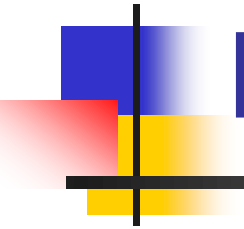


Module-II

Symmetric Key Cryptography

Symmetric Key Cryptography

Module-II

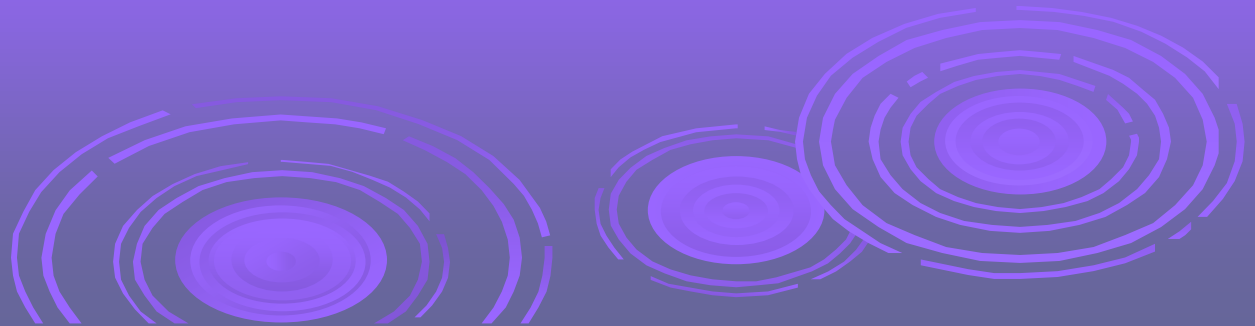
Contents:

29-3 MODERN CIPHERS

The traditional symmetric-key ciphers that we have studied so far are character-oriented ciphers. With the advent of the computer, we need bit-oriented ciphers. This is because the information to be encrypted is not just text; it can also consist of numbers, graphics, audio, and video data. It is convenient to convert these types of data into a stream of bits, to encrypt the stream, and then to send the encrypted stream. A modern block cipher can be either a block cipher or a stream cipher.

Modern Block Ciphers

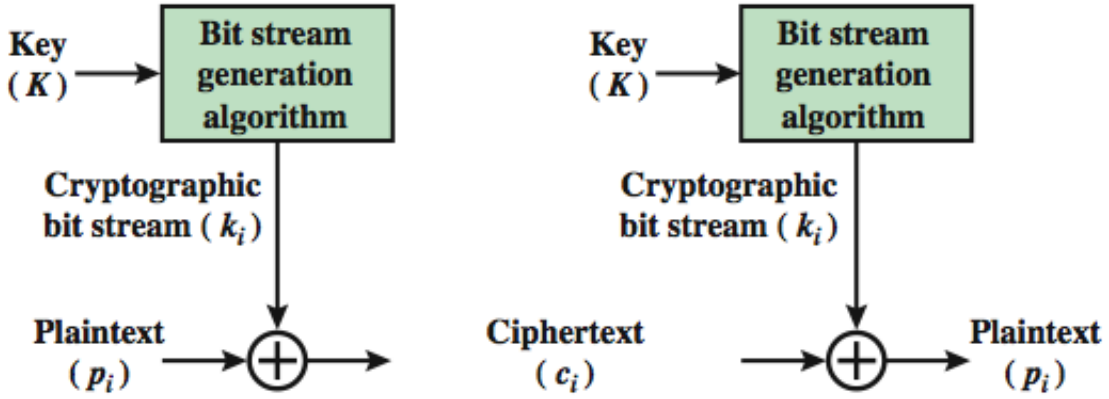
- now look at modern block ciphers
- one of the most widely used types of cryptographic algorithms
- provide secrecy /authentication services
- focus on DES (Data Encryption Standard)
- to illustrate block cipher design principles



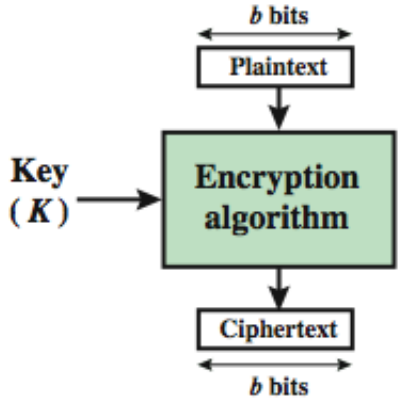
Block vs Stream Ciphers

- block ciphers process messages in blocks, each of which is then en/decrypted
- like a substitution on very big characters
  - 64-bits or more
- stream ciphers process messages a bit or byte at a time when en/decrypting
- many current ciphers are block ciphers
  - better analysed
  - broader range of applications

Block vs Stream Ciphers



(a) Stream Cipher Using Algorithmic Bit Stream Generator

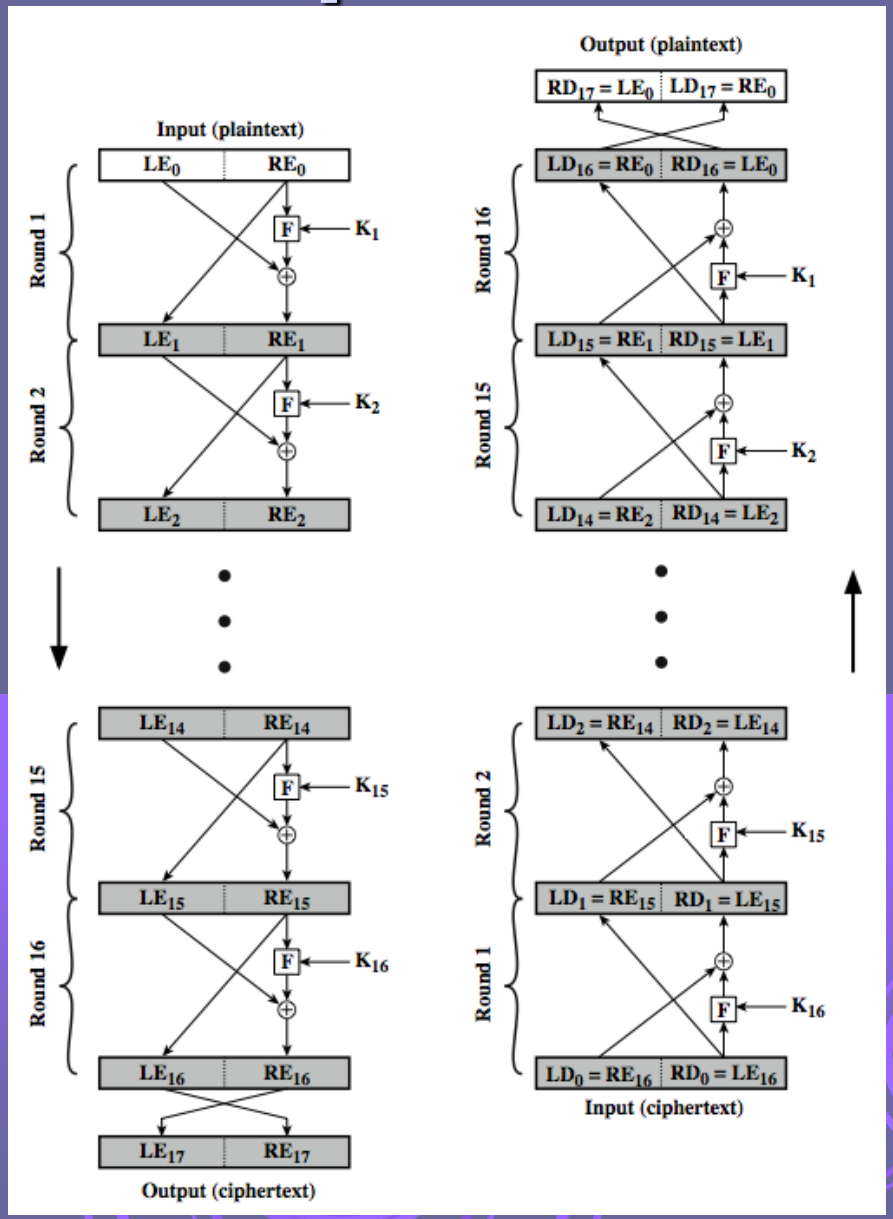


(b) Block Cipher

Feistel Cipher Structure

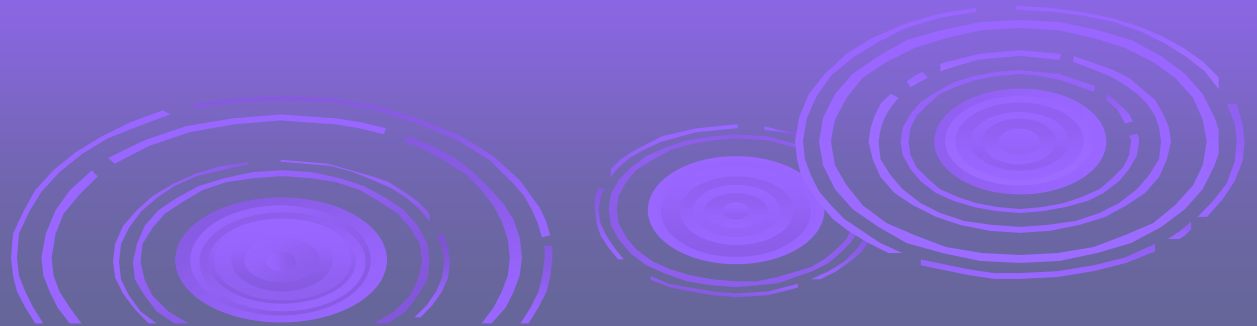
- Horst Feistel devised the F**eistel cipher**
  - based on concept of invertible product cipher
- partitions input block into two halves
  - process through multiple rounds which
  - perform a substitution on left data half
  - based on round function of right half & subkey
  - then have permutation swapping halves

Feistel Cipher Structure



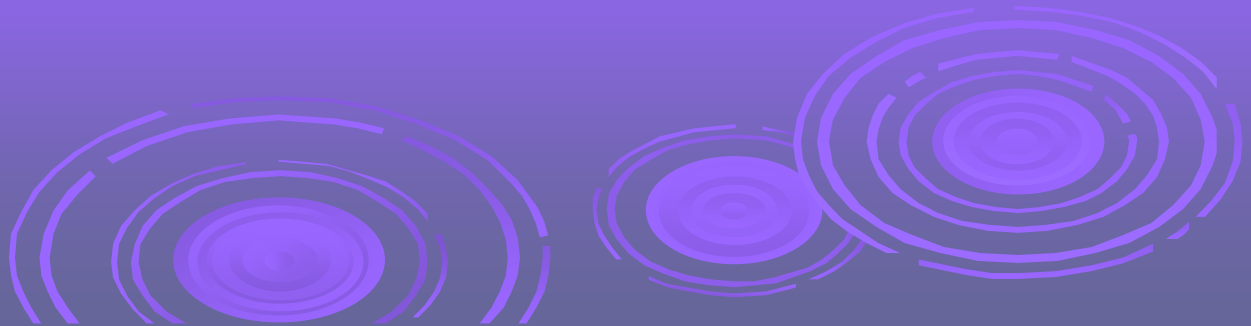
Feistel Cipher Design Elements

- block size
- key size
- number of rounds
- subkey generation algorithm
- round function
- fast software en/decryption
- ease of analysis



Data Encryption Standard (DES)

- most widely used block cipher in world
- adopted in 1977 by NBS (now NIST)
  - as FIPS PUB 46
- encrypts 64-bit data using 56-bit key
- has widespread use
- has been considerable controversy over its security



DES History

- IBM developed Lucifer cipher
  - by team led by Feistel in late 60's
  - used 64-bit data blocks with 128-bit key
- then redeveloped as a commercial cipher with input from NSA and others
- in 1973 NBS issued request for proposals for a national cipher standard
- IBM submitted their revised Lucifer which was eventually accepted as the DES

DES Design Controversy

- although DES standard is public
- was considerable controversy over design
  - in choice of 56-bit key (vs Lucifer 128-bit)
  - and because design criteria were classified
- subsequent events and public analysis show in fact design was appropriate
- use of DES has flourished
  - especially in financial applications
  - still standardised for legacy application use

