

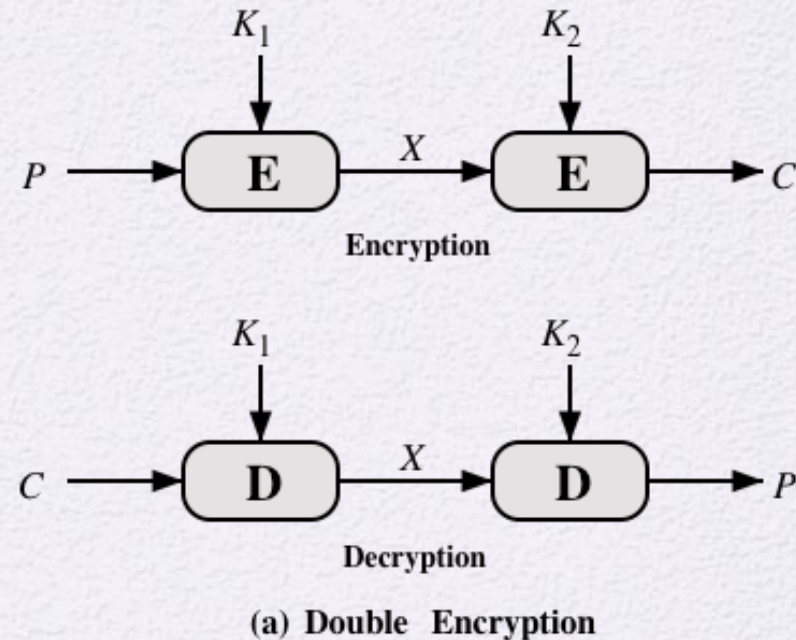


Chapter 7

Block Cipher Operation

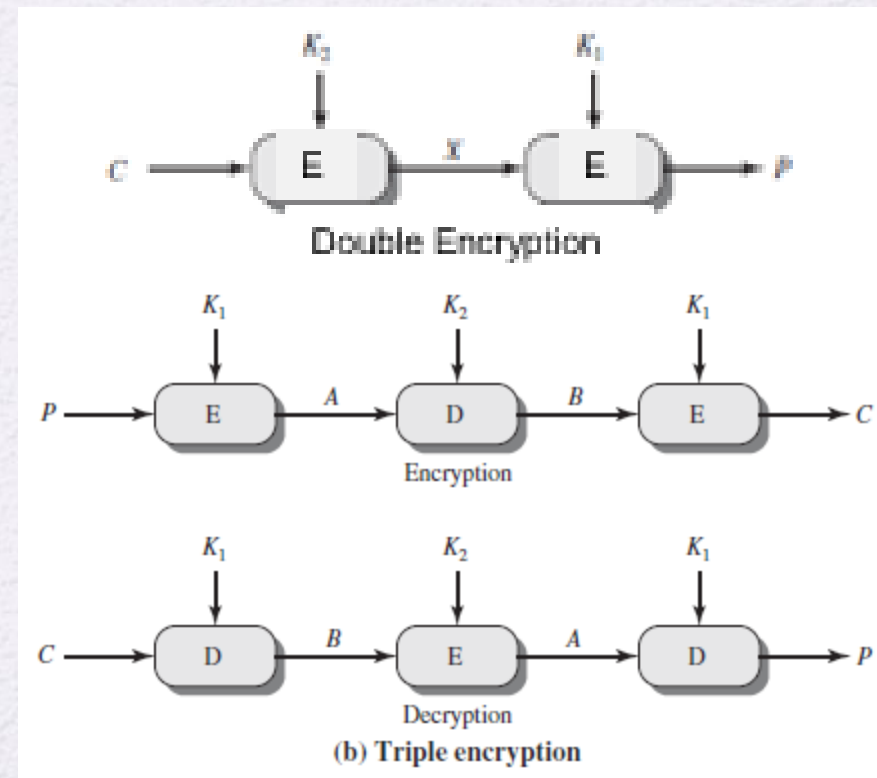
Background: Approaches to Address DES Weakness

1. Stronger Cipher e.g., AES
2. Keep DES and its investment in software but use:
 - Multiple Encryptions
 - Multiple Keys

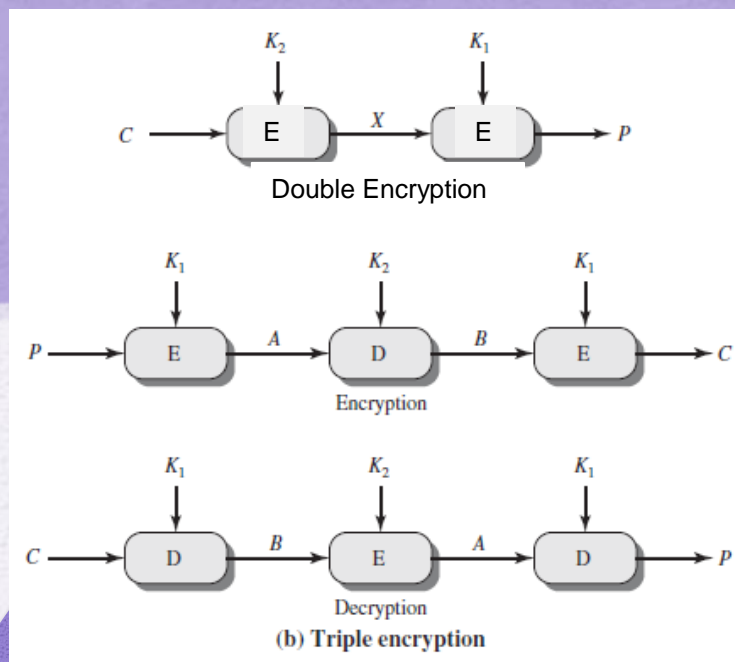


DES Alternatives

- To address potential vulnerability of DES to a brute-force attack:
 1. New Algorithm: AES
 2. Multiple encryption of DES
 - Double DES with two keys
 - 3DES with two keys
 - 3DES with 3 keys



