Chapter 7

Block Cipher Operation

Multiple Encryption & DES

- clear a replacement for DES was needed
  - theoretical attacks that can break it
  - demonstrated exhaustive key search attacks
- AES is a new cipher alternative
- prior to this alternative was to use multiple encryption with DES implementations
- Triple-DES is the chosen form

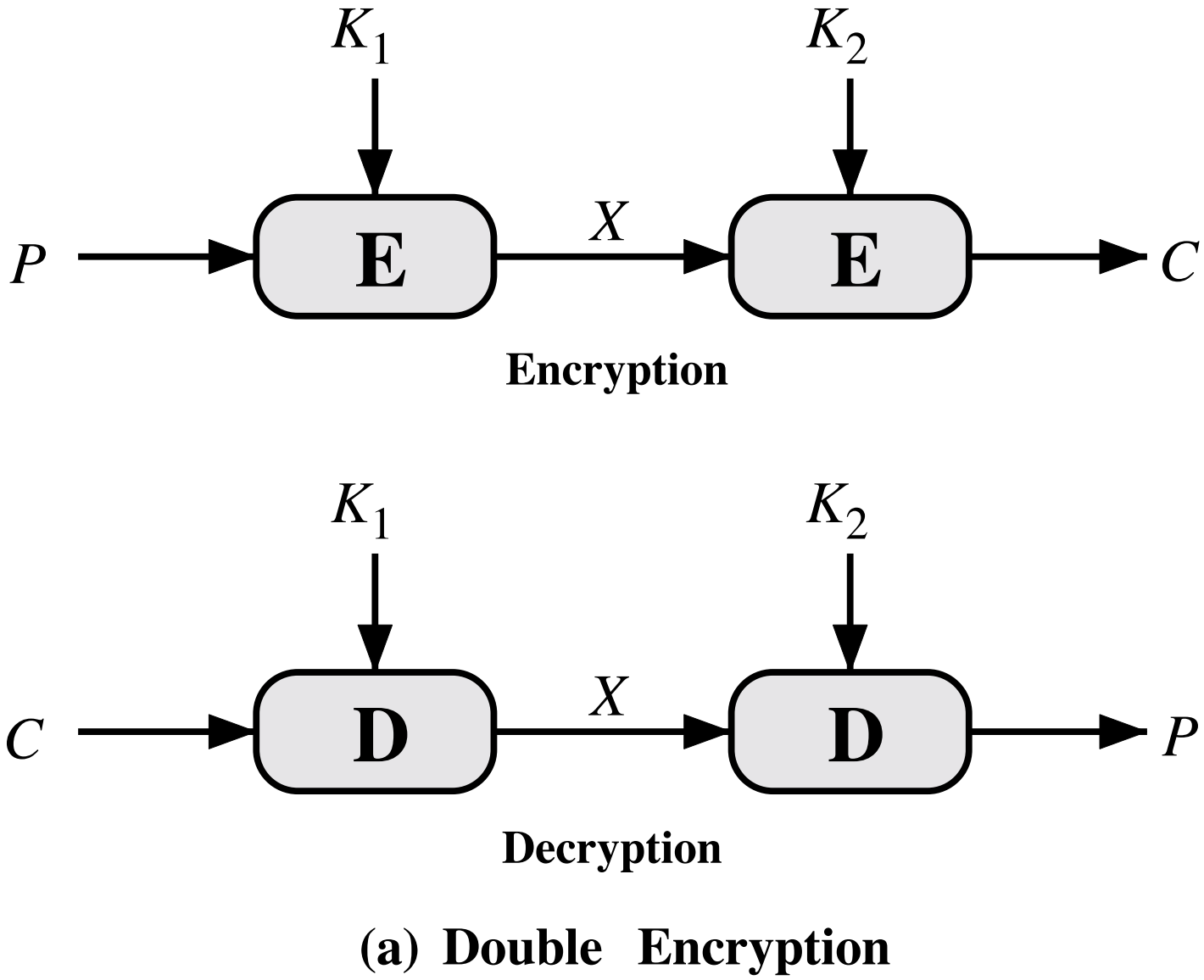


Figure 7.1 Multiple Encryption

Double-DES?

- could use 2 DES encrypts on each block
  - $C = E_{K2}(E_{K1}(P))$
- issue of reduction to single stage
- and have “meet-in-the-middle” attack
  - works whenever use a cipher twice
  - since $X = E_{K1}(P) = D_{K2}(C)$
  - attack by encrypting P with all keys and store
  - then decrypt C with keys and match X value
  - can show takes $O(2^{56})$ steps

Meet-in-the-Middle Attack

The use of double DES results in a mapping that is not equivalent to a single DES encryption

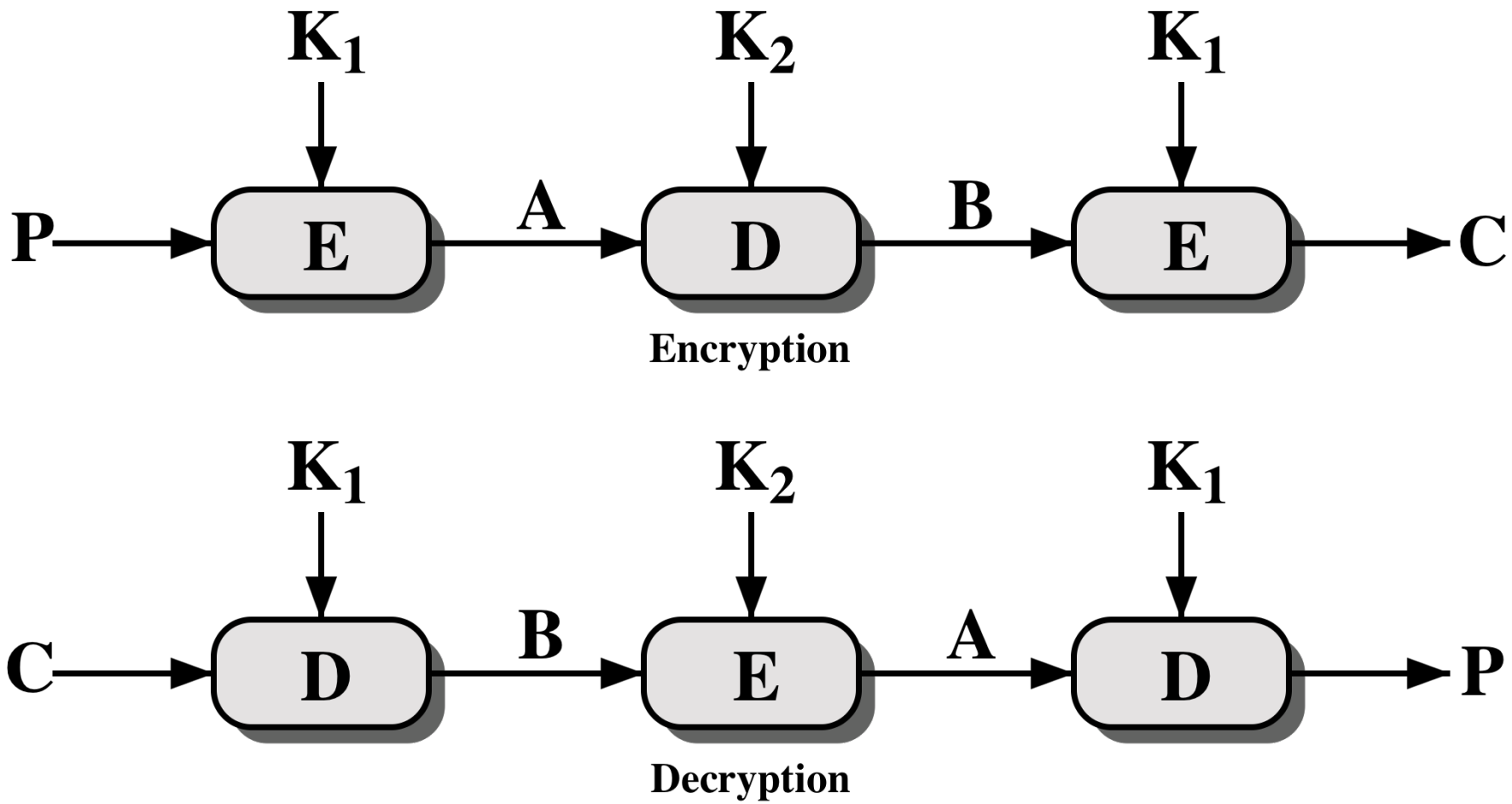
The meet-in-the-middle attack algorithm will attack this scheme and does not depend on any particular property of DES but will work against any block encryption cipher

Triple-DES with Two-Keys

- Obvious counter to the meet-in-the-middle attack is to use three stages of encryption with three different keys
  - This raises the cost of the meet-in-the-middle attack to 2^{112}, which is beyond what is practical
  - Has the drawback of requiring a key length of 56 x 3 = 168 bits, which may be somewhat unwieldy
  - As an alternative Tuchman proposed a triple encryption method that uses only two keys
- 3DES with two keys is a relatively popular alternative to DES and has been adopted for use in the key management standards ANSI X9.17 and ISO 8732

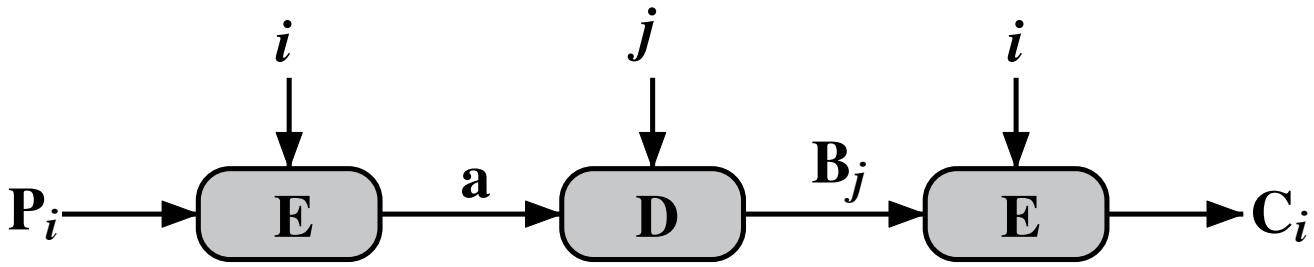
Triple-DES with Two-Keys

- hence can use 2 keys with E-D-E sequence
  - $C = E_{K1}(D_{K2}(E_{K1}(P)))$
  - encrypt & decrypt equivalent in security
  - if $K1=K2$ then can work with single DES
- standardized in ANSI X9.17 & ISO8732
- no current known practical attacks
  - several proposed impractical attacks might become basis of future attacks

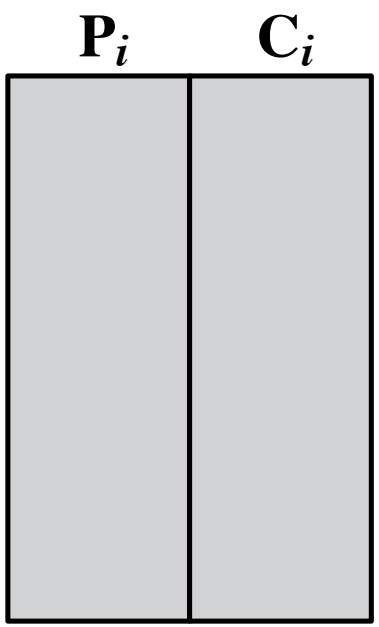


(b) Triple Encryption

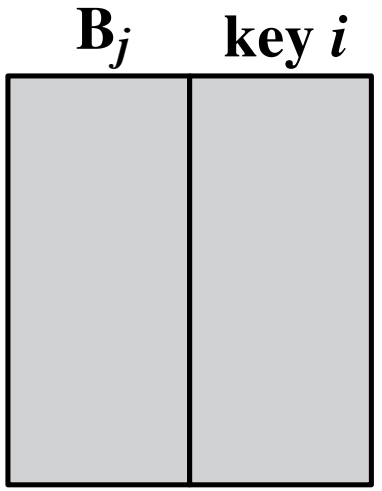
Figure 7.1 Multiple Encryption



(a) Two-key Triple Encryption with Candidate Pair of Keys



(b) Table of n known plaintext-ciphertext pairs, sorted on P



(c) Table of intermediate values and candidate keys

Figure 7.2 Known-Plaintext Attack on Triple DES

Triple DES with Three Keys

- Although are no practical attacks on two-key Triple-DES have some indications. Many researchers now feel that three-key 3DES is the preferred alternative

Three-key 3DES has an effective key length of 168 bits and is defined as:	• $C = E(K_3, D(K_2, E(K_1, P)))$
Backward compatibility with DES is provided by putting:	• $K_3 = K_2$ or $K_1 = K_2$

- A number of Internet-based applications have adopted three-key 3DES including PGP and S/MIME

Modes of Operation

- A technique for enhancing the effect of a cryptographic algorithm or adapting the algorithm for an application
- To apply a block cipher in a variety of applications, five *modes of operation* have been defined by NIST
  - The five modes are intended to cover a wide variety of applications of encryption for which a block cipher could be used
  - These modes are intended for use with any symmetric block cipher, including triple DES and AES

Table 7.1 Block Cipher Modes of Operation

Mode	Description	Typical Application
Electronic Codebook (ECB)	Each block of plaintext bits is encoded independently using the same key.	•Secure transmission of single values (e.g., an encryption key)
Cipher Block Chaining (CBC)	The input to the encryption algorithm is the XOR of the next block of plaintext and the preceding block of ciphertext.	•General-purpose block-oriented transmission •Authentication
Cipher Feedback (CFB)	Input is processed *s* bits at a time. Preceding ciphertext is used as input to the encryption algorithm to produce pseudorandom output, which is XORed with plaintext to produce next unit of ciphertext.	•General-purpose stream-oriented transmission •Authentication
Output Feedback (OFB)	Similar to CFB, except that the input to the encryption algorithm is the preceding encryption output, and full blocks are used.	•Stream-oriented transmission over noisy channel (e.g., satellite communication)
Counter (CTR)	Each block of plaintext is XORed with an encrypted counter. The counter is incremented for each subsequent block.	•General-purpose block-oriented transmission •Useful for high-speed requirements