Figure 29.7 *A modern block cipher*

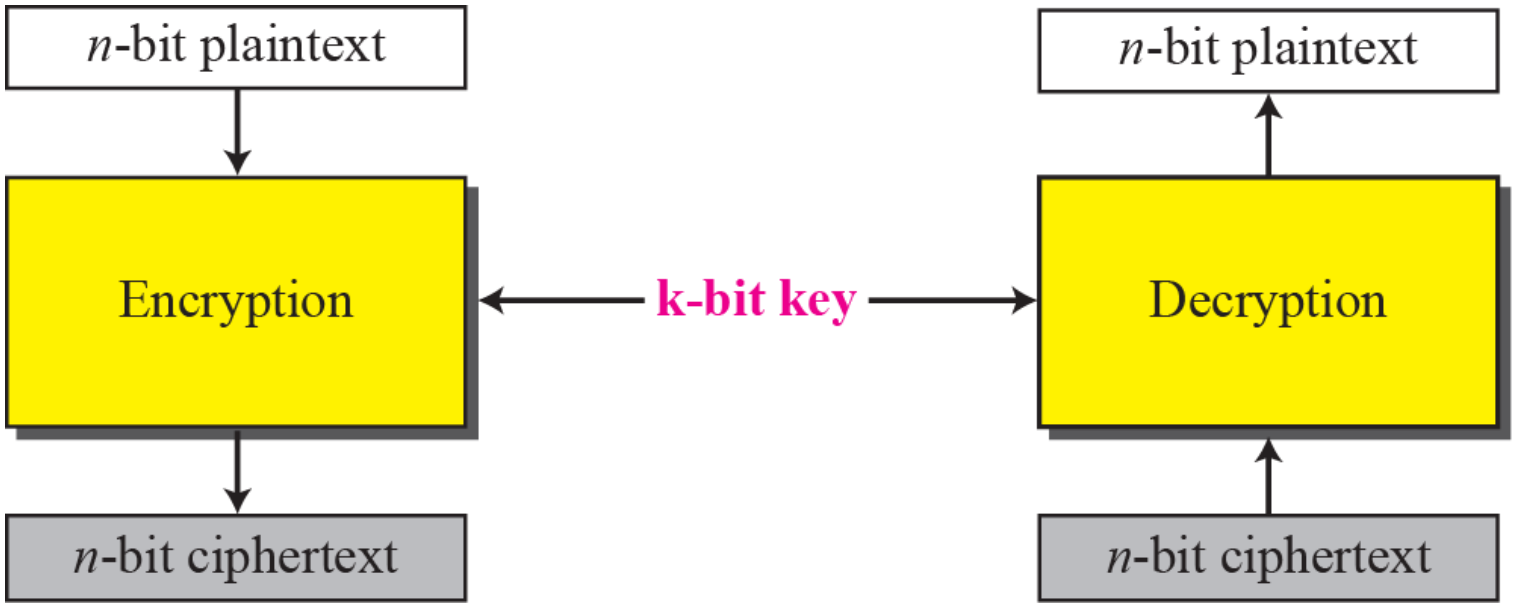


Figure 29.8 *Components of a modern block cipher*

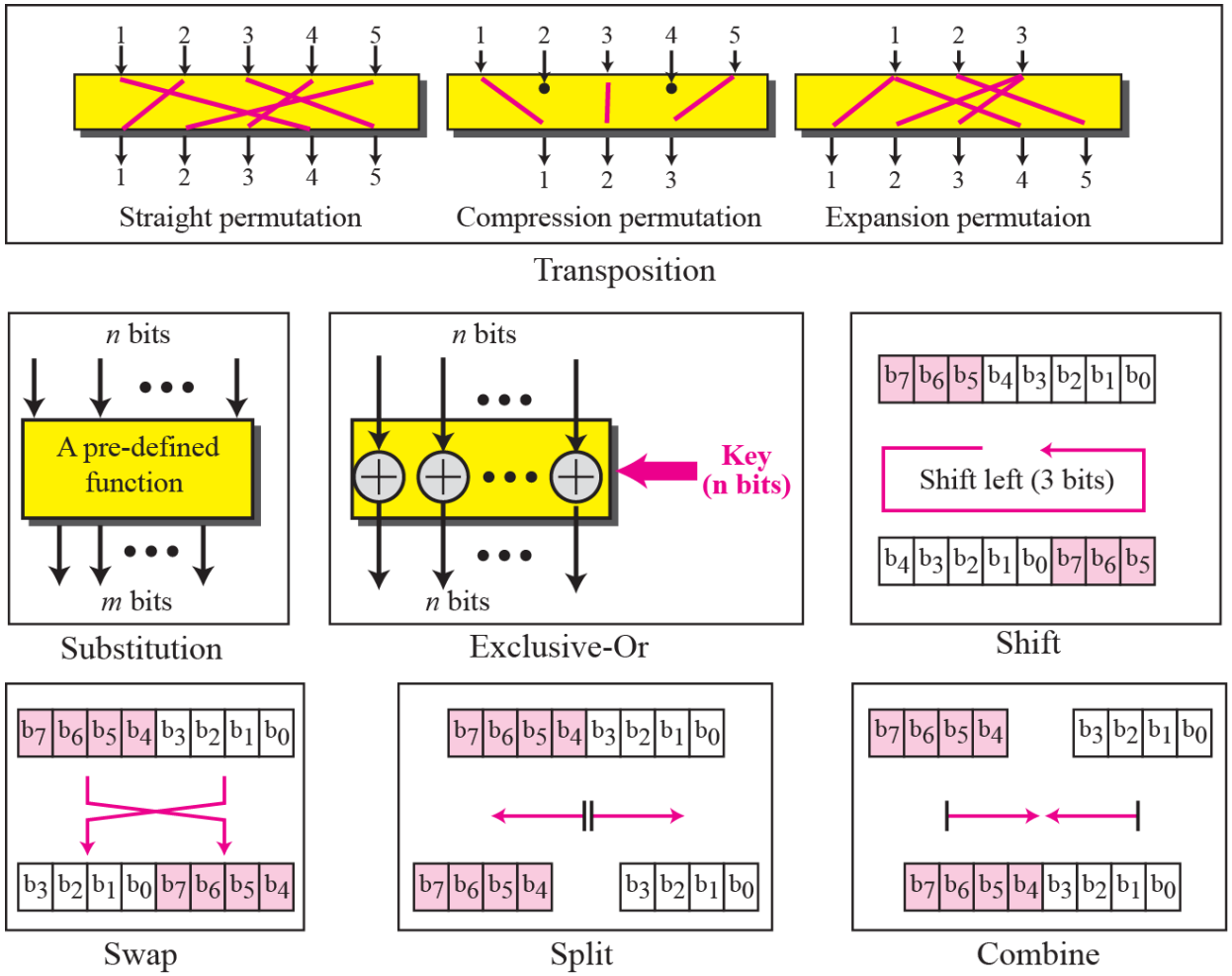


Figure 29.9 *General structure of DES*

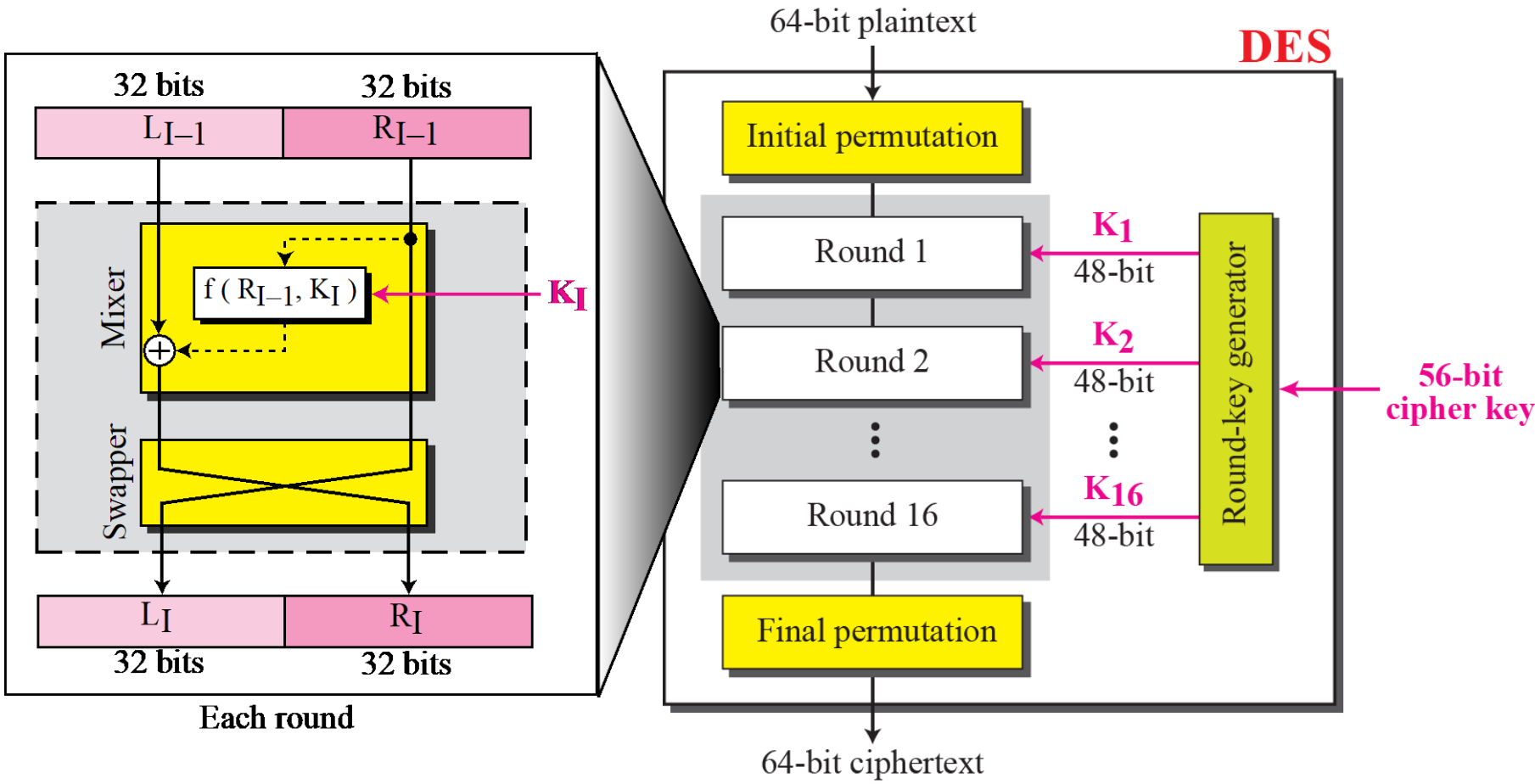
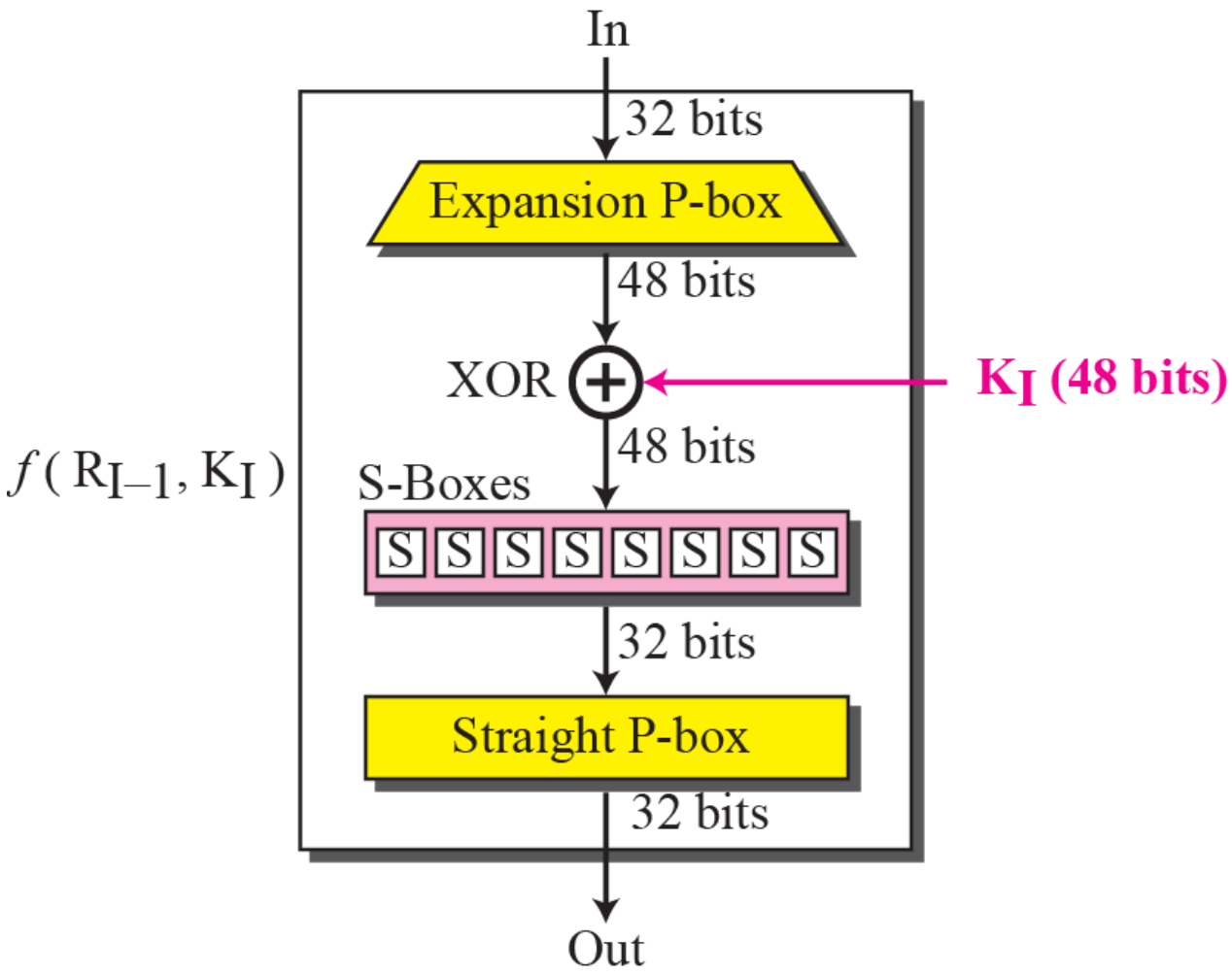


Figure 29.10 *DES function*



DES Round in Full

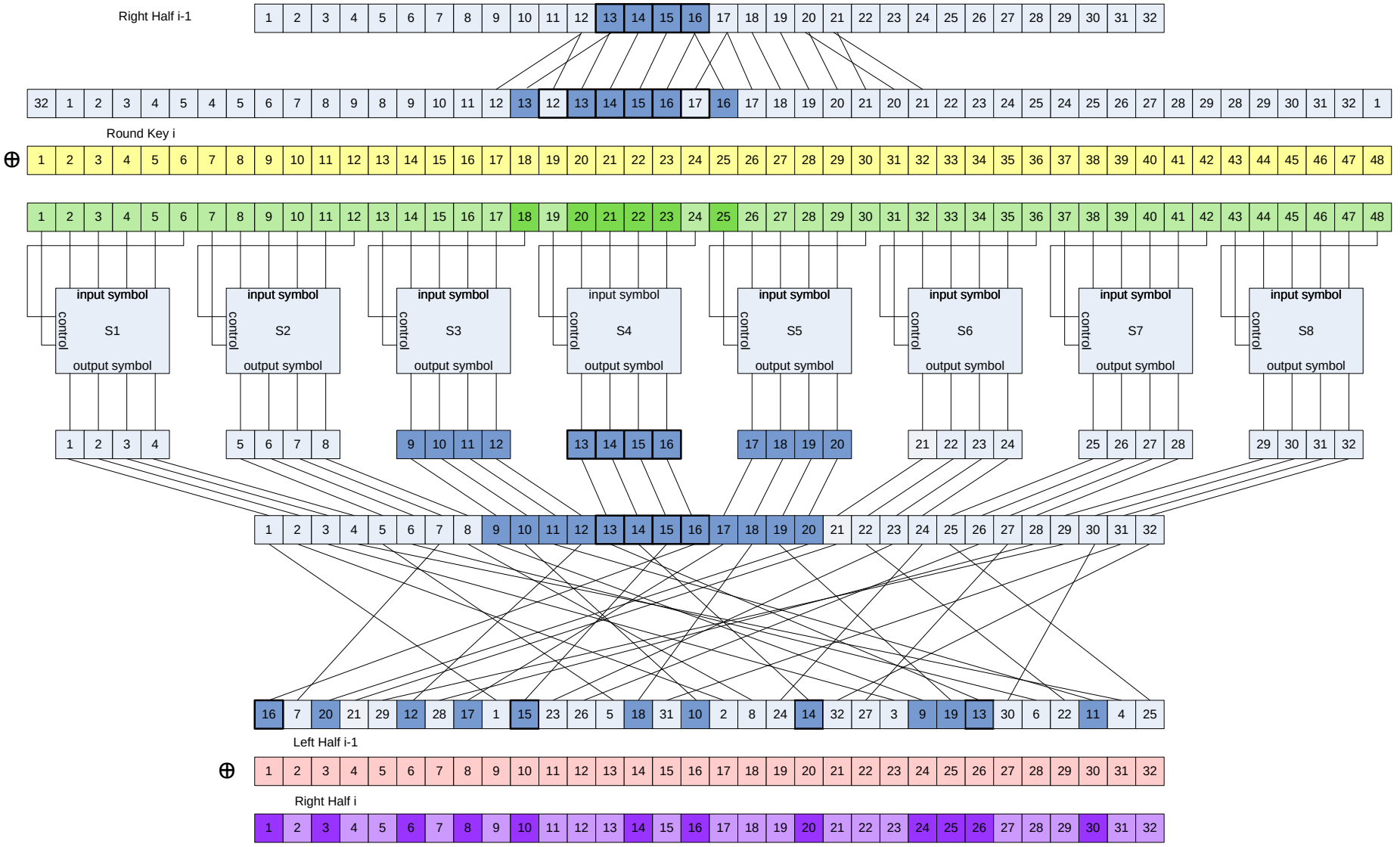
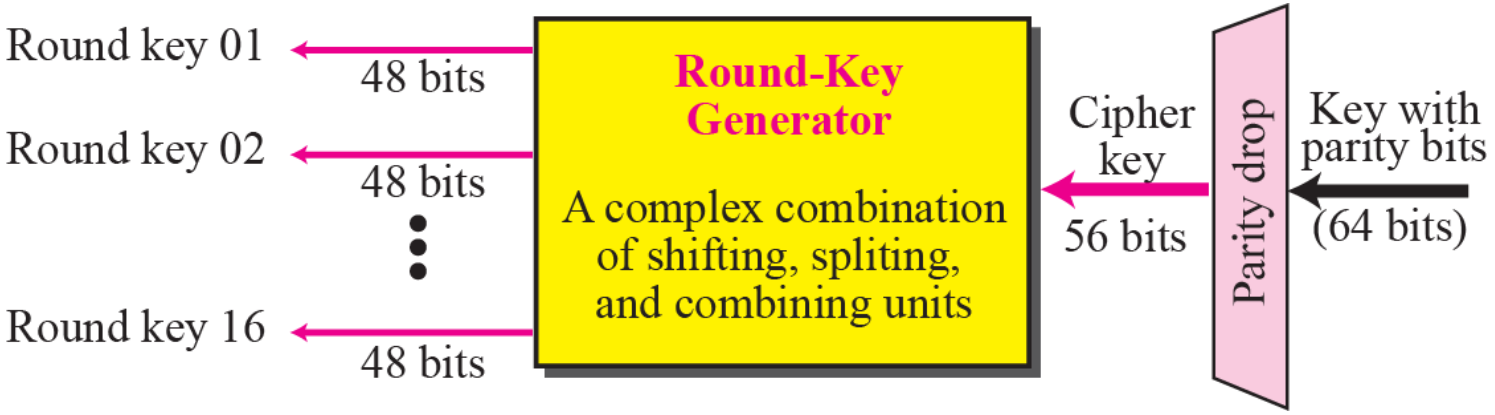


Figure 29.11 *Key generation*



Example 29.4

We choose a random plaintext block, a random key, and a computer program to determine what the ciphertext block would be (all in hexadecimal):

Plaintext:	Key:	CipherText:
123456ABCD132536	AABB09182736CCDD	C0B7A8D05F3A829C

Example 29.5

To check the effectiveness of DES, when a single bit is changed in the input, let us use two different plaintexts with only one single bit difference. The two ciphertexts are completely different without even changing the key:

Plaintext:	Key:	Ciphertext:
0000000000000000	22234512987ABB23	4789FD476E82A5F1
Plaintext:	Key:	Ciphertext:
0000000000000001	22234512987ABB23	0A4ED5C15A63FEA3

Although the two plaintext blocks differ only in the rightmost bit, the ciphertext blocks differ in 29 bits.

