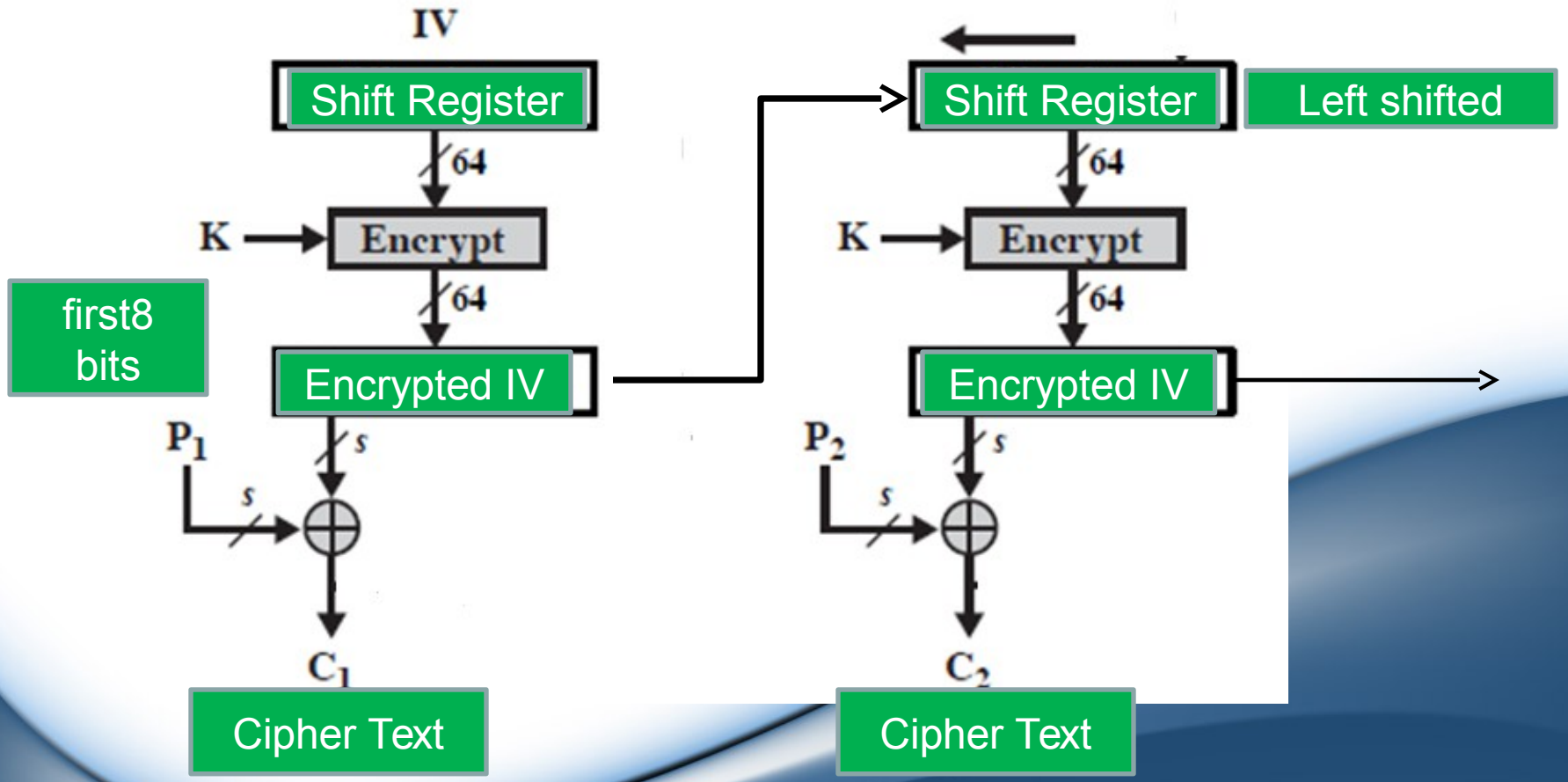
Step-2 : Take leftmost *j* bits of IV and first *j* bits of plain text and perform XOR operations.

Step-3 : Now the cipher bits so generated are left shifted in the **Shift Register**.


IV
Shift Register
64
K
Encrypt
64
Encrypted IV
first8 bits
P_1
first8 bits
s
s
XOR
s
Cipher text
Shift Register
Left shifted
64
K
Encrypt
64
Encrypted IV
P_2
s
s
C_2

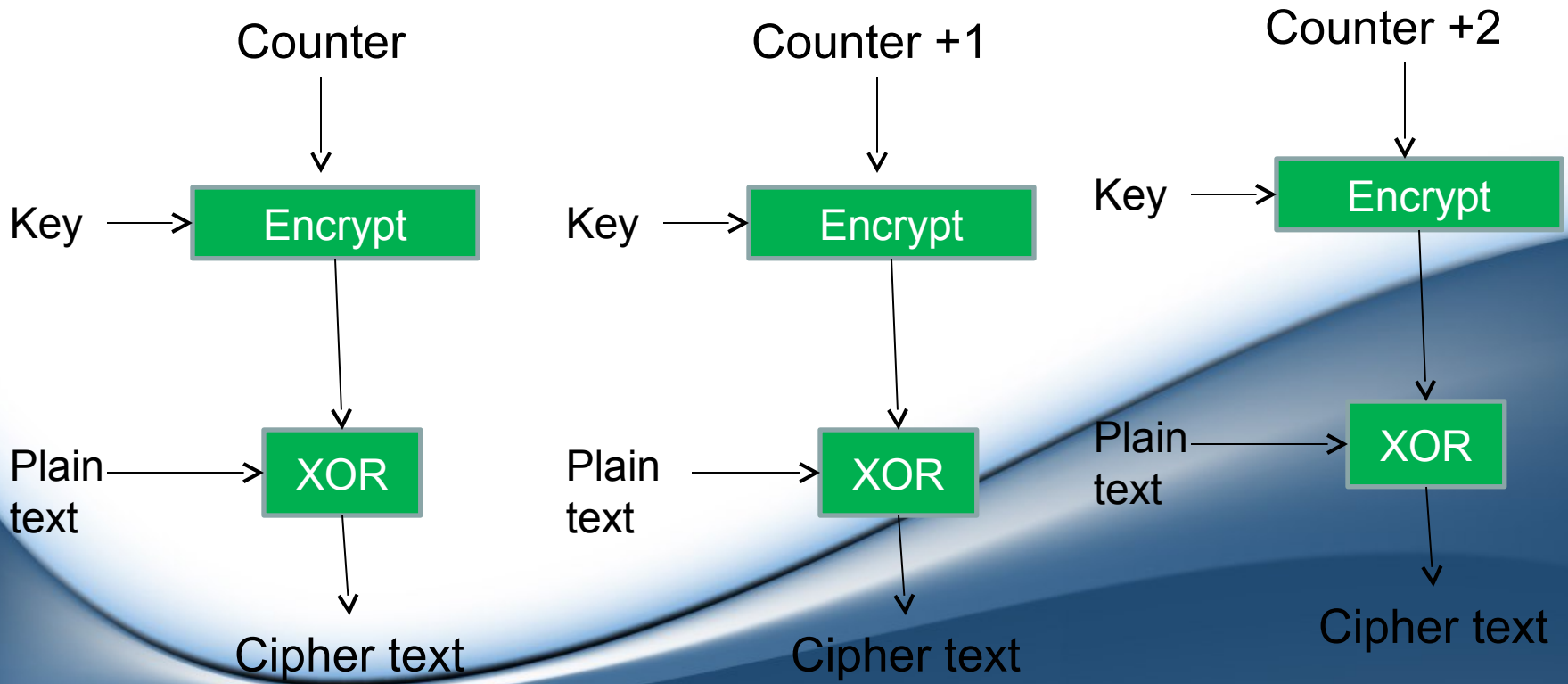
Output Feedback (OFB)

- The OFB is extremely similar to CFB.
- The only difference is that instead of feeding cipher text to the next encryption, the OFB feeds the encrypted IV to the next encryption step.