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

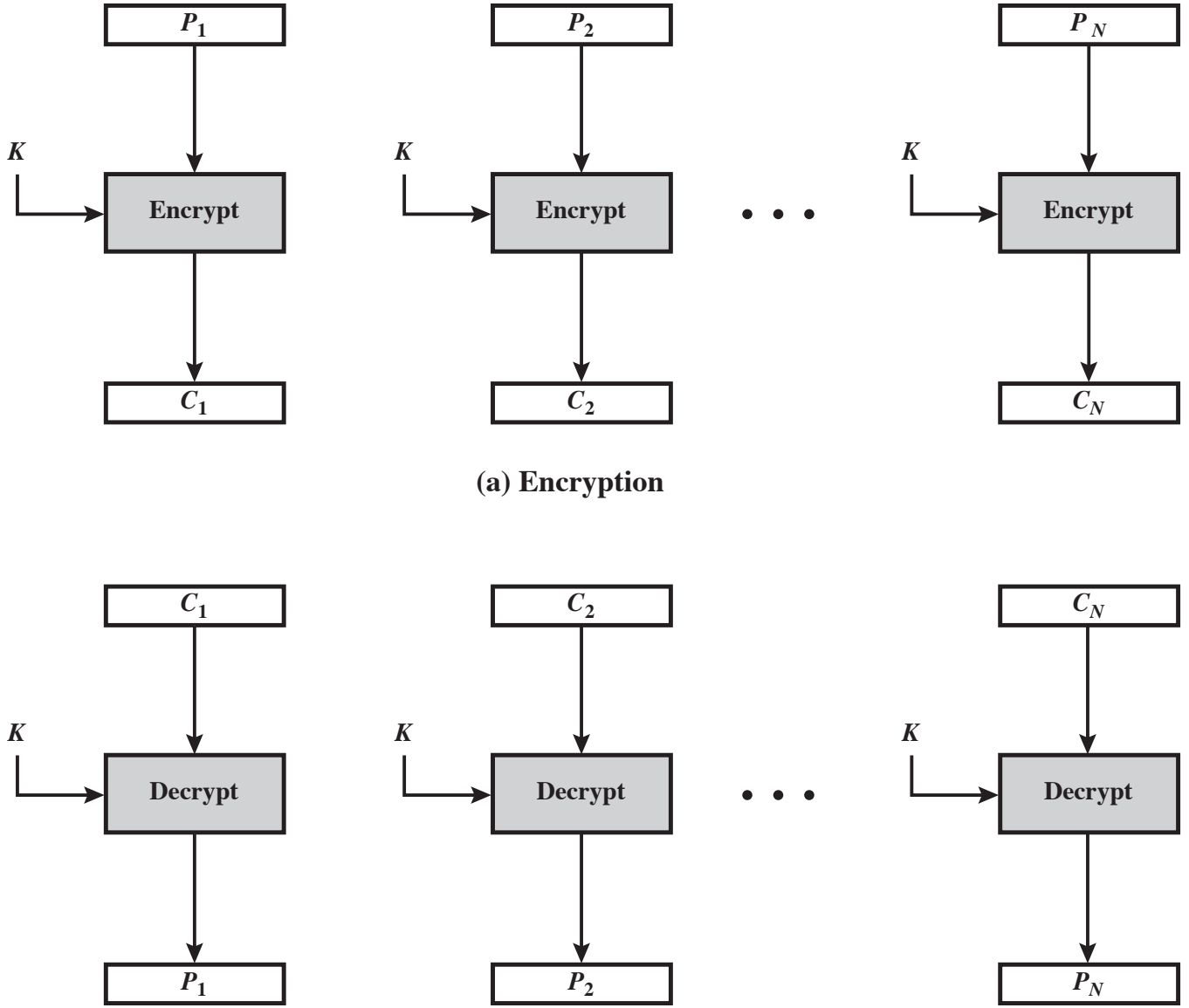


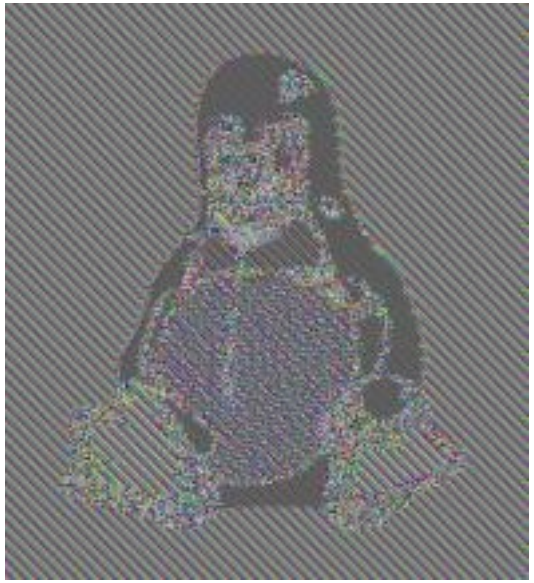
Figure 7.3 Electronic Codebook (ECB) Mode

Advantages and Limitations of ECB

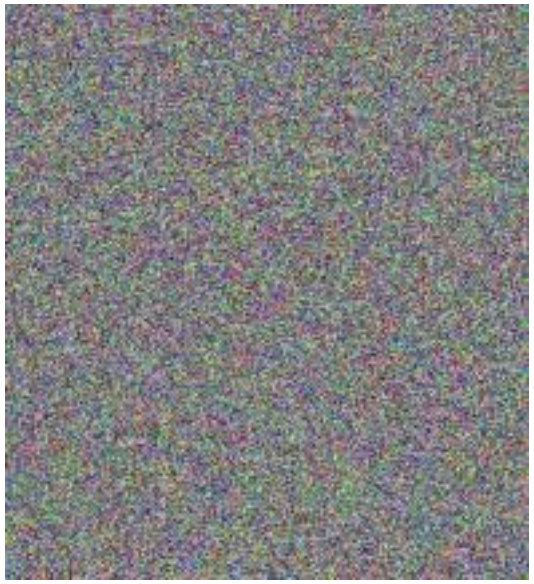
- message repetitions may show in ciphertext
  - if aligned with message block
  - particularly with data such graphics
  - or with messages that change very little, which become a code-book analysis problem
- weakness is due to the encrypted message blocks being independent
- main use is sending a few blocks of data

ECB Demo on an Image

Original image



Encrypted using ECB mode



Modes other than ECB result in pseudo-randomness

Criteria and properties for evaluating and constructing block cipher modes of operation that are superior to ECB:

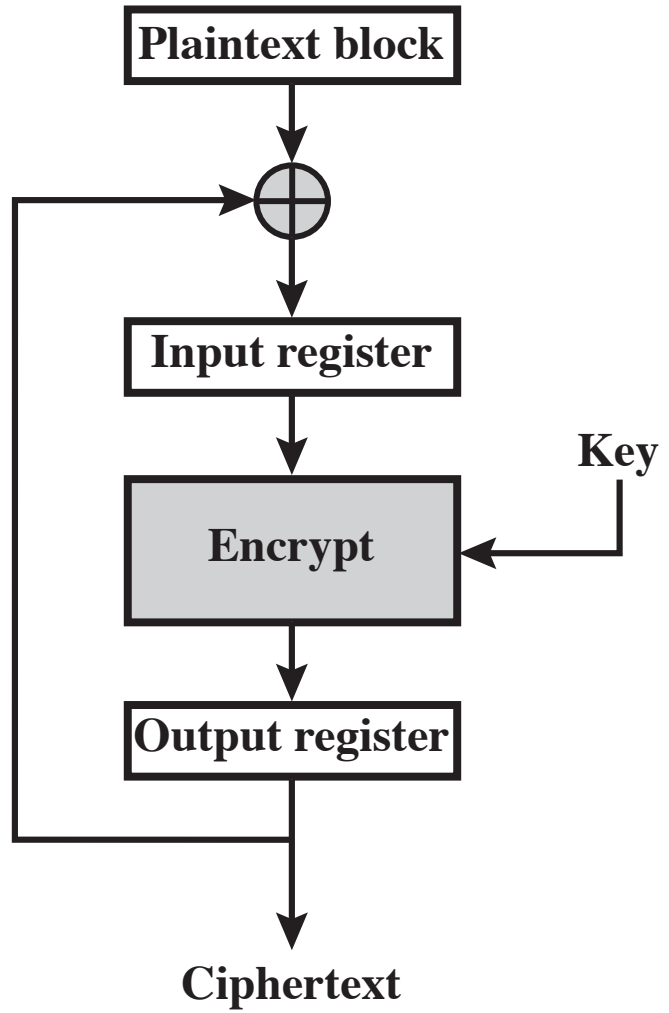
- Overhead
- Error recovery
- Error propagation
- Diffusion
- Security

Cipher Block Chaining (CBC)

- message is broken into blocks
- linked together in encryption operation
- each previous cipher blocks is chained with current plaintext block, hence name
- use Initial Vector (IV) to start process

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

- uses: bulk data encryption, authentication



(a) Cipher block chaining (CBC) mode

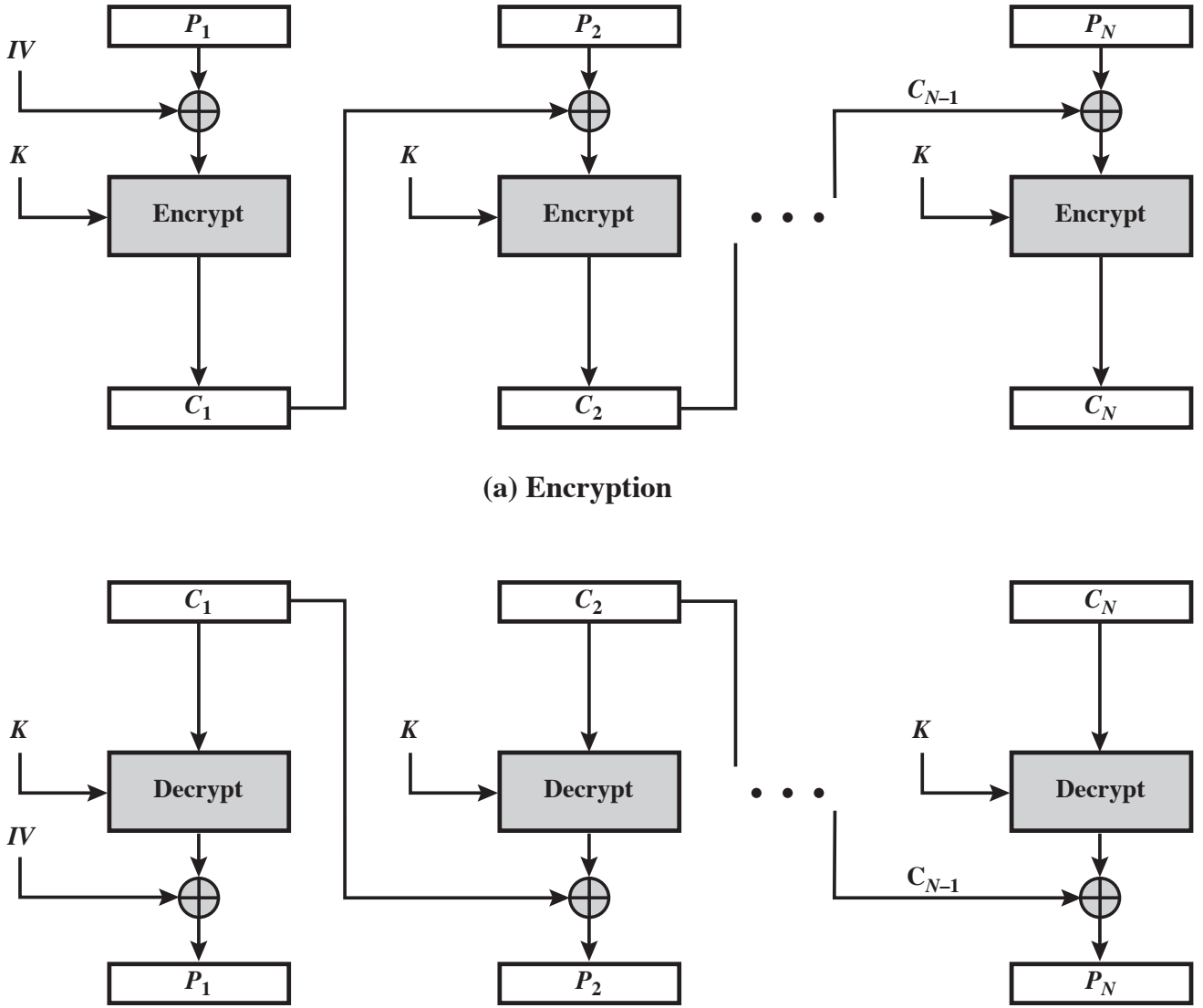


Figure 7.4 Cipher Block Chaining (CBC) Mode

Message Padding

- at end of message must handle a possible last short block
  - which is not as large as blocksize of cipher
  - pad either with known non-data value (eg nulls)
  - or pad last block along with count of pad size
    - eg. [b1 b2 b3 0 0 0 0 5]
    - means have 3 data bytes, then 5 bytes pad+count
  - this may require an extra entire block over those in message
- there are other, more esoteric modes, which avoid the need for an extra block