Advance Encryption Standard

Topics

- **Origin of AES**
- Basic AES
- Inside Algorithm
- Final Notes

Origins

- A replacement for DES was needed
  - Key size is too small
- Can use Triple-DES – but slow, small block
- US NIST issued call for ciphers in 1997
- 15 candidates accepted in Jun 98
- 5 were shortlisted in Aug 99

AES Competition Requirements

- symmetric key block cipher
- 128-bit data, 128/192/256-bit keys
- Stronger & faster than Triple-DES
- Provide full specification & design details

AES Evaluation Criteria

- criteria
  - general security
  - ease of software & hardware implementation
  - implementation attacks
  - flexibility (in en/decrypt, keying, other factors)

AES Shortlist

- After testing and evaluation, shortlist in Aug-99
  - MARS (IBM) - complex, fast, high security margin
  - RC6 (USA) - v. simple, v. fast, low security margin
  - Rijndael (Belgium) - clean, fast, good security margin
  - Serpent (Euro) - slow, clean, v. high security margin
  - Twofish (USA) - complex, v. fast, high security margin

Rijndae: pronounce “Rain-Dahl”

The AES Cipher - Rijndael

- Rijndael was selected as the AES in Oct-2000
  - Designed by Vincent Rijmen and Joan Daemen in Belgium
  - Issued as FIPS PUB 197 standard in Nov-2001

V. Rijmen

- An **iterative** rather than **Feistel** cipher
  - processes data as block of 4 columns of 4 bytes (128 bits)
  - operates on entire data block in every round

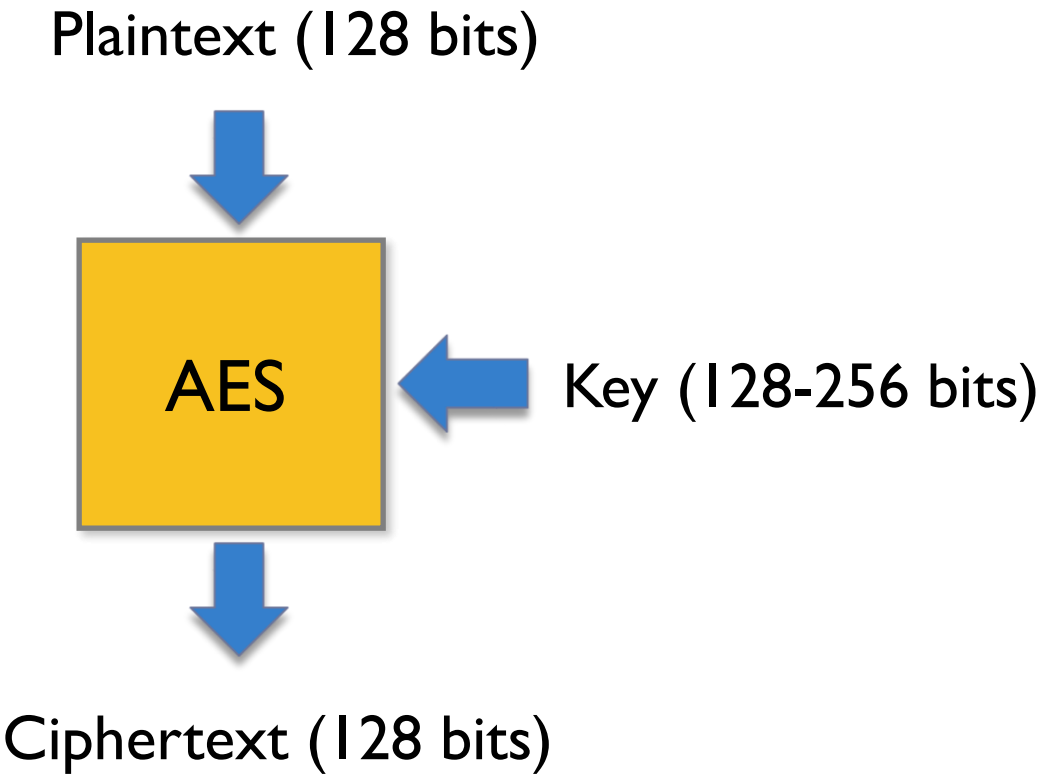
J. Daemen

- Rijndael design:
  - simplicity
  - has 128/192/256 bit keys, 128 bits data
  - resistant against known attacks
  - speed and code compactness on many CPUs

Topics

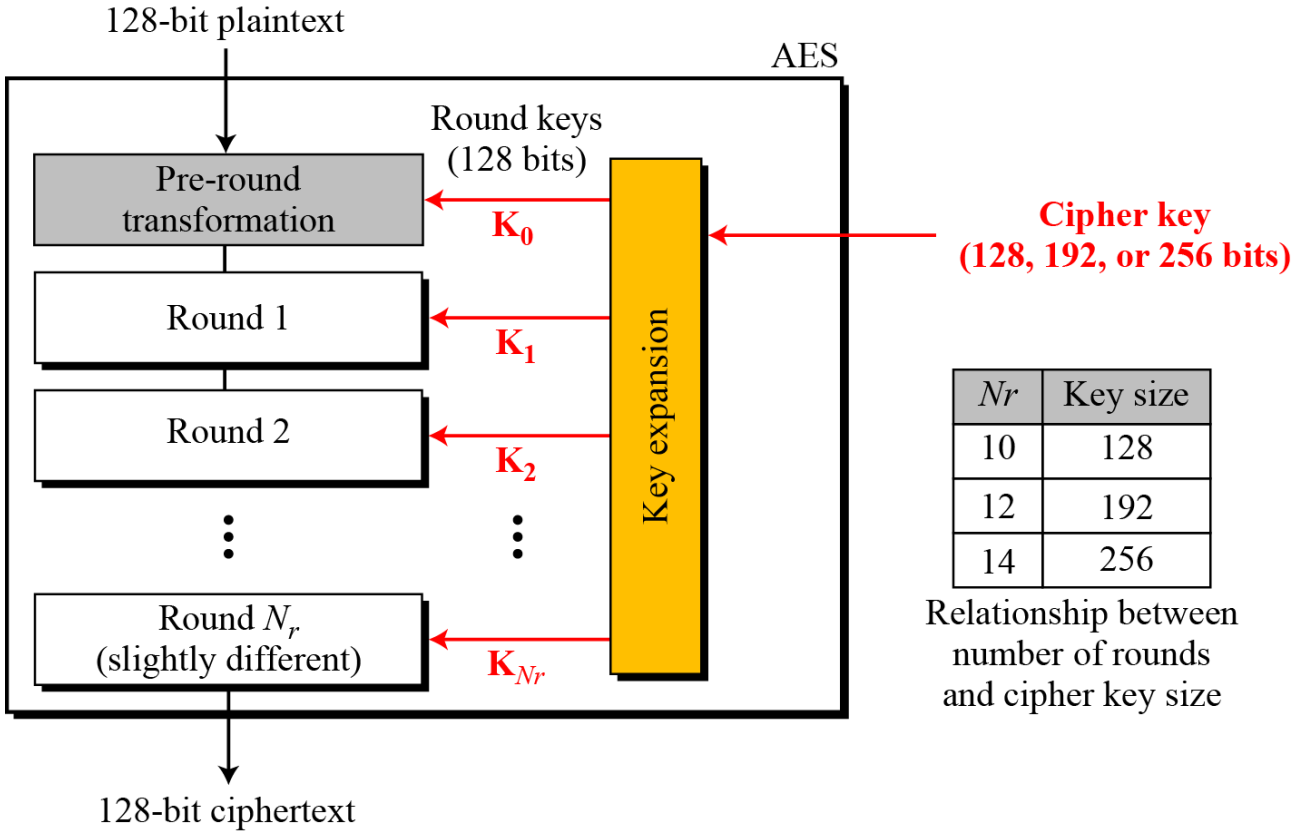
- Origin of AES
- **Basic AES**
- Inside Algorithm
- Final Notes

AES Conceptual Scheme



Multiple rounds

- Rounds are (almost) identical
  - First and last round are a little different



Nr	Key size
10	128
12	192
14	256

Relationship between number of rounds and cipher key size

High Level Description

Key Expansion

- Round keys are derived from the cipher key using Rijndael's key schedule

Initial Round

- AddRoundKey : Each byte of the state is combined with the round key using bitwise xor

Rounds

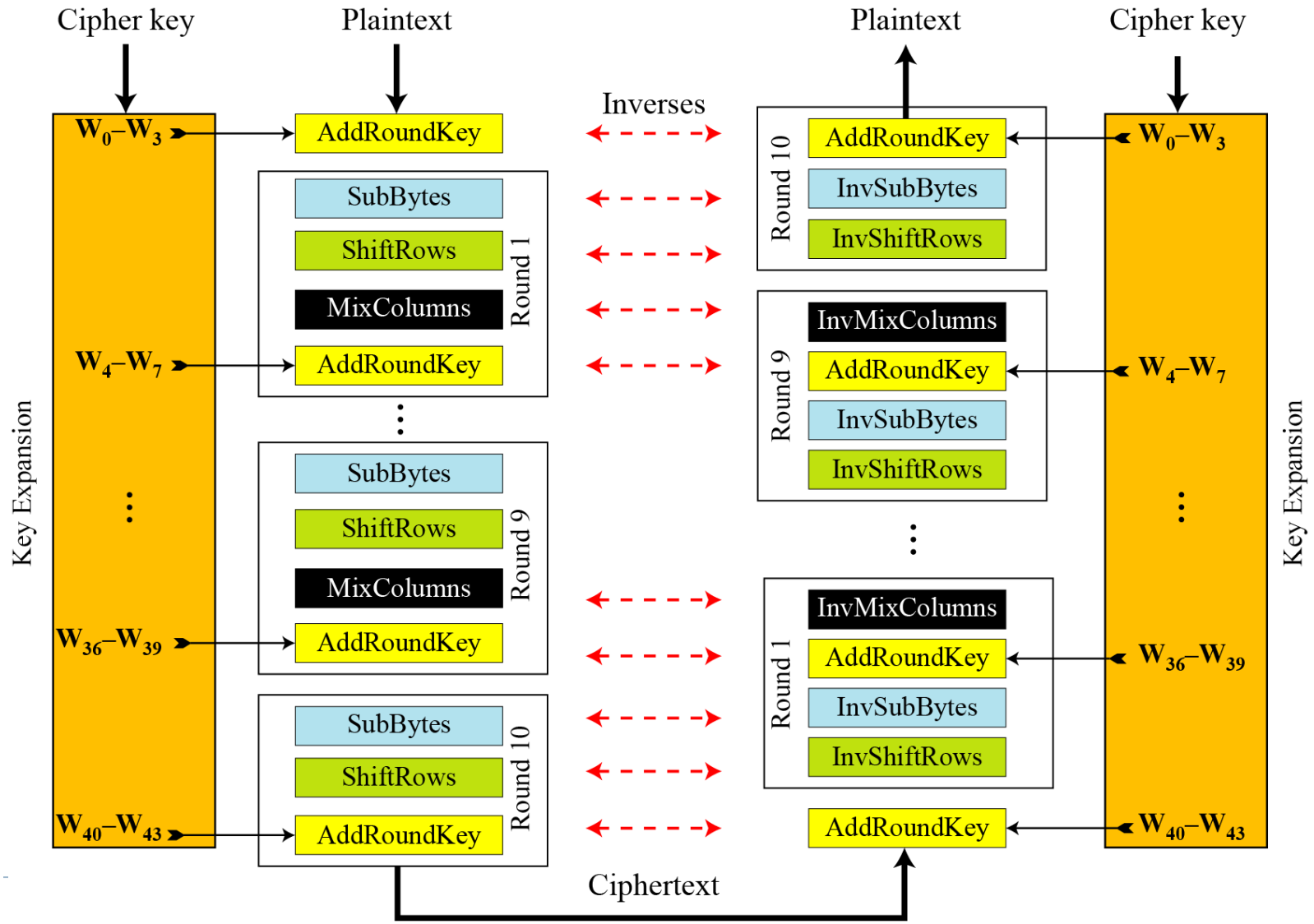
- SubBytes : non-linear substitution step
- ShiftRows : transposition step
- MixColumns : mixing operation of each column.
- AddRoundKey

Final Round

- SubBytes
- ShiftRows
- AddRoundKey

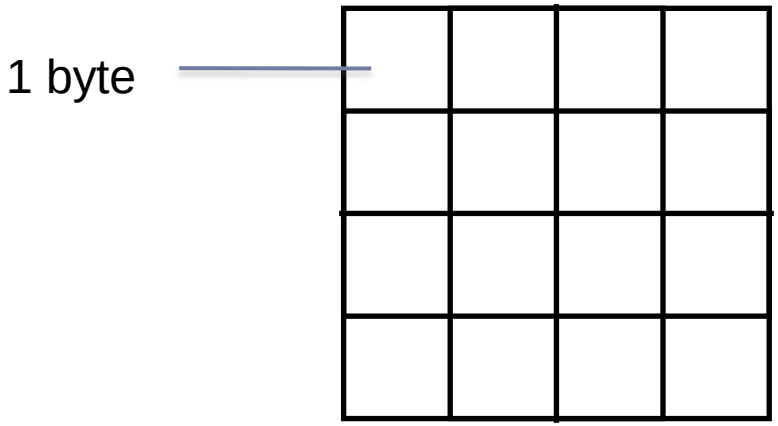
No MixColumns

Overall Structure

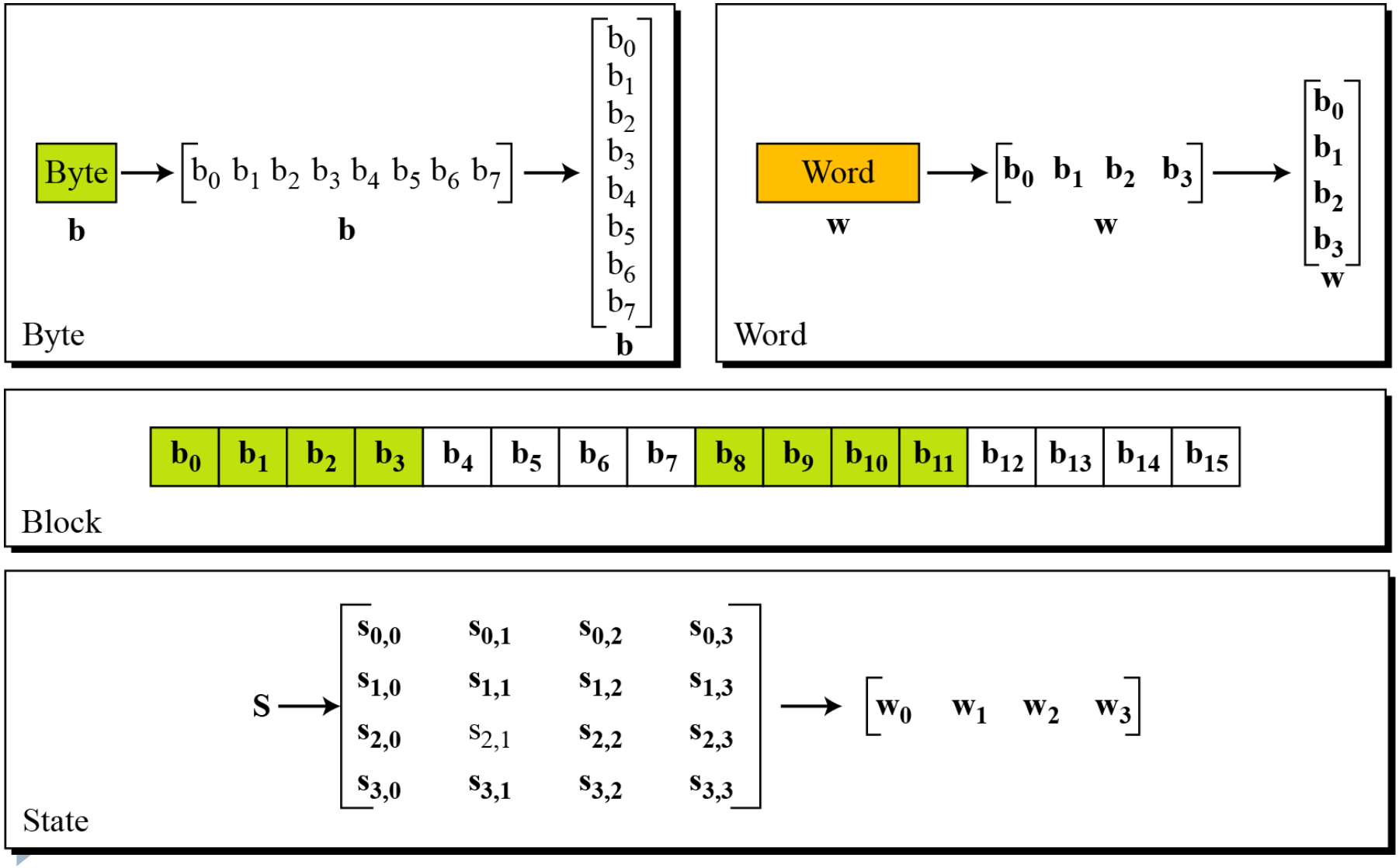


128-bit values

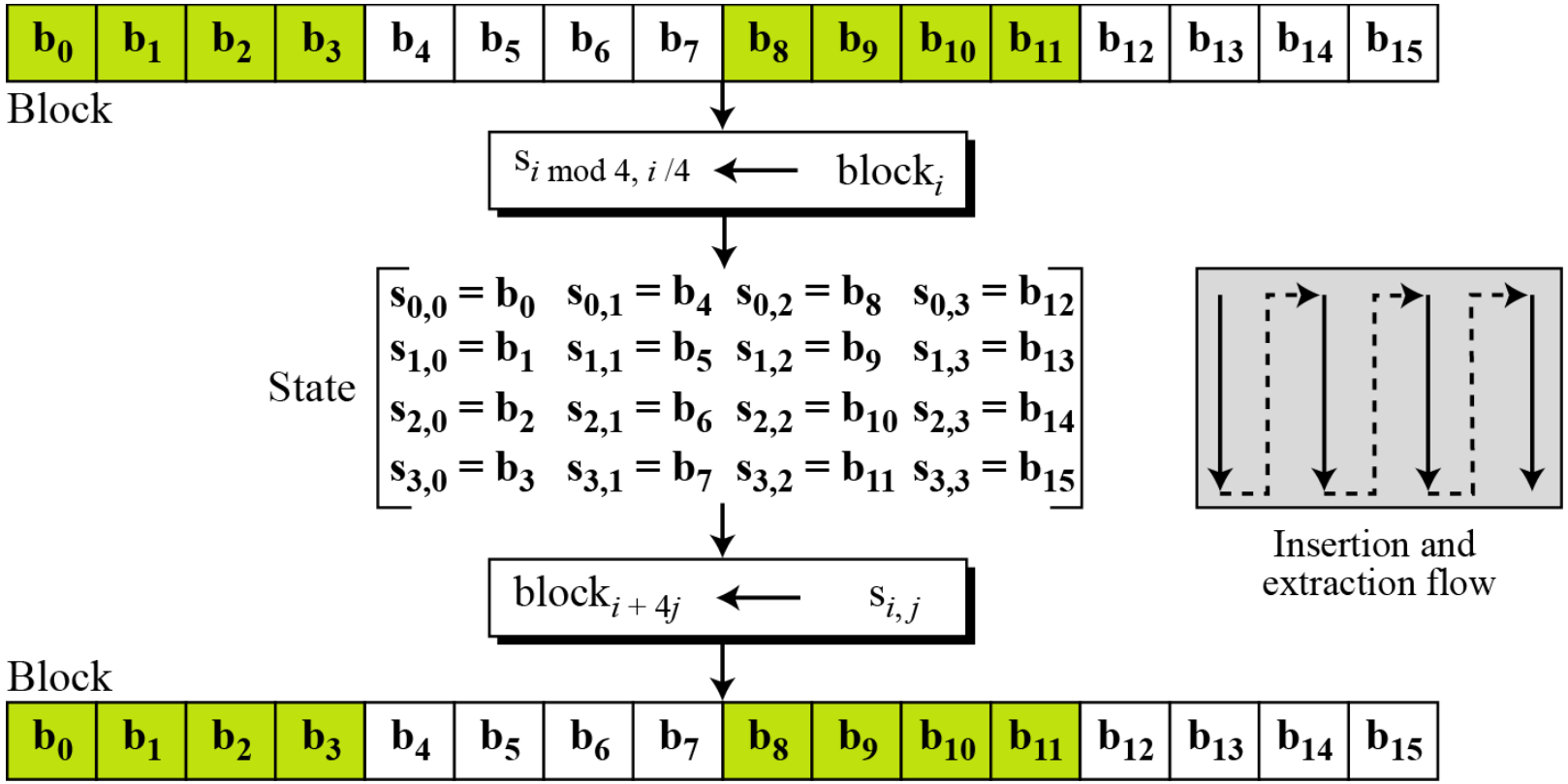
- Data block viewed as 4-by-4 table of bytes
- Represented as 4 by 4 matrix of 8-bit bytes.
- Key is expanded to array of 32 bits words



