CRYPTOGRAPHY 1

Fifth Lecture – Modern Cryptography Asymmetric

Assistant Professor Dr.

Sufyan Salim Mahmood

2024 - 2025

Asymmetric cryptography

Asymmetric-key cryptography is also called Public key cryptography. It is the cryptographic algorithm which uses pairs keys.

The keys are known as public and private keys. Public key is used to encrypt the data and private key is used to decrypt the data.

Asymmetric cryptography

Both the key pairs are generated using cryptographic algorithms.

The security of public key cryptography depends on keeping the private key secret and the public key can be shared and distributed publicly.

Asymmetric cryptography

When someone wants to send an encrypted message, they pull the intended recipient's public key from a public directory and use it to encrypt the message before sending it. The recipient of the message can decrypt the message using their related private key.

Asymmetric cryptography

Examples of these algorithms include

- RSA
- Diffie-Hellman
- Elliptic Curve Cryptography (ECC)

Asymmetric cryptography

Many protocols rely on asymmetric cryptography including

- Transport Layer Security (TLS)
- Secure Sockets Layer (SSL)

Asymmetric cryptography

Number of required keys is 2N

Asymmetric cryptography

- Asymmetric encryption offers several benefits over other encryption techniques, including:
- Security: Asymmetric encryption provides strong security for data, as the private key used for decryption is kept secret and not shared with anyone.
- This makes it difficult for unauthorized users to access the data encrypted with asymmetric keys.

Asymmetric cryptography

- Authentication: Asymmetric encryption can also be used for authentication, as the public key can be used to verify the identity of the sender of the message, and only the corresponding private key can decrypt it.
- This helps to prevent fraud and protect against malicious attacks.

Asymmetric cryptography

- Key distribution: Asymmetric encryption eliminates the need for a secure channel to distribute keys, as each user has a unique public-private key pair, and only the private key can decrypt messages encrypted with the public key.
- This makes it easier to distribute keys and manage access to encrypted data.

Asymmetric cryptography

- Non-repudiation: Asymmetric encryption provides non-repudiation, meaning that the sender of a message cannot deny having sent it, as the message can be traced back to their unique private key, and only the intended recipient's public key can decrypt it.

Asymmetric cryptography

- However, the downside of Asymmetric encryption is that it is slower and more complex to implement than symmetric encryption.

CRYPTOGRAPHY 1

Sixth Lecture –

Stream Cipher

Assistant Professor Dr.

Sufyan Salim Mahmood

2024 - 2025

Introduction

- A stream cipher is an encryption technique that works bit by bit to transform plain text into code that's unreadable to anyone without the proper key.

How do stream ciphers work?

- All cryptographic methods aim to scramble data to hide it from outsiders. Stream ciphers work on each bit of data in a message rather than chunking the message into groups and encrypting them in blocks.

How do stream ciphers work?

- Stream ciphers rely on:
- **Plaintext.** You must have a message you'd like to encode.
- **Keystreams.** A set of random characters replaces those in the plaintext. They could be numbers, letters, or symbols.
- **Ciphertext.** This is the encoded message.

How do stream ciphers work?

- Generating a key is a complicated mathematical process.
- Bits of plaintext enter the stream cipher, and the cipher manipulates each bit with the mathematical formula.
- The resulting text is completely scrambled, and the recipient can't read it without the proper key.

How do stream ciphers work?

- For example, the first bit in Person A's 10-bit message will be XOR-ed with the first bit of the keystream. If the two digits are the same, the XOR operator will produce a zero. If the two are different -- i.e., a combination of 1 and 0 -- the XOR operator will produce a 1. This is part of what makes stream cipher encryption so fast.

How do stream ciphers work?

- Decryption of the ciphertext can happen in a manner similar to how the plaintext encryption occurs. This time, instead of the data and keystream being XOR-ed, the ciphertext and the keystream are XOR-ed.

Advantages of a stream cipher

Speed of encryption tops the list of advantages for stream ciphers. Once a stream cipher makes a key, the encryption and decryption process is almost instantaneous.

Advantages of a stream cipher

The hardware complexity of a stream cipher is quite low -- meaning a wider range of technologies can facilitate this mode of operation.

Advantages of a stream cipher

- ✓ This type of encryption technology is faster than any other technique.
- ✓ It is very easy to use, and complicated hardware is not required.
- ✓ The data can be sent bit by bit instead of waiting for everything to be done.

Advantages of a stream cipher

- ✔ Stream Cipher makes cryptanalysis very difficult. Cryptanalysis means decoding the encrypted text without access to the secret key. It is very secure. It is very difficult to decrypt a Stream Cipher text message without the secret key.
- ✔ If you use a longer keystream, Stream Cipher can safeguard the message from Brute Force Attacks. Brute Force Attacks are attacks in which the attacker tries to guess the secret key by using hit and trial.

Disadvantages of a stream cipher

- Stream ciphers are vulnerable to attack if the same key is used twice (depth of two) or more.

Disadvantages of a stream cipher

- Say we send messages *A* and *B* of the same length, both encrypted using same key, *K*. The stream cipher produces a string of bits *C(K)* the same length as the messages.

Disadvantages of a stream cipher

- *E(A) = A xor C*
- *E(B) = B xor C*
- where *xor* is performed bit by bit.

Disadvantages of a stream cipher

- An adversary has intercepted *E(A)* and *E(B)*.
- They can easily compute:
- *E(A) xor E(B)*

Stream Cipher Examples

Stream ciphers are widely used in various cryptographic applications. Let's see some common stream ciphers:

Stream Cipher Examples

RC4 (Rivest Cipher 4): RC4 is one of the most widely used stream ciphers.

- It was designed by Ron Rivest in 1987 and has been used in protocols such as SSL/TLS and WEP.
- RC4 generates a pseudorandom keystream by using a permutation of all 256 possible bytes.
- However, RC4 has been found to have several vulnerabilities and is now considered insecure for most purposes.

Stream Cipher Examples

Salsa20: Salsa20 is another stream cipher designed by Daniel J. Bernstein.

It uses a 256-bit key and a 64-bit nonce to generate the keystream.

Salsa20 has a simple and efficient design, making it suitable for both software and hardware implementations.

It has been used in various applications, including the Sodium crypto library.

Stream Cipher Examples

ChaCha20: ChaCha20 is a modern stream cipher designed by Daniel J. Bernstein.

It is based on the Salsa20 cipher and uses a 256-bit key and a 96-bit nonce (number used once) to generate the keystream.

ChaCha20 is known for its simplicity, speed, and security.

It has been adopted by various protocols and applications, including TLS 1.3 and the WireGuard VPN protocol.

Stream Cipher Examples

Grain: Grain is a family of stream ciphers designed for resource-constrained environments, such as hardware implementations and low-power devices. Grain ciphers use a combination of linear feedback shift registers (LFSRs) and nonlinear functions to generate the keystream. Different versions of Grain, such as Grain-128a and Grain-128AEAD, offer different key sizes and authentication capabilities.

Stream Cipher Examples

HC-128 and HC-256: HC-128 and HC-256 are stream ciphers designed by Hongjun Wu. They use a 128-bit and 256-bit key, respectively, along with a 128-bit initialization vector. HC-128 and HC-256 have a large internal state and use a combination of permutations and feedback functions to generate the keystream. They are designed to be fast and secure, offering good resistance against various cryptanalytic attacks.

CRYPTOGRAPHY 1

Seventh Lecture –

Block Cipher

Assistant Professor Dr.

Sufyan Salim Mahmood

2024 - 2025

Introduction

- **Block cipher** is an encryption algorithm that takes a fixed size of input say *b* bits and produces a ciphertext of *b* bits again.

Introduction

- The size of block is fixed in the given scheme.
- The choice of block size does not directly affect to the strength of encryption scheme.
- The strength of cipher depends up on the key length.

Block Size

- Though any size of block is acceptable, following the coming aspects while selecting a size of a block.