Advantages and Limitations of CBC

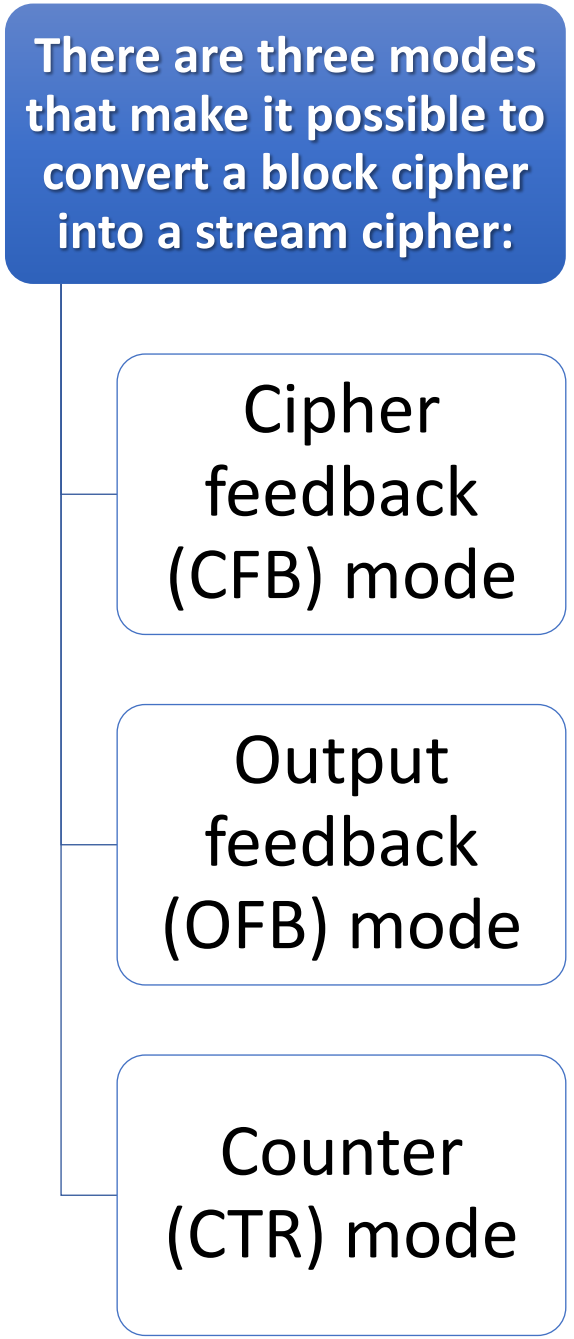
- a ciphertext block depends on **all** blocks before it
- any change to a block affects all following ciphertext blocks
- need **Initialization Vector** (IV)
  - which must be known to sender & receiver
  - if sent in clear, attacker can change bits of first block, and change IV to compensate
  - hence IV must either be a fixed value (as in EFTPOS)
  - or must be sent encrypted in ECB mode before rest of message

Stream Modes of Operation

- block modes encrypt entire block
- may need to operate on smaller units
  - real time data
- convert block cipher into stream cipher
  - cipher feedback (CFB) mode
  - output feedback (OFB) mode
  - counter (CTR) mode
- use block cipher as some form of **pseudo-random number** generator

Cipher Feedback Mode

- For AES, DES, or any block cipher, encryption is performed on a block of *b* bits
  - In the case of DES *b* = 64
  - In the case of AES *b* = 128

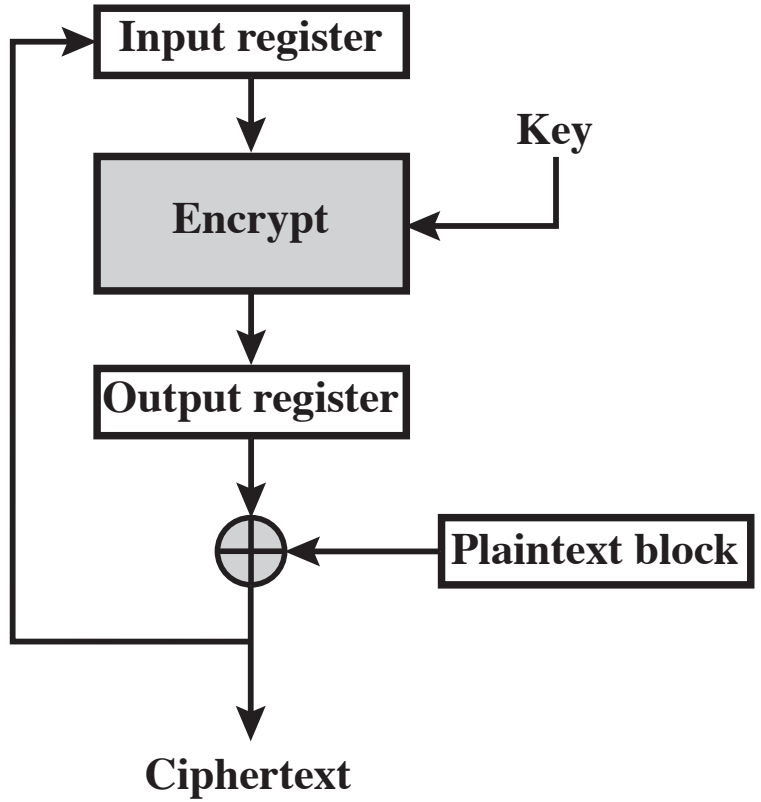


Cipher FeedBack (CFB)

- message is treated as a stream of bits
- added to the output of the block cipher
- result is feed back for next stage (hence name)
- standard allows any number of bit (1,8, 64 or 128 etc) to be feed back
  - denoted CFB-1, CFB-8, CFB-64, CFB-128 etc
- most efficient to use all bits in block (64 or 128)