Data Unit


Byte
b
b0 b1 b2 b3 b4 b5 b6 b7
b
b0
b1
b2
b3
b4
b5
b6
b7
b
Byte
Word
w
b0 b1 b2 b3
w
b0
b1
b2
b3
w
Word
b0 b1 b2 b3 b4 b5 b6 b7 b8 b9 b10 b11 b12 b13 b14 b15
Block
S
s0,0 s0,1 s0,2 s0,3
s1,0 s1,1 s1,2 s1,3
s2,0 s2,1 s2,2 s2,3
s3,0 s3,1 s3,2 s3,3
w0 w1 w2 w3
State

Unit Transformation



Changing Plaintext to State

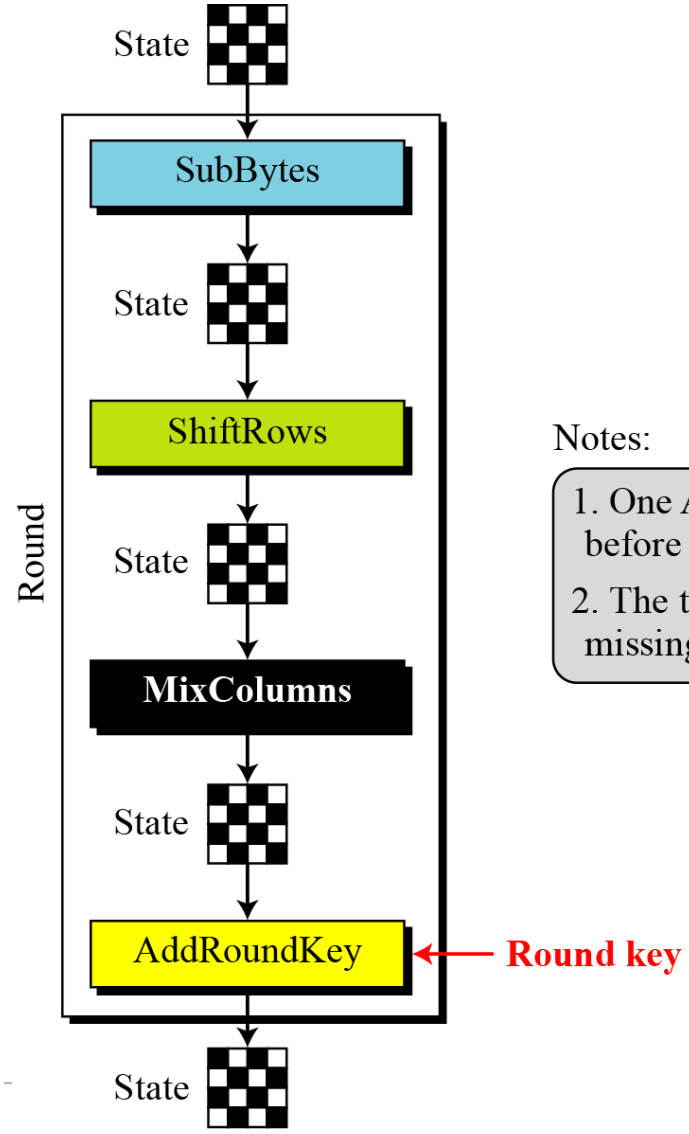
Text	A	E	S	U	S	E	S	A	M	A	T	R	I	X	Z	Z
Hexadecimal	00	04	12	14	12	04	12	00	0C	00	13	11	08	23	19	19

$$\begin{bmatrix} 00 & 12 & 0C & 08 \\ 04 & 04 & 00 & 23 \\ 12 & 12 & 13 & 19 \\ 14 & 00 & 11 & 19 \end{bmatrix}$$ State

Topics

- Origin of AES
- Basic AES
- **Inside Algorithm**
- Final Notes

Details of Each Round



Notes:

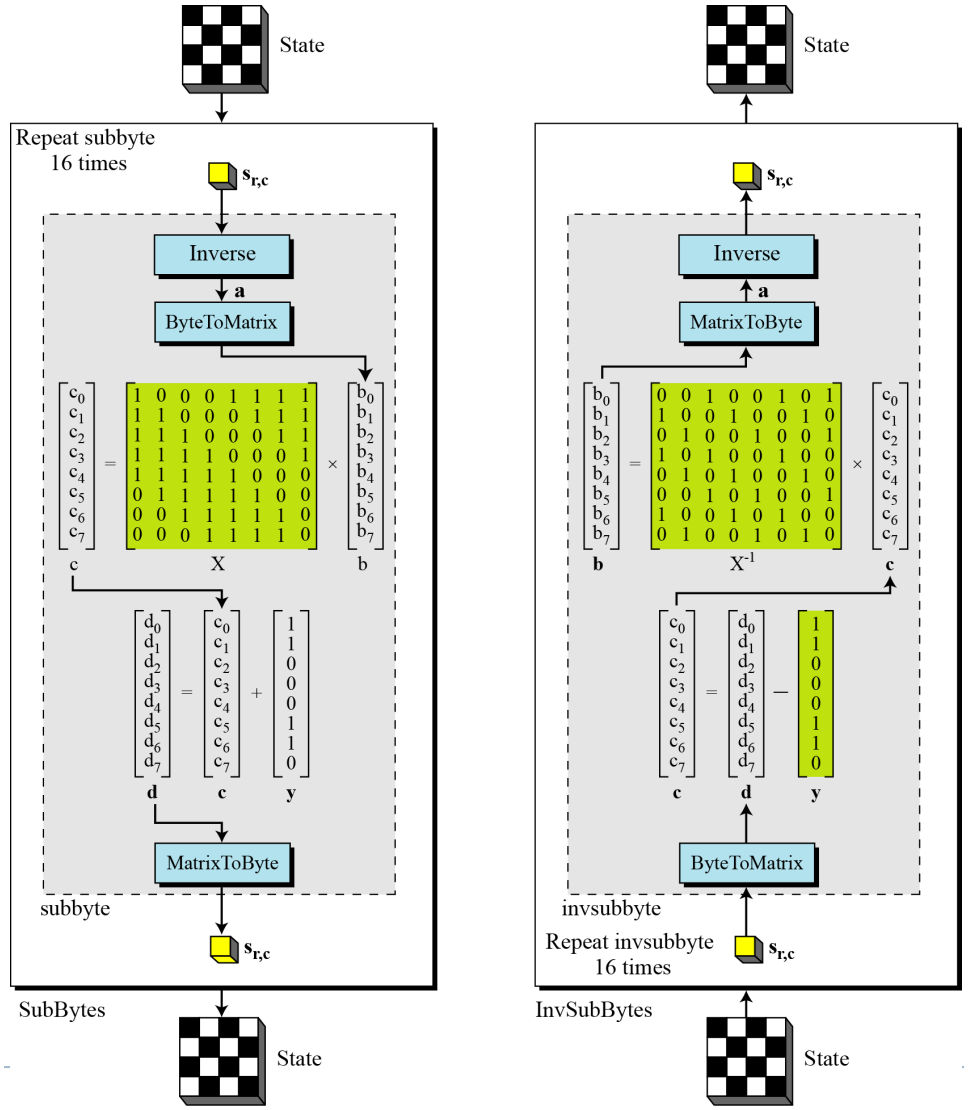
1. One AddRoundKey is applied before the first round.
2. The third transformation is missing in the last round.

SubBytes: Byte Substitution

- A simple substitution of each byte
  - provide a confusion
- Uses one S-box of 16x16 bytes containing a permutation of all 256 8-bit values
- Each byte of state is replaced by byte indexed by row (left 4-bits) & column (right 4-bits)
  - eg. byte {95} is replaced by byte in row 9 column 5
  - which has value {2A}
- S-box constructed using defined transformation of values in Galois Field- $GF(2^8)$

Galois : pronounce “Gal-Wa”

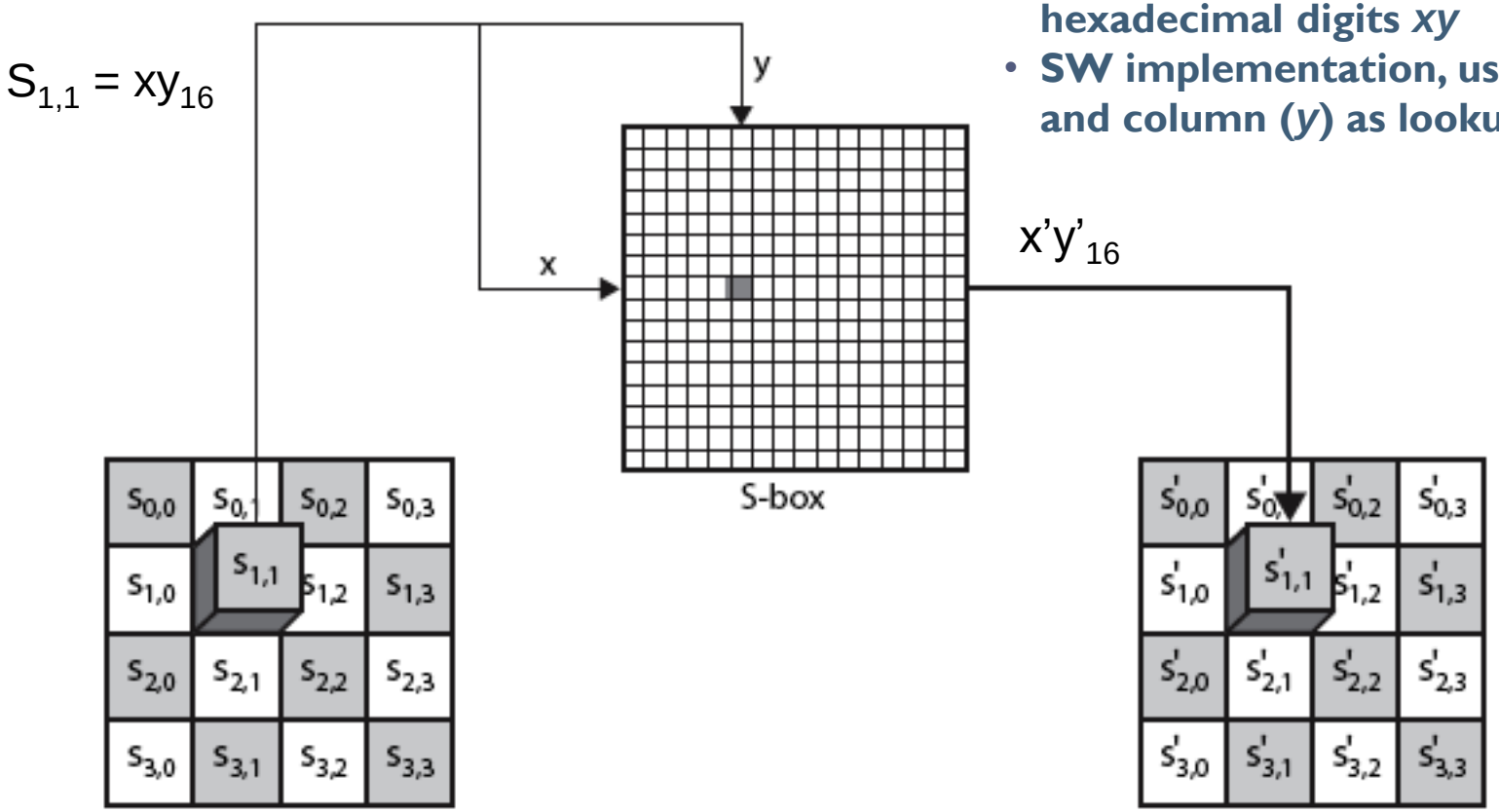
SubBytes and InvSubBytes



SubBytes Operation

- The SubBytes operation involves 16 independent byte-to-byte transformations.

- **Interpret the byte as two hexadecimal digits *xy***
- **SW implementation, use row (*x*) and column (*y*) as lookup pointer**



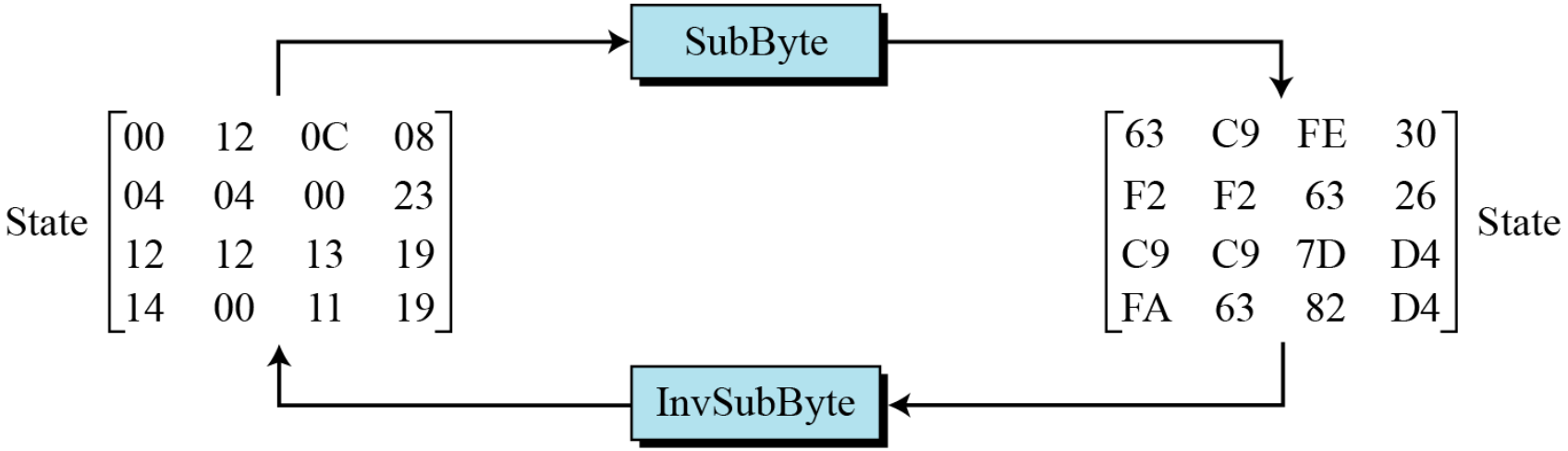
SubBytes Table

- Implement by Table Lookup

		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

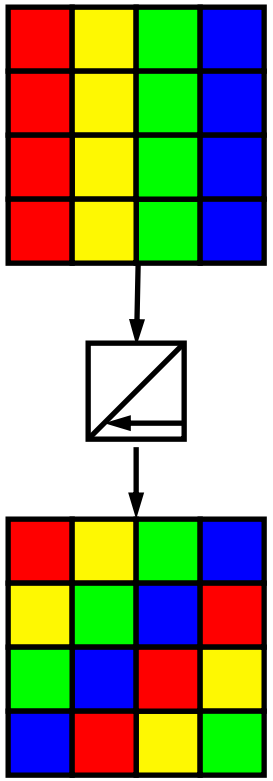
Sample SubByte Transformation

- The SubBytes and InvSubBytes transformations are inverses of each other.