Block Size

- **Avoid very small block size** – Say a block size is m bits. Then the possible plaintext bits combinations are then 2^m.

Block Size

- If the attacker discovers the plain text blocks corresponding to some previously sent ciphertext blocks, then the attacker can launch a type of 'dictionary attack' by building up a dictionary of plaintext/ciphertext pairs sent using that encryption key.

Block Size

- A larger block size makes attack harder as the dictionary needs to be larger.

Block Size

- **Multiples of 8 bit** – A preferred block size is a multiple of 8 as it is easy for implementation as most computer processor handle data in multiple of 8 bits.

Mode of Operations

- If the input is larger than *b* bits it can be divided further.
- For different applications and uses, there are several modes of operations for a block cipher.

Why Mode of Operations

- Block ciphers are deterministic, i.e. given a key and plaintext block as input they will always generate the same output.
- So we need to have a mode of operation to convert fewer or more bytes than the block into ciphertext.

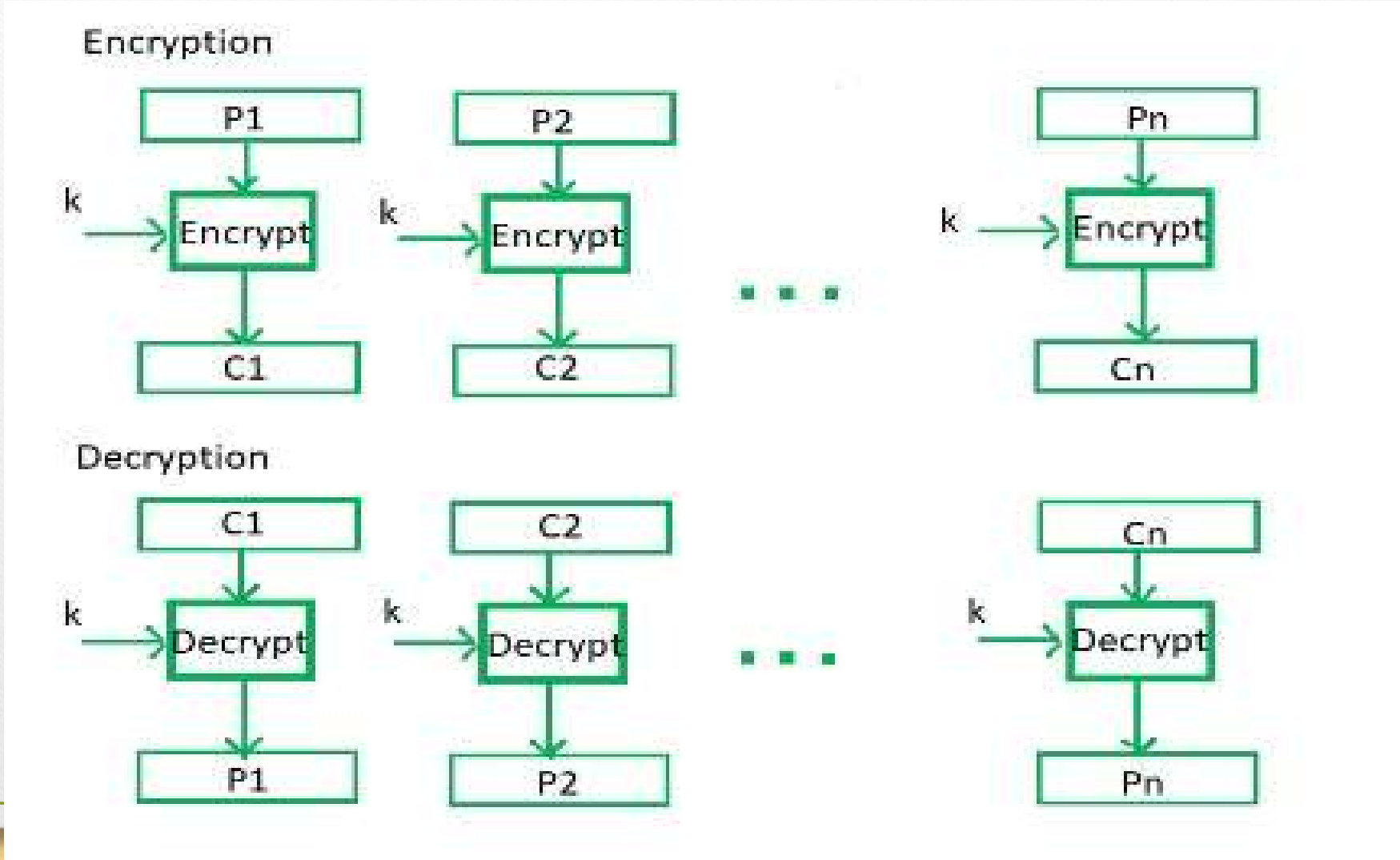
Mode of Operations

- Electronic Code Book
- Cipher Block Chaining
- Cipher Feedback Mode
- Output Feedback Mode
- Counter Mode

Electronic Code Book (ECB)

- Electronic code book is the easiest block cipher mode of functioning.
- It is easier because of direct encryption of each block of input plaintext and output is in form of blocks of encrypted ciphertext.
- Generally, if a message is larger than b bits in size, it can be broken down into a bunch of blocks and the procedure is repeated

Electronic Code Book (ECB)



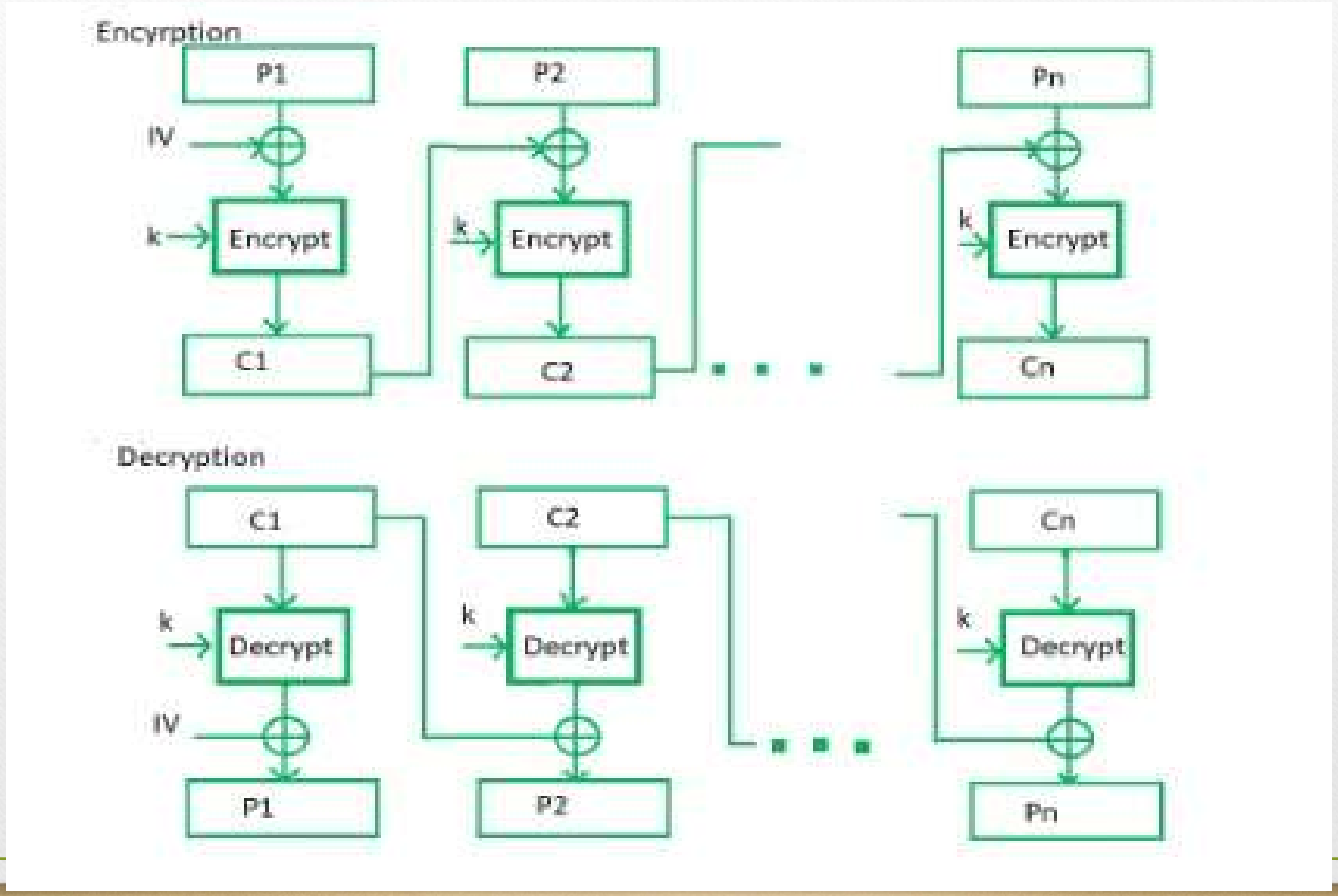
Electronic Code Book (ECB)