IV
Shift Register
64
K
Encrypt
64
first8 bits
Encrypted IV
P1
s
s
C1
Cipher Text
Shift Register
Left shifted
64
K
Encrypt
64
Encrypted IV
P2
s
s
C2
Cipher Text

Counter (CTR)

- The CTR is similar to OFB.
- Instead of IV, it uses a sequence number called counters as inputs to the algorithm.
- Usually , a counter is a constant.
- To fill the register , the counter is incremented



Uses

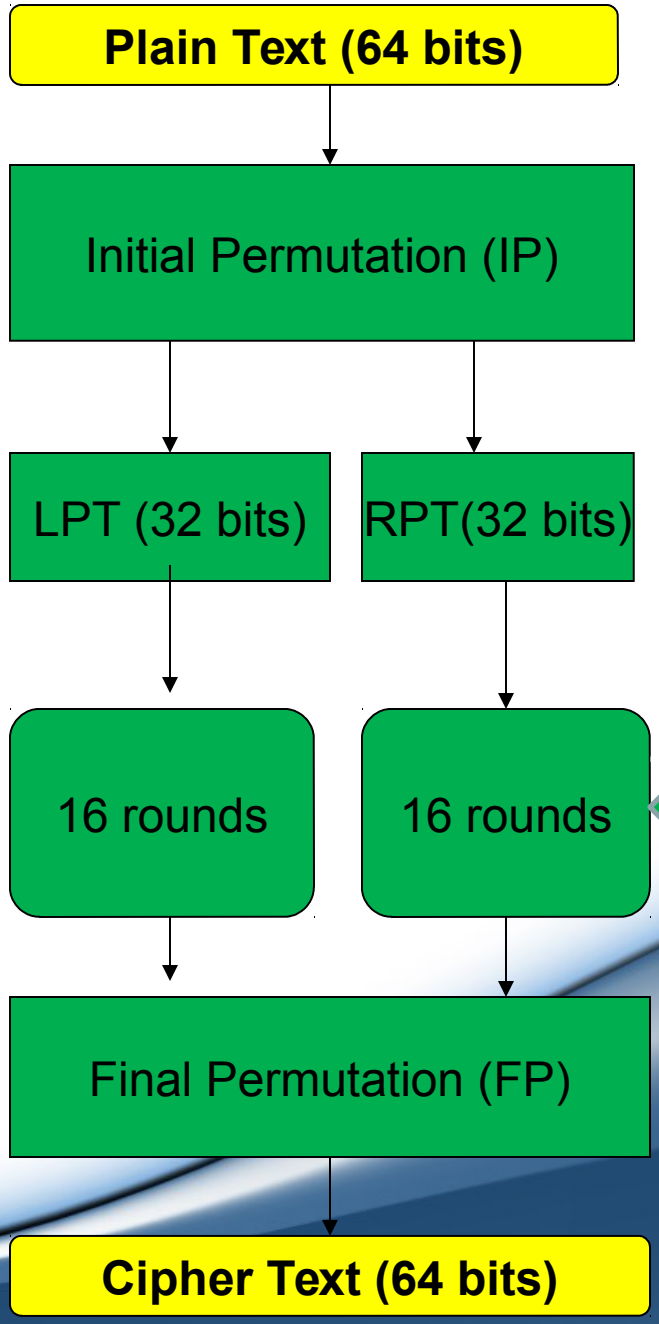
Algorithm	Usage
ECB	Transmitting a single value in a secure fashion (eg: password or key used for encryption)
CBC	Encrypting blocks of text authentication
CFB	Transmitting encrypted stream of data authentication
OFB	Transmitting encrypted stream of data
CTR	Block – oriented transmissions applications needing high speed

Symmetric Key Cryptography Algorithms

DES (Data Encryption Standard)

Designer(s)	IBM	
First published		1975
Derived from	Lucifer	

- DES is a block cipher.
- It breaks the plain text into **64-bits block**.
- Each such block is **an input to the DES** encryption.
- The key used here is of **56-bits**. (initial key consists of 64 bits. However before the DES process even starts, the key is compressed to 56-bits.)
- The compression of keys is done by **discarding every 8th, 16th, 24th, 32nd, 40th, 48th, 56th and 64th bit**.



Performed only once where:

1st bit is replaced with 58th

2nd is replaced with 50th

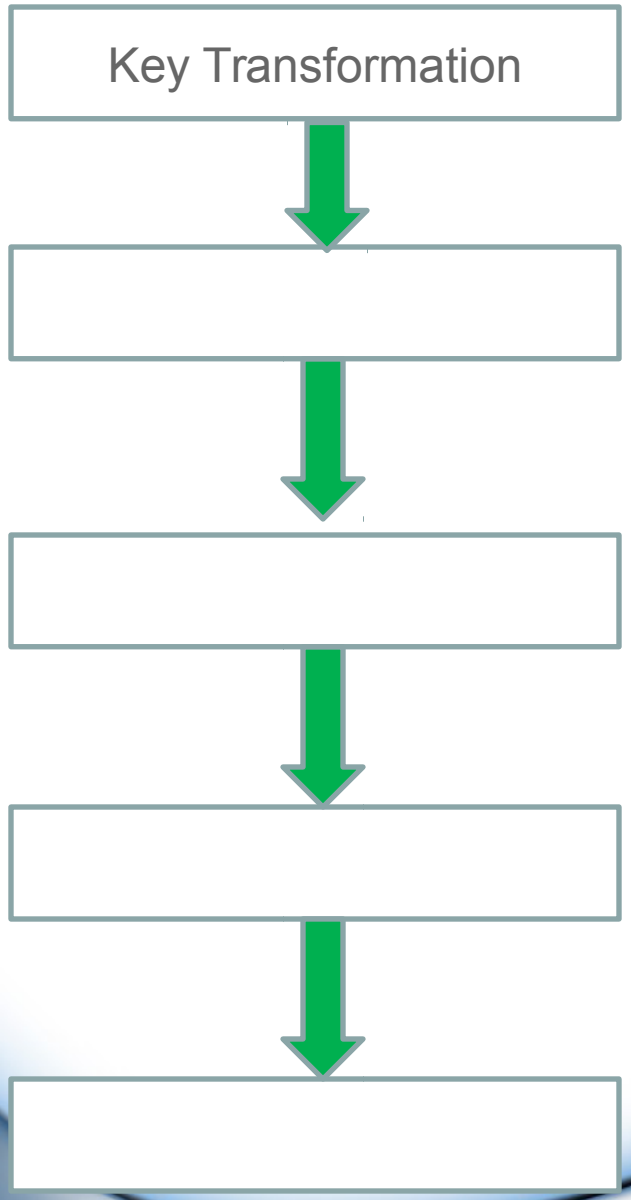
…..

Performed only once where:

1st bit is replaced with 40th

2nd is replaced with 8th

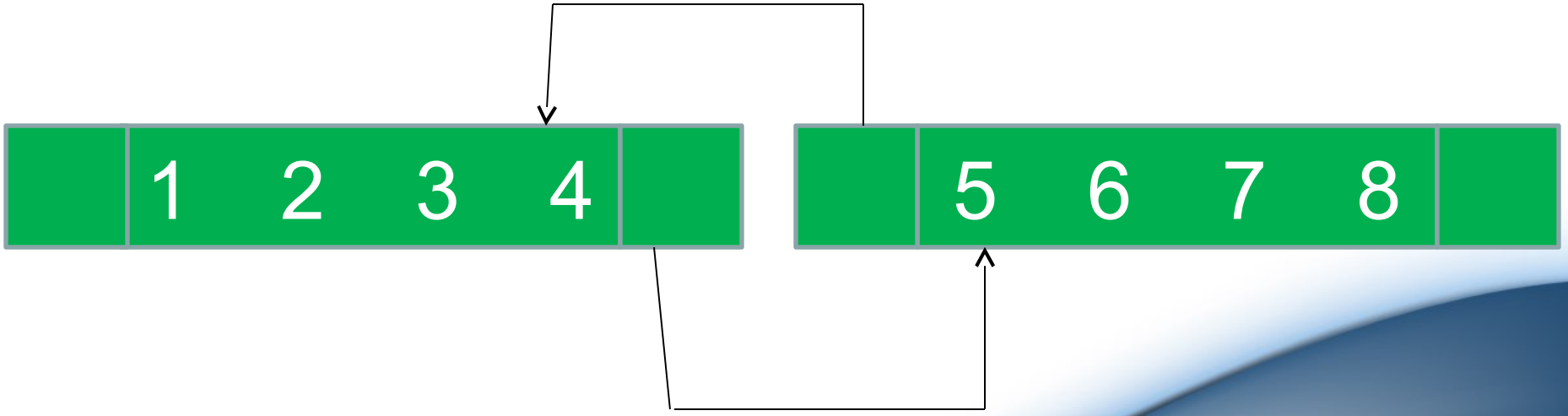
3rd is replaced with 48th…..