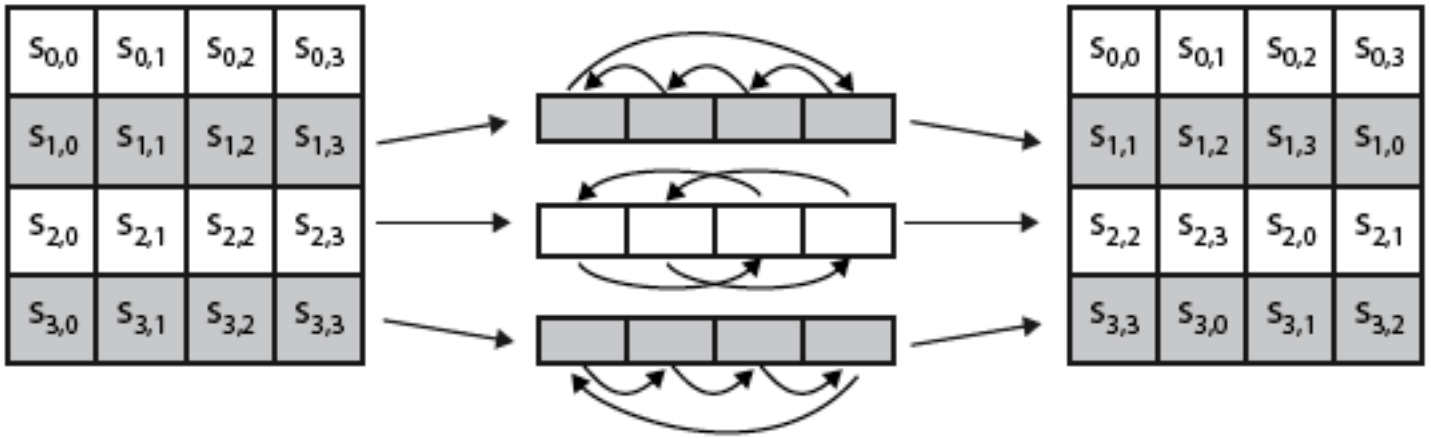


ShiftRows

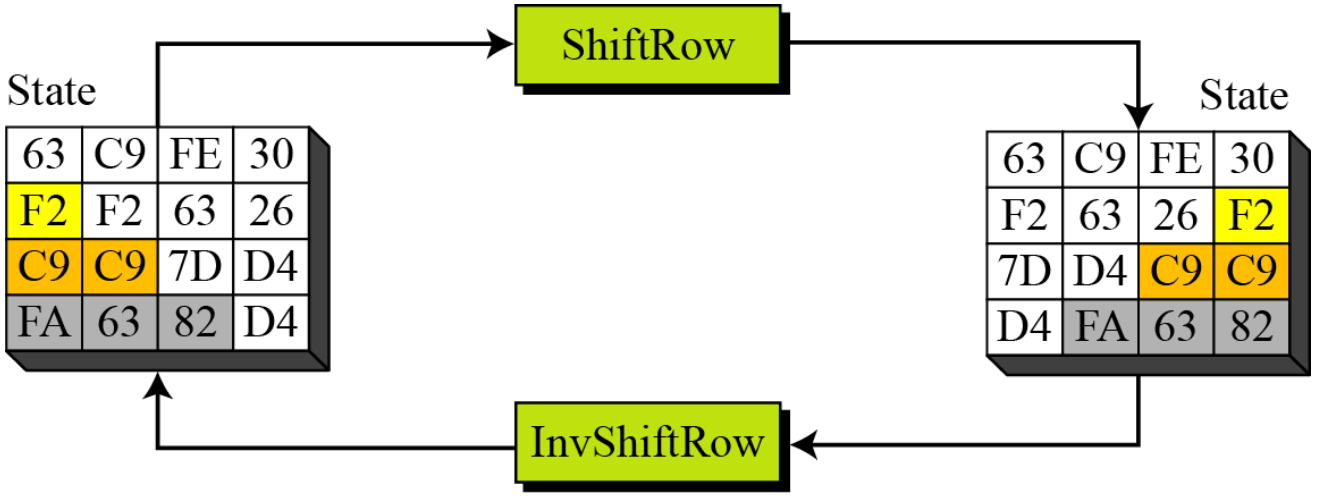
- Shifting, which permutes the bytes.
- A circular byte shift in each each
  - 1st row is unchanged
  - 2nd row does 1 byte circular shift to left
  - 3rd row does 2 byte circular shift to left
  - 4th row does 3 byte circular shift to left
- In the encryption, the transformation is called ShiftRows
- In the decryption, the transformation is called InvShiftRows and the shifting is to the right



ShiftRows Scheme

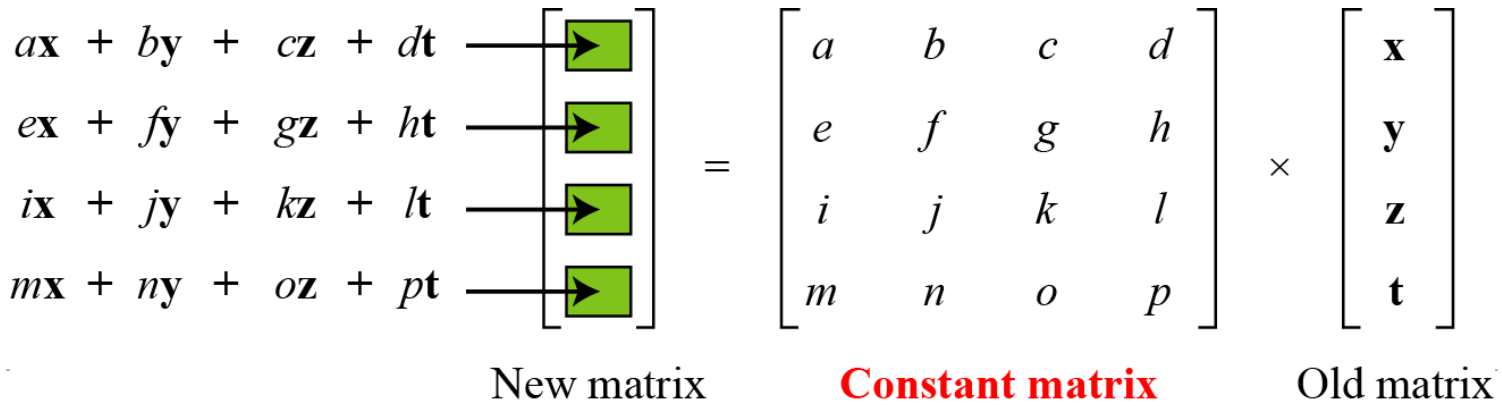


ShiftRows and InvShiftRows

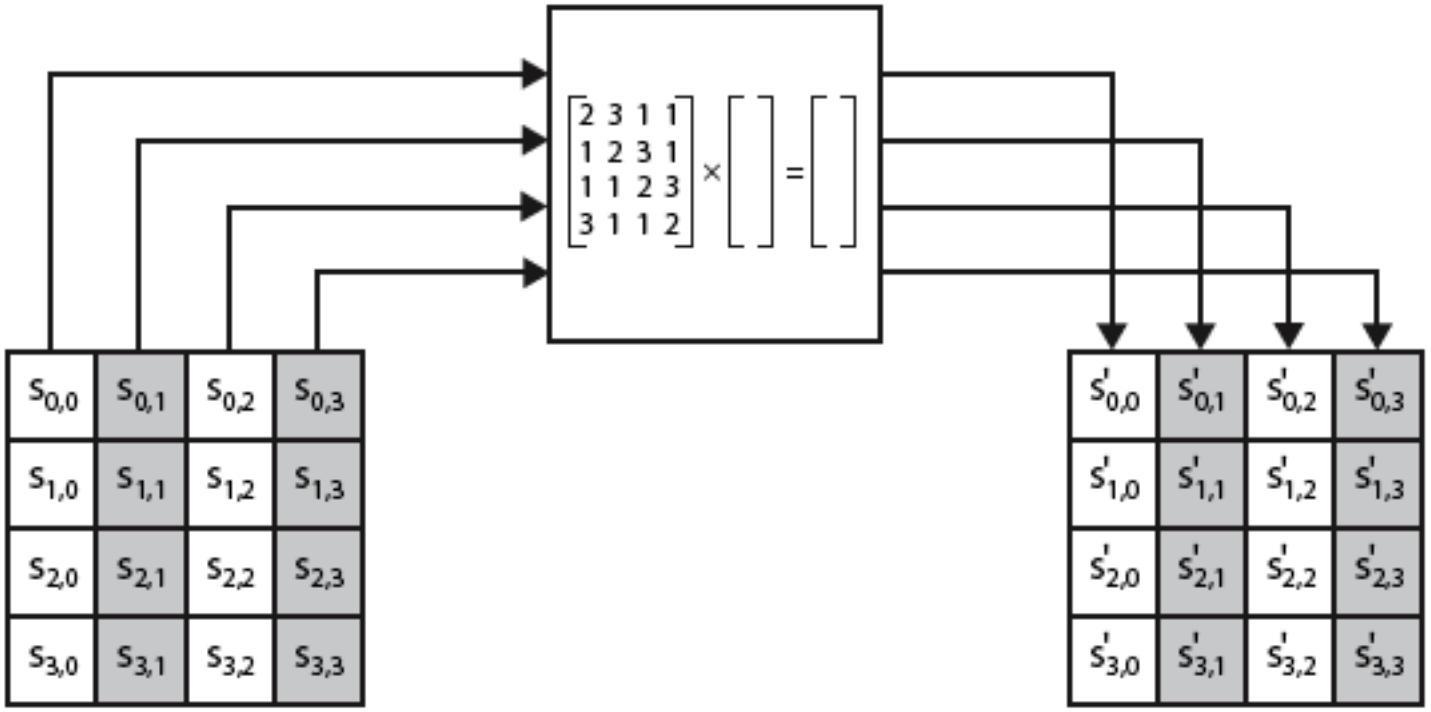


MixColumns

- ShiftRows and MixColumns provide diffusion to the cipher
- Each column is processed separately
- Each byte is replaced by a value dependent on all 4 bytes in the column
- Effectively a matrix multiplication in GF(2^8)



MixClumns Scheme

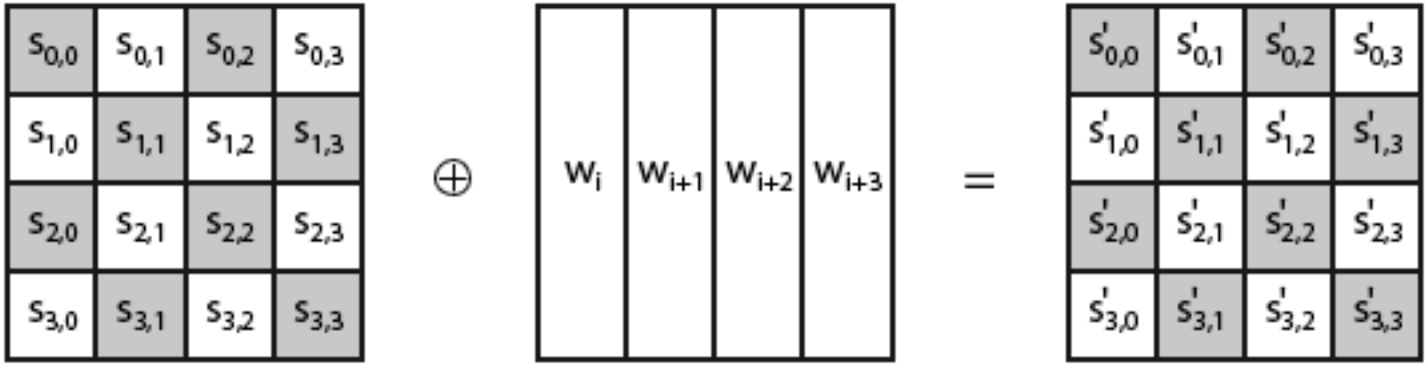


The MixColumns transformation operates at the column level; it transforms each column of the state to a new column.

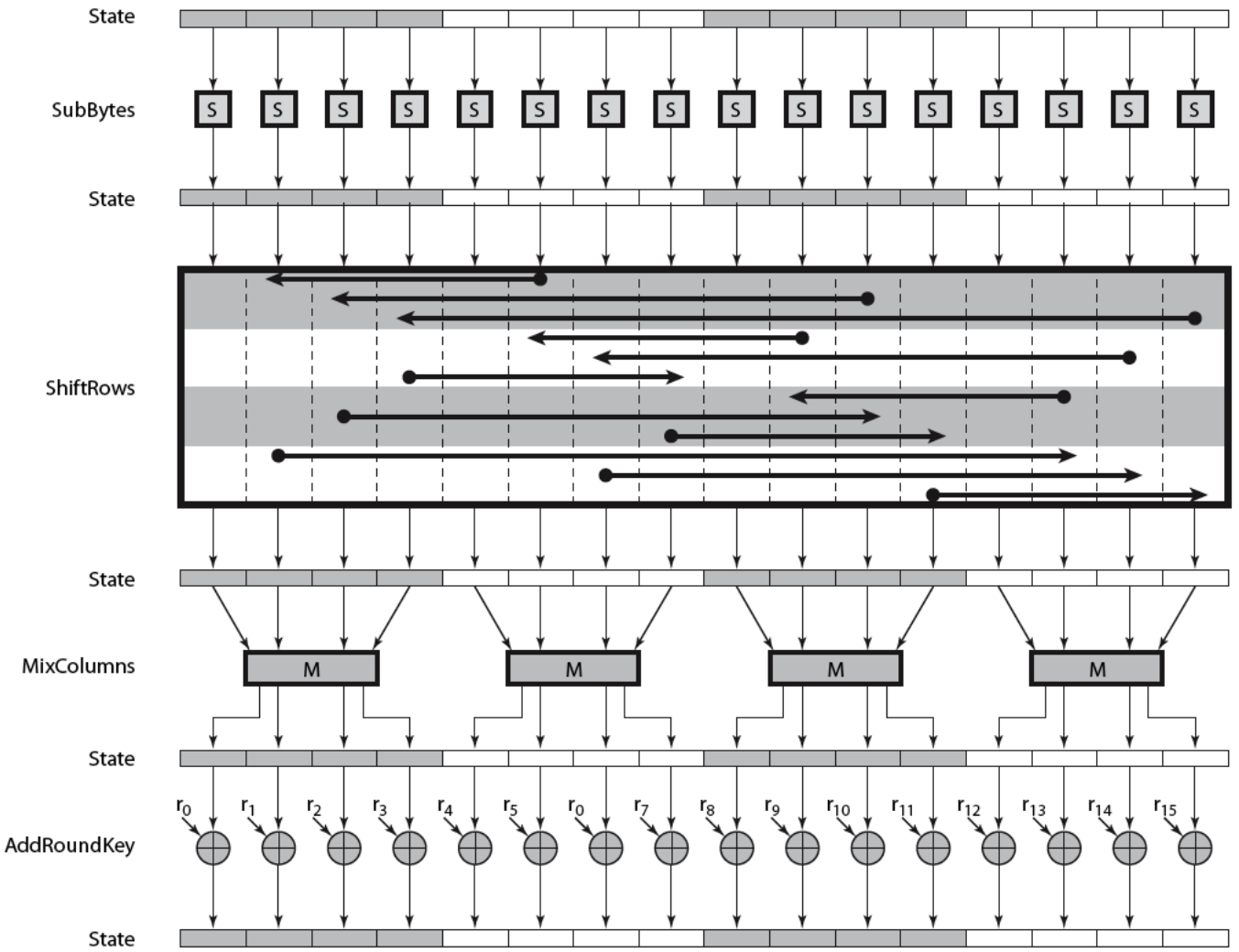
AddRoundKey

- XOR state with 128-bits of the round key
- AddRoundKey proceeds one column at a time.
  - adds a round key word with each state column matrix
  - the operation is matrix addition
- Inverse for decryption identical
  - since XOR own inverse, with reversed keys
- Designed to be as simple as possible

AddRoundKey Scheme



AES Round


State
SubBytes
S
State
ShiftRows
State
MixColumns
M
State
AddRoundKey
r_0 r_1 r_2 r_3 r_4 r_5 r_0 r_7 r_8 r_9 r_{10} r_{11} r_{12} r_{13} r_{14} r_{15}
State

Topics

- Origin of AES
- Basic AES
- Inside Algorithm
- **Final Notes**

AES Security

- AES was designed after DES.
- Most of the known attacks on DES were already tested on AES.
- Brute-Force Attack
  - AES is definitely more secure than DES due to the larger-size key.