Double DES: Meet-in-the-Middle Attack



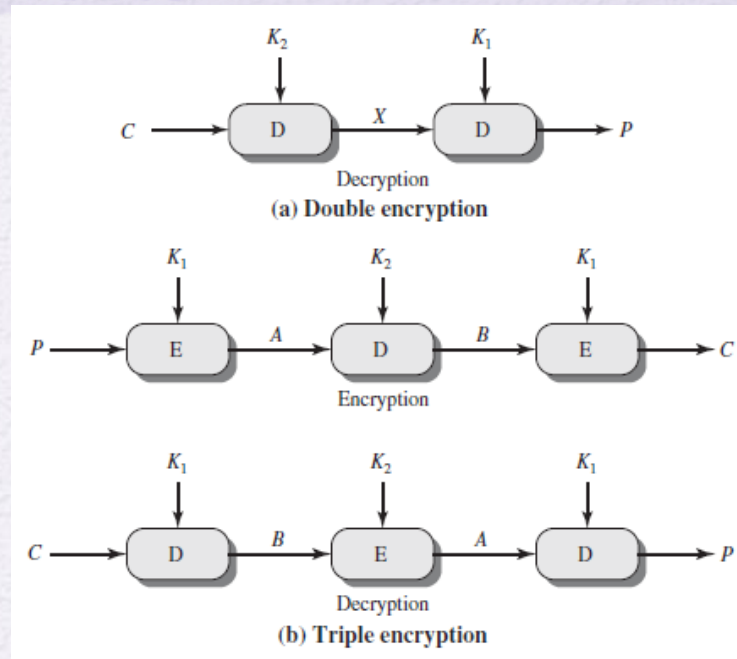
The use of double DES results in a mapping that is not equivalent to a single DES encryption

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 $56 \times 3 = 168$ 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 Three Keys

- 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:

$$\bullet C = E(K_3, D(K_2, E(K_1, P)))$$

Backward compatibility with DES is provided by putting:

$$\bullet K_3 = K_2 \text{ or } K_1 = K_2$$

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

Modes of Operation

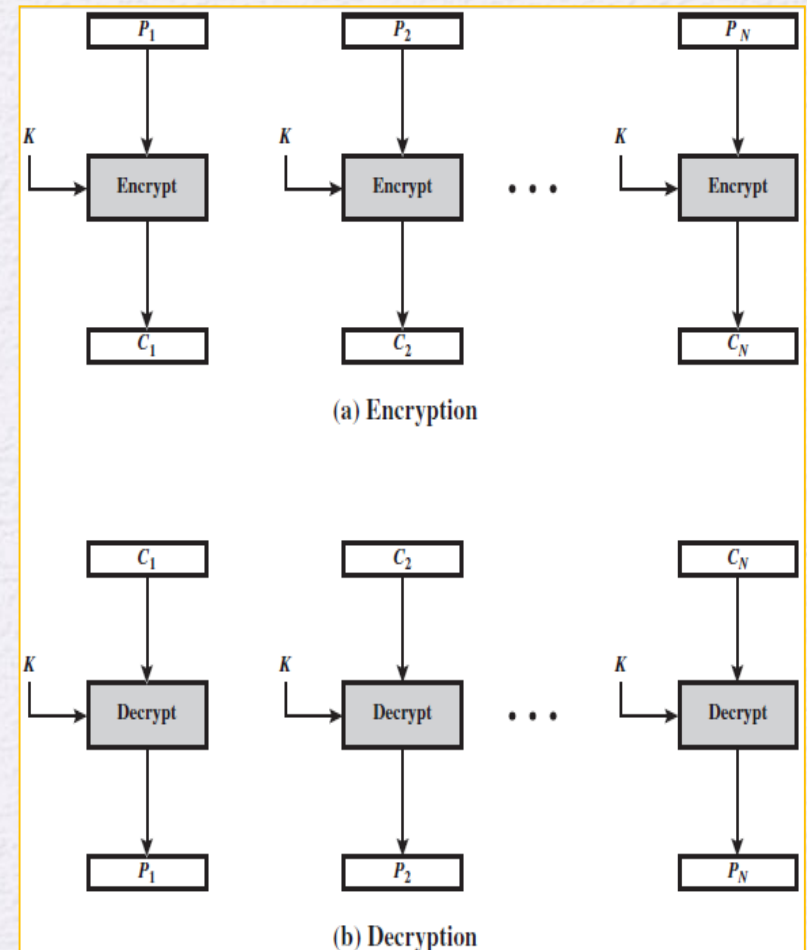
- If multiple blocks of plaintext are encrypted using the same key, a number of security threats arise.
- Mode of operation:
 - A technique for enhancing the effect of a cryptographic algorithm repeatedly apply a cipher's single-block operation to securely transform amounts of data larger than a block
- Five *modes of operation* have been defined by NIST
 - These modes are intended for use with any symmetric block cipher, including triple DES and AES

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 Code (ECB) Mode

- Plaintext is handled **one block (b-bit) at a time** and each block of plaintext is encrypted using the **same key**.
- For a message longer than b-bits, the procedure is simply to **break the message into blocks, padding** the last block if necessary.
- The term codebook is used because, for a given key, there is a unique ciphertext for every b-bit block of plaintext.
 - You can imagine a gigantic codebook in which there is an entry for every possible b-bit plaintext pattern showing its corresponding ciphertext.



ECB	$C_j = E(K, P_j)$	$j = 1, \dots, N$	$P_j = D(K, C_j)$	$j = 1, \dots, N$
-----	-------------------	-------------------	-------------------	-------------------