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

- uses: stream data encryption, authentication



(b) Cipher feedback (CFB) mode

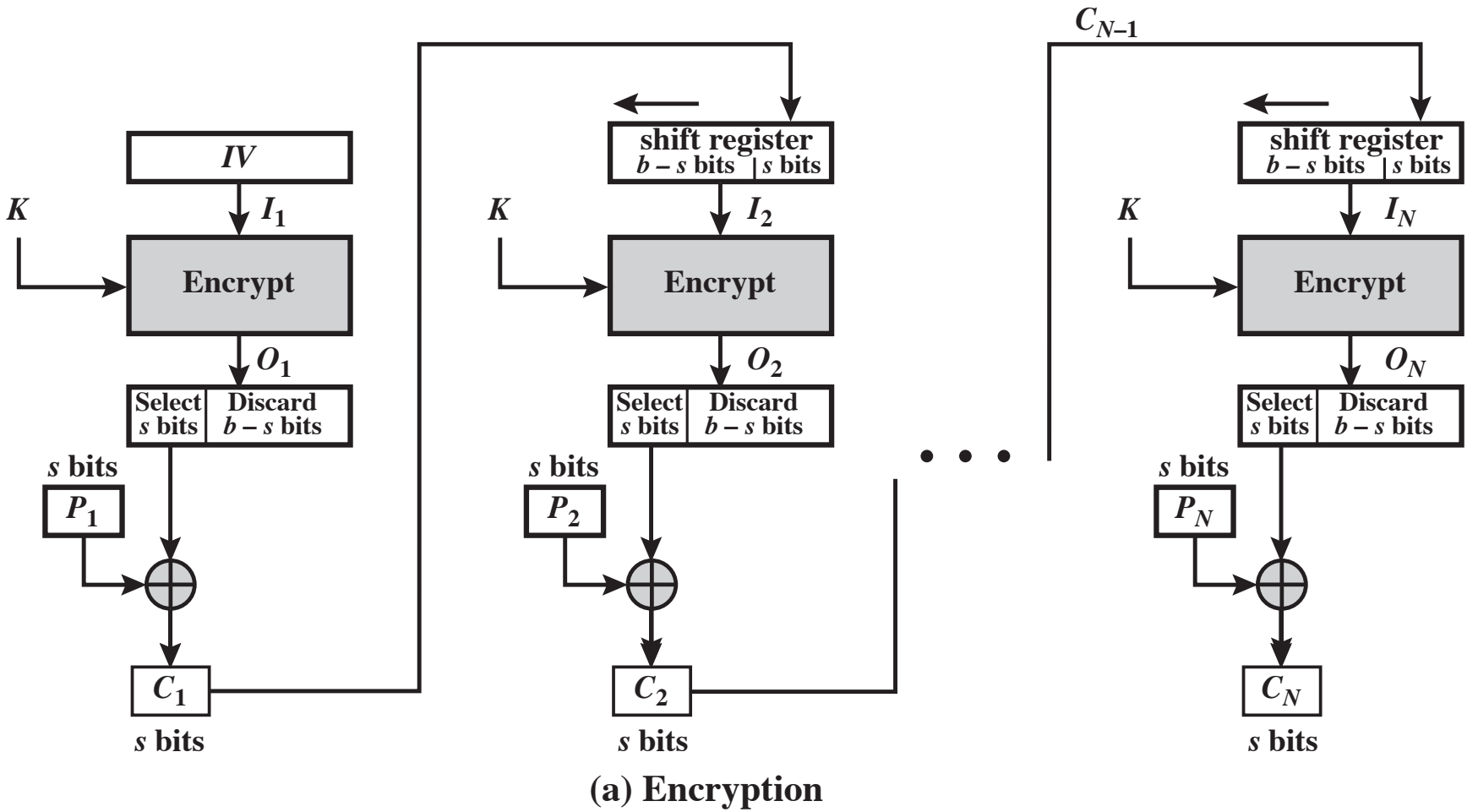


Figure 7.5 *s*-bit Cipher Feedback (CFB) Mode

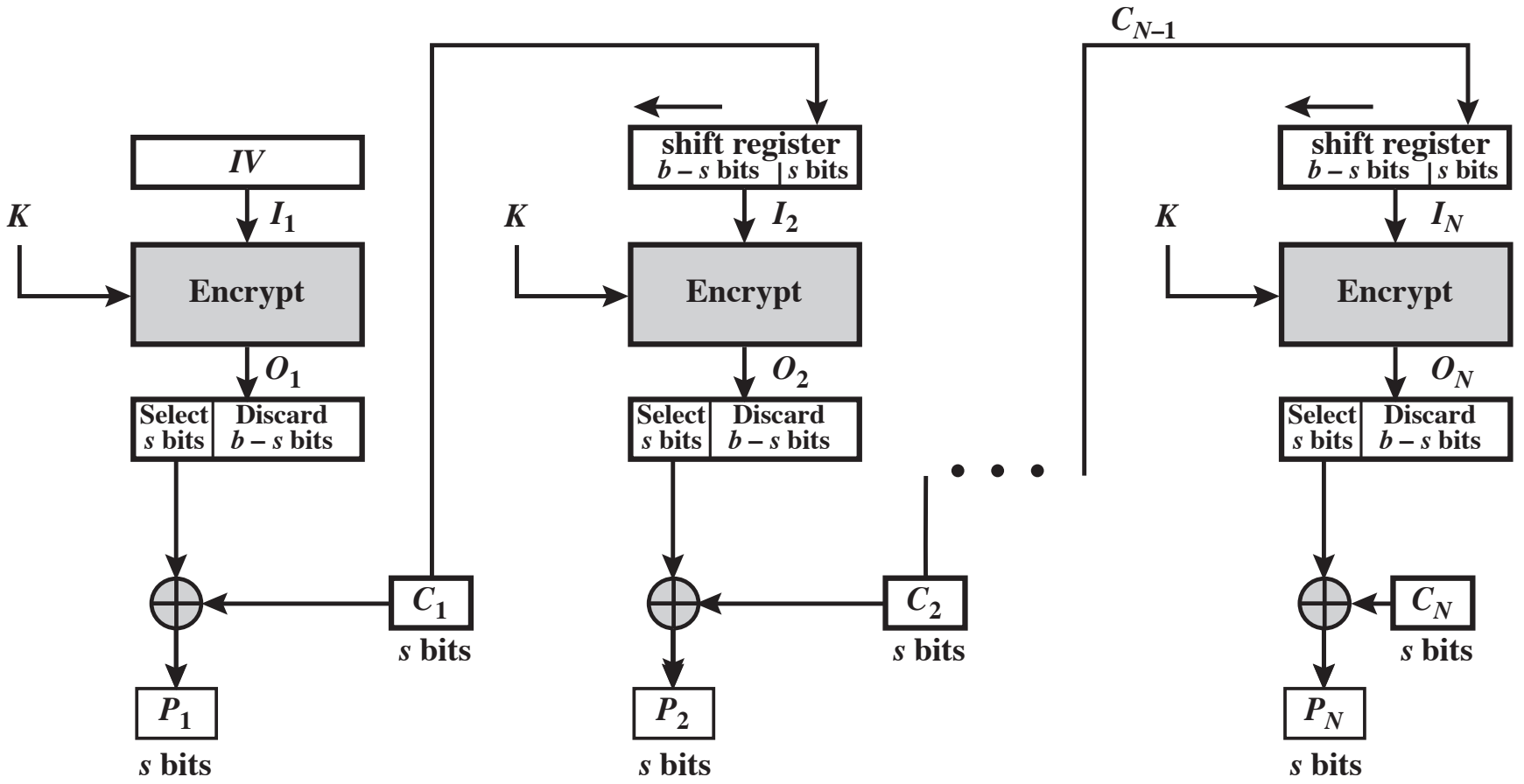


Figure 7.5 *s*-bit Cipher Feedback (CFB) Mode

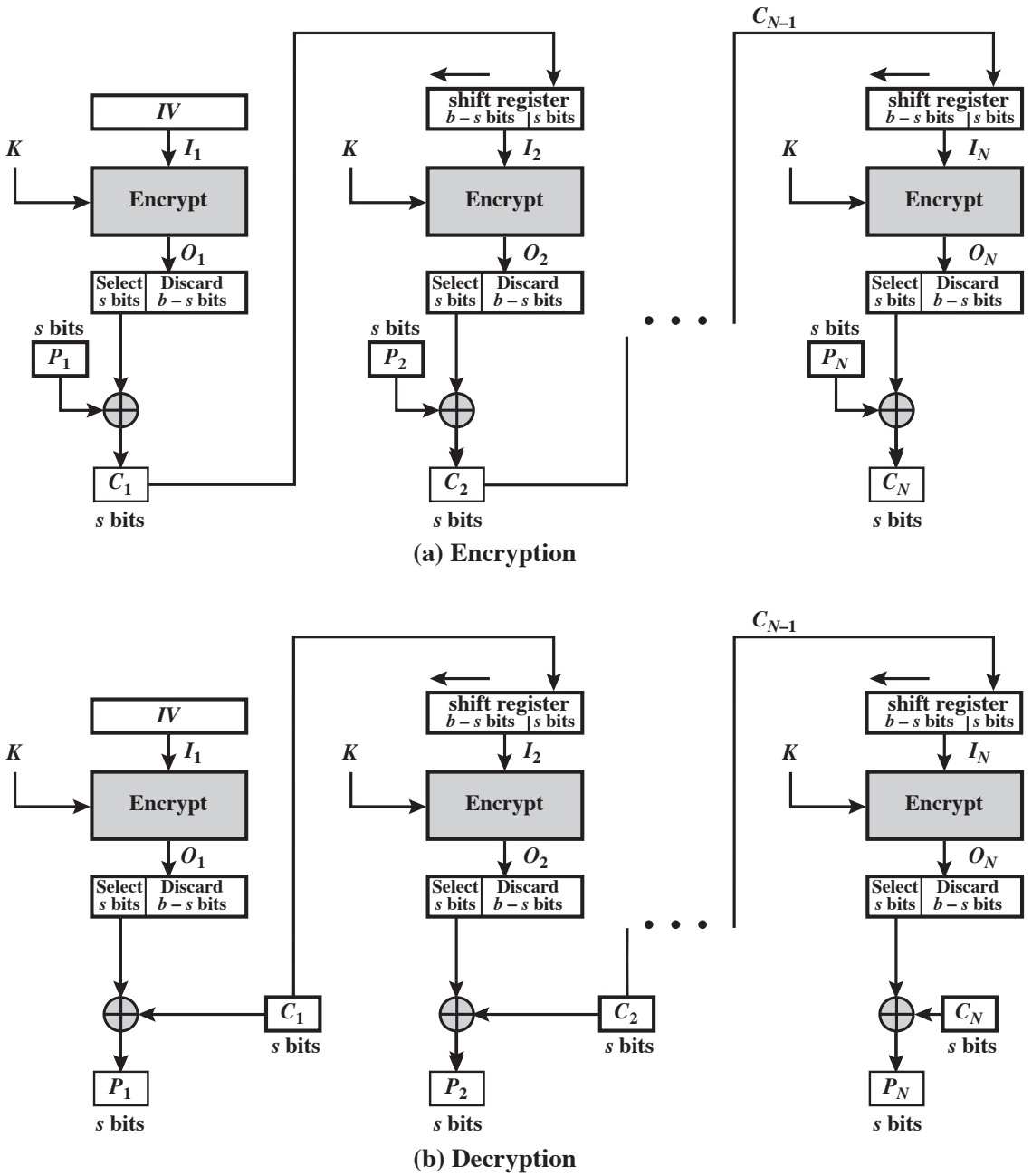


Figure 7.5 *s*-bit Cipher Feedback (CFB) Mode

Advantages and Limitations of CFB

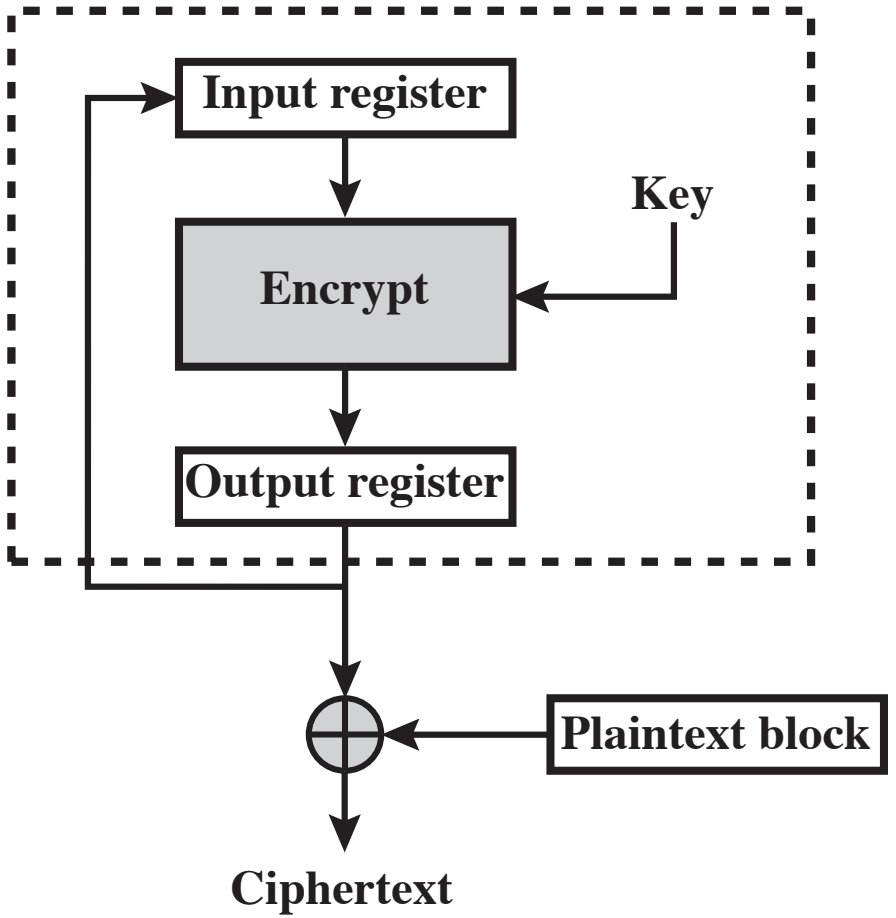
- appropriate when data arrives in bits/bytes
- most common stream mode
- limitation is need to stall while do block encryption after every n-bits
- note that the block cipher is used in **encryption** mode at **both** ends
- errors propogate for several blocks after the error

Output FeedBack (OFB)

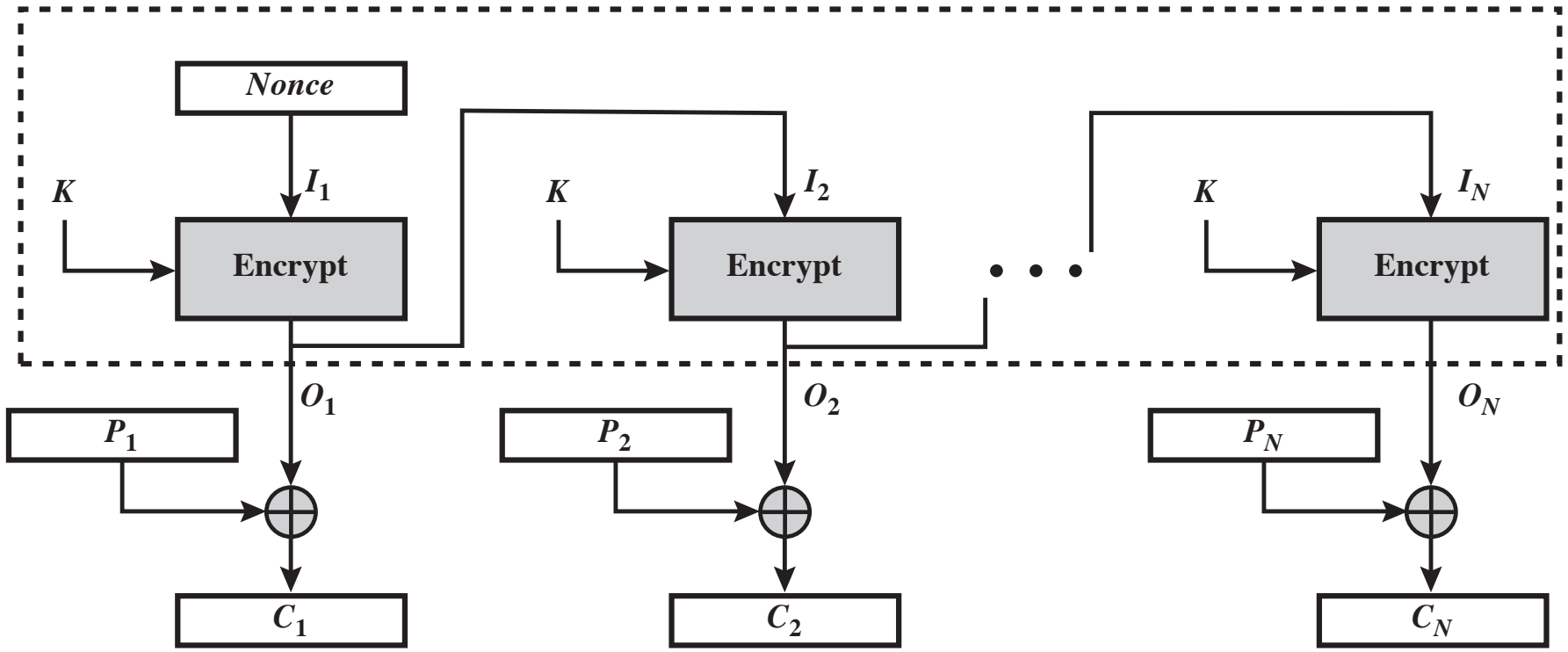
- message is treated as a stream of bits
- output of cipher is added to message
- output is then feed back (hence name)
- feedback is independent of message
- can be computed in advance