- Advantages of using ECB
  - Parallel encryption of blocks of bits is possible, thus it is a faster way of encryption.
  - Simple way of the block cipher.
- Disadvantages of using ECB –
  - Prone to cryptanalysis since there is a direct relationship between plaintext and ciphertext.

Cipher block chaining (CBC)

- Cipher block chaining or CBC is an advancement made on ECB since ECB compromises some security requirements.
- In CBC, the previous cipher block is given as input to the next encryption algorithm after XOR with the original plaintext block.
- In a nutshell here, a cipher block is produced by encrypting an XOR output of the previous cipher block and present plaintext block.

Cipher block chaining (CBC)



Cipher block chaining (CBC)

- **Advantages of CBC**

  CBC works well for input greater than b bits.

  CBC is a good authentication mechanism.

  Better resistive nature towards cryptanalysis than ECB.

- **Disadvantages of CBC**

  Parallel encryption is not possible since every encryption requires a previous cipher.

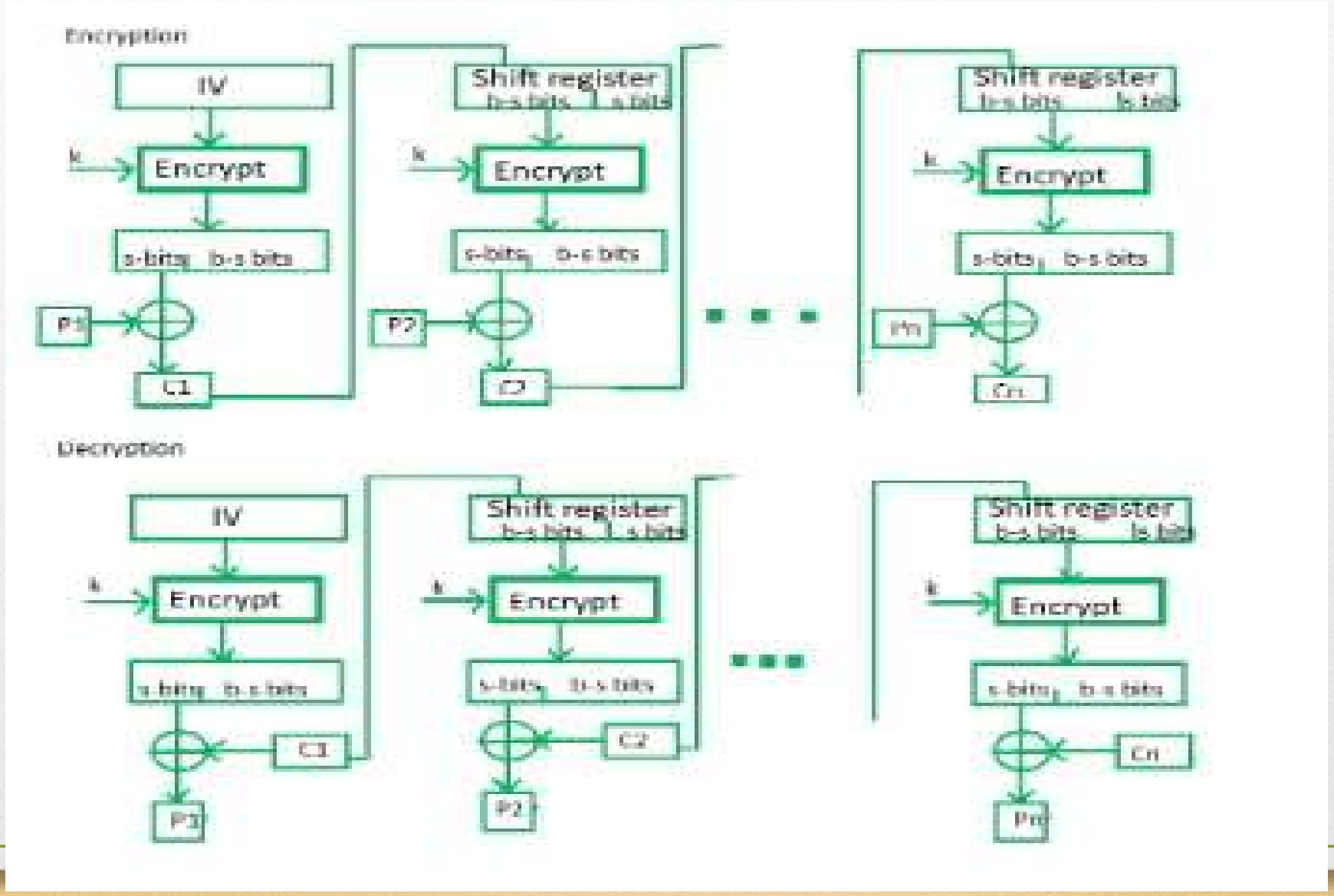
Cipher Feedback Mode (CFB)

- In this mode the cipher is given as feedback to the next block of encryption with some new specifications: first, an initial vector IV is used for first encryption and output bits are divided as a set of *s* and *b*-*s* bits.
- The left-hand side *s* bits are selected along with plaintext bits which an XOR operation is applied.

Cipher Feedback Mode (CFB)

- The result is given as input to a shift register having b-s bits to lhs,s bits to rhs and the process continues.
- The encryption and decryption process for the same is shown below, both of them use encryption algorithms.

Cipher Feedback Mode (CFB)