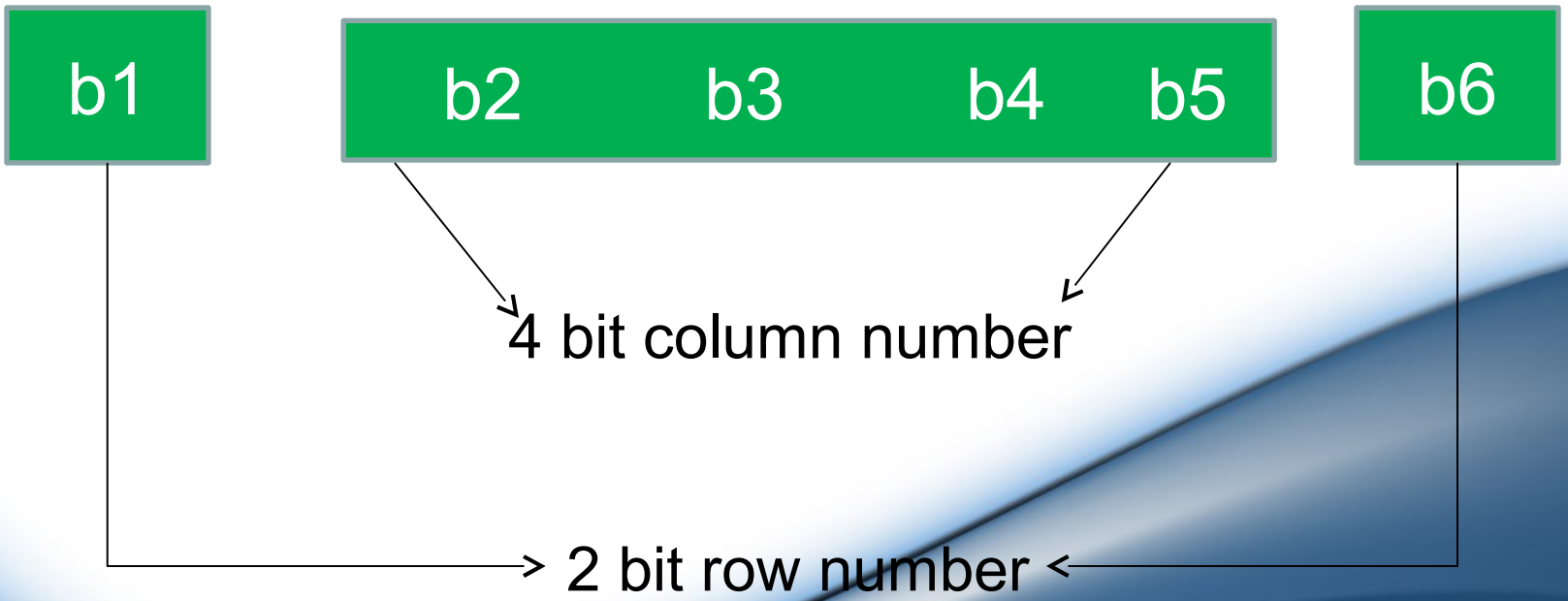
Details of one round in DES

- Out of 56 bits key, a subkey of 48 bits is generated.
- This is done by dividing 56 bits into 2 half i.e. 28 bits.
- These are circularly shifted by 1 or 2 position depending on round
- If round 1,2,9,16, the shift is done by 1 position.
- Thus it is also called as compression permutation.
- The RPT is expanded from 32 bits to 48 bits and they are also permuted.
- 32 bit block is divided into 8 blocks, each consisting of 4 bits.
- These 4 bit block is expanded into 6 bit block by repeating the bit.
- Now the 48 bit key and 48 bit RPT block are XORed.
- The 48 bit block from the above step is divided 8 block each of 6 bits.
- Each 6 bit block are send to S-box to get a 4 bits block.
- The S- box can be imagined as table of 4 rows and 16 columns.
- The intersection of row and column gives a 4-bit output.
- The 32 bit block are permuted using a P-box.
- Here there is a simple permutation of bits
- At this point, the LPT is XORed with the permuted RPT.
- Now they are swaped as LPT becomes RPT and RPT becomes LPT.
- Then they are combined to get 64 bits and send to next step.

Expansion Process:

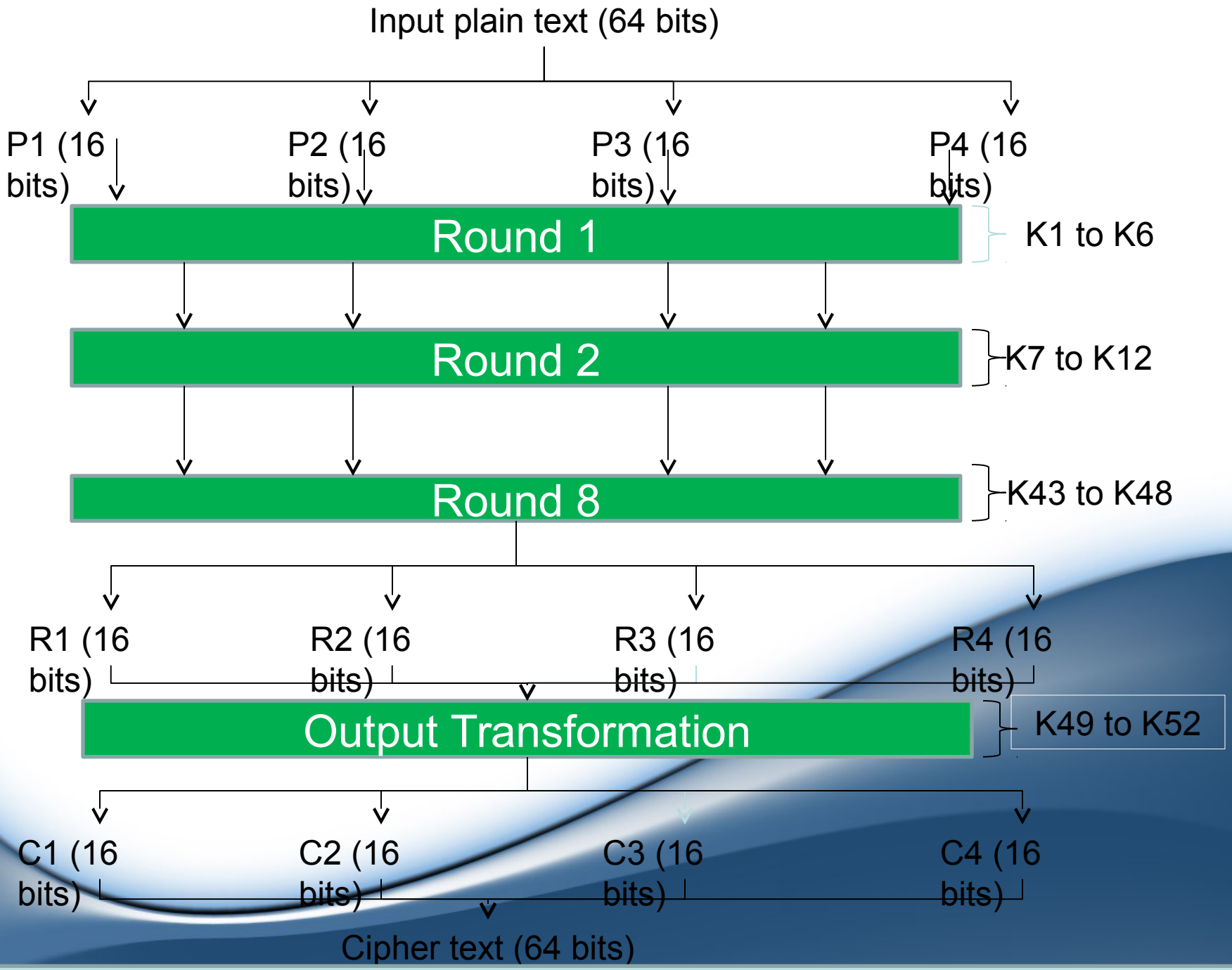


Selecting an entry in S-Box :



IDEA (International Data Encryption Algorithm)

- Strongest cryptographic algorithm.
- IDEA developed in 1990 but authorized in 1992.
- IDEA is patented. Need license to use.
- Email privacy PGP (Pretty Good Privacy) is based on IDEA.
- It works on 64 bits block. And further divided into 4 blocks of 16 bits.
- It has 8 rounds.
- The key length is 128 bits.
- Out of which 6 subkeys are generated for each round of 16 bits each


Input plain text (64 bits)
P1 (16 bits)
P2 (16 bits)
P3 (16 bits)
P4 (16 bits)
Round 1
K1 to K6
Round 2
K7 to K12
Round 8
K43 to K48
R1 (16 bits)
R2 (16 bits)
R3 (16 bits)
R4 (16 bits)
Output Transformation
K49 to K52
C1 (16 bits)
C2 (16 bits)
C3 (16 bits)
C4 (16 bits)
Cipher text (64 bits)

Step: 1 Multiply* P1 and K1

Step:2 Add* P2 and K2

Step :3 Add* P3 and K3

Step:4 Multiply* P4 and K4

Step:5 XOR the results of Step 1 & 3

Step:6 XOR the results of Step 2 & 4

Step:7 Multiply* the results of Step 5 & K5

Step:8 Add* the results of Step 6 & 7

Step:9 Multiply* the results of Step 8 & K6

Step:10 Add* the results of Step 7 & 9

Step:11 XOR the results of Step 1 & 9

Step:12 XOR the results of Step 3 & 9

Step:13 XOR the results of Step 2 & 10

Step:14 XOR the results of Step 4 & 10

Details of one round in IDEA

Binary Addition of two 16-bit numbers:

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

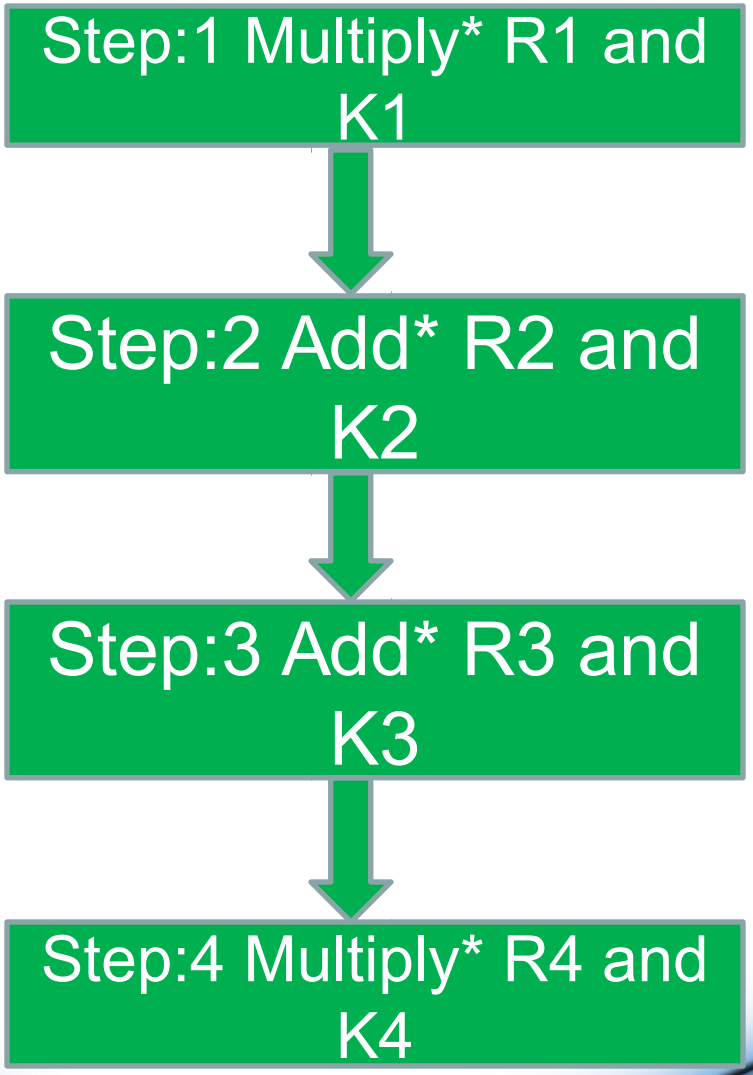
\+

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

---

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

mod 65536



Details of output transformation process

Sub-key generation of a round

Total key length is of 128 bits. Each key is of 16 bits

1 round has 6 subkeys.

So 8 rounds has 6x8= 48 subkeys

+

Output transformation = 4 subkeys

52 subkeys

How to generate 52 subkeys from 128 bits?

For 1st round, we need 6 subkeys, 6x16 = 96 bits

So now we have 128 - 96 = 32 bits left

For 2nd round, we can use 32 bits to generate 2 subkeys. But what about remaining 4 subkeys?

For the remaining 4 subkeys, 64 bits can be used from the fresh 128 bits.

The same process is used for generation of all 52 subkeys.

RC4 algorithm

- The official name of the algorithm is 'Rivest Cipher 4'.
- Developed by Ron Rivest in 1987.
- RC4 is a stream cipher.
- RC4 has become part of WEP and WPA encryption technologies for wireless cards

- In RC4, the key is of variable length consisting of 1 to 256 bytes.
- This key is used to initialize a *state vector* S of 256-byte.
- For encryption or decryption, one of the 256 bytes of S is selected and processed.
- After this S is permuted once again.
- There are two processes:
- **Initialization of S**
- **Stream generation**

Initialization of S

- Choose a key (K) of length between 1 to 256 bytes.
- Set the values in the state vector S equal to the values from 0 to 255 in an ascending order as: S[0]=0, S[1]=1, S[2]=2…..
- Create another temporary array T.
- If the key length is 256 bytes, then copy the key as it is.
- If the key length is < 256, then copy K to T and fill the remaining positions of T by repeating K.

Following steps are executed:

```
Initialize state vector S
For i=0 to 255
   S[i]=i;
Copy the contents of K array to T
T[i] = K[i mod keylen]
```

Permutation of S :

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

Stream Generation

- When S is ready with initialization & permutation, the initial key array K is discarded.
- Now for every step, we swap S[i] with another byte in S.

The logic is as follows:

```
i=0;
j=0;
While (true)
    i=(i+1) mod 256;
    j=(j+S[i]) mod 256;
    swap(S[i],S[j]);
    t=(S[i]+S[j]) mod 256;
    k=S[t];
```

After this, k is XORed with the next byte of plain text.