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

- uses: stream encryption on noisy channels

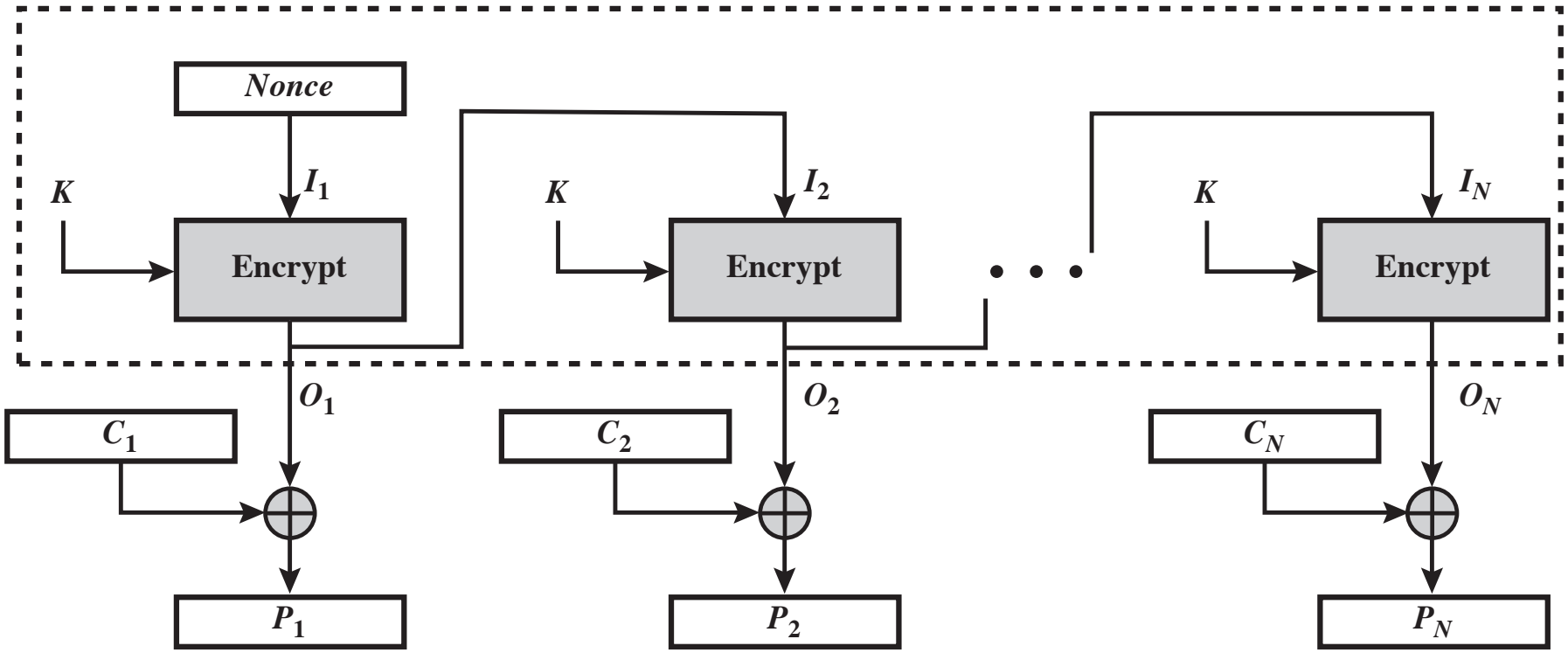


(c) Output feedback (OFB) mode



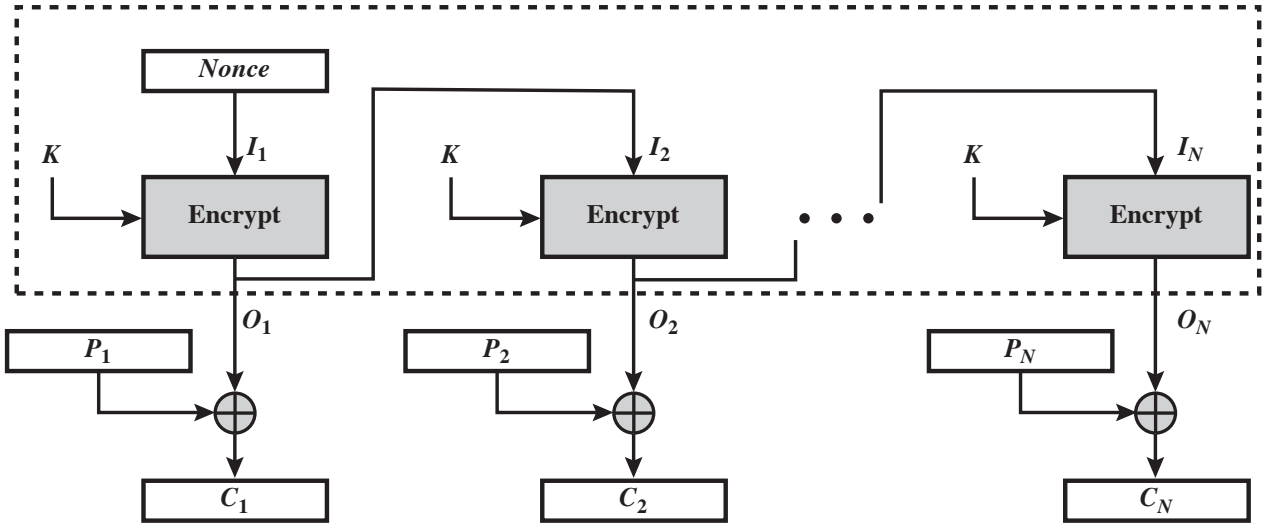
(a) Encryption

Figure 7.6 Output Feedback (OFB) Mode

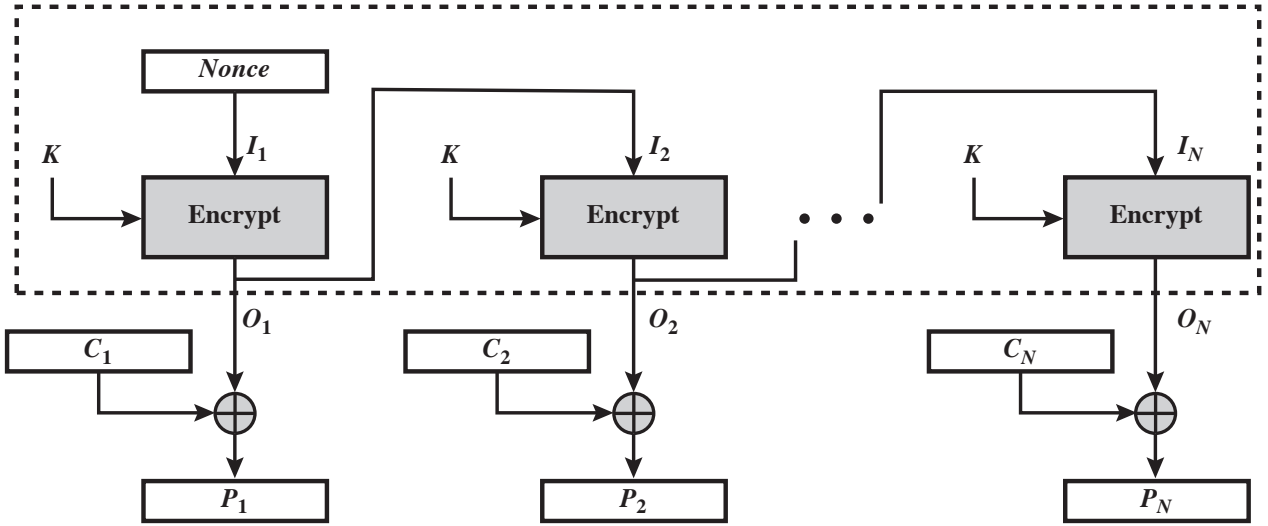


(b) Decryption

Figure 7.6 Output Feedback (OFB) Mode



(a) Encryption



(b) Decryption

Figure 7.6 Output Feedback (OFB) Mode

Advantages and Limitations of OFB

- needs an IV which is unique for each use
  - if ever reuse attacker can recover outputs
- bit errors do not propagate
- more vulnerable to message stream modification
- sender & receiver must remain in sync
- only use with full block feedback
  - subsequent research has shown that only **full block feedback** (ie CFB-64 or CFB-128) should ever be used

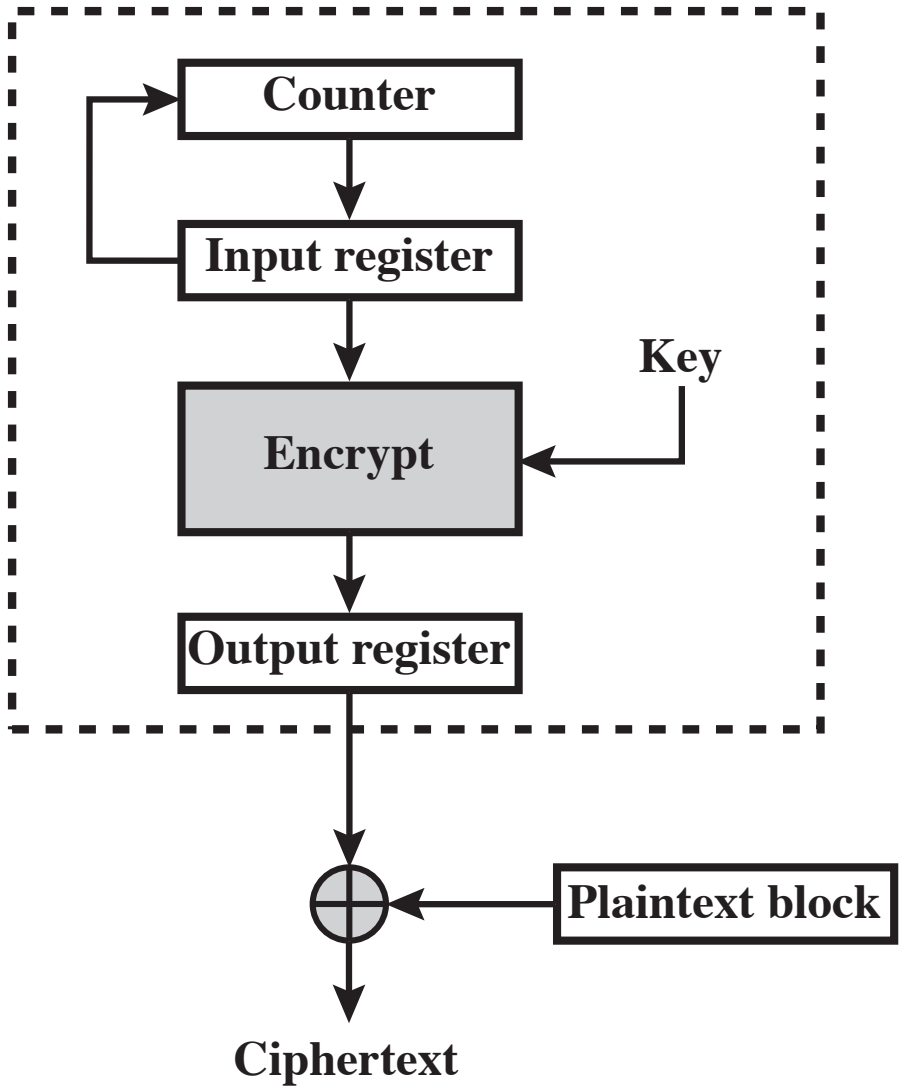
Counter (CTR)

- A “new” mode, though proposed early on
- Similar to OFB but encrypts counter value rather than any feedback value
- Must have a different key & counter value for every plaintext block (never reused)