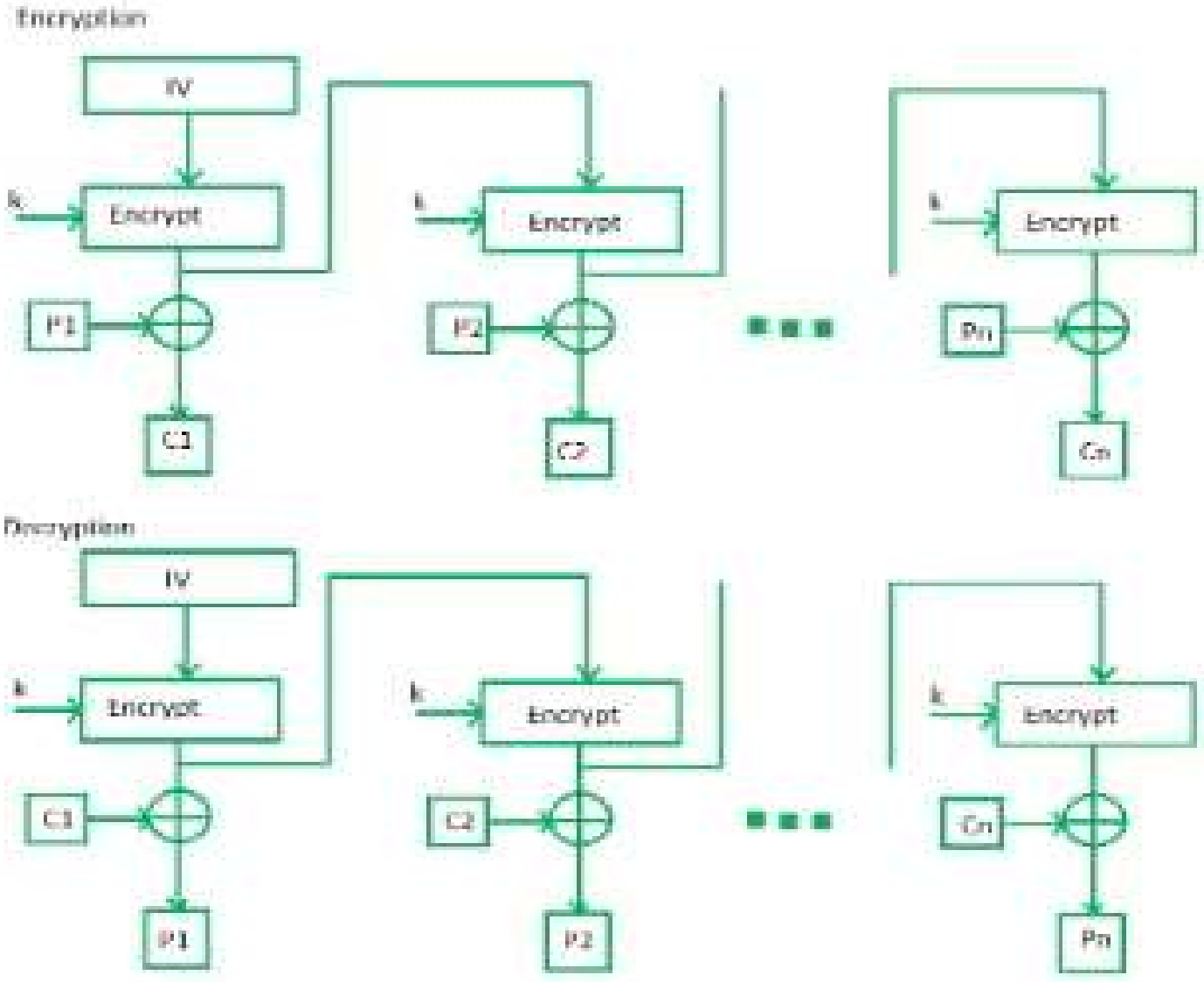
Cipher Feedback Mode (CFB)

- **Advantages of CFB**
  - Since, there is some data loss due to the use of shift register, thus it is difficult for applying cryptanalysis.
- **Disadvantages of using CFB**
  - The drawbacks of CFB are the same as those of CBC mode.

Output Feedback Mode (OFM)

- The output feedback mode follows nearly the same process as the Cipher Feedback mode except that it sends the encrypted output as feedback instead of the actual cipher which is XOR output.
- In this output feedback mode, all bits of the block are sent instead of sending selected s bits. The Output Feedback mode of block cipher holds great resistance towards bit transmission errors. It also decreases the dependency or relationship of the cipher on the plaintext.

Output Feedback Mode (OFB)



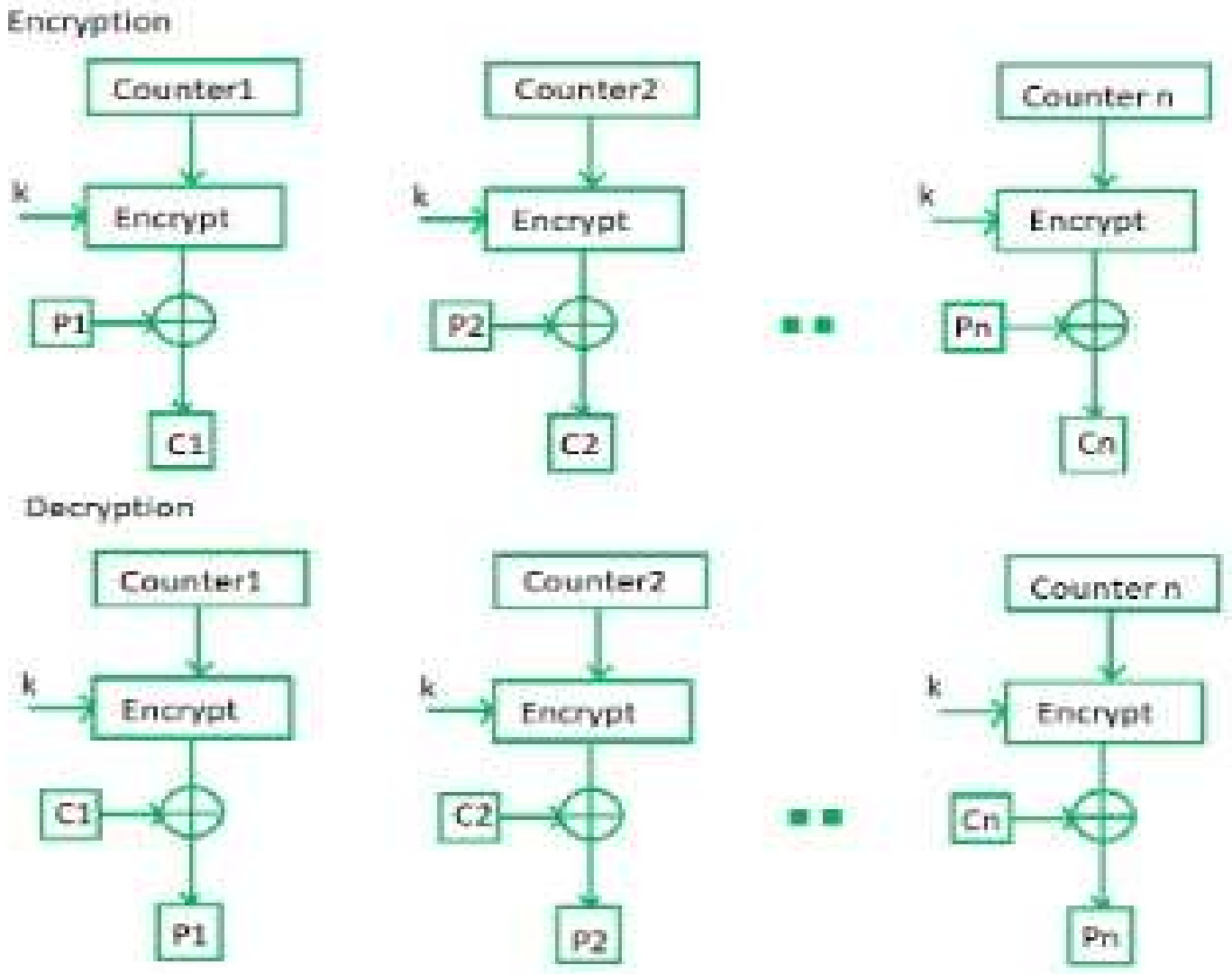
Output Feedback Mode (OFB)

- **Advantages of OFB –**
  - In the case of CFB, a single bit error in a block is propagated to all subsequent blocks. This problem is solved by OFB as it is free from bit errors in the plaintext block.
- **Disadvantages of OFB-**
  - It is more susceptible to a message stream modification attack than CFB.

Counter Mode (CTR)

- The Counter Mode or CTR is a simple counter-based block cipher implementation. Every time a counter-initiated value is encrypted and given as input to XOR with plaintext which results in ciphertext block. The CTR mode is independent of feedback use and thus can be implemented in parallel.

Counter Mode (CTR)



Counter Mode (CTR)

- **Advantages of Counter –**
- Since there is a different counter value for each block, the direct plaintext and ciphertext relationship is avoided. This means that the same plain text can map to different ciphertext.
- Parallel execution of encryption is possible as outputs from previous stages are not chained as in the case of CBC.