ECB Mode

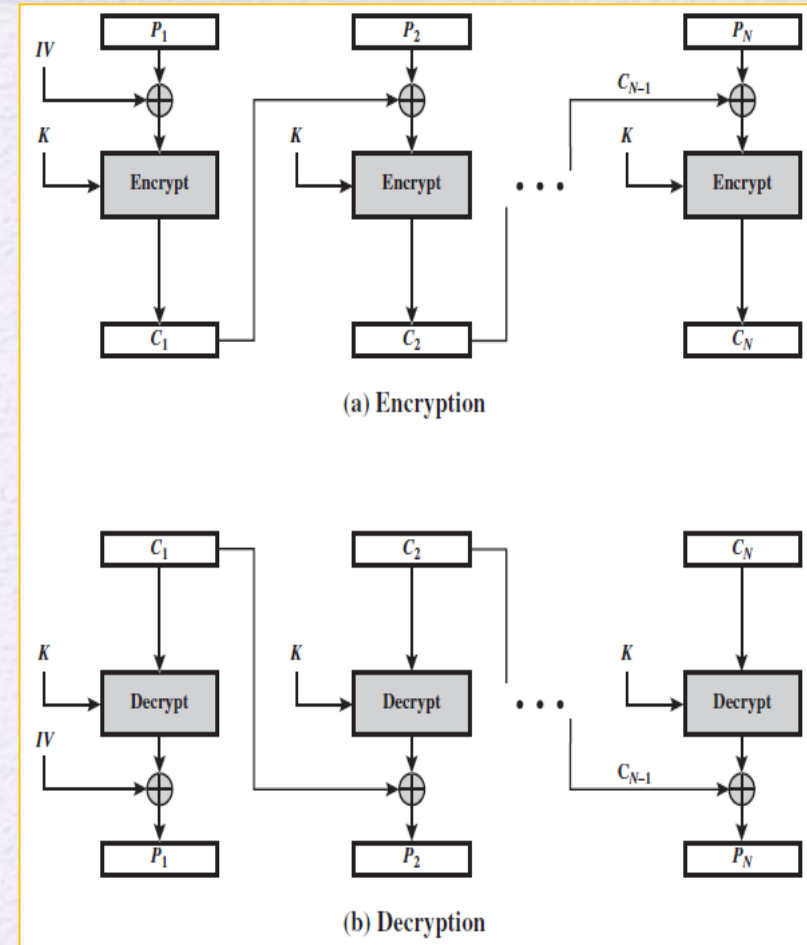
- The ECB method is ideal for a short amount of data
- Disadvantages:
 - The same plaintext block, if repeated, produces same ciphertext blocks
 - For lengthy messages, the ECB mode may not be secure.
 - If the message is highly structured, it may be possible for a **cryptanalyst to exploit these regularities.**

Properties for evaluating Block cipher modes of operation that are better than ECB

- Overhead
 - **The additional operations** for the encryption and decryption operation when compared to encrypting and decrypting in the ECB mode.
- Error could occur during the transmission of a cipher text. Two important properties:
 - Error recovery: the property that an **error in the i -th ciphertext block is inherited by only a few plaintext blocks** after which the mode **resynchronizes**.
 - Error propagation: the property that an **error in the i -th ciphertext block is inherited by the i -th and all subsequent plaintext blocks**.
- Diffusion
 - How the **plaintext statistics are reflected in the ciphertext**. Low entropy plaintext blocks should not be reflected in the ciphertext blocks. Roughly, low entropy equates to predictability or lack of randomness.
- Security
 - Whether or not the **ciphertext blocks leak information about the plaintext blocks**.

Cipher Block Chaining (CBC) Mode

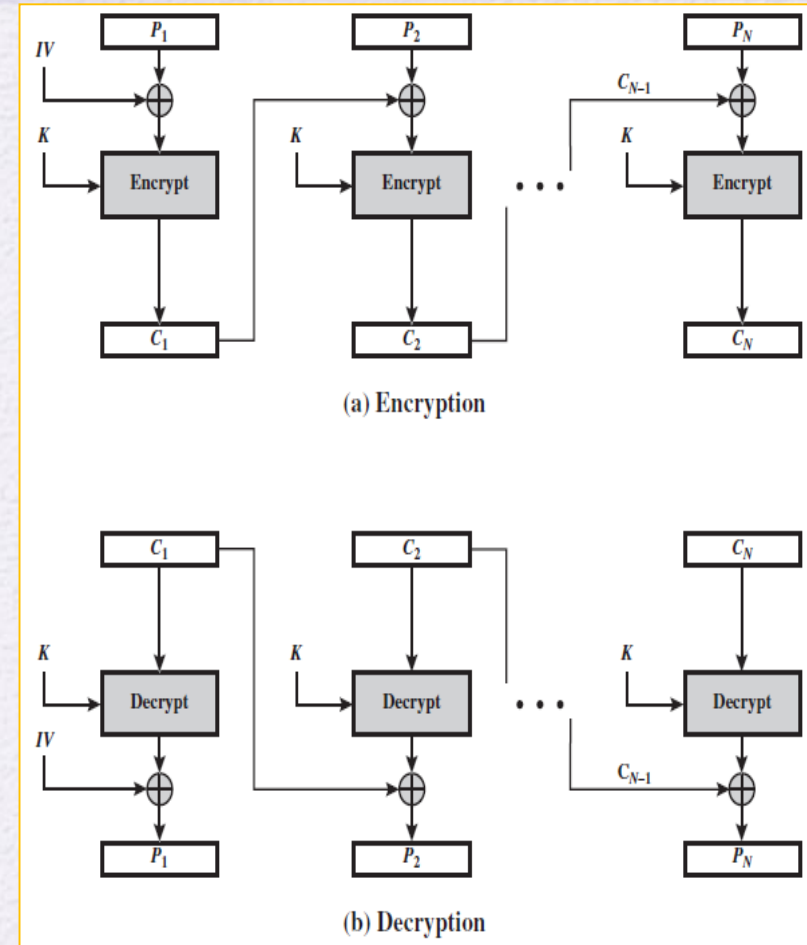
- There are two main ideas behind the *Cipher Block Chaining (CBC)* mode:
 - the encryption of all blocks are “chained together” such that ciphertext y_i depends not only on block x_i but on all previous plaintext blocks as well.
 - the encryption is randomized by using an initialization vector (IV).



CBC	$C_1 = E(K, [P_1 \oplus IV])$	$P_1 = D(K, C_1) \oplus IV$
	$C_j = E(K, [P_j \oplus C_{j-1}]) \quad j = 2, \dots, N$	$P_j = D(K, C_j) \oplus C_{j-1} \quad j = 2, \dots, N$

Cipher Block Chaining (CBC) Mode

- Input to the encryption algorithm is the XOR of the current plaintext block and the preceding ciphertext block; the same key is used for each block.
- To produce the first block of ciphertext, an initialization vector (IV) is XORed with the first block of plaintext.
 - IV is transmitted as first block to be used in decryption.
- As with the ECB mode, the CBC mode requires that the last block be **padded** to a full bits if it is a partial block.
- Advantages:**
 - The same plaintext block, if repeated, produces different ciphertext blocks
 - CBC mode can be used for authentication.