Implementation Aspects

- The algorithms used in AES are so simple that they can be easily implemented using cheap processors and a minimum amount of memory.
- Very efficient
- Implementation was a key factor in its selection as the AES cipher
- AES animation:
  - http://www.cs.bc.edu/~straubin/cs381-05/blockciphers/rijndael_ingles2004.swf

Modes of Operation

Topics

- **Overview of Modes of Operation**
- ECB, CBC, CFB, OFB, CTR
- Notes and Remarks on each modes

Modes of Operation

- Block ciphers encrypt fixed size blocks
  - eg. DES encrypts 64-bit blocks, with 56-bit key
- Need way to use in practise, given usually have arbitrary amount of information to encrypt
  - Partition message into separate block for ciphering
- A **mode of operation** describes the process of encrypting each of these blocks **under a single key**
- Some modes may use randomized addition input value

Quick History

1981

- Early modes of operation: **ECB, CBC, CFB, OFB**
  - DES Modes of operation

    http://www.itl.nist.gov/fipspubs/fip81.htm

2001

- Revised and including **CTR** mode and AES
  - Recommendation for Block Cipher Modes of Operation

    http://csrc.nist.gov/publications/nistpubs/800-38a/sp800-38a.pdf

2010

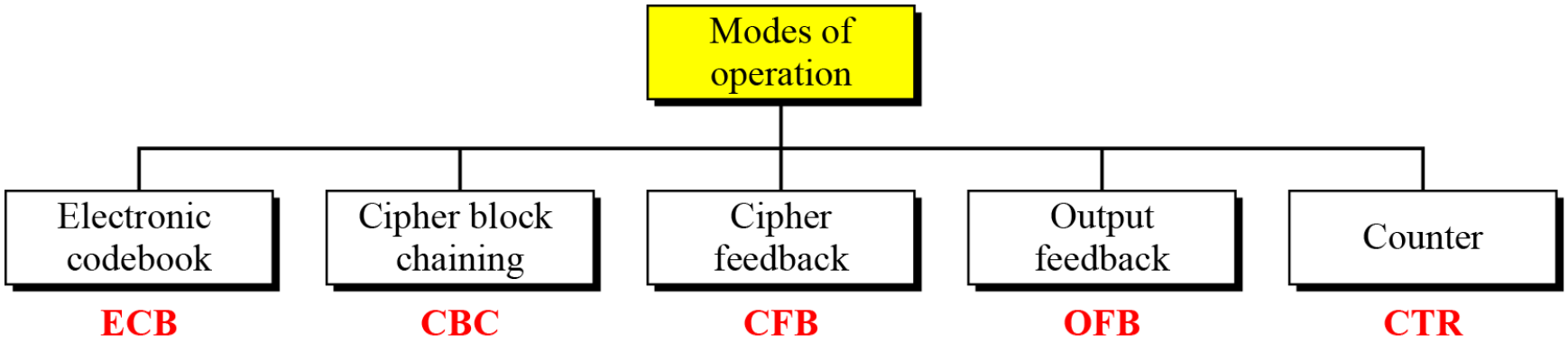
- New Mode : **XTS-AES**
  - Recommendation for Block Cipher Modes of Operation: The XTS-AES Mode for Confidentiality on Storage Devices

    http://csrc.nist.gov/publications/nistpubs/800-38E/nist-sp-800-38E.pdf

Modes of operation are nowadays defined by a number of national and internationally recognized standards bodies such as ISO, IEEE, ANSI and IETF. The most influential source is the US NIST

Modes of Operation Taxonomy

- Current well-known modes of operation



Moe Technical Notes

- Initialize Vector (IV)
  - a block of bits to randomize the encryption and hence to produce distinct ciphertext
- Nonce : Number (used) Once
  - Random of psuedorandom number to ensure that past communications can not be reused in replay attacks
  - Some also refer to initialize vector as nonce
- Padding
  - final block may require a padding to fit a block size
  - Method
    - Add null Bytes
    - Add 0x80 and many 0x00
    - Add the *n* bytes with value *n*

Electronic Codebook Book (ECB)

- Message is broken into independent blocks which are encrypted
- Each block is a value which is substituted, like a codebook, hence name
- Each block is encoded independently of the other blocks

$$C_i = E_K (P_i)$$

- Uses: secure transmission of single values

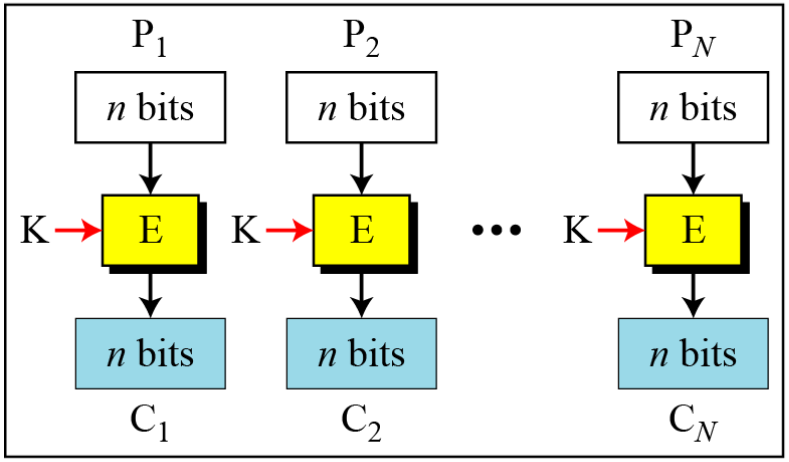
Topics

- Overview of Modes of Operation
- **EBC, CBC, CFB, OFB, CTR**
- Notes and Remarks on each modes

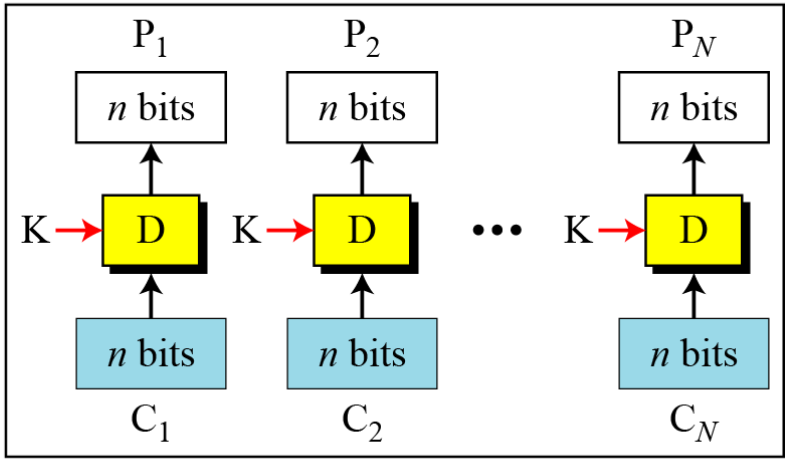
ECB Scheme

Encryption: $C_i = E_K (P_i)$ Decryption: $P_i = D_K (C_i)$

E: Encryption D: Decryption
P_i: Plaintext block i C_i: Ciphertext block i
K: Secret key



Encryption



Decryption

Remarks on ECB

- Strength: it's simple.
- Weakness:
  - Repetitive information contained in the plaintext may show in the ciphertext, if aligned with blocks.
  - If the same message is encrypted (with the same key) and sent twice, their ciphertext are the same.
- Typical application:
  - secure transmission of short pieces of information (e.g. a temporary encryption key)

Cipher Block Chaining (CBC)

- Solve security deficiencies in ECB
  - Repeated same plaintext block result different ciphertext block
- Each previous cipher blocks is chained to be input with current plaintext block, hence name
- Use Initial Vector (IV) to start process

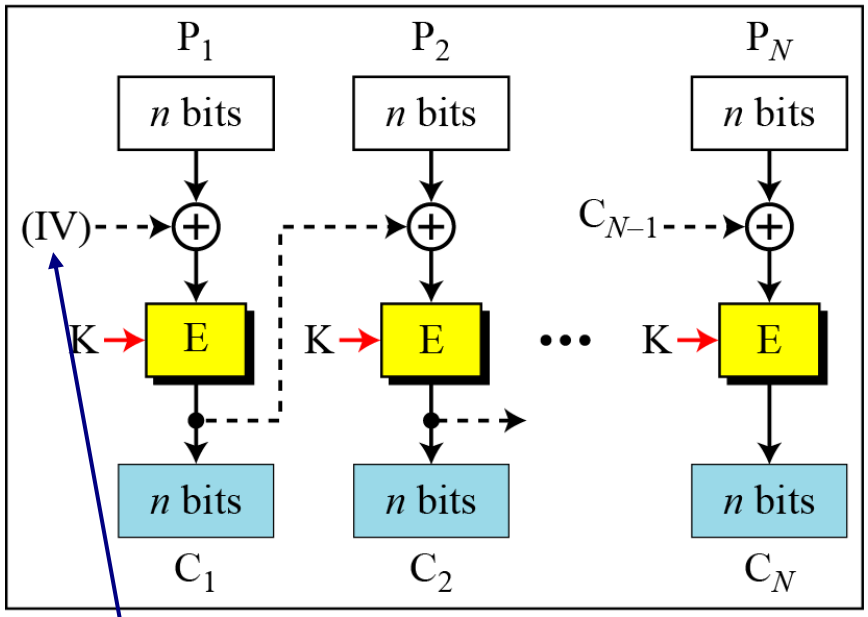
$$C_i = E_K (P_i \text{ XOR } C_{i-1})$$
$$C_0 = IV$$

- Uses: bulk data encryption, authentication

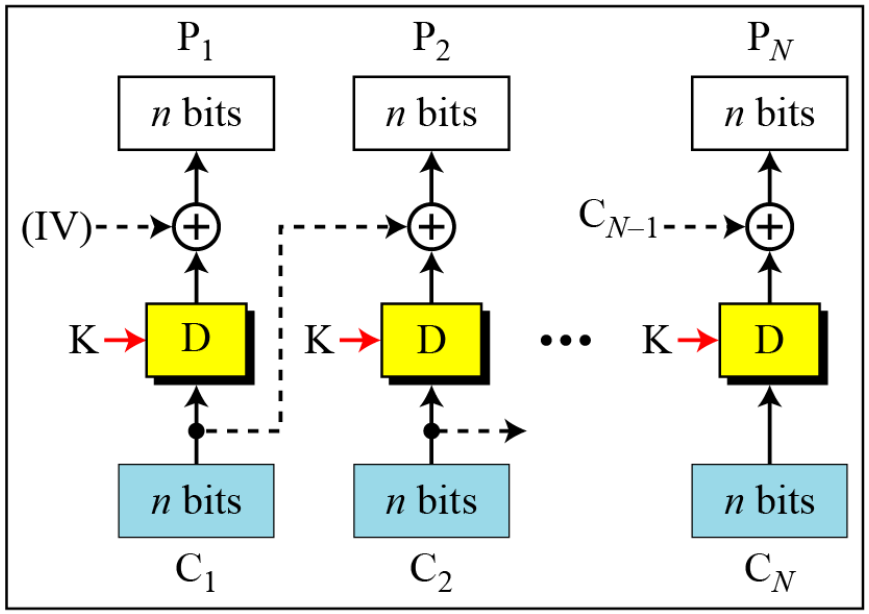
CBC scheme

E: Encryption
P_i: Plaintext block i
K: Secret key

D : Decryption
C_i : Ciphertext block i
IV: Initial vector (C_0)



Encryption



Decryption

Encryption:

$C_0 = IV$

$C_i = E_K (P_i \oplus C_{i-1})$

Decryption:

$C_0 = IV$

$P_i = D_K (C_i) \oplus C_{i-1}$

Remarks on CBC

- The encryption of a block depends on the current and **all** blocks before it.
- So, repeated plaintext blocks are encrypted differently.
- Initialization Vector (IV)
  - May sent encrypted in ECB mode before the rest of ciphertext

Cipher FeedBack (CFB)

- Use Initial Vector to start process
-
- Encrypt previous ciphertext , then combined with the plaintext block using X-OR to produce the current ciphertext
- Cipher is fed back (hence name) to concatenate with the rest of IV
- Plaintext is treated as a stream of bits
  - Any number of bit (1, 8 or 64 or whatever) to be feed back (denoted CFB-1, CFB-8, CFB-64)
- Relation between plaintext and ciphertext

$$C_i = P_i \text{ XOR } \text{SelectLeft}(E_K(\text{ShiftLeft}(C_{i-1})))$$
$$C_0 = IV$$

- Uses: stream data encryption, authentication

CFB Scheme

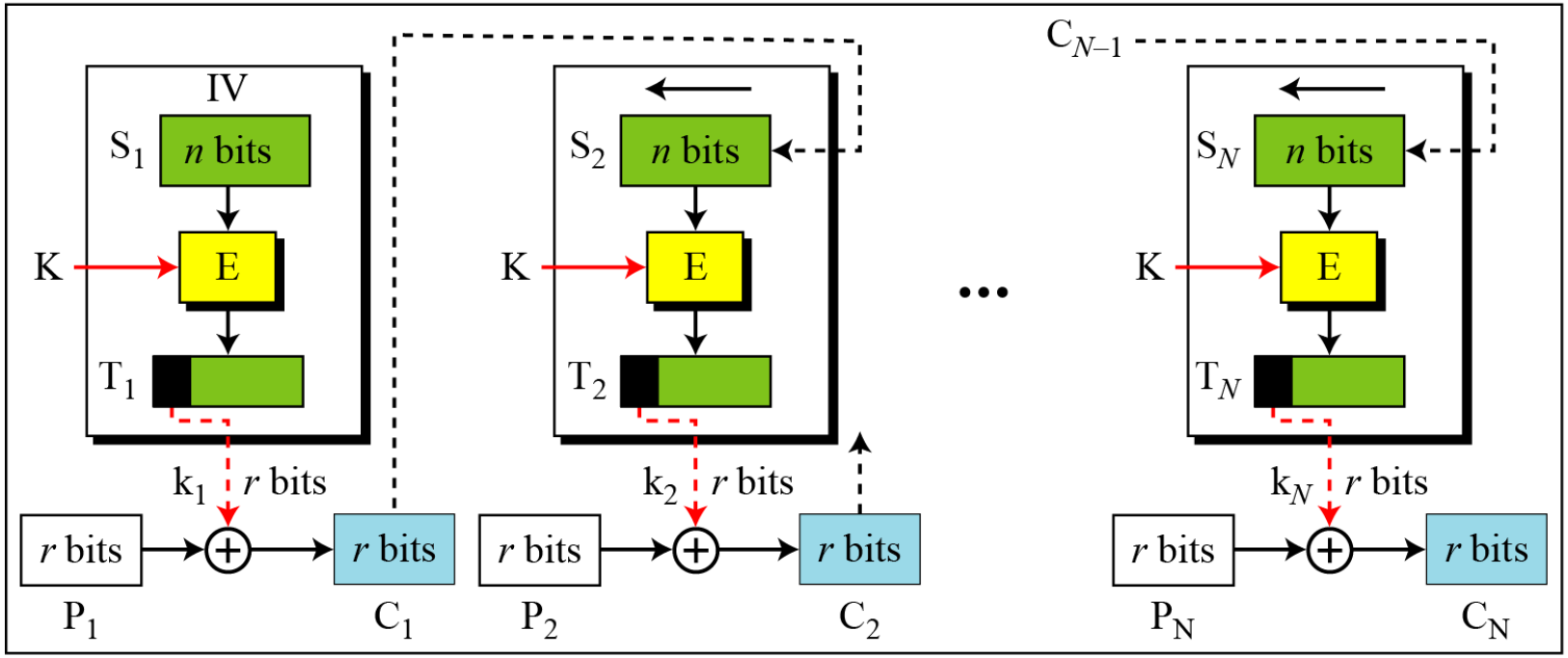
Encryption: $C_i = P_i \oplus \text{SelectLeft}_r \{E_K [\text{ShiftLeft}_r (S_{i-1}) \mid C_{i-1})]\}$

Decryption: $P_i = C_i \oplus \text{SelectLeft}_r \{E_K [\text{ShiftLeft}_r (S_{i-1}) \mid C_{i-1})]\}$