Counter Mode (CTR)

- **Disadvantages of Counter-**
- The fact that CTR mode requires a synchronous counter at both the transmitter and the receiver is a severe drawback.
- The recovery of plaintext is erroneous when synchronization is lost.

Applications of Block Cipher

1. Data Encryption: Block Ciphers are widely used for the encryption of private and sensitive data such as passwords, credit card details and other information that is transmitted or stored for a communication. This encryption process converts a plain data into non-readable and complex form. Encrypted data can be decrypted only by the authorised person with the private keys.

Applications of Block Cipher

2. File and Disk Encryption: Block Ciphers are used for encryption of entire files and disks in order to protect their contents and restrict from unauthorised users. The disk encryption softwares such as BitLocker, TrueCrypt aslo uses block cipher to encrypt data and make it secure.

Applications of Block Cipher

3. Virtual Private Networks (VPN): Virtual Private Networks (VPN) use block cipher for the encryption of data that is being transmitted between the two communicating devices over the internet. This process makes sure that data is not accessed by unauthorised person when it is being transmitted to another user.

Applications of Block Cipher

4. Secure Sockets Layer (SSL) and Transport Layer Security (TLS): SSL and TLS protocols use block ciphers for encryption of data that is transmitted between web browsers and servers over the internet. This encryption process provides security to confidential data such as login credentials, card information etc.

Applications of Block Cipher

5. Digital Signatures: Block ciphers are used in the digital signature algorithms, to provide authenticity and integrity to the digital documents. This encryption process generates the unique signature for each document that is used for verifying the authenticity and detecting if any malicious activity is detected.

CRYPTOGRAPHY 1

Eighth Lecture –

AES

Assistant Professor Dr.

Sufyan Salim Mahmood

2024 - 2025

Advanced Encryption Standard (AES)

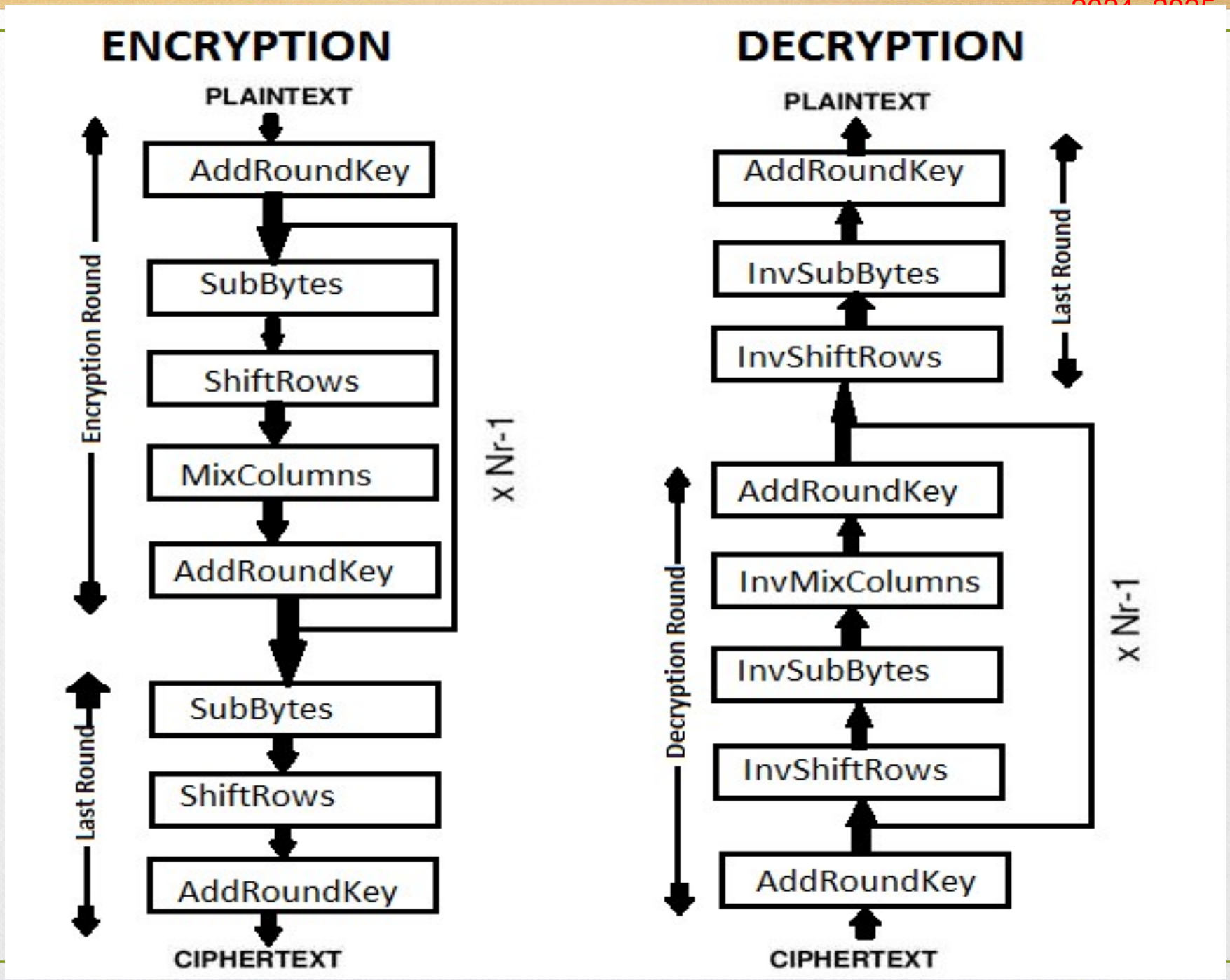
- Advanced Encryption Standard (AES) algorithm is one on the most common and widely symmetric block cipher algorithm used in worldwide. This algorithm has an own particular structure to encrypt and decrypt sensitive data and is applied in hardware and software all over the world. It is extremely difficult to hackers to get the real data when encrypting by AES algorithm.

Advanced Encryption Standard (AES)

- Till date is not any evidence to crake this algorithm. AES has the ability to deal with three different key sizes such as AES 128, 192 and 256 bit and each of this ciphers has 128 bit block size.