CBC	$C_1 = E(K, [P_1 \oplus IV])$	$P_1 = D(K, C_1) \oplus IV$
	$C_j = E(K, [P_j \oplus C_{j-1}]) \quad j = 2, \dots, N$	$P_j = D(K, C_j) \oplus C_{j-1} \quad j = 2, \dots, N$

Cipher Feedback Mode

- Block cipher can be converted into a stream cipher, using one of the three modes: cipher feedback (CFB) mode, output feedback (OFB) mode, and counter (CTR) mode.
- Why stream cipher?
 - Because it eliminates the need to **pad** a message to be an integral number of blocks. It also can operate in real time.
- 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 $b = 128$

There are three modes that make it possible to convert a block cipher into a stream cipher:

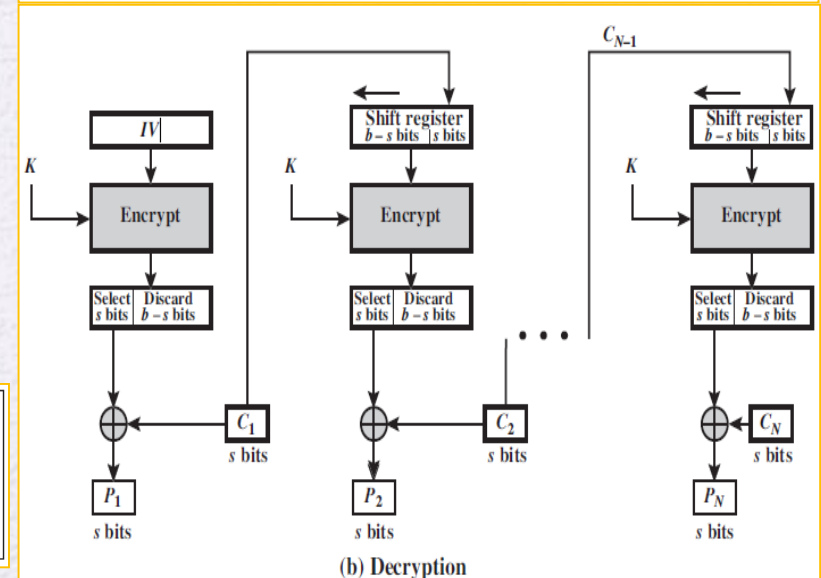
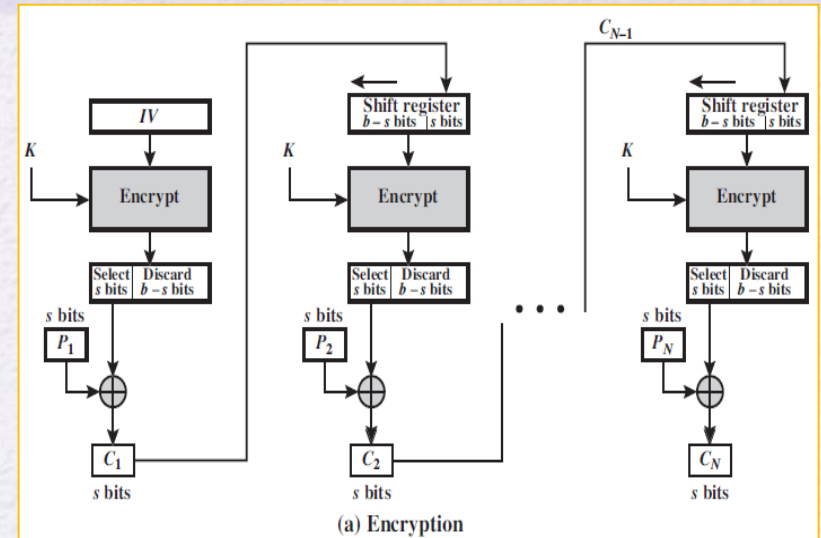
Cipher feedback (CFB) mode

Output feedback (OFB) mode

Counter (CTR) mode

Cipher Feedback (CFB) Mode

- CFB is illustrated in the figure:
 - Plaintext is divided into s -bits ($s < \text{block size}$, e.g. $s=8$)
 - As with CBC, the units of plaintext are chained together, so that the ciphertext of any plaintext unit is a function of all the preceding plaintext.



CFB	$I_1 = IV$	$I_1 = IV$
	$I_j = \text{LSB}_{b-s}(I_{j-1}) \parallel C_{j-1} \quad j = 2, \dots, N$	$I_j = \text{LSB}_{b-s}(I_{j-1}) \parallel C_{j-1} \quad j = 2, \dots, N$
	$O_j = E(K, I_j) \quad j = 1, \dots, N$	$O_j = E(K, I_j) \quad j = 1, \dots, N$
	$C_j = P_j \oplus \text{MSB}_s(O_j) \quad j = 1, \dots, N$	$P_j = C_j \oplus \text{MSB}_s(O_j) \quad j = 1, \dots, N$

Output Feedback (OFB) Mode

OFB is similar to CFB, but with following differences:

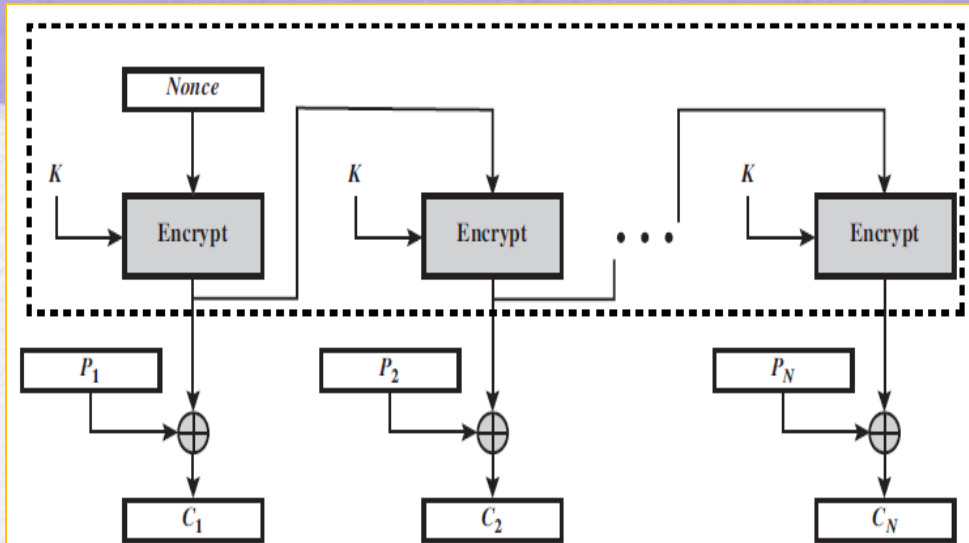
- Feedback:** For OFB, the output of the encryption function is fed back. In CFB, the output of the XOR unit is fed back.
- OFB mode operates on **full blocks**, whereas CFB operates on an **s-bit subset**.