E : Encryption
P_i: Plaintext block i
K: Secret key

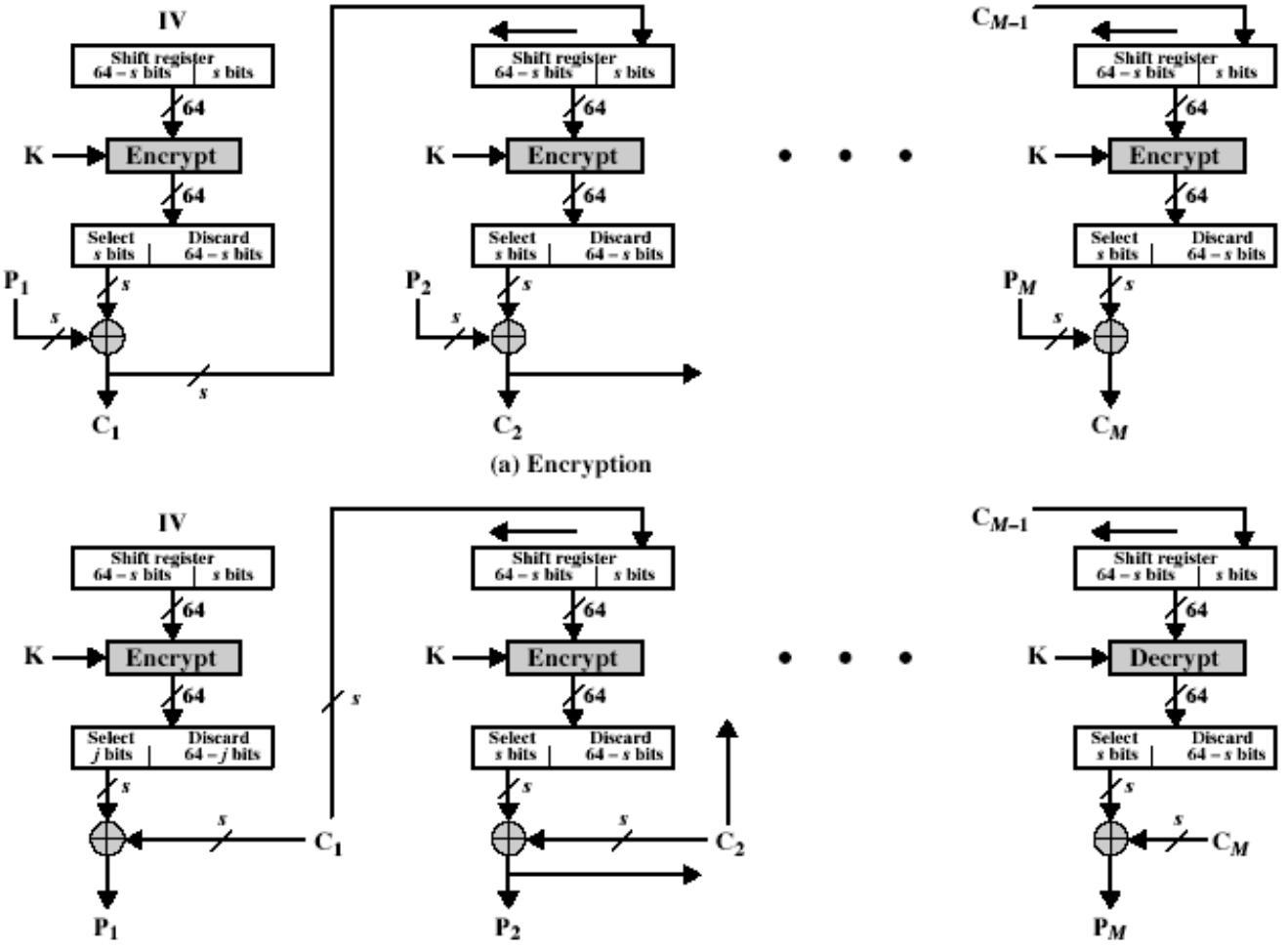
D : Decryption
C_i: Ciphertext block i
IV: Initial vector (S_1)

S_i: Shift register
T_i: Temporary register



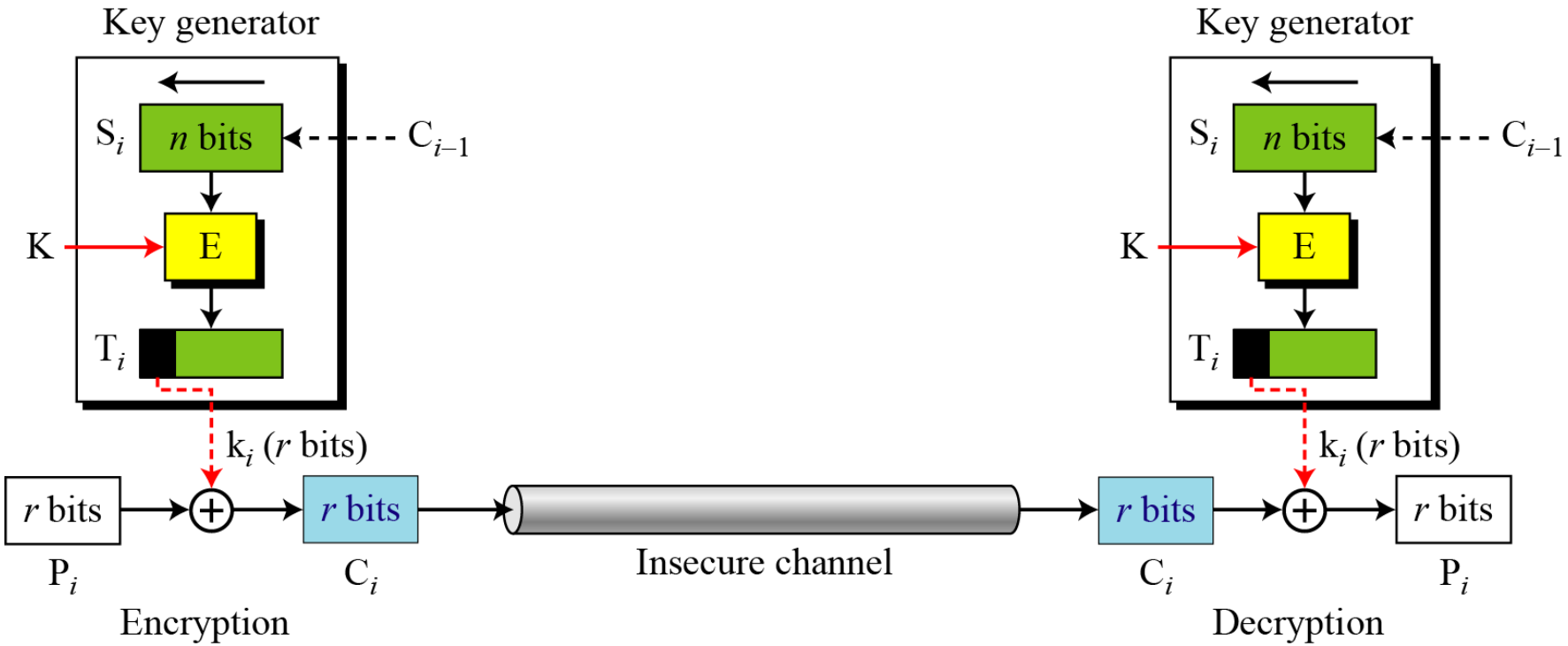
Encryption

CFB Encryption/Decryption



CFB as a Stream Cipher

- In CFB mode, encipherment and decipherment use the encryption function of the underlying block cipher.



Remark on CFB

- The block cipher is used as a stream cipher.
  - enable to encrypt any number of bits e.g. single bits or single characters (bytes)
  - S=1 : bit stream cipher
  - S=8 : character stream cipher)

- A ciphertext segment depends on the current and all preceding plaintext segments.

- A corrupted ciphertext segment during transmission will affect the current and next several plaintext segments.

Output FeedBack (OFB)

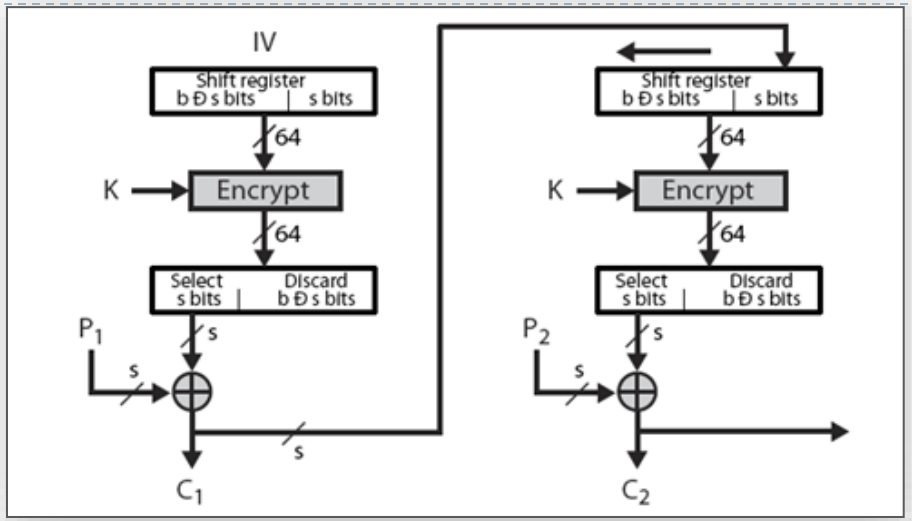
- Very similar to CFB
- But output of the encryption function output of cipher is fed back (hence name), instead of ciphertext
- Feedback is independent of message
- Relation between plaintext and ciphertext

$$C_i = P_i \text{ XOR } O_i$$
$$O_i = E_K(O_{i-1})$$
$$O_0 = IV$$

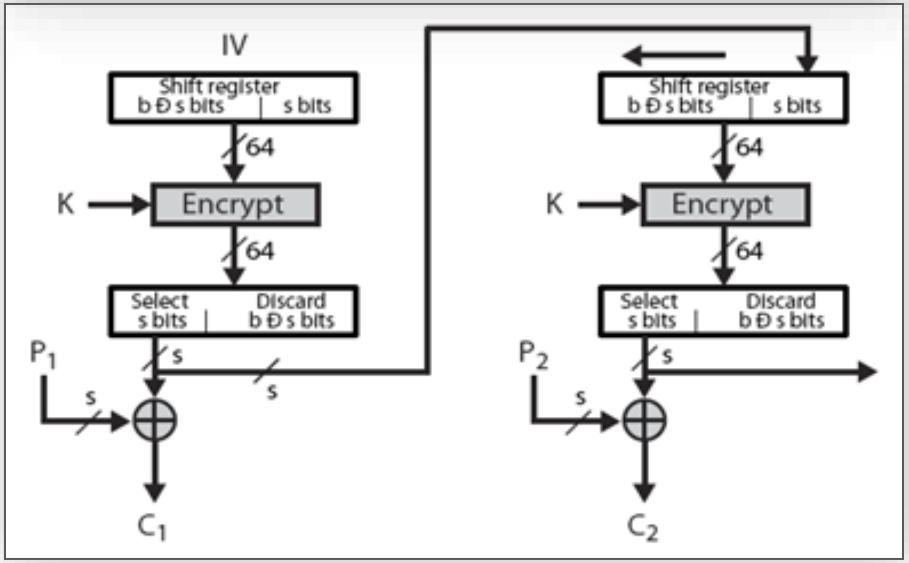
- Uses: stream encryption over noisy channels

CFB V.S. OFB

Cipher Feedback



Output Feedback

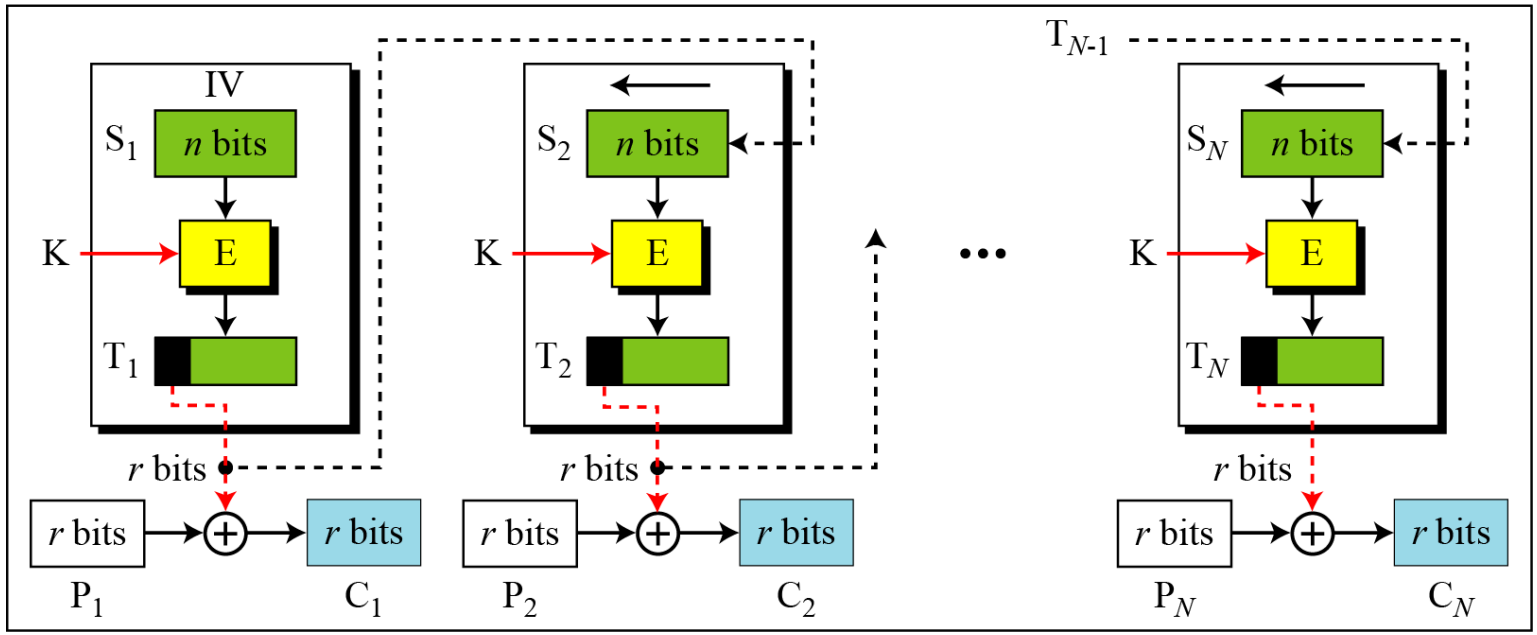


OFB Scheme

E : Encryption
P_i: Plaintext block i
K: Secret key

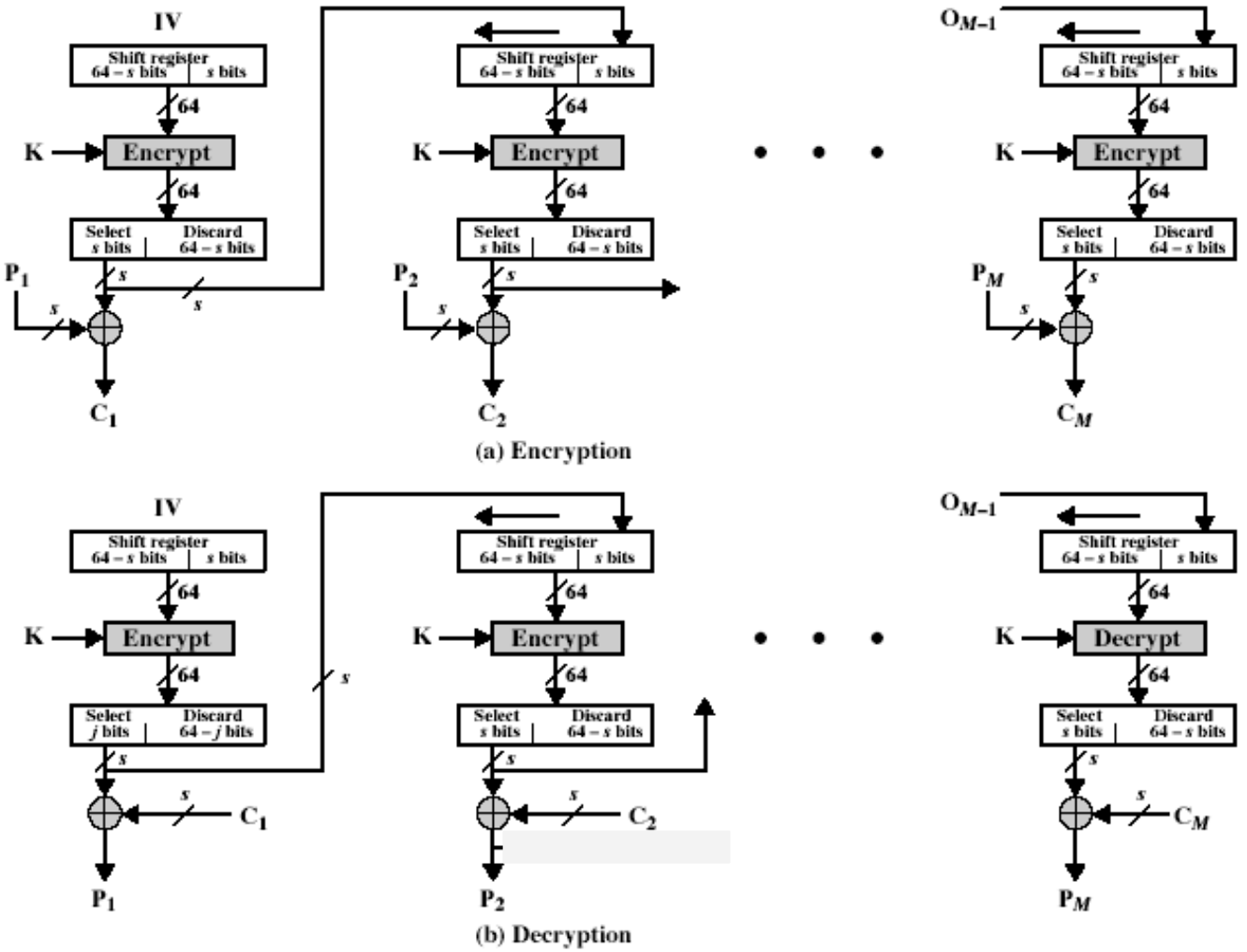
D : Decryption
C_i: Ciphertext block i
IV: Initial vector (S_1)