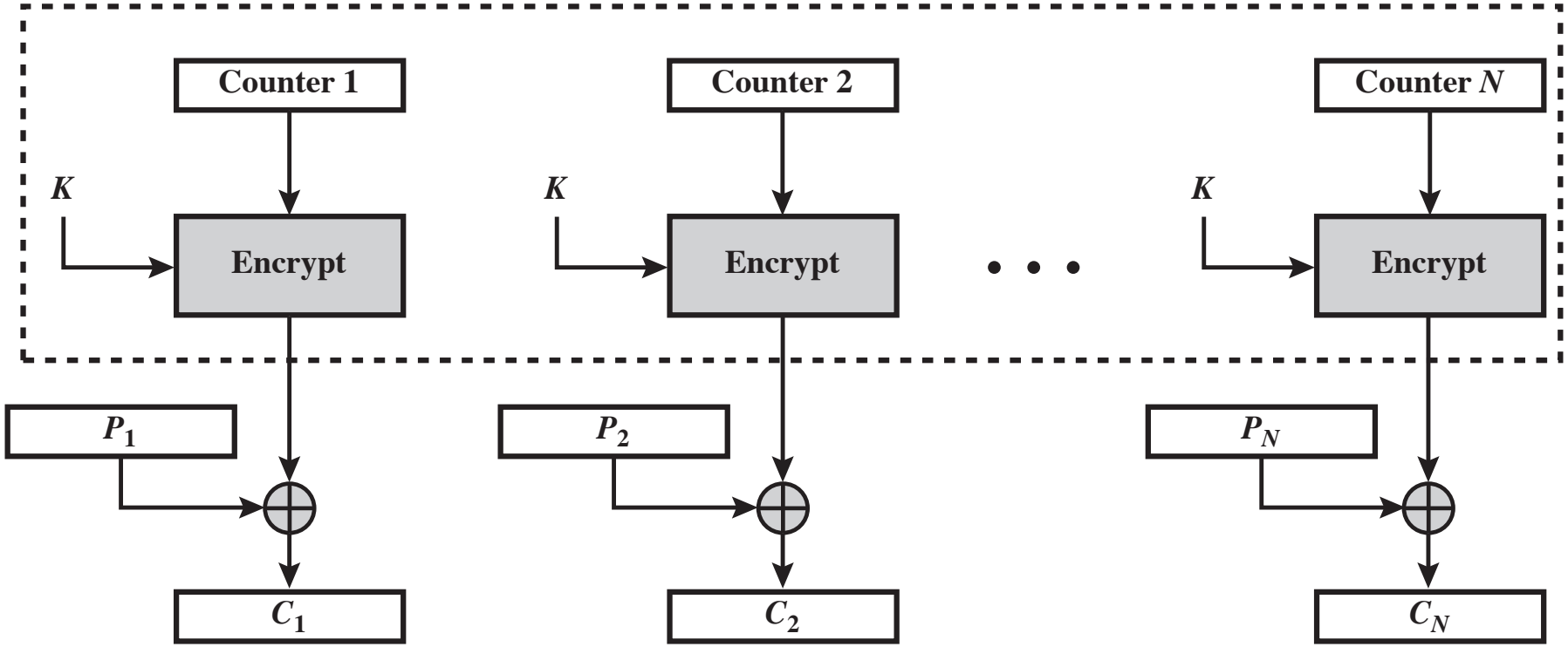
$O_i = E_K(i)$

$C_i = P_i \text{ XOR } O_i$

- Uses: high-speed network encryptions

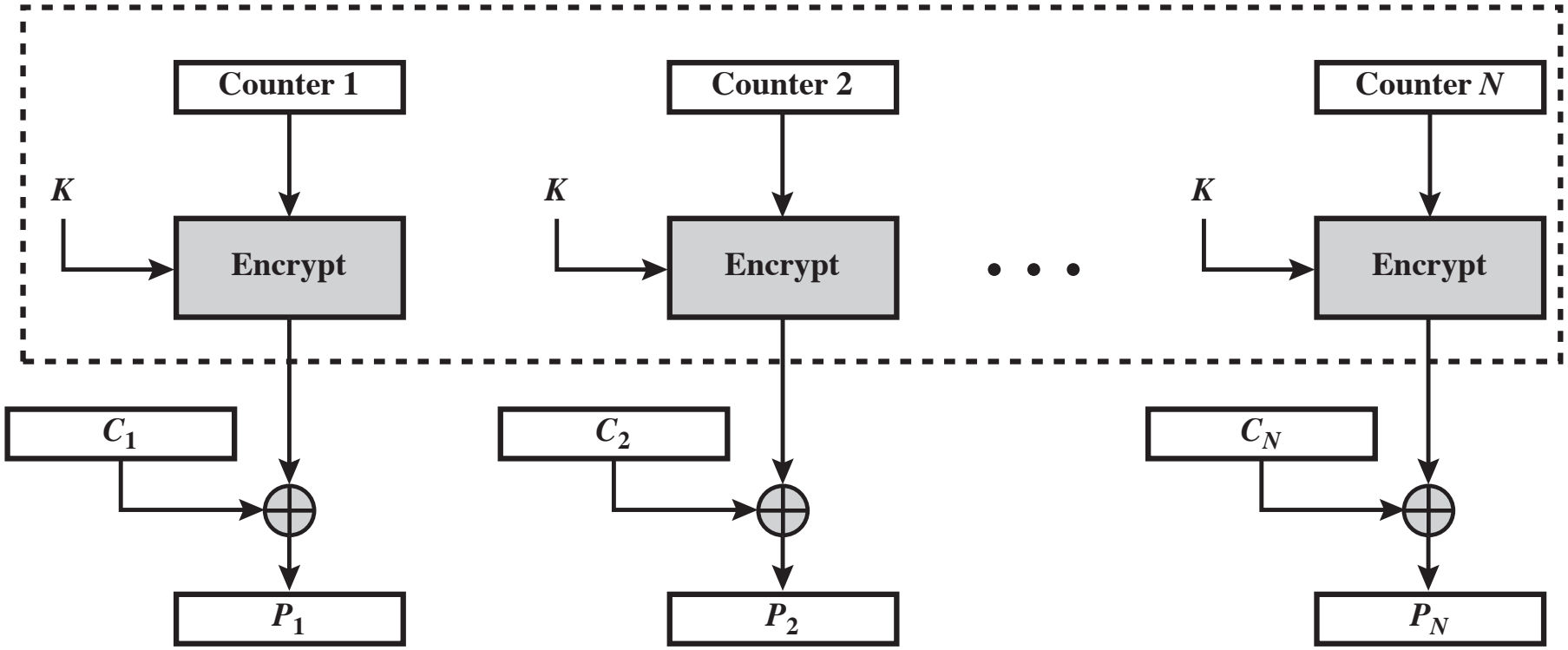


(d) Counter (CTR) mode



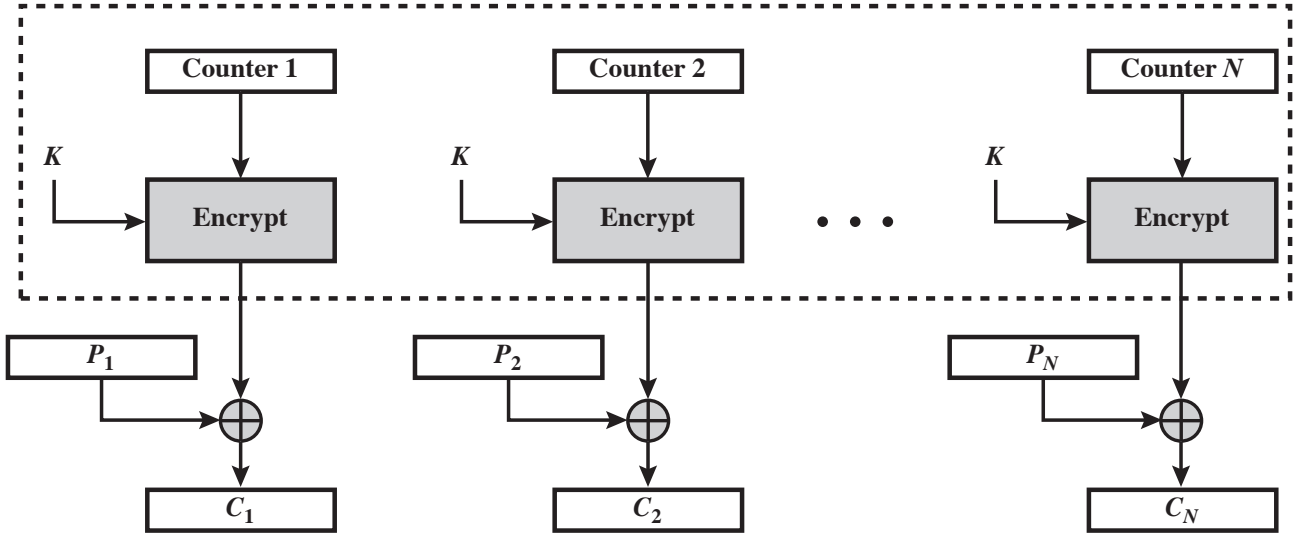
(a) Encryption

Figure 7.7 Counter (CTR) Mode

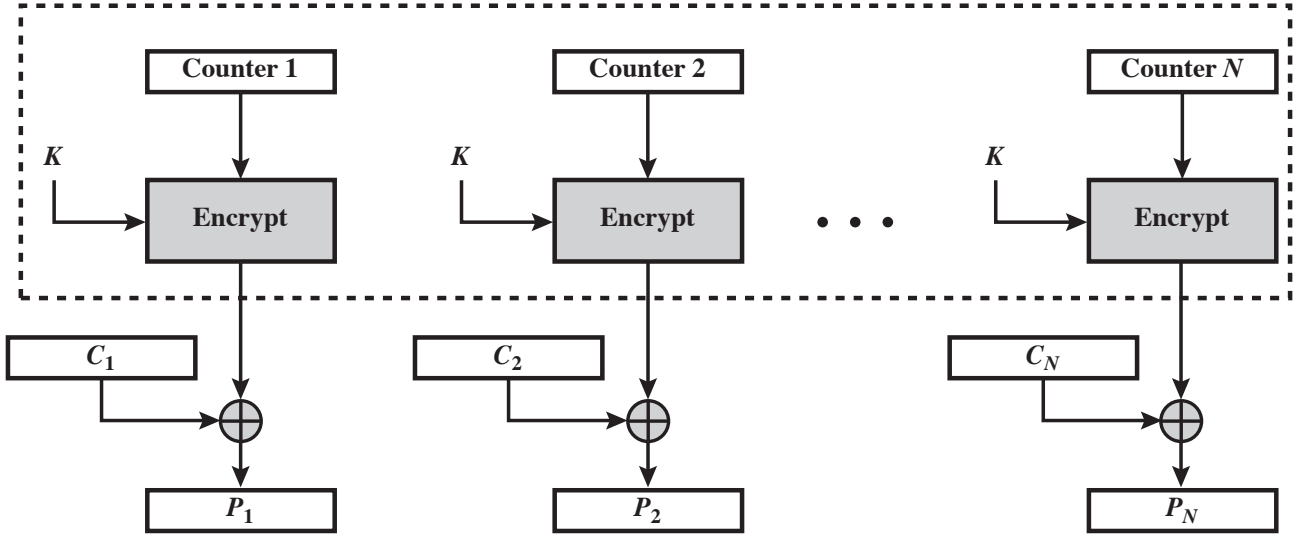


(b) Decryption

Figure 7.7 Counter (CTR) Mode



(a) Encryption



(b) Decryption

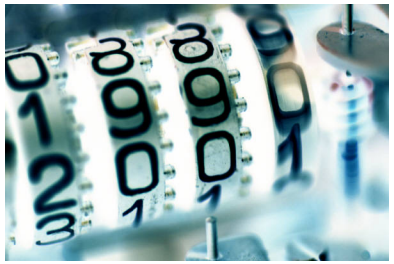
Figure 7.7 Counter (CTR) Mode

Advantages and Limitations of CTR

- Efficiency
  - can do parallel encryptions in h/w or s/w
  - can preprocess in advance of need
  - good for bursty high speed links
- Random access to encrypted data blocks
- Provable security (good as other modes)
- But must ensure never reuse key/counter values, otherwise could break (cf OFB)

Advantages of CTR

- Hardware efficiency
- Software efficiency
- Preprocessing
- Random access
- Provable security
- Simplicity



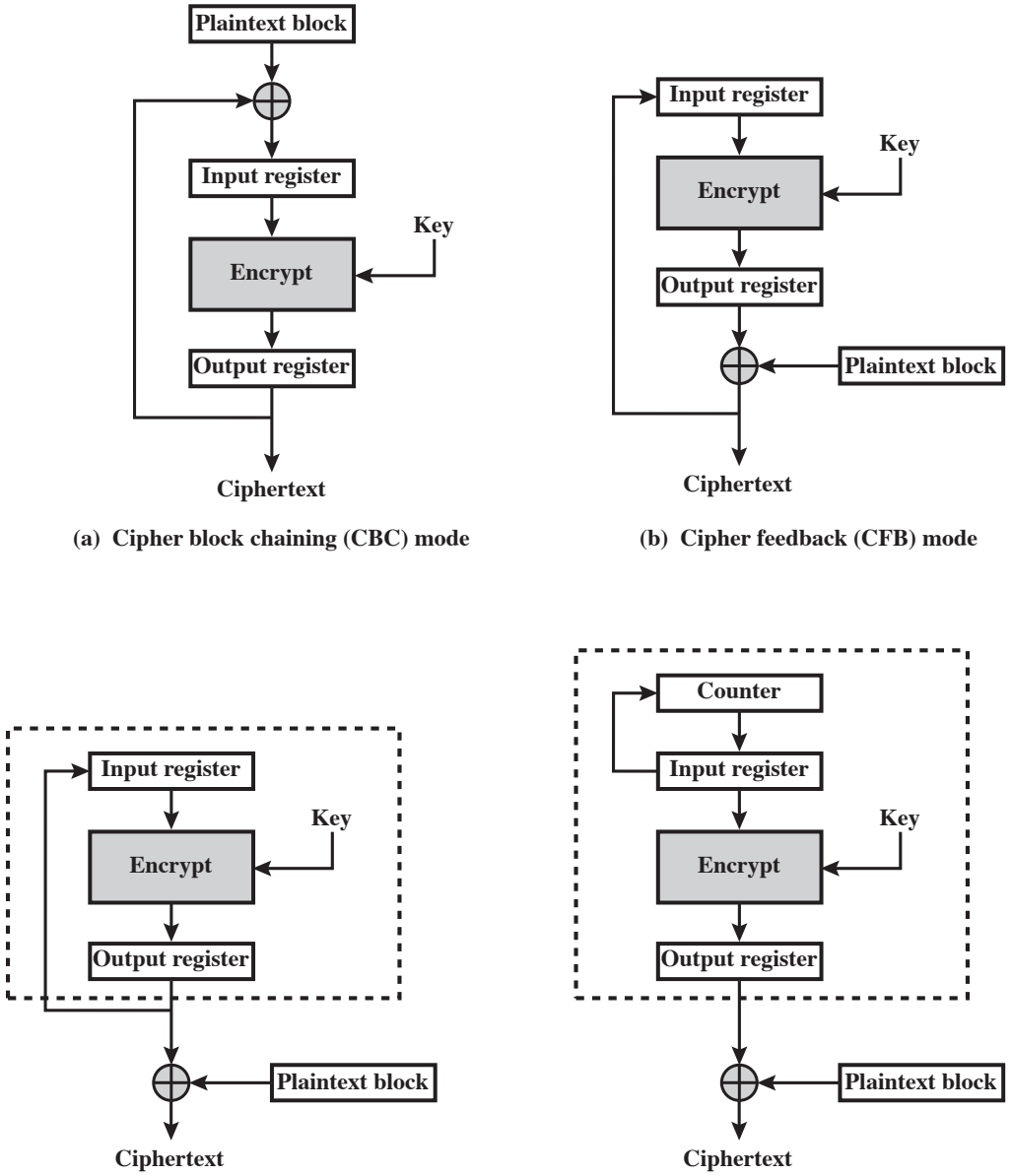


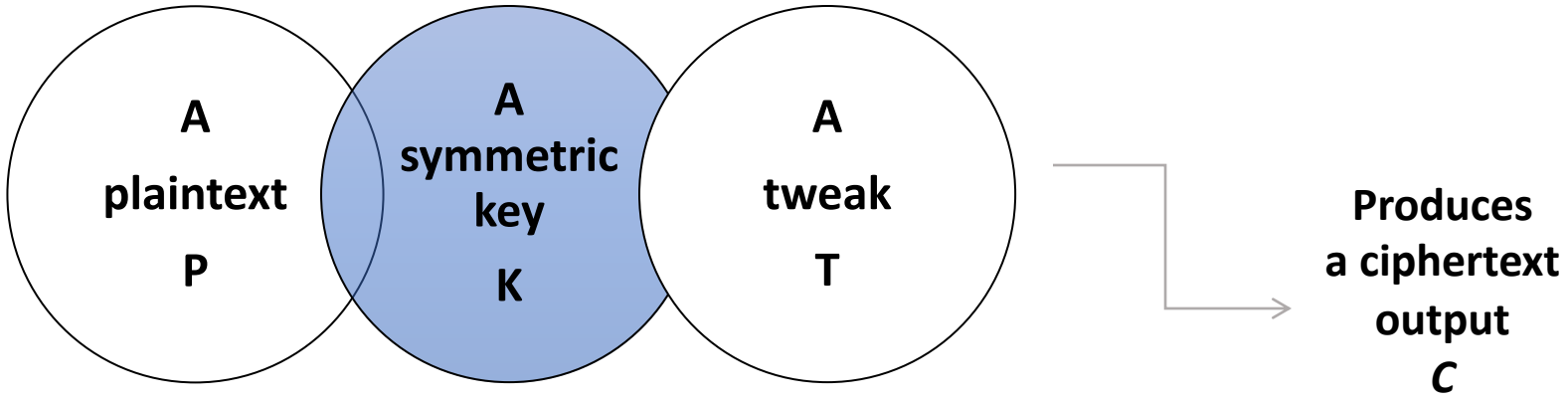
Figure 7.8 Feedback Characteristic of Modes of Operation

XTS-AES Mode for Block-Oriented Storage Devices

- Approved as an additional block cipher mode of operation by NIST in 2010
- Mode is also an IEEE Standard, IEEE Std 1619-2007
  - Standard describes a method of encryption for data stored in sector-based devices where the threat model includes possible access to stored data by the adversary
  - Has received widespread industry support

Tweakable Block Ciphers

- XTS-AES mode is based on the concept of a *tweakable block cipher*
- General structure:
  - Has three inputs:



- Tweak need not be kept secret
  - Purpose is to provide variability

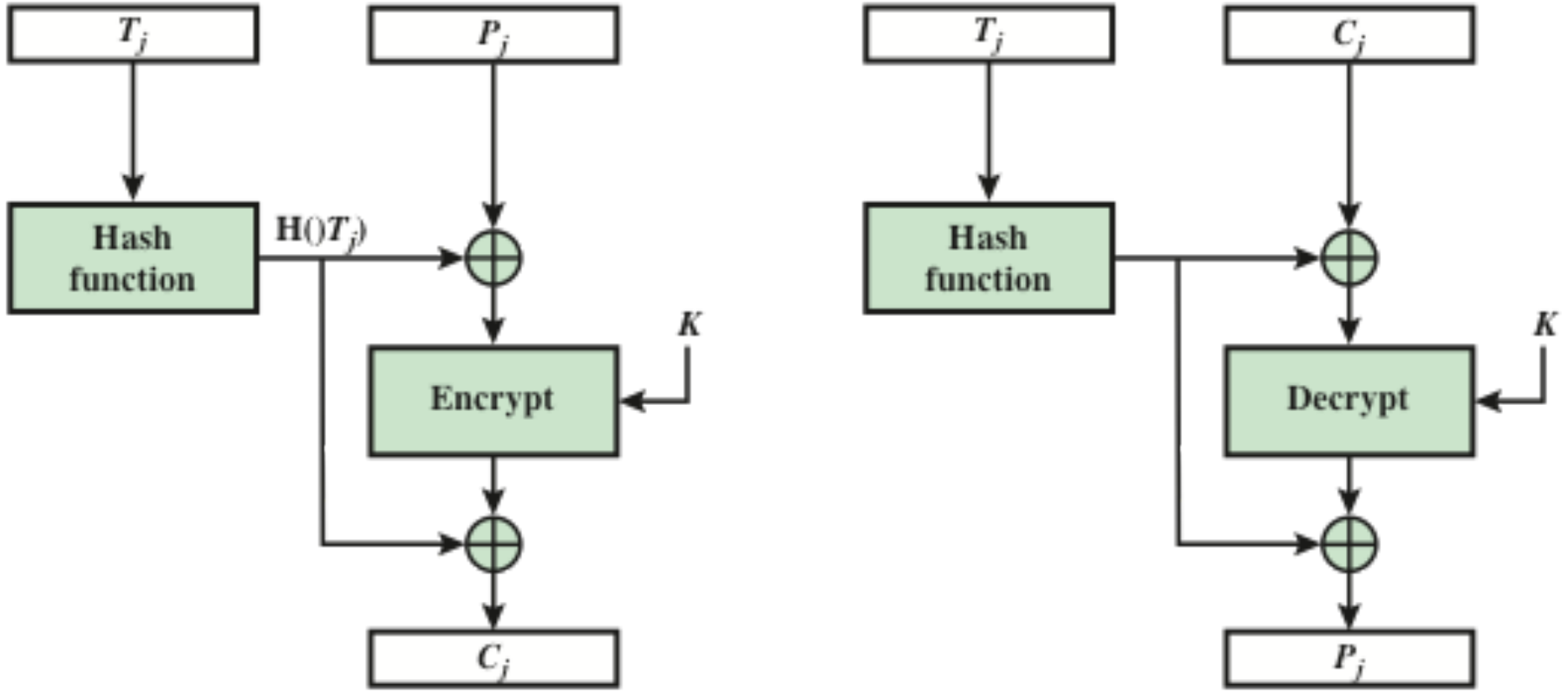


Figure 7.9 Tweakable Block Cipher

XTS-AES Mode for Block-Oriented Storage Devices

- Approved as an additional block cipher mode of operation by NIST in 2010
- Mode is also an IEEE Standard, IEEE Std 1619-2007
  - Standard describes a method of encryption for data stored in sector-based devices where the threat model includes possible access to stored data by the adversary
  - Has received widespread industry support
- concept of tweakable block cipher
- different requirements to transmitted data