IV must be unique to each execution (i.e., nonce) why? The sequence of encryption output blocks, O_i , depends only on the key and the IV and does not depend on the plaintext.

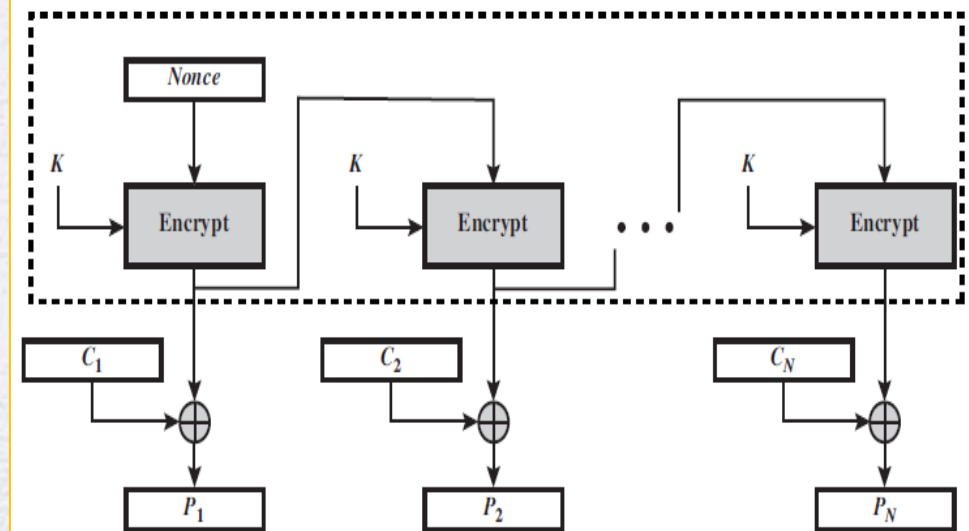
One advantage of the OFB method is that bit **errors in transmission do not propagate**.

The disadvantage of OFB is that it is:

- repeatedly encrypting the initialization vector may produce the same state that has occurred before. This is an unlikely situation, but in such a case, the plaintext will start to be encrypted by the same data as it was previously.
- more vulnerable to a message stream modification attack than is CFB



(a) Encryption

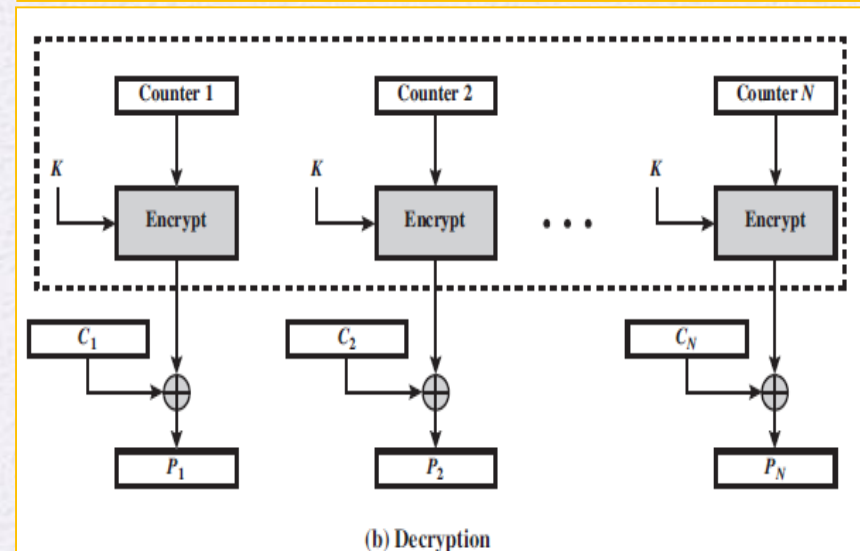
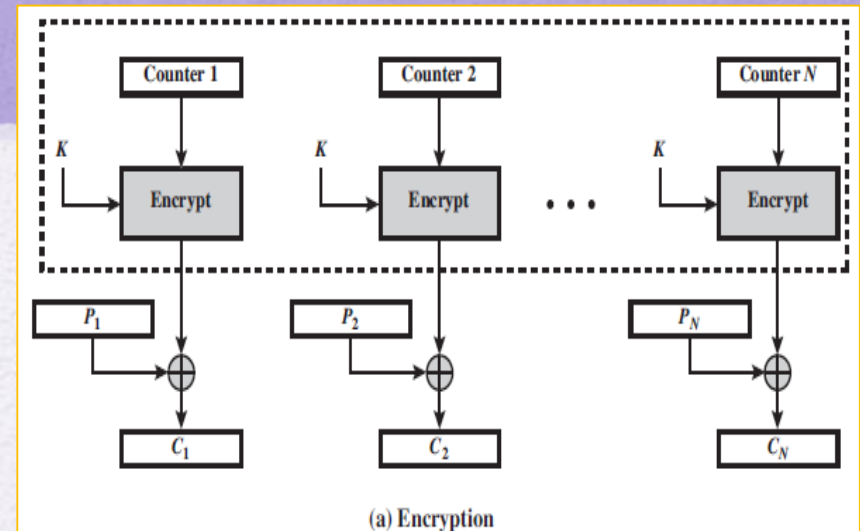


(b) Decryption

OFB	$I_1 = \text{Nonce}$	$I_1 = \text{Nonce}$
	$I_j = O_{j-1} \quad j = 2, \dots, N$	$I_j = \text{LSB}_{b-s}(I_{j-1}) \parallel C_{j-1} \quad j = 2, \dots, N$
	$O_j = E(K, I_j) \quad j = 1, \dots, N$	$O_j = E(K, I_j) \quad j = 1, \dots, N$
	$C_j = P_j \oplus O_j \quad j = 1, \dots, N-1$	$P_j = C_j \oplus O_j \quad j = 1, \dots, N-1$
	$C_N^* = P_N^* \oplus \text{MSB}_a(O_N)$	$P_N^* = C_N^* \oplus \text{MSB}_a(O_N)$