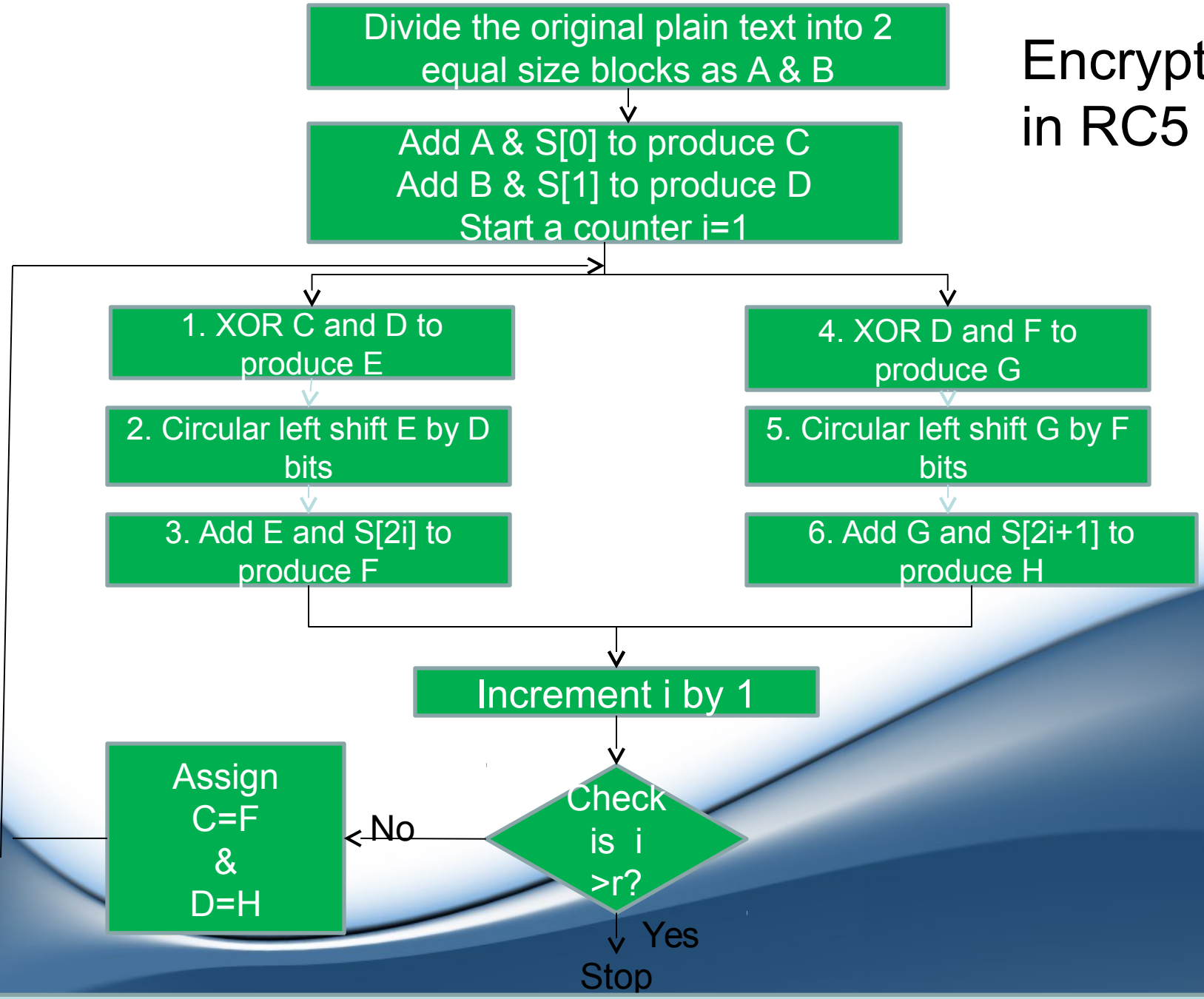
RC5

- RC5 is a symmetric key block –encryption algorithm developed by Ron Rivest.
- RC5 is quite fast as it uses only primitive computer operations(add, XOR, shift etc).
- In RC5, input plain text block size(word size), number of rounds and number of 8-bit bytes(octets) of key, all can be of variable length.

Parameter	Allowed Values
Word size in bits(encryption of 2-word blocks at a time)	16, 32, 64
Number of rounds	0-255
Number of 8-bit bytes(octets)	0-255

- A particular instance of the RC5 algorithm is denoted as RC5-*w/r/b*,
- Where w= word size in bits,
- r= number of rounds,
- b=number of octets
- Rivest has suggested RC5-*32/12/16* as the minimum safety version

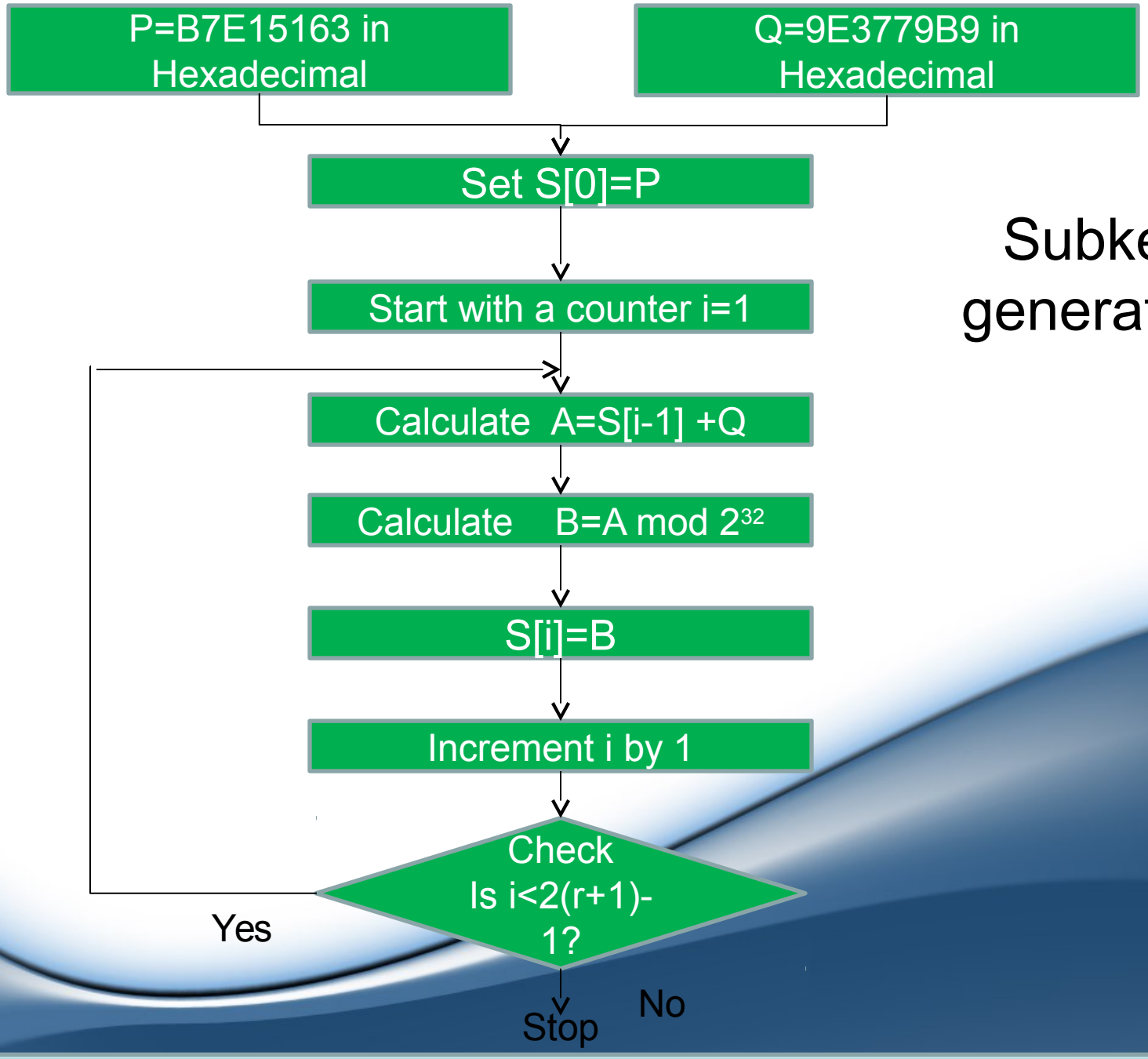
Encryption in RC5



Subkey creation

- This is a 2 step process:
- Step:1 The subkeys S[0], S[1]…… are generated
- Step:2 The original key is called L. In this step, the subkeys (S[0], S[1]….) is mixed with subportions of original key i.e. L[0], L[1]…..