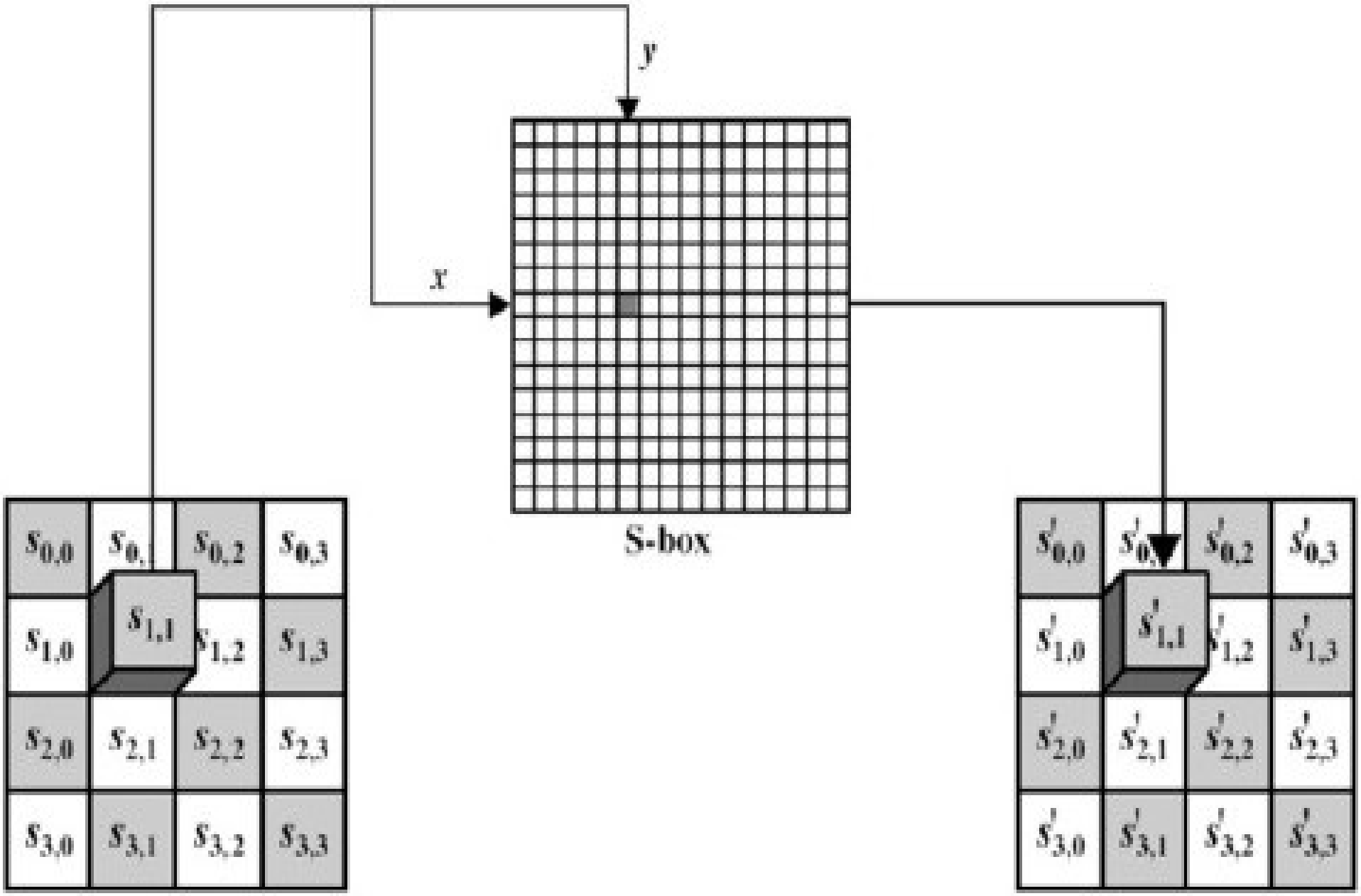
BASIC STRUCTURE OF AES Algorithm

- It is based on two common techniques to encrypt and decrypt data knowns as substitution and permutation network (SPN). SPN is a number of mathematical operations that are carried out in block cipher algorithms. AES has the ability to deal with 128 bits (16 bytes) as a fixed plaintext block size. These 16 bytes are represented in 4x4 matrix and AES operates on a matrix of bytes.


ENCRYPTION
PLAINTEXT
AddRoundKey
SubBytes
ShiftRows
MixColumns
AddRoundKey
SubBytes
ShiftRows
AddRoundKey
CIPHERTEXT
Encryption Round
Last Round
x Nr-1
DECRYPTION
PLAINTEXT
AddRoundKey
InvSubBytes
InvShiftRows
AddRoundKey
InvMixColumns
InvSubBytes
InvShiftRows
AddRoundKey
CIPHERTEXT
Last Round
Decryption Round
x Nr-1

Encryption Processes

- Each round consists of the following four steps to encrypt 128 bit block
- 1) Substitute bytes
- 2) Shift rows
- 3) Mix columns
- 4) Add round key.
- The last step consists of the previous three steps only except Mixcolumns.


y
x
S-box
$s_{0,0}$ $s_{0,1}$ $s_{0,2}$ $s_{0,3}$
$s_{1,0}$ $s_{1,1}$ $s_{1,2}$ $s_{1,3}$
$s_{2,0}$ $s_{2,1}$ $s_{2,2}$ $s_{2,3}$
$s_{3,0}$ $s_{3,1}$ $s_{3,2}$ $s_{3,3}$
$s'_{0,0}$ $s'_{0,1}$ $s'_{0,2}$ $s'_{0,3}$
$s'_{1,0}$ $s'_{1,1}$ $s'_{1,2}$ $s'_{1,3}$
$s'_{2,0}$ $s'_{2,1}$ $s'_{2,2}$ $s'_{2,3}$
$s'_{3,0}$ $s'_{3,1}$ $s'_{3,2}$ $s'_{3,3}$

(a) 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

(b) Inverse S-box

x \ y	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	52	09	6A	D5	30	36	A5	38	BF	40	A3	9E	81	F3	D7	FB
1	7C	E3	39	82	9B	2F	FF	87	34	8E	43	44	C4	DE	E9	CB
2	54	7B	94	32	A6	C2	23	3D	EE	4C	95	0B	42	FA	C3	4E
3	08	2E	A1	66	28	D9	24	B2	76	5B	A2	49	6D	8B	D1	25
4	72	F8	F6	64	86	68	98	16	D4	A4	5C	CC	5D	65	B6	92
5	6C	70	48	50	FD	ED	B9	DA	5E	15	46	57	A7	8D	9D	84
6	90	D8	AB	00	8C	BC	D3	0A	F7	E4	58	05	B8	B3	45	06
7	D0	2C	1E	8F	CA	3F	0F	02	C1	AF	BD	03	01	13	8A	6B
8	3A	91	11	41	4F	67	DC	EA	97	F2	CF	CE	F0	B4	E6	73
9	96	AC	74	22	E7	AD	35	85	E2	F9	37	E8	1C	75	DF	6E
A	47	F1	1A	71	1D	29	C5	89	6F	B7	62	0E	AA	18	BE	1B
B	FC	56	3E	4B	C6	D2	79	20	9A	DB	C0	FE	78	CD	5A	F4
C	1F	DD	A8	33	88	07	C7	31	B1	12	10	59	27	80	EC	5F
D	60	51	7F	A9	19	B5	4A	0D	2D	E5	7A	9F	93	C9	9C	EF
E	A0	E0	3B	4D	AE	2A	F5	B0	C8	EB	BB	3C	83	53	99	61
F	17	2B	04	7E	BA	77	D6	26	E1	69	14	63	55	21	0C	7D

Shift Rows

- STEP 2: This step (known as ShiftRows) is shown in following figure.

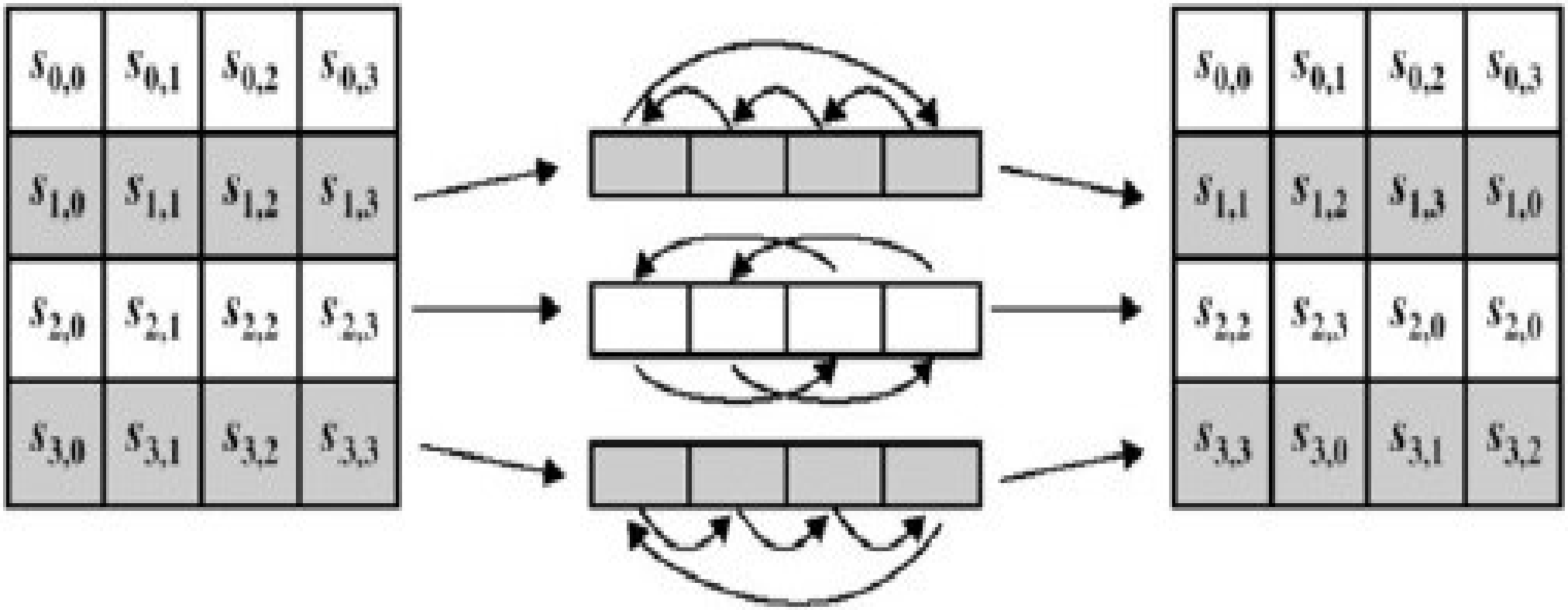
This is a simple permutation and it works as follow:

• The first row of **state** is *not* altered.