Counter (CTR) Mode

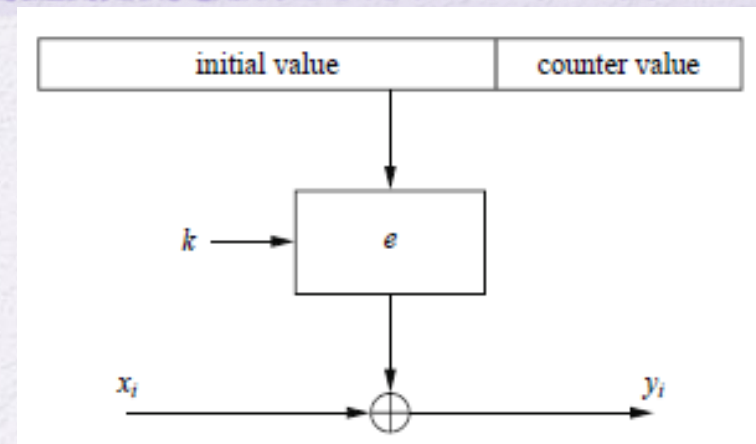
- CTR mode uses a counter where:
 - Counter size** equal to the plaintext block size.
 - Typically, the counter is **initialized** to some value and then **incremented** by 1 for each subsequent block (modulo 2^b , where b is the block size).
- For the last plaintext block, which may be a partial block of u bits, the most significant u bits of the last output block are used for the XOR operation; the remaining bits are discarded.
 - Unlike ECB, CBC, and CFB: **no padding**.
- As with the OFB mode, the initial counter value must be a **nonce**;
 - T_i must be different for all of the messages encrypted using the **same key**.
 - Further, all T_i values across all messages must be unique.
 - One way to ensure the uniqueness of counter values is to continue to increment the counter value by 1 across messages.
- If a counter value is used multiple times, then the confidentiality of all of the plaintext blocks corresponding to that counter value may be compromised.



CTR	$C_j = P_j \oplus E(K, T_j) \quad j = 1, \dots, N-1$	$P_j = C_j \oplus E(K, T_j) \quad j = 1, \dots, N-1$
	$C_N^* = P_N^* \oplus \text{MSB}_u[E(K, T_N)]$	$P_N^* = C_N^* \oplus \text{MSB}_u[E(K, T_N)]$

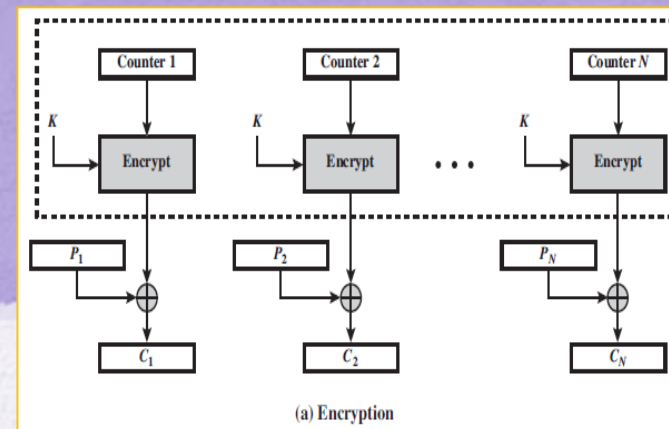
Counter (CTR) Mode

- One method to assure uniqueness of the count value, is to use initial value (IV) as shown in the figure. The initial value is changed when the counter completes its counting range.
- How often we should change IV? Lets go over an example.
- **Example:**
 - Assume a block size of 128 bits, such as an AES.
 - We construct 128-bit IV+Count as follows:
 - 96-bit initial value (IV)
 - 32-bit counter, which is initialized to zero.
 - For every block that is encrypted during the session, the counter is incremented but the IV stays the same.
 - In this case, the number of blocks we can encrypt without choosing a new IV is the range of the counter = $2^{32} = 4$ Giga blocks
 - Since every block consists of 16 bytes, a maximum of ≈ 64 Gigabytes, can be encrypted before a new IV must be generated.



Counter (CTR) Mode

Advantages



- **Efficiency:** encryption (or decryption) in CTR mode can be done in parallel (in hardware or software) on multiple blocks of plaintext or ciphertext.
 - For the chaining modes, the algorithm must complete the computation on one block before beginning on the next block.
- **Preprocessing:** The execution of the underlying encryption algorithm does not depend on input of the plaintext or ciphertext.
- **Random access:** The t -th block of plaintext or ciphertext can be processed in random-access fashion.
- **Simplicity:** CTR mode requires only the implementation of the encryption only (since decryption is the same.)
- **Provable security**