Subkey generation



Subkey Mixing

i=j=0;
A=B=0;

Do 3n times where n is the maximum of 2(r+1) and c (last subkey position of L)

A=S[i]=(S[i]+A+B)<<<3
B=L[i]=(L[i]+A+B)<<<(A+B)

i=(i+1) mod 2(r+1)
j=(j+1) mod c

End do

BLOWFISH

- Developed by Bruce Schneier.
- Encryption rate on 32-bit microprocessors is 26 clock cycles per byte.
- Blowfish can execute in less than 5 KB memory.
- It uses primitive operations such as addition, XOR and table lookup, making its design and implementation simple.
- It has variable length key up to 448 bits long.
- Blowfish suits applications where the key remains constant.

- Blowfish encrypts 64-bit blocks with a variable- length key.
- It contains 2 parts:
- Subkey generation: this process converts the key up to 448 bits long to subkeys totaling 4168 bits.
- Data encryption: this process involves the iteration of a simple function 16 times. Each round contains a key-dependent permutation and data-dependent substitution

Subkey generation

- The key size ranges from 32 bits to 448 bits. In other words, the key size is 1 to 14 words (1 word = 32 bits or 4 bytes). These keys are stored in an array as K1, K2, K3…..Kn where $1 \leq n \leq 14$
- A P-array is created consisting of 18 32-bit subkeys followed by 4 S-boxes of 256 32-bit entries. These P-array and S-boxes are called as subkeys.
- As P1,P2….P18 and S1,0,S1,1,S1,2…..S1,255. Same for S2,0,S2,1,S2,2….S2,255.
- Schneier recommends the use of the bits of the hexadecimal fractional part of constant (π) pi for all the above values.
- Do a bitwise XOR of P1 with K1, P2 with K2……P18 with K4. Repeat the keys when exhausted.
- Now take a 64-bit block with all initialized to 0. Now use the Blowfish algorithm to generate subkeys themselves.
- When above block is encrypted, it produces a 64-bit cipher text.
- This 64-bit cipher text is divided into equal halves of 32-bit and are replaced with P1 and P2. this will again produce a 64-bit cipher text. Again these are divided into 32-bit block and replaced with P3 and P4. this process goes on to fill P-array & S-boxes.
- Once the final subkeys are ready, the Blowfish algorithm is used to encrypt the plain text.

Encryption & Decryption

- The encryption of a 64- bit block of input plain text X is shown in algorithmic fashion:
- Divide X into two blocks XL (32 bits) and XR (32 bits).
- For i=1 to 16
- XL= XL XOR Pi
- XR= F(XL) XOR XR
- Swap (XL,XR)
- Next i
- Swap XL,XR (undo the last step)
- XL= XL XOR P18
- Combine XL and XR back into X.

- Function F is as follows:
- Divide the 32 bit XL block into four 8 bit block named as a,b,c,d
- Compute F(a,b,c,d)=((S1,a+S2,b) XOR S3,c)+S4,d.

AES (Advanced Encryption Standard)

- AES competition
  - Started in January 1997 by NIST
  - 4-year cooperation between
    - U.S. Government
    - Private Industry
    - Academia
- Why?
  - Replace 3DES
  - Provide an unclassified, publicly disclosed encryption algorithm, available royalty-free, worldwide

The Finalists

- **MARS**
  - IBM
- **RC6**
  - RSA Laboratories
- **Rijndael**
  - Joan Daemen (Proton World International) and
  - Vincent Rijmen (Katholieke Universiteit Leuven)
- **Serpent**
  - Ross Anderson (University of Cambridge),
  - Eli Biham (Technion), and
  - Lars Knudsen (University of California San Diego)
- **Twofish**
  - Bruce Schneier, John Kelsey, and Niels Ferguson (Counterpane, Inc.),
  - Doug Whiting (Hi/fn, Inc.),
  - David Wagner (University of California Berkeley), and
  - Chris Hall (Princeton University)

Got selected as winner

- AES uses the mathematical concept of Galois Field
- Byte notation for the element
- Has its own arithmetic operations.
- Rijndael supports key length and plain block size from 128 bits to 256 bits.
- In general two versions of AES are used:
- 128 bit of plain text block combined with 128 bits key block.
- 128 bit of plain text block combined with 256 bits key block.
- The minimum no. of rounds is 10 and maximum is 14.

- **One-time initialization process**

a.) Expand the 16- byte(128 bits) key to get the actual *key block* to be used

b.) Do one-time initialization of the 16-byte plain text block (called State)