Storage Encryption Requirements

- The requirements for encrypting stored data, also referred to as “data at rest”, differ somewhat from those for transmitted data
- The P1619 standard was designed to have the following characteristics:
  - The ciphertext is freely available for an attacker
  - The data layout is not changed on the storage medium and in transit
  - Data are accessed in fixed sized blocks, independently from each other
  - Encryption is performed in 16-byte blocks, independently from each other
  - There are no other metadata used, except the location of the data blocks within the whole data set
  - The same plaintext is encrypted to different ciphertexts at different locations, but always to the same ciphertext when written to the same location again
  - A standard conformant device can be constructed for decryption of data encrypted by another standard conformant device

XTS-AES Operation on Single Block

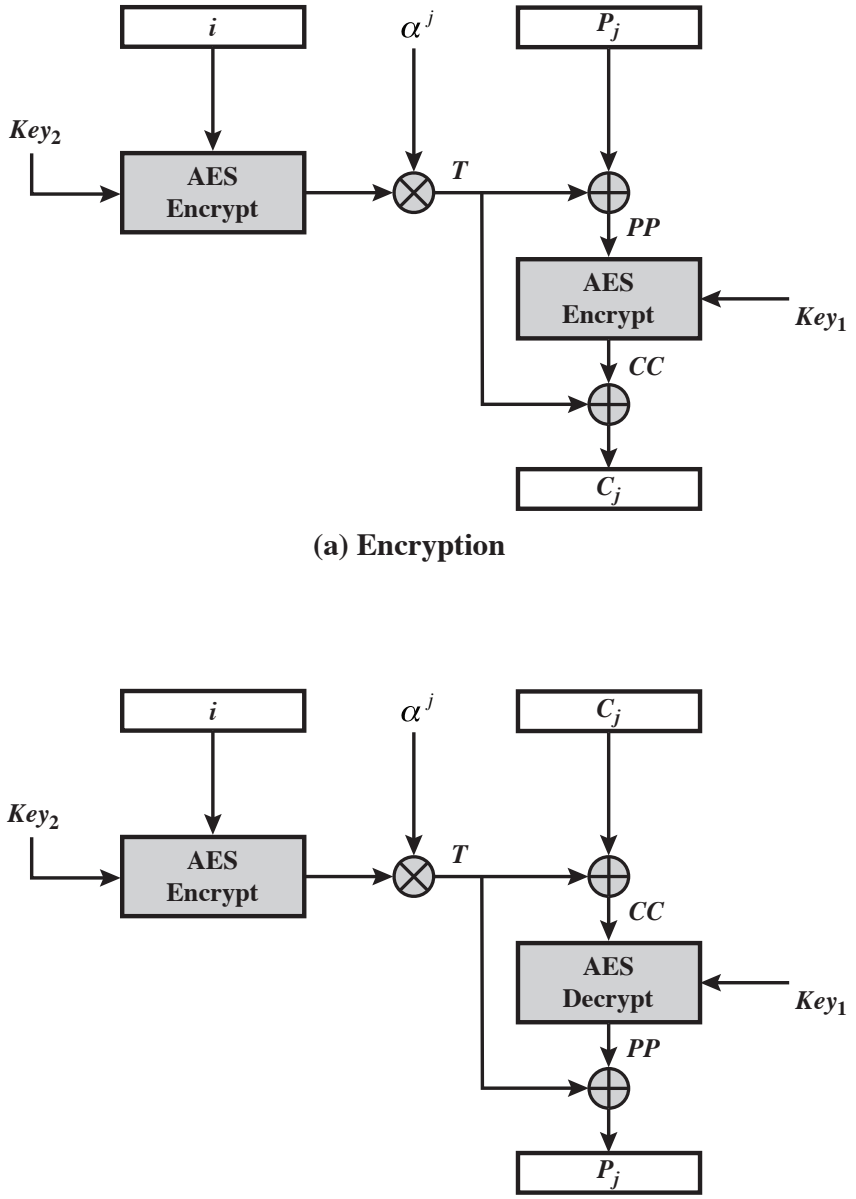
- uses AES twice for each block

$$T_j = E_{K2}(i) \otimes \alpha^j$$

$$C_j = E_{K1}(P_j \text{ XOR } T_j) \text{ XOR } T_j$$

where *i* is tweak & *j* is sector no

- each sector may have multiple blocks



α	A primitive element of GF(2^{128}) that corresponds to polynomial x (i.e., 0000 . . . 010_2).
α^j	α multiplied by itself j times, in GF(2^{128}).
$\oplus$	Bitwise XOR.
$\otimes$	Modular multiplication of two polynomials with binary coefficients modulo $x^{128} + x^7 + x^2 + x + 1$. Thus, this is multiplication in GF(2^{128}).

Figure 7.10 XTS-AES Operation on Single Block

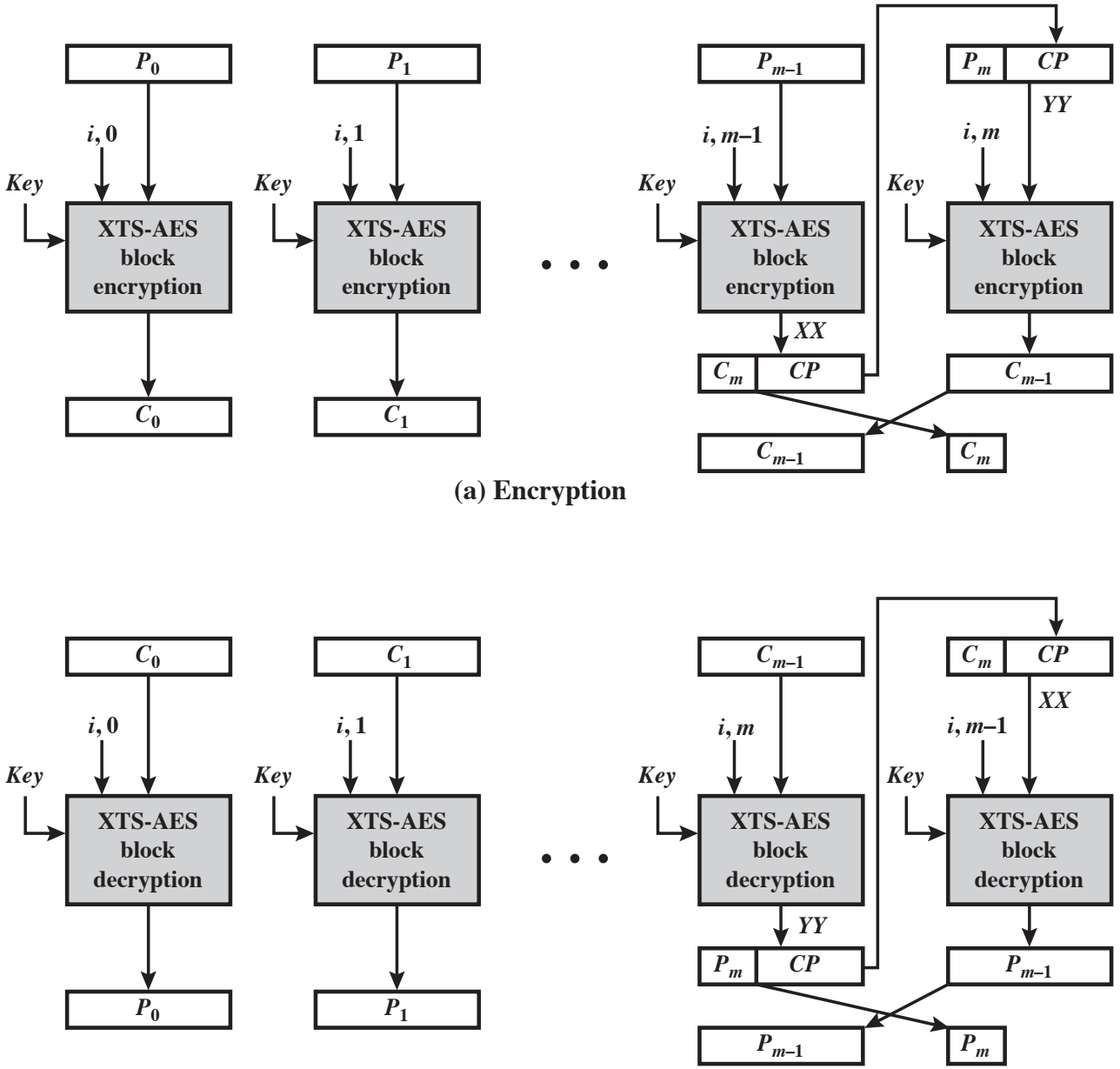


Figure 7.11 XTS-AES Mode

Advantages and Limitations of XTS-AES

- efficiency
  - can do parallel encryptions in h/w or s/w
  - random access to encrypted data blocks
- has both nonce & counter
- addresses security concerned related to stored data

Format-Preserving Encryption (FPE)

- Refers to any encryption technique that takes a plaintext in a given format and produces a ciphertext in the same format
  - For example:
    credit cards consist of 16 decimal digits.
    An FPE that can accept this type of input would produce a ciphertext output of 16 decimal digits.
    (Note that the ciphertext need not be, and in fact in unlikely to be, a valid credit card number.)
    But it will have the same format and can be stored in the same way as credit card number plaintext.

Table 7.2

Comparison of Format-Preserving Encryption and AES

	Credit Card	Tax ID	Bank Account Number
Plaintext	8123 4512 3456 6780	219-09-9999	800N2982K-22
FPE	8123 4521 7292 6780	078-05-1120	709G9242H-35
AES (hex)	af41326466add24 c86abd8aa525db7a	7b9af4f3f218ab25 07c7376869313afa	9720ec7f793096ff d37141242e1c51bd

Motivation

FPE facilitates the retrofitting of encryption technology **to legacy applications**, where a conventional encryption mode might not be feasible because it would disrupt data fields/pathways

FPE has emerged as a useful cryptographic tool, whose applications include financial-information security, data sanitization, and transparent encryption of fields in legacy databases

The principal benefit of FPE is that it enables protection of particular data elements, while still enabling workflows that were in place before FPE was in use

- No database schema changes and minimal application changes are required
- Only applications that need to see the plaintext of a data element need to be modified and generally these modifications will be minimal

Some examples of legacy applications where FPE is desirable are:

- COBOL data-processing applications
- Database applications
- FPE-encrypted characters can be significantly compressed for efficient transmission

Difficulties in Designing an FPE

- A general-purpose standardized FPE should meet a number of requirements:
  - The ciphertext is of the same length and format as the plaintext
  - It should be adaptable to work with a variety of character and number types
  - It should work with variable plaintext length
  - Security strength should be comparable to that achieved with AES
  - Security should be strong even for very small plaintext lengths

e.g. Decimal digit strings of max length of 32 digits

- Represented in 16 bytes (128 bits)
  - encoding each digit as its four bits binary value (e.g., 6 is encoded as 0101)
- Plaintext input X = 128-bit binary string of 4-bit decimal digits X[1] . . . X[16]
  - Padding out to the left (most significant) with zeros, if less than 16 digits long
- With key K, ciphertext Y = $AES_K(X)$ is a string of length 16 of 4-bit elements.
  - Some of the entries in Y may be > 9 (e.g., 1100)
  - To form ciphertext Z in the required format, calculate
    $Z[i] = Y[i] \bmod 10, \text{ for } 1 \leq i \leq 16$
- However, no reversibility
  - many- to-one function, e.g. 12 mod 10 = 2 mod 10 = 2

16-digit credit card number (CCN)

- first six digits provide the issuer identification number (IIN)
- final digit is a check digit to catch typographical errors or other mistakes
- remaining nine digits are the user's account number
- the last four digits is often left in the clear (for receipt double check)
  - leaves only six digits for encryption
- To use tweakable block cipher
- the tweak for CCNs could be the first two and last four digits of the CCN

CCN	Tweak	Plaintext	Plaintext + Tweak
4012 8812 3456 1884	401884	123456	524230
5105 1012 3456 6782	516782	123456	639138

Character Strings

- The NIST, and the other FPE algorithms that have been proposed, are used with plaintext consisting of a string of elements, called characters
- A finite set of two or more symbols is called an *alphabet*
- The elements of an alphabet are called *characters*
- A *character string* is a finite sequence of characters from an alphabet
- Individual characters may repeat in the string
- The number of different characters in an alphabet is called the *base* (also referred to as the *radix)* of the alphabet