• The second row is shifted 1 bytes to the left in a circular manner.

• The third row is shifted 2 bytes to the left in a circular manner.

• The fourth row is shifted 3 bytes to the left in a circular manner.


$S_{0,0}$ $S_{0,1}$ $S_{0,2}$ $S_{0,3}$
$S_{1,0}$ $S_{1,1}$ $S_{1,2}$ $S_{1,3}$
$S_{2,0}$ $S_{2,1}$ $S_{2,2}$ $S_{2,3}$
$S_{3,0}$ $S_{3,1}$ $S_{3,2}$ $S_{3,3}$
$S_{0,0}$ $S_{0,1}$ $S_{0,2}$ $S_{0,3}$
$S_{1,1}$ $S_{1,2}$ $S_{1,3}$ $S_{1,0}$
$S_{2,2}$ $S_{2,3}$ $S_{2,0}$ $S_{2,0}$
$S_{3,3}$ $S_{3,0}$ $S_{3,1}$ $S_{3,2}$

Shift Rows

- The Inverse Shift Rows transformation (known as InvShiftRows) performs these circular shifts in the opposite direction for each of the last three rows (the first row was
- unaltered to begin with).

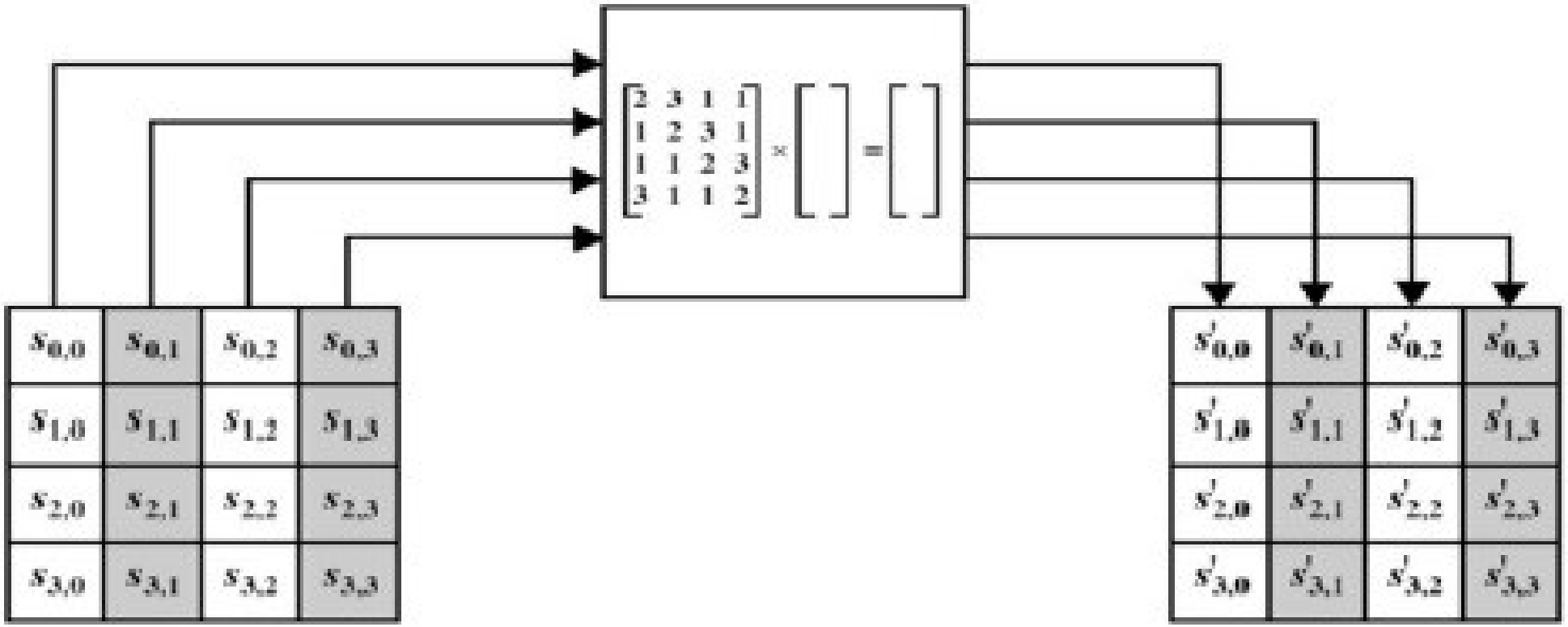
Mix Columns

- STEP 3: This step (known as MixColumn) is basically a substitution but it makes use of arithmetic of GF(2^8). Each column is operated on individually. Each byte of a column is mapped into a new value that is a function of all four bytes in the column. The transformation can be determined by the following matrix multiplication on state.

$$\begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} s_{0,0} & s_{0,1} & s_{0,2} & s_{0,3} \\ s_{1,0} & s_{1,1} & s_{1,2} & s_{1,3} \\ s_{2,0} & s_{2,1} & s_{2,2} & s_{2,3} \\ s_{3,0} & s_{3,1} & s_{3,2} & s_{3,3} \end{bmatrix} = \begin{bmatrix} s'_{0,0} & s'_{0,1} & s'_{0,2} & s'_{0,3} \\ s'_{1,0} & s'_{1,1} & s'_{1,2} & s'_{1,3} \\ s'_{2,0} & s'_{2,1} & s'_{2,2} & s'_{2,3} \\ s'_{3,0} & s'_{3,1} & s'_{3,2} & s'_{3,3} \end{bmatrix}$$

$$s'_{0,j} = (2 \bullet s_{0,j}) \oplus (3 \bullet s_{1,j}) \oplus s_{2,j} \oplus s_{3,j}$$
$$s'_{1,j} = s_{0,j} \oplus (2 \bullet s_{1,j}) \oplus (3 \bullet s_{2,j}) \oplus s_{3,j}$$
$$s'_{2,j} = s_{0,j} \oplus s_{1,j} \oplus (2 \bullet s_{2,j}) \oplus (3 \bullet s_{3,j})$$
$$s'_{3,j} = (3 \bullet s_{0,j}) \oplus s_{1,j} \oplus s_{2,j} \oplus (2 \bullet s_{3,j})$$

where $\bullet$ denotes multiplication over the finite field GF(2^8).