S_i: Shift register
T_i: Temporary register



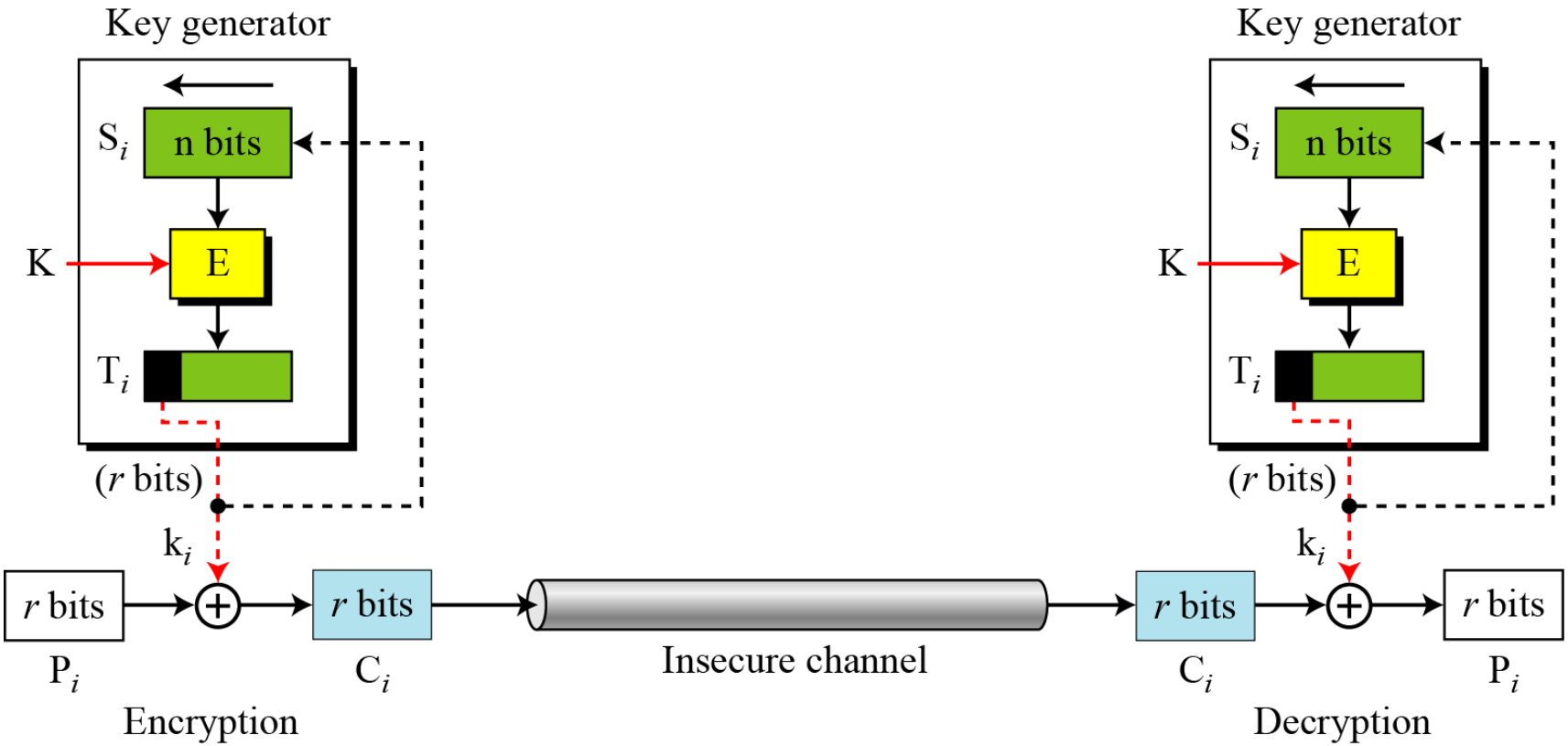
Encryption

OFB Encryption and Decryption



OFB as a Stream Cipher

- In OFB mode, encipherment and decipherment use the encryption function of the underlying block cipher.



Remarks on OFB

- Each bit in the ciphertext is independent of the previous bit or bits. This avoids error propagation
- Pre-compute of forward cipher is possible
- Security issue
  - when j^{th} plaintext is known, the j^{th} output of the forward cipher function will be known
  - Easily cover j^{th} plaintext block of other message with the same IV
- Require that the IV is a nonce

Counter (CTR)

- Encrypts counter value with the key rather than any feedback value (no feedback)
- Counter for each plaintext will be different
  - can be any function which produces a sequence which is guaranteed not to repeat for a long time
- Relation

$$C_i = P_i \text{ XOR } O_i$$
$$O_i = E_K(i)$$

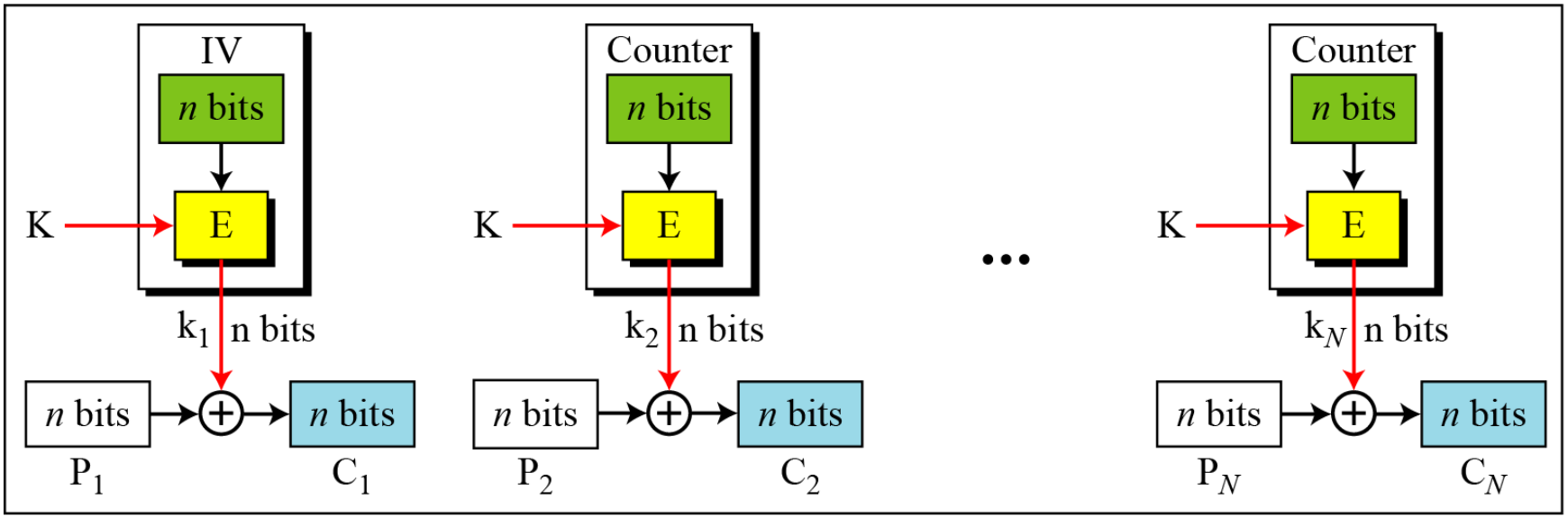
- Uses: high-speed network encryptions

CTR Scheme

E : Encryption
P_i : Plaintext block *i*
K : Secret key

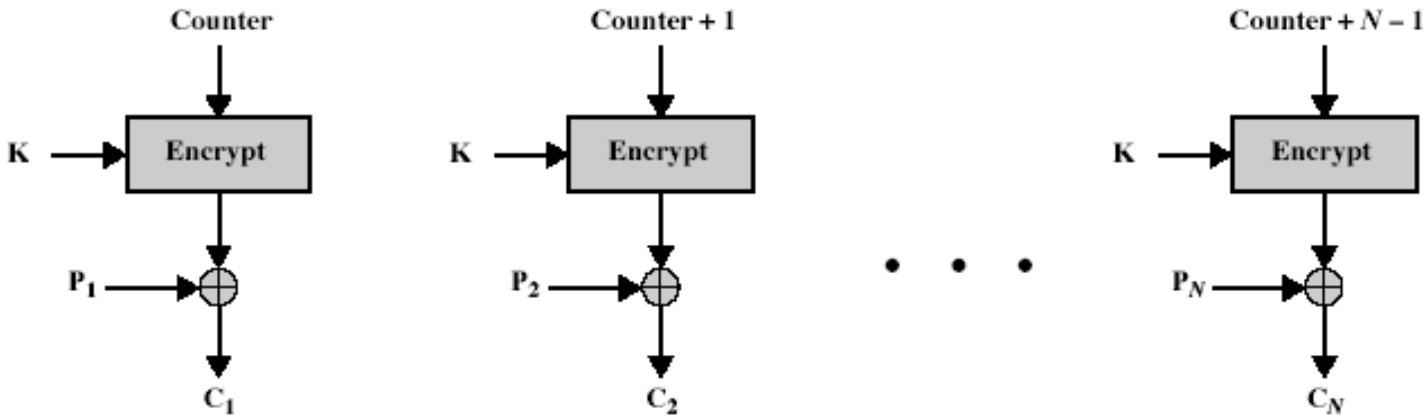
IV: Initialization vector
C_i : Ciphertext block *i*
k_i : Encryption key *i*

The counter is incremented for each block.

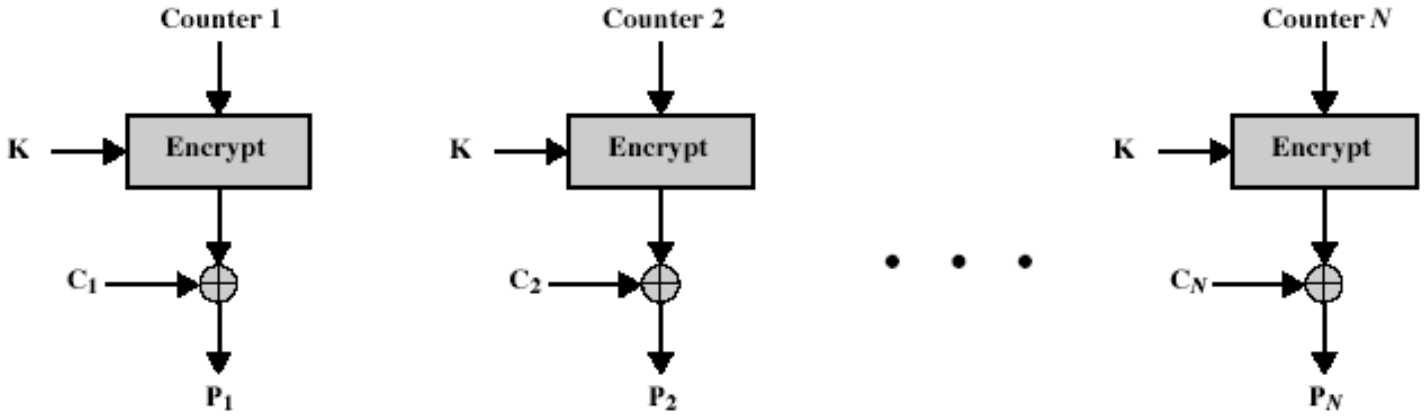


Encryption

CTR Encryption and Decryption

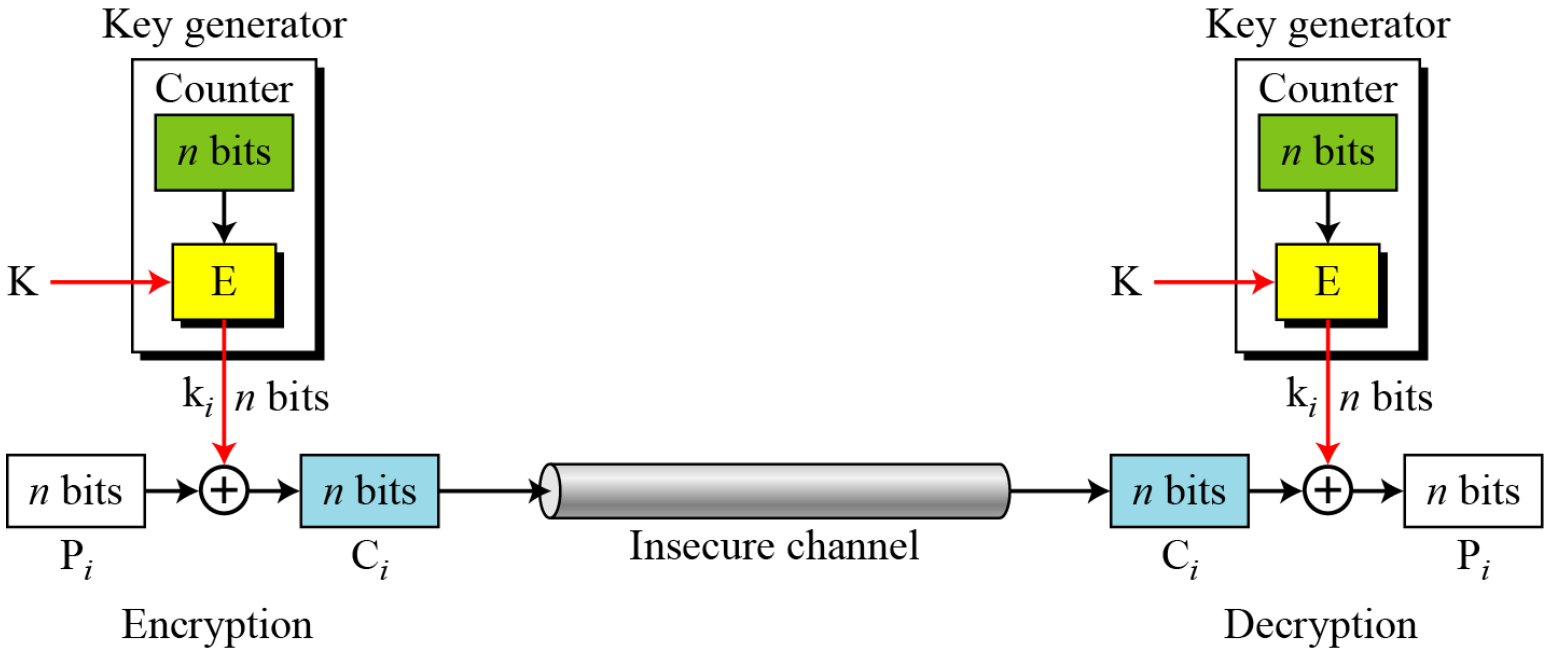


(a) Encryption



(b) Decryption

OFB as a Stream Cipher



Remark on CTR

- Strengthes:
  - Needs only the encryption algorithm
  - Random access to encrypted data blocks
    - blocks can be processed (encrypted or decrypted) in parallel
  - Simple; fast encryption/decryption

- Counter must be
  - Must be unknown and unpredictable
  - pseudo-randomness in the key stream is a goal

Topics

- Overview of Modes of Operation
- EBC, CBC, CFB, OFB, CTR
- **Notes and Remarks on each modes**

Remark on each mode

- Basically two types:
  - block cipher
  - stream cipher
- CBC is an excellent block cipher
- CFB, OFB, and CTR are stream ciphers
- CTR is faster because simpler and it allows parallel processing

Modes and IV

- An IV has different security requirements than a key
- Generally, an IV will not be reused under the same key
- CBC and CFB
  - reusing an IV leaks some information about the first block of plaintext, and about any common prefix shared by the two messages
- OFB and CTR
  - reusing an IV completely destroys security

CBC and CTR comparison

CBC	CTR
Padding needed	No padding
No parallel processing	Parallel processing
Separate encryption and decryption functions	Encryption function alone is enough
Random IV or a nonce	Unique nonce
Nonce reuse leaks some information about initial plaintext block	Nonce reuse will leak information about the entire message

Comparison of Different Modes

Operation Mode	*Description*	*Type of Result*	*Data Unit Size*
ECB	Each n-bit block is encrypted independently with the same cipher key.	Block cipher	n
CBC	Same as ECB, but each block is first exclusive-ored with the previous ciphertext.	Block cipher	n
CFB	Each r-bit block is exclusive-ored with an r-bit key, which is part of previous cipher text	Stream cipher	$r \leq n$
OFB	Same as CFB, but the shift register is updated by the previous r-bit key.	Stream cipher	$r \leq n$
CTR	Same as OFB, but a counter is used instead of a shift register.	Stream cipher	n

Comparison of Modes

Mode	Description	Application
ECB	64-bit plaintext block encoded separately	Secure transmission of encryption key
CBC	64-bit plaintext blocks are XORed with preceding 64-bit ciphertext	Commonly used method. Used for authentication
CFB	s bits are processed at a time and used similar to CBC	Primary stream cipher. Used for authentication

Comparison of Modes

Mode	Description	Application
OFB	Similar to CFB except that the output is fed back	Stream cipher well suited for transmission over noisy channels
CTR	Key calculated using the nonce and the counter value. Counter is incremented for each block	General purpose block oriented transmission. Used for high-speed communications

Final Notes

- ECB, CBC, OFB, CFB, CTR, and XTS modes only provide confidentiality
- To ensure an encrypted message is not accidentally modified or maliciously tampered requires a separate Message Authentication Code (MAC)
- Several MAC schemes
  - HMAC, CMAC and GMAC
- But.. compositing a confidentiality mode with an authenticity mode could be difficult and error prone
- New modes combined confidentiality and data integrity into a single cryptographic primitive
  - CCM, GCM, CWC, EAX, IAPM and OCB

Q&A