ASCII Table

0	*NUL*	16	*DLE*	32	SPC	48	0	64	@	80	P	96	`	112	p
1	*SOH*	17	*DC1*	33	!	49	1	65	A	81	Q	97	a	113	q
2	*STX*	18	*DC2*	34	"	50	2	66	B	82	R	98	b	114	r
3	*ETX*	19	*DC3*	35	#	51	3	67	C	83	S	99	c	115	s
4	*EOT*	20	*DC4*	36	$	52	4	68	D	84	T	100	d	116	t
5	*ENQ*	21	*NAK*	37	%	53	5	69	E	85	U	101	e	117	u
6	*ACK*	22	*SYN*	38	&	54	6	70	F	86	V	102	f	118	v
7	*BEL*	23	*ETB*	39	'	55	7	71	G	87	W	103	g	119	w
8	*BS*	24	*CAN*	40	(	56	8	72	H	88	X	104	h	120	x
9	*HT*	25	*EM*	41	)	57	9	73	I	89	Y	105	i	121	y
10	*LF*	26	*SUB*	42	*	58	:	74	J	90	Z	106	j	122	z
11	*VT*	27	*ESC*	43	+	59	;	75	K	91	[	107	k	123	{
12	*FF*	28	*FS*	44	,	60	<	76	L	92	\	108	l	124	|
13	*CR*	29	*GS*	45	-	61	=	77	M	93	]	109	m	125	}
14	*SO*	30	*RS*	46	.	62	>	78	N	94	^	110	n	126	~
15	*SI*	31	*US*	47	/	63	?	79	O	95	_	111	o	127	DEL

Table 7.3 Notation and Parameters Used in FPE Algorithms

(a) Notation

$[x]^s$	Converts an integer into a byte string; it is the string of s bytes that encodes the number x, with $0 \le x < 2^{8s}$. The equivalent notation is $\mathsf{STR}_2^{8s}(x)$.
LEN(X)	Length of the character string X.
$\text{NUM}_{radix}(X)$	Converts strings to numbers. The number that the numeral string X represents in base *radix*, with the most significant character first. In other words, it is the non-negative integer less than $radix^{\text{LEN}(X)}$ whose most-significant-character-first representation in base *radix* is X.
$\text{PRF}_K(X)$	A pseudorandom function that produces a 128-bit output with X as the input, using encryption key K.
$\mathsf{STR}_{radix}^m(x)$	Given a nonnegative integer x less than $radix^m$, this function produces a representation of x as a string of m characters in base *radix*, with the most significant character first.
$[i \,..\, j]$	The set of integers between two integers i and j, including i and j.
$X[i \,..\, j]$	The substring of characters of a string X from $X[i]$ to $X[j]$, including $X[i]$ and $X[j]$.
REV(X)	Given a bit string, X, the string that consists of the bits of X in reverse order.

Table 7.3
Notation and Parameters Used in FPE Algorithms

(b) Parameters

radix	The base, or number of characters, in a given plaintext alphabet.
tweak	Input parameter to the encryption and decryption functions whose confidentiality is not protected by the mode.
tweakradix	The base for tweak strings
minlen	Minimum message length, in characters.
maxlen	Maximum message length, in characters.
maxTlen	Maximum tweak length

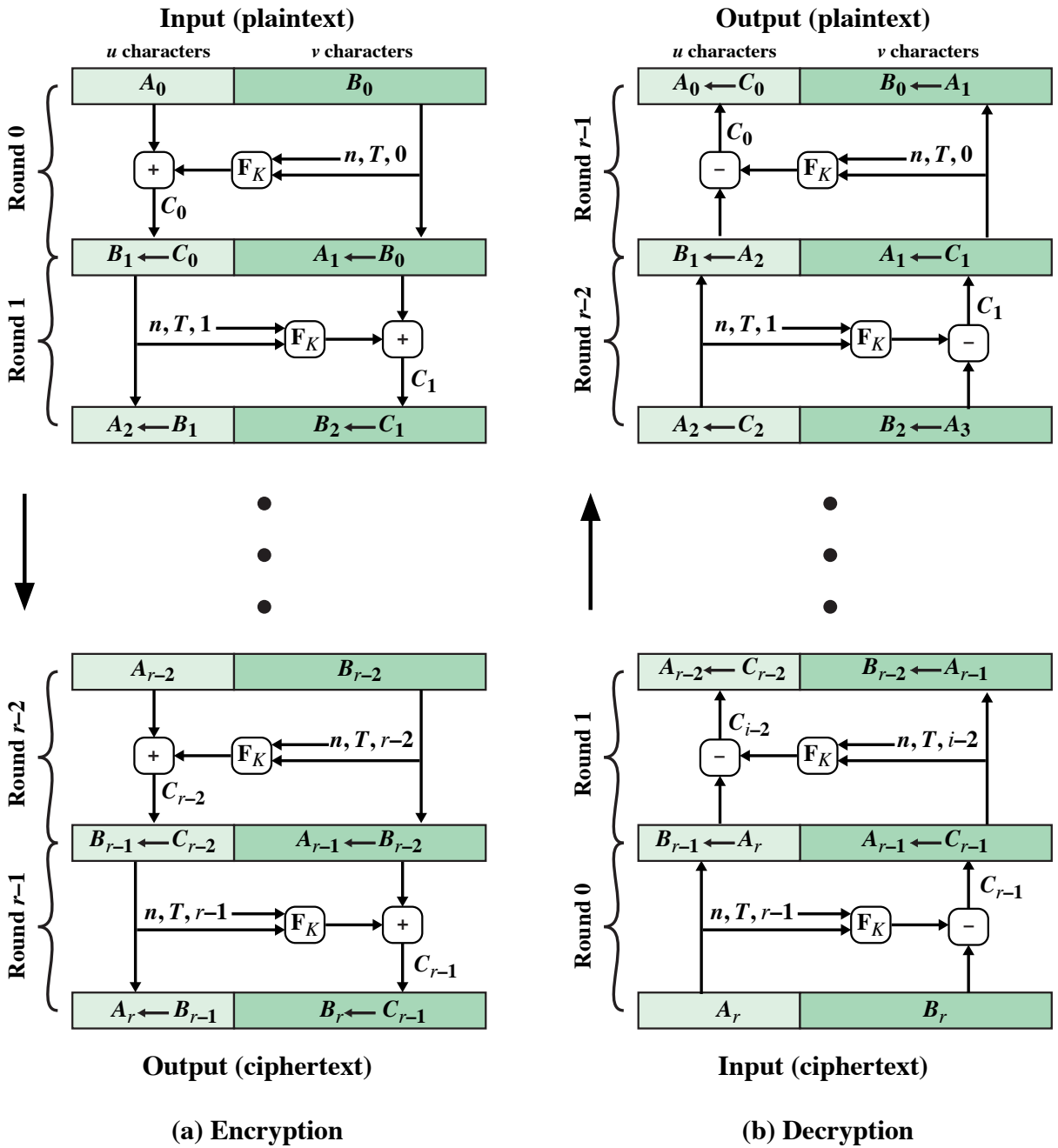


Figure 7.12 Feistel Structure for Format-Preserving Encryption

NIST Methods for Format-Preserving Encryption

The three methods FF1, FF2, and FF3 all use the Feistel structure in Figure 7.12, but employ somewhat different round functions F_K, which are built using AES.

Important differences are the following:

- FF1 supports the greatest range of lengths for the plaintext character string and the tweak
  - the round function uses a cipher-block-chaining (CBC) style of encryption, whereas FF2 and FF3 employ simple electronic codebook (ECB) encryption.
- FF2 uses a subkey generated from the encryption key and the tweak, whereas FF1 and FF3 use the encryption key directly.
  - The use of a subkey may help protect the original key from side-channel analysis
    - an attack based on information gained from the physical implementation of a cryptosystem, rather than brute force or cryptanalysis, e.g. based on power consumption or execution time.
- FF3 offers the lowest round count, eight, compared to ten for FF1 and FF2, and is the least flexible in the tweaks that it supports.

t
Prerequisites:
Approved, 128-bit block cipher, CIPH;
Key, *K*, for the block cipher;

Input:
Nonempty bit string, *X*, such that LEN(*X*) = is a multiple of 128.
Output:
128-bit block, *Y*

Steps:
1. Let m = LEN(X)/128.
2. Partition X into m 128-bit blocks $X_1,\ldots, X_m$, so that $X = X_1 \| \ldots \| X_m$
3. Let $Y_0 = [0]^{16}$
4. For j from 1 to m:
5. let $Y_j = \text{CIPH}_K(Y_{j-1} \oplus X_j)$.
6. Return Y_m.

Figure 7.13 Algorithm PRF(X)

Prerequisites:
Approved, 128-bit block cipher, CIPH;
Key, K, for the block cipher;
Base, *radix*, for the character alphabet;
Range of supported message lengths, [minlen .. maxlen];
Maximum byte length for tweaks, maxTlen.

Inputs:

Character string, X, in base radix of length n such that $n \in$ [minlen .. maxlen];

Tweak T, a byte string of byte length t, such that $t \in$ [0 .. maxTlen].

Output:
Character string, Y, such that LEN(Y) = n.

Steps:

1. Let $u = \lfloor n/2 \rfloor$; $v = n - u$.
2. Let $A = X[1 .. u]$; $B = X[u + 1 .. n]$.
3. Let $b = \lceil \lceil v \, \text{LOG}_2(radix) \rceil / 8 \rceil$; $d = 4 \lceil b/4 \rceil + 4$
4. Let $P = [1]^1 \,||\, [2]^1 \,||\, [1]^1 \,||\, [radix]^3 \,||\, [10]^1 \,||\, [u \bmod 256]^1 \,||\, [n]^4 \,||\, [t]^4$.
5. For i from 0 to 9:
   i. Let $Q = T \,||\, [0]^{(-t-b-1) \bmod 16} \,||\, [i]^1 \,||\, [\text{NUM}_{radix}(B)]^b$.
   ii. Let $R = \text{PRF}_K(P \,||\, Q)$.
   iii. Let S be the first d bytes of the following string of $\lceil d/16 \rceil$ 128-bit blocks:
   $R \,||\, \text{CIPH}_K(R \oplus [1]^{16}) \,||\, \text{CIPH}_K(R \oplus [2]^{16}) \,||\, \ldots \,||\, \text{CIPH}_K(R \oplus [\lceil d/16 \rceil - 1]^{16})$.
   iv. Let $y = \text{NUM}_2(S)$.
   v. If i is even, let $m = u$; else, let $m = v$.
   vi Let $c = (\text{NUM}_{radix}(A) + y) \bmod radix^m$.
   vii. Let $C = \text{STR}^m_{radix}(c)$.
   viii. Let $A = B$.
   ix. Let $B = C$.
6. Return $Y = A \,||\, B$.

Figure 7.14 Algorithm FF1 (FFX[Radix])

Approved, 128-bit block cipher, CIPH;
Key, K, for the block cipher;
Base, *tweakradix*, for the tweak character alphabet;
Range of supported message lengths, [*minlen* .. *maxlen*]
Maximum supported tweak length, *maxTlen*.

Inputs:

Numeral string, X, in base *radix*, of length n such that $n \in$ [*minlen* .. *maxlen*];

Tweak numeral string, T, in base *tweakradix*, of length t such that $t \in$ [0 .. *maxTlen*].

Output:

Numeral string, Y, such that LEN(Y) = n.

Steps:

1. Let $u = \lfloor n/2 \rfloor$; $v = n - u$.
2. Let $A = X[1 .. u]$; $B = X[u + 1 .. n]$.
3. If $t>0$, $P = [radix]^1 \,||\, [t]^1 \,||\, [n]^1 \,||\, [\text{NUM}_{tweakradix}(T)]^{13}$;
   else $P = [radix]^1 \,||\, [0]^1 \,||\, [n]^1 \,||\, [0]^{13}$.
4. Let $J = \text{CIPH}_K(P)$
5. For i from 0 to 9:
   i. Let $Q \leftarrow [i]^1 \,||\, [\text{NUM}_{radix}(B)]^{15}$
   ii. Let $Y \leftarrow \text{CIPH}_J(Q)$.
   iii. Let $y \leftarrow \text{NUM}_2(Y)$.
   iv. If i is even, let $m = u$; else, let $m = v$.
   v. Let $c = (\text{NUM}_{radix}(A) + y) \bmod radix^m$.
   vi. Let $C = \text{STR}^m_{radix}(c)$.
   vii. Let $A = B$.
   viii. Let $B = C$.
6. Return $Y = A \,||\, B$.

Figure 7.15 Algorithm FF2 (VAES3)

Approved, 128-bit block cipher, CIPH;
Key, K, for the block cipher;
Base, *radix*, for the character alphabet such that $radix \in [2 .. 2^{16}]$;
Range of supported message lengths, [*minlen* .. *maxlen*],
such that $minlen \geq 2$ and $maxlen \leq 2\lfloor \log_{radix}(2^{96}) \rfloor$.

Inputs:
Numeral string, X, in base *radix* of length n such that $n \in [minlen .. maxlen]$;
Tweak bit string, T, such that LEN(T) = 64.
Output:
Numeral string, Y, such that LEN(Y) = n.

Steps:
1. Let $u = \lceil n/2 \rceil$; $v = n - u$.
2. Let $A = X[1 .. u]$; $B = X[u + 1 .. n]$.
3. Let $T_L = T[0..31]$ and $T_R = T[32..63]$
4. For i from 0 to 7:
   i. If i is even, let $m = u$ and $W = T_R$, else let $m = v$ and $W = T_L$.
   ii. Let $P = \text{REV}([\text{NUM}_{radix}(\text{REV}(B))]^{12}) \,||\, [W \oplus \text{REV}([i]^4])$.
   iii. Let $Y = \text{CIPH}_K(P)$.
   iv. Let $y = \text{NUM}_2(\text{REV}(Y))$.
   v. Let $c = (\text{NUM}_{radix}(\text{REV}(A)) + y) \bmod radix^m$.
   vi. Let $C = \text{REV}(\text{STR}^m_{radix}(c))$.
   vii. Let $A = B$.
   viii. Let $B = C$.
5. Return $A \,||\, B$.

Figure 7.16 Algorithm FF3 (BPS-BC)

Summary

- Multiple encryption and triple DES
  - Double DES
  - Triple DES with two keys
  - Triple DES with three keys
- Electronic codebook
- Cipher block chaining mode
- Format-preserving encryption
  - Motivation
  - Difficulties in designing
  - Feistel structure
  - NIST methods

- Cipher feedback mode
- Output feedback mode
- Counter mode
- XTS-AES mode for block-oriented storage devices
  - Tweakable block ciphers
  - Storage encryption requirements
  - Operation on a single block
  - Operation on a sector