2 3 1 1
1 2 3 1
1 1 2 3
3 1 1 2
×
=
$s_{0,0}$ $s_{0,1}$ $s_{0,2}$ $s_{0,3}$
$s_{1,0}$ $s_{1,1}$ $s_{1,2}$ $s_{1,3}$
$s_{2,0}$ $s_{2,1}$ $s_{2,2}$ $s_{2,3}$
$s_{3,0}$ $s_{3,1}$ $s_{3,2}$ $s_{3,3}$
$s'_{0,0}$ $s'_{0,1}$ $s'_{0,2}$ $s'_{0,3}$
$s'_{1,0}$ $s'_{1,1}$ $s'_{1,2}$ $s'_{1,3}$
$s'_{2,0}$ $s'_{2,1}$ $s'_{2,2}$ $s'_{2,3}$
$s'_{3,0}$ $s'_{3,1}$ $s'_{3,2}$ $s'_{3,3}$

The InvMixColumns is defined by the following matrix multiplication:

$$\begin{bmatrix} 0E & 0B & 0D & 09 \\ 09 & 0E & 0B & 0D \\ 0D & 09 & 0E & 0B \\ 0B & 0D & 09 & 0E \end{bmatrix} \begin{bmatrix} s_{0,0} & s_{0,1} & s_{0,2} & s_{0,3} \\ s_{1,0} & s_{1,1} & s_{1,2} & s_{1,3} \\ s_{2,0} & s_{2,1} & s_{2,2} & s_{2,3} \\ s_{3,0} & s_{3,1} & s_{3,2} & s_{3,3} \end{bmatrix} = \begin{bmatrix} s'_{0,0} & s'_{0,1} & s'_{0,2} & s'_{0,3} \\ s'_{1,0} & s'_{1,1} & s'_{1,2} & s'_{1,3} \\ s'_{2,0} & s'_{2,1} & s'_{2,2} & s'_{2,3} \\ s'_{3,0} & s'_{3,1} & s'_{3,2} & s'_{3,3} \end{bmatrix}$$

Add RoundKey

- STEP 4: This step called AddRoundKey for adding the round key to the output of the previous step during the forward process. The corresponding step during decryption is denoted InvAddRoundKey for inverse add round key transformation.

Add RoundKey

- Each round has its own round key that is derived from the original 128-bit encryption key in the manner described in this section. One of the four steps of each round, for both encryption and decryption, involves XORing of the round key with the state array.

Add RoundKey

- In the same manner as the 128-bit input block is arranged in the form of a state array, the algorithm first arranges the 16 bytes of the encryption key in the form of a 4×4 array of bytes.

Add RoundKey

$$\begin{bmatrix} k_0 & k_4 & k_8 & k_{12} \\ k_1 & k_5 & k_9 & k_{13} \\ k_2 & k_6 & k_{10} & k_{14} \\ k_3 & k_7 & k_{11} & k_{15} \end{bmatrix}$$

$$\Downarrow$$

$$\begin{bmatrix} w_0 & w_1 & w_2 & w_3 \end{bmatrix}$$

Add RoundKey

- The first four bytes of the encryption key constitute the word w0, the next four bytes the word w1, and so on.
- The algorithm subsequently expands the words [w0,w1,w2,w3] into a 44-word key schedule that can be labeled w0, w1, w2, w3, ..., w43

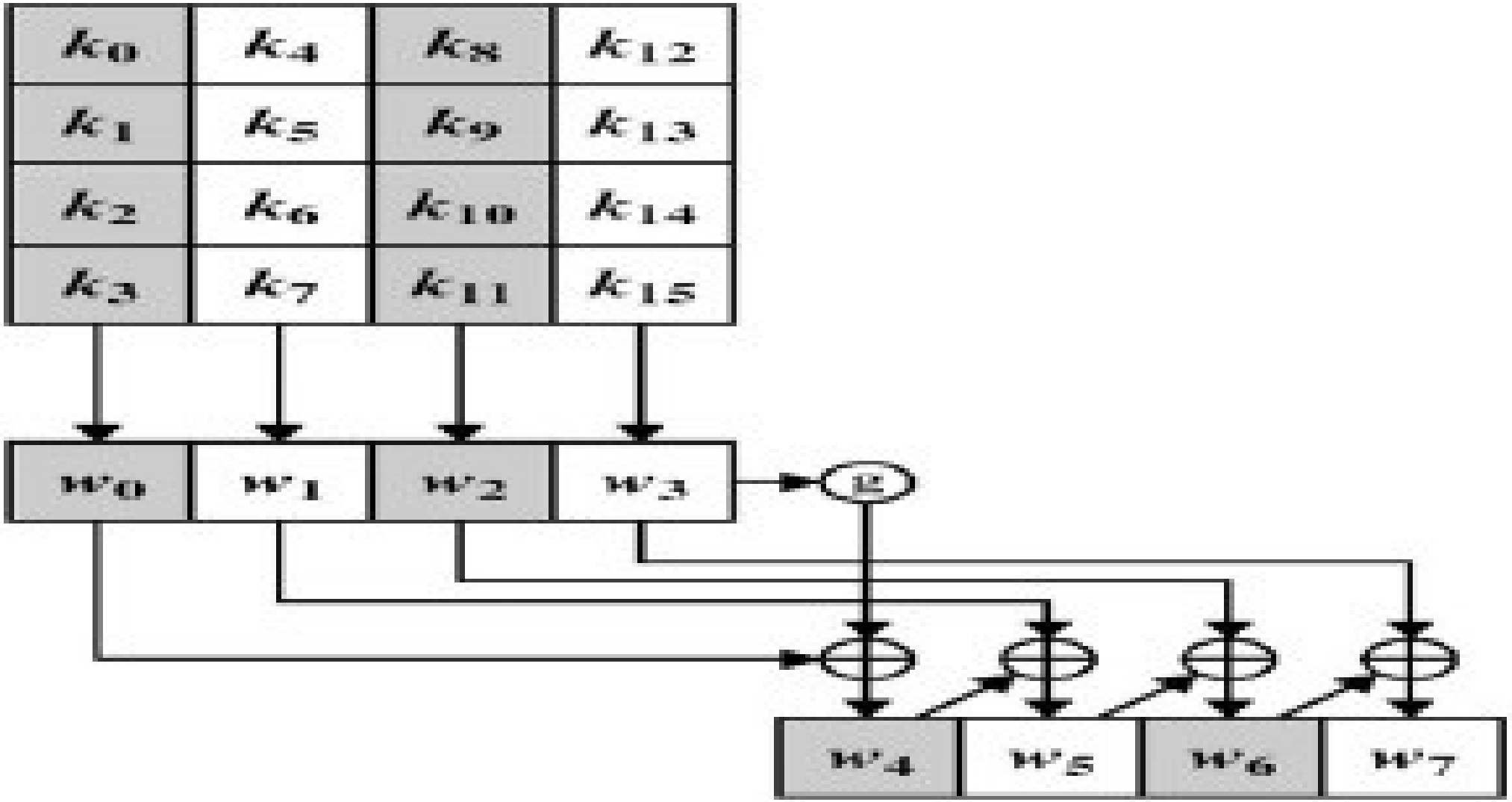
Add RoundKey

- the words [w0,w1,w2,w3] are bitwise XOR'ed with the input block before the round-based processing begins.
- The remaining 40 words of the key schedule are used four words at a time in each of the 10 rounds.
- The above two statements are also true for decryption, except for the fact that we now reverse the order of the words in the key schedule.

Add RoundKey

- Now comes the difficult part: How does the Key Expansion Algorithm expand four words w0,w1,w2,w3 into the 44 words w0,w1,w2,w3,w4,w5, ……..,w43 ?

Add RoundKey



Add RoundKey

- The function g consists of the following subfunctions
- **1. RotWord** performs a one-byte circular left shift on a word. This means that an input word [b0, b1, b2, b3] is transformed into [b1, b2, b3, b0].
- 2. **SubWord** performs a byte substitution on each byte of its input word, using the s-box described earlier.
- 3. The result of steps 1 and 2 is XORed with round constant, Rcon[j].

Add RoundKey

- The round constant is a word in which the three rightmost bytes are always 0. Thus the effect of an XOR of a word with Rcon is to only perform an XOR on the leftmost byte of the word. The round constant is different for each round and is defined as Rcon[j] = (RC[J], 0,0,0), with RC[1]= 1, RC[j]= 2• RC[j −1] and with multiplication defined over the field GF(28).