c.) XOR the State with key block

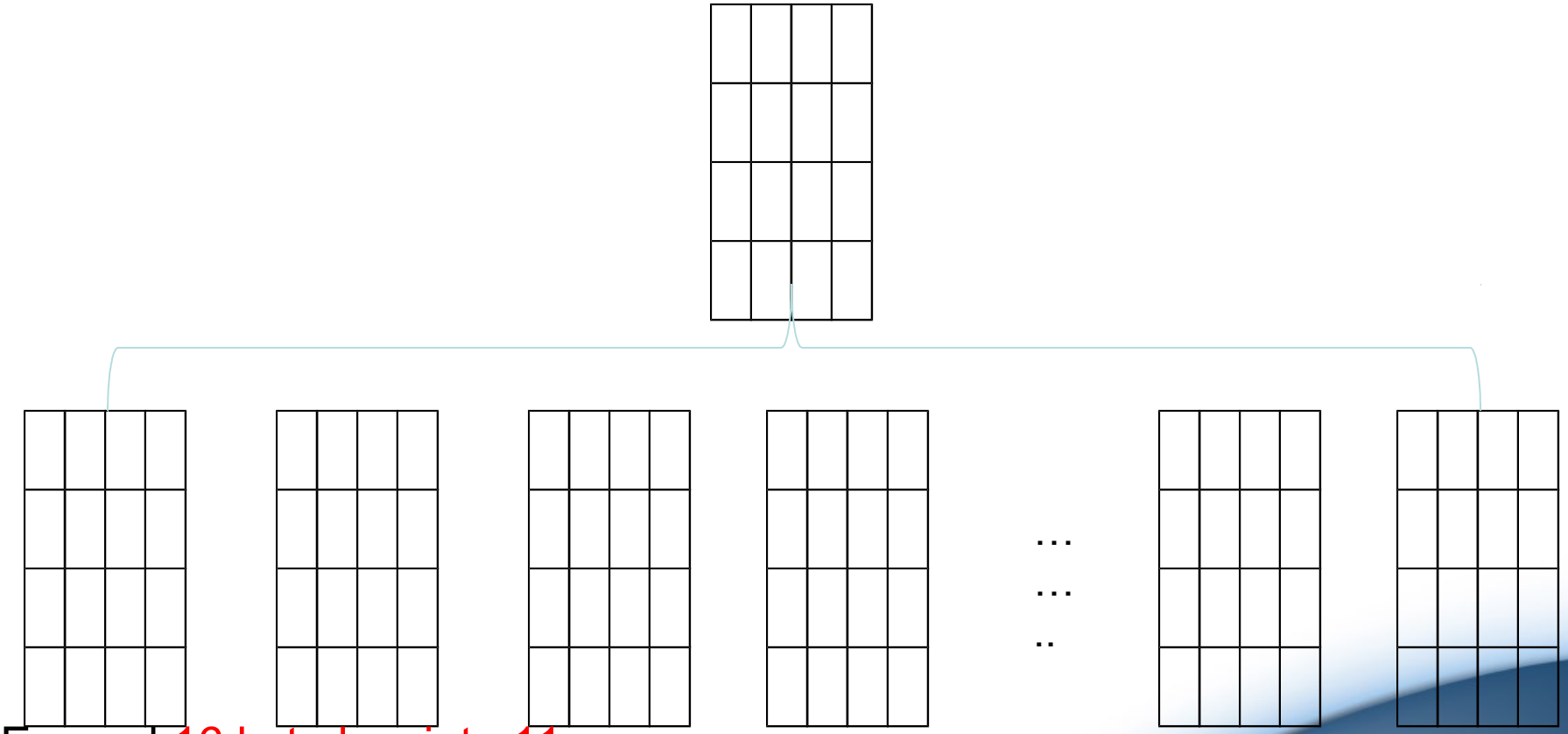
- **For each round, do the following**

a.) Apply S-box to each of the plain –text bytes

b.) Rotate row k of the plain text block(i.e State) by k bytes

c.) Perform a mix columns operation

d.) XOR the state with the key block



- Expand 16 byte key into 11 arrays.
- 1st array is used in initialization process and other 10 arrays are used in the rounds, one array per round
- Each array contains 4 rows and 4 columns.
- A word =4 bytes. Therefore 16 byte key (4 word) will be expanded into 176 byte key(44 words)
- Firstly, the 16 byte key is copied as it is in 1st array, and the remaining array are filled depending on the preceding blocks.

To fill the arrays, following algorithm is used:

```
Expandkey (byte K[16], word W[44])
  {
      word temp;
 // copy all the contents for 1st word as it is

      for(i=0 ; i<4 ; i++)
        W[i]=K[4*i],   K[4*i+1],   K[4*i+2],   K[4*i+3];

// now populate the remaining array

    for(i=4 ; i<44 ; i++)
     {   temp=W[i-1];
      if (i mod 4 ==0)
         temp=Substitute(Rotate(temp)) XOR Constant(i/4);

      W[i]=W[i-4] XOR temp;
    }
}
```

Circular left shift on the contents by one byte

Performs a byte substitution on each byte by using S-box

The output of Substitute is XORed with constant value which in turn is based on round number

S-box

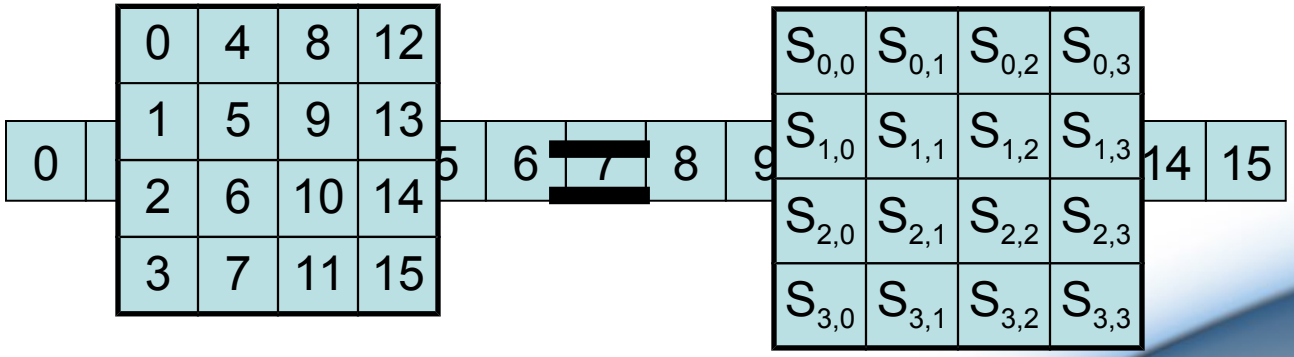
x \ y	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
0	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
1	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
2	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
3	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
4	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
5	53	d1	00	ed	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

Constant Values:

Round Number	1	2	3	4	5	6	7	8	9	10
Value of constant to be used in Hex	01	02	04	08	10	20	40	80	1B	36

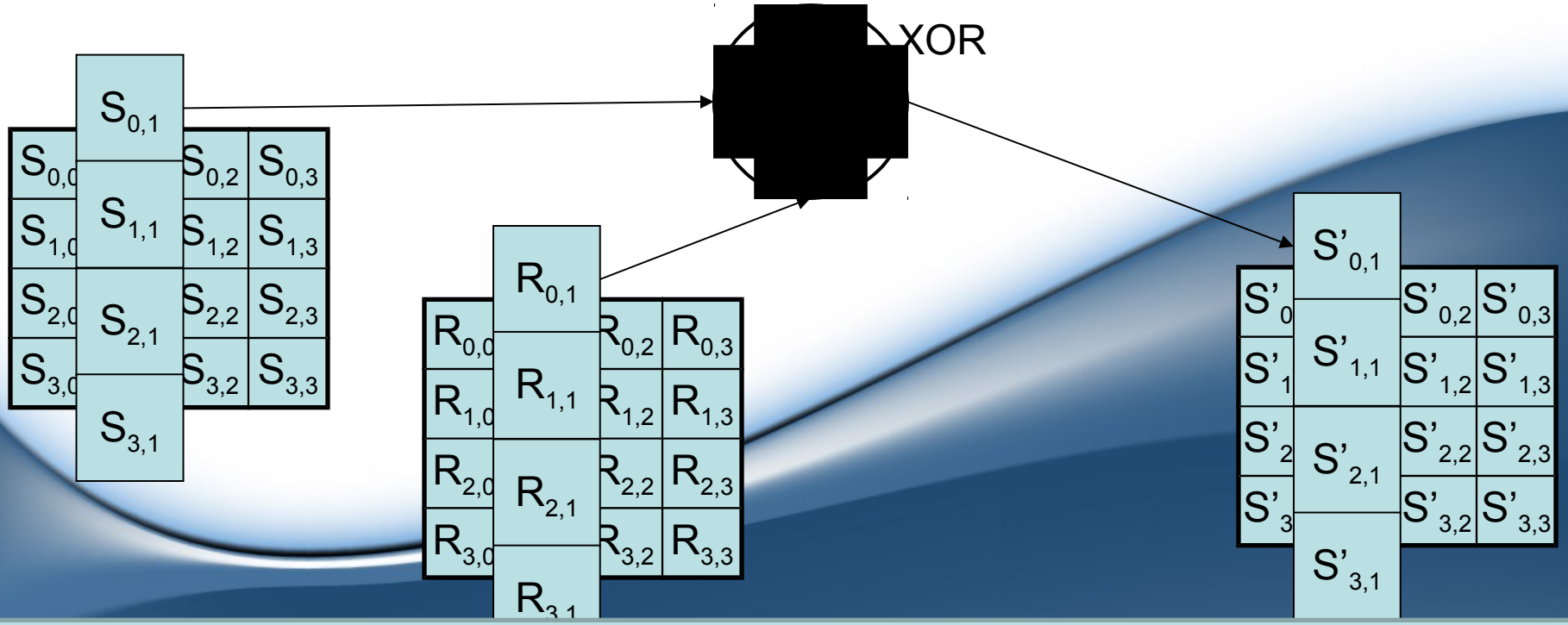
One-time initialization of the 16-byte plain text block (called State)

Input block:



XOR the State with key block

- XOR each byte of the key block with its corresponding byte in the state array



Apply S-box to each of the plain –text bytes

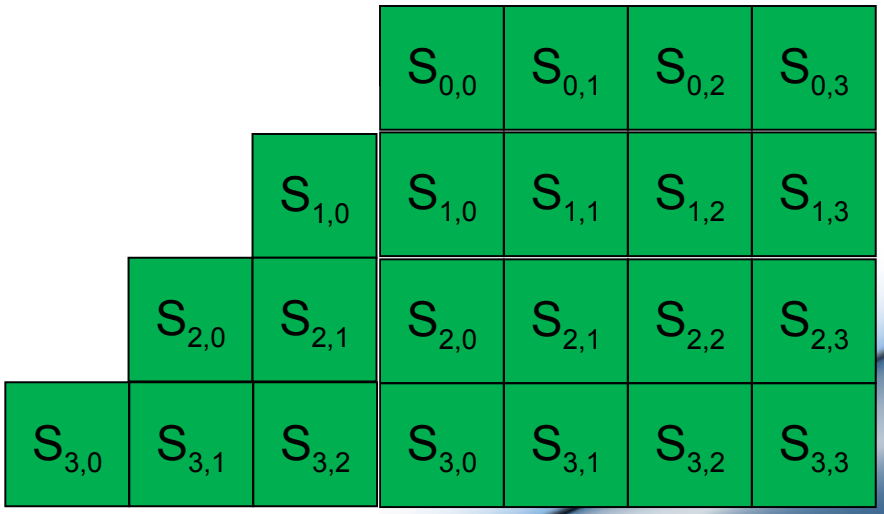
00	44	88	CC
11	55	99	DD
22	66	AA	EE
33	77	BB	FF

- Replace each byte in the state array with its corresponding value from the S-Box

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

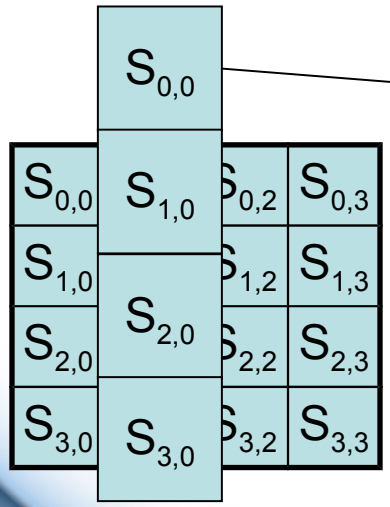
Rotate row k of the plain text block by k bytes

- Rotate row 0 with 0 bytes.
- Rotate row 1 with 1 bytes.
- Rotate row 2 with 2 bytes.
- Rotate row 3 with 3 bytes.



Perform a mix columns operation

- Apply MixColumn transformation to each column



b1 = (b1 * 2) XOR (b2 * 3) XOR (b3 * 1) XOR (b4 * 1)

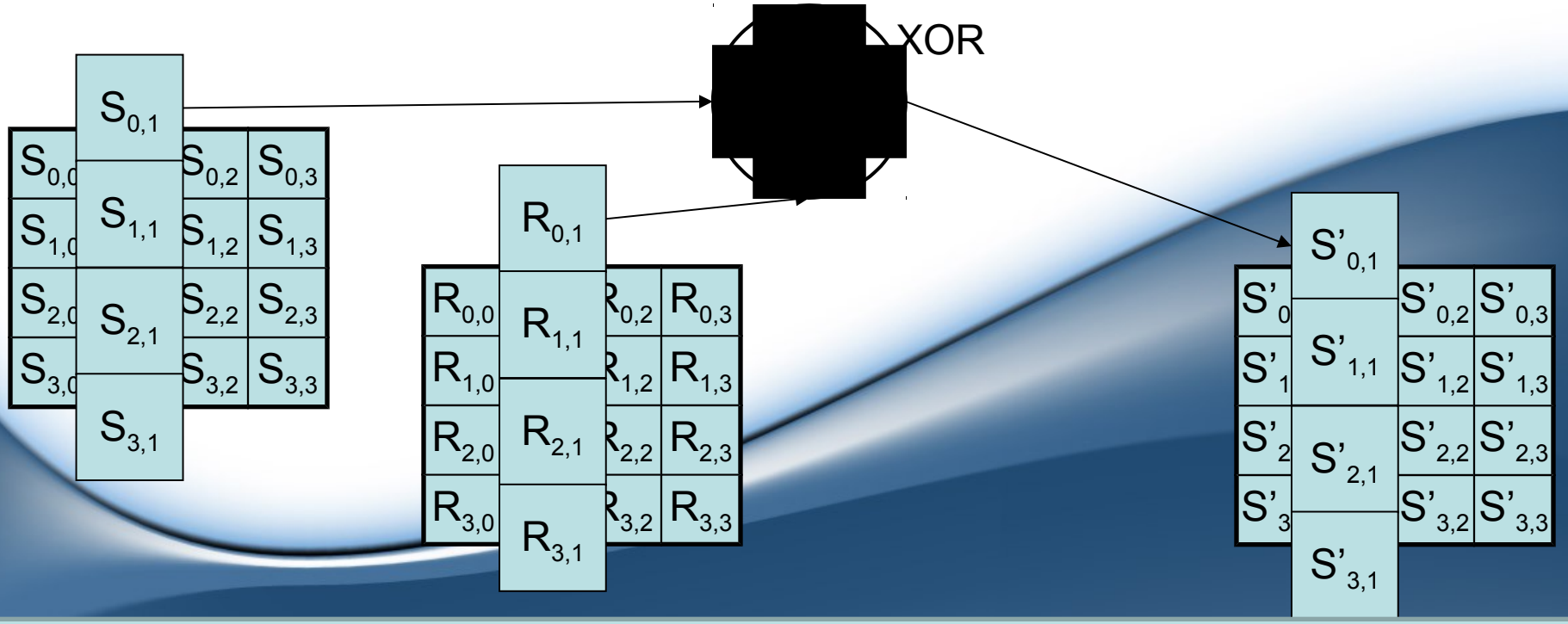
b2 = (b1 * 1) XOR (b2 * 2) XOR (b3 * 3) XOR (b4 * 1)

b3 = (b1 * 1) XOR (b2 * 1) XOR (b3 * 2) XOR (b4 * 3)

b4 = (b1 * 3) XOR (b2 * 1) XOR (b3 * 1) XOR (b4 * 2)

XOR the State with key block

- XOR each byte of the key block with its corresponding byte in the state array



Diffie-Hellman Key Exchange Algorithm

P	Q
• Prime number ***'n'***	• Prime number ***'g'***
Exchange n and g	
• Random number '**x**'	• Random number '**y**'
• Calculate A as :- $A=g^x \bmod n$	• Calculate B as:- $B=g^y \bmod n$
Exchange A and B	
• Computes the key K1 as:-	• Computes the key K2 as:-
• $K1=B^x \bmod n$	• $K2=A^y \bmod n$

K1=K2

For Example

P

n=11

x=3

Exchange n and g

Q

g=7

y=6

- Calculate A as :-

  $A=7^3 \bmod 11$

  $=343 \bmod 11$

  $= 2$

- Calculate B as:-

  $B=7^6 \bmod 11$

  $=117649 \bmod 11$

  $=4$

Exchange A and B

- Computes the key K1 as:-
- $K1=4^3 \bmod 11$

  $= 64 \bmod 11$

  $=9$

- Computes the key K2 as:-
- $K2=2^6 \bmod 11$

  $=64 \bmod 11$

  $=9$

K1=K2

Man in the Middle Attack(MITM)

- n=11,g=7 x=3

- n=11,g=7 x=8,y=6

- n=11,g=7 y=9

- $A=7^3 \bmod 11=2$
- $A=7^8 \bmod 11=\mathbf{9}$, $B=7^6 \bmod 11=\mathbf{4}$
- $7^9 \bmod 11=8$

Alice
A=2, B=4

Tom
A=2, B=8

Bob
A=9, B=8

A=2

A=9

Intercept

B=8

B=4

Intercept

Continued..

- A=2 , B=**4**

  K1=4^3 mod 11=**9**

- A=2, B=8

  K1=8^8 mod 11=5

  K2=2^6 mod 11=9

- A=**9** , B=8

  K2=9^9 mod 11=**5**

- **MITM attack is also known as:**
  - Bucket-brigade attack
  - Fire brigade attack
  - Monkey-in-the-middle attack
  - Session hijacking
  - TCP hijacking
  - TCP session hijacking

Thank You