The following is read material

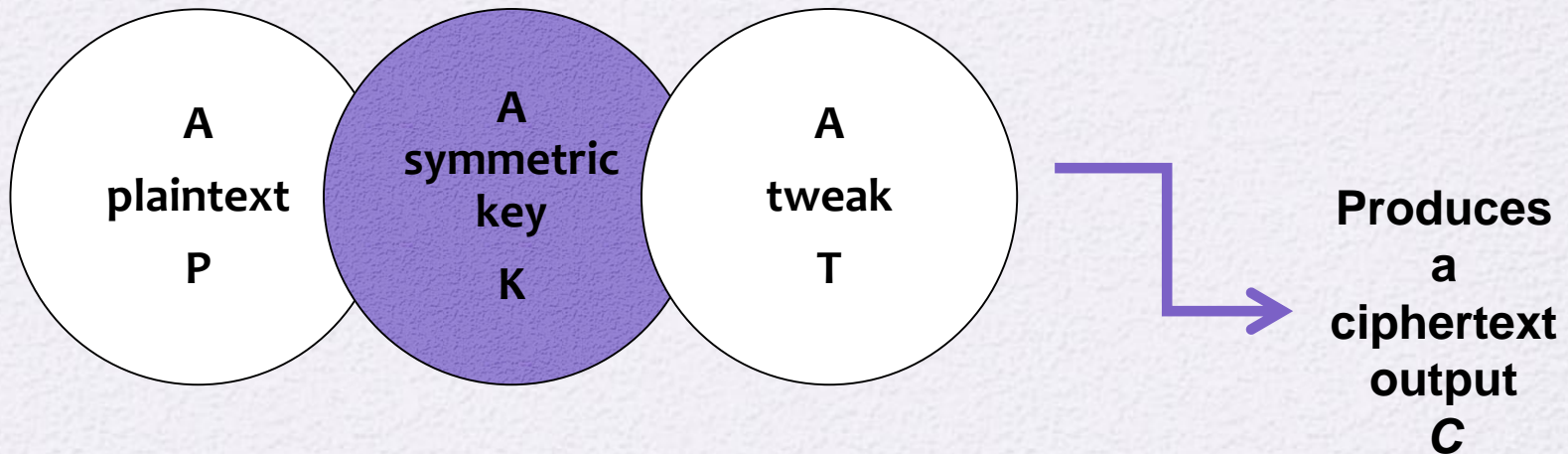
- This is reading material

XTS-AES Mode for Block-Oriented Storage Devices

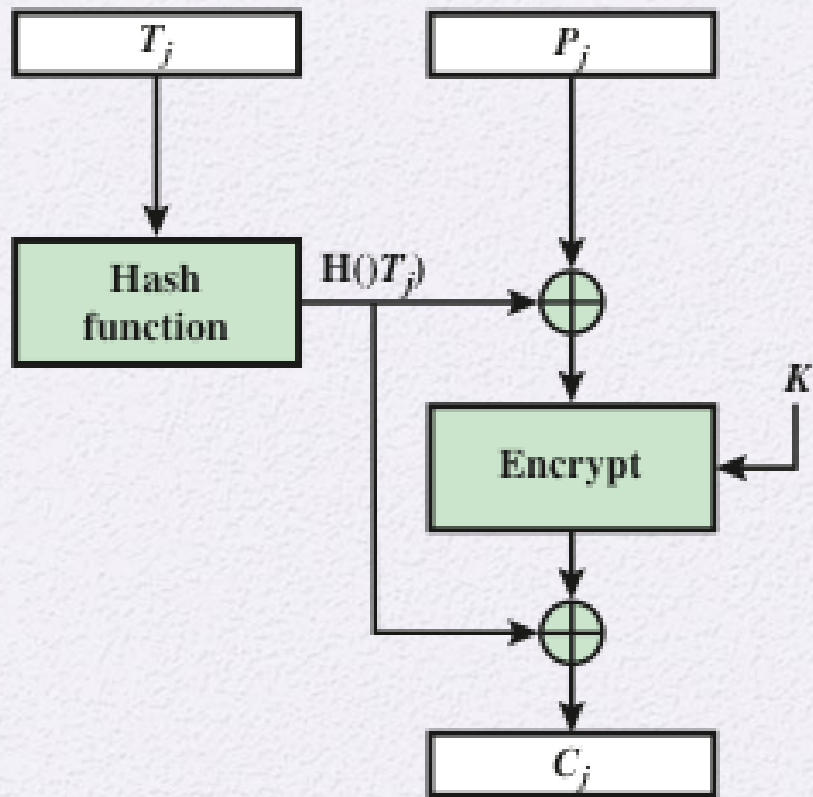
- Approved as an additional block cipher mode of operation by NIST in 2010
- Mode is also an IEEE Standard, IEEE Std 1619-2007
 - Standard describes a method of encryption for data stored in sector-based devices where the threat model includes possible access to stored data by the adversary
 - Has received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

Tweakable Block Ciphers

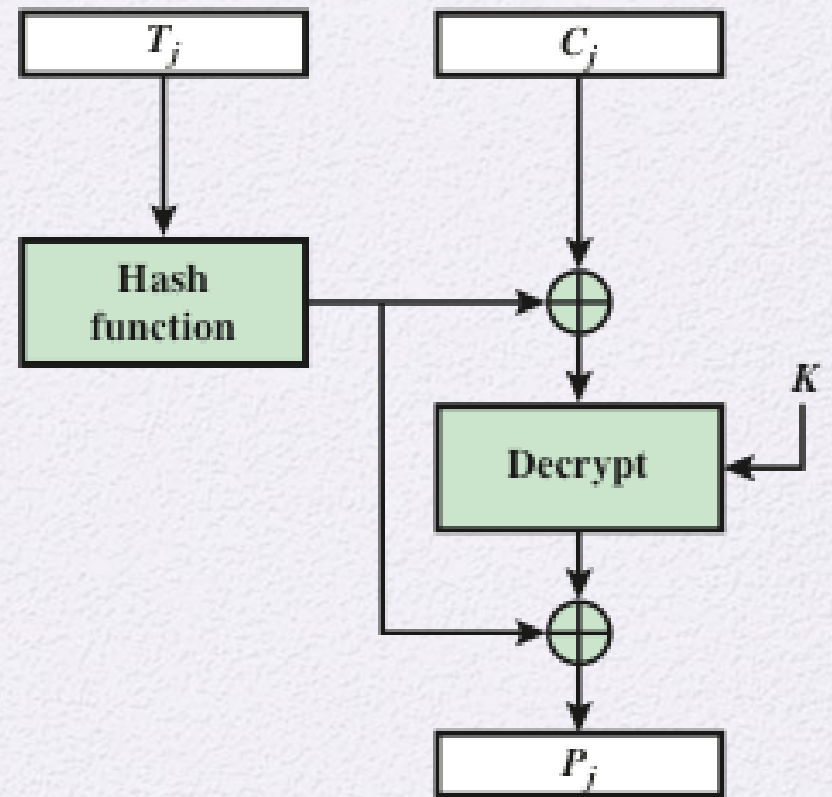
- XTS-AES mode is based on the concept of a *tweakable block cipher*
- General structure:
 - Has three inputs:



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



(a) Encryption



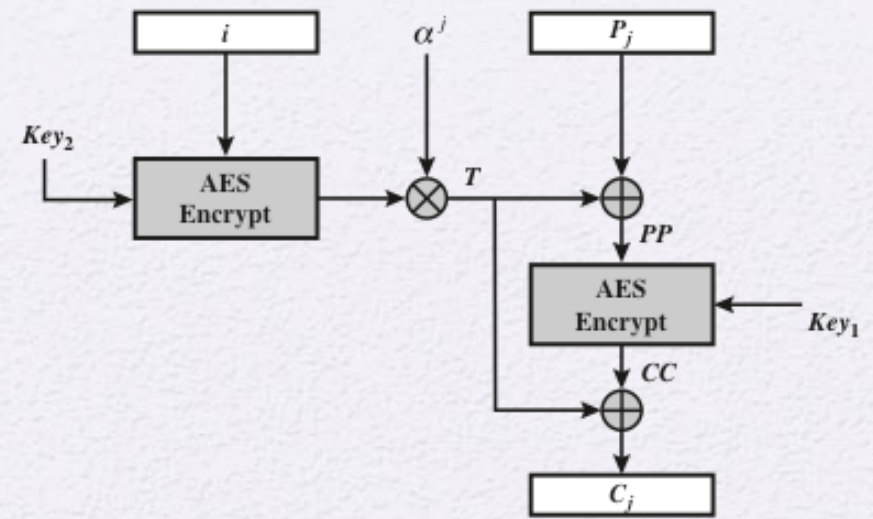
(a) Decryption

Figure 7.9 Tweakable Block Cipher

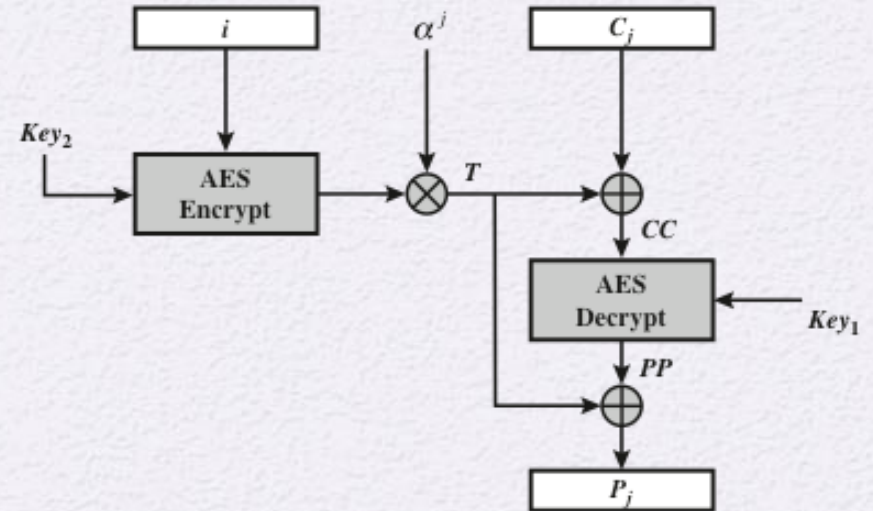
Storage Encryption Requirements

- 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

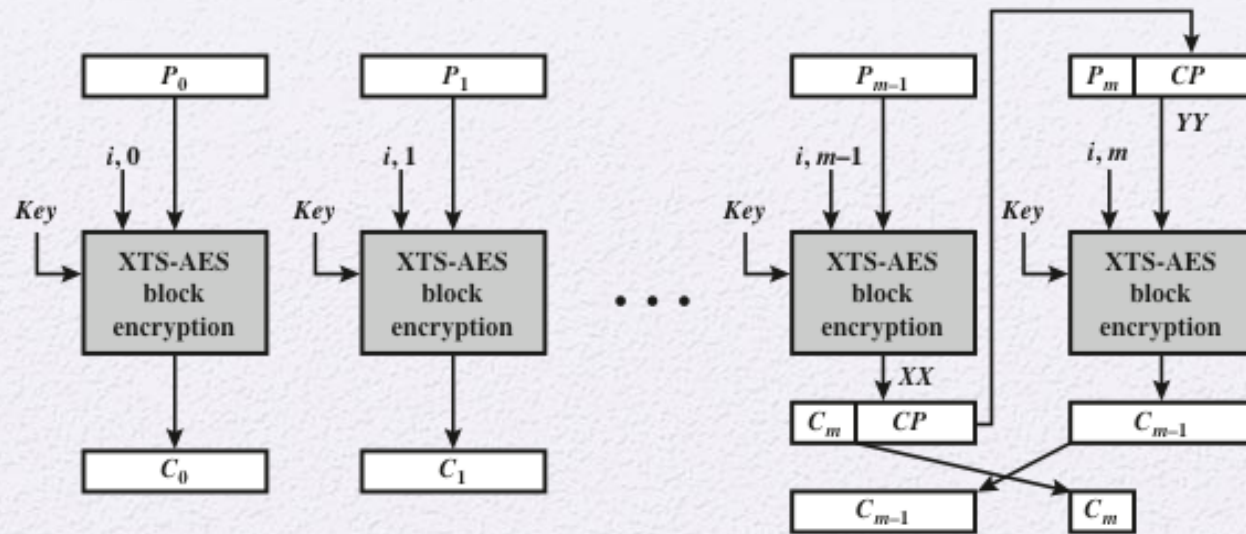


(a) Encryption

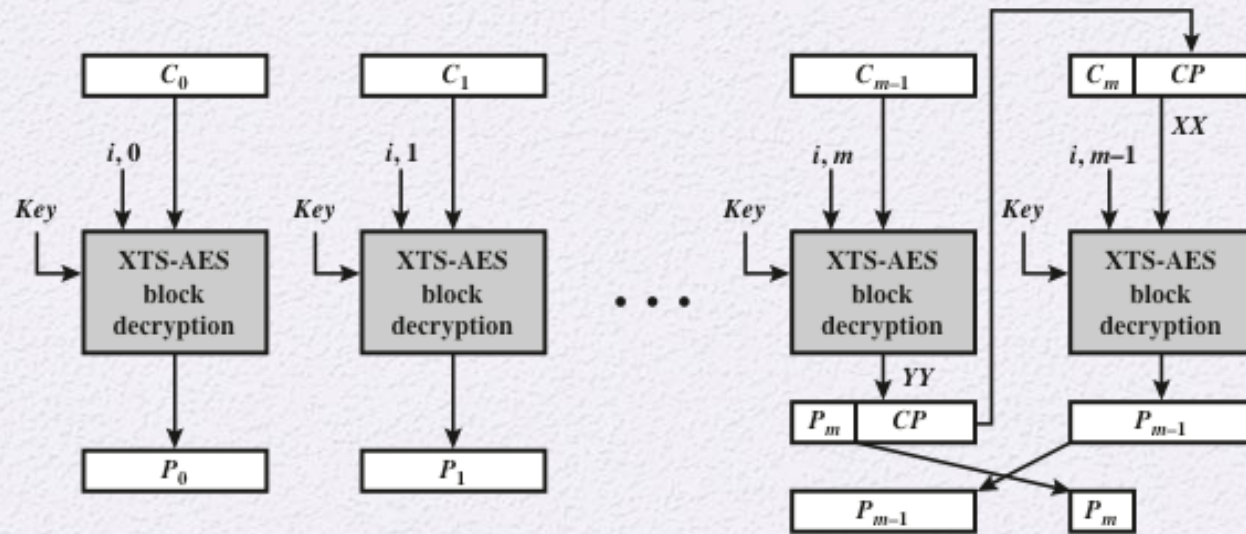


(b) Decryption

Figure 7.10 XTS-AES Operation on Single Block



(a) Encryption



(b) Decryption

Figure 7.11 XTS-AES Mode

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 is 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)	af411326466add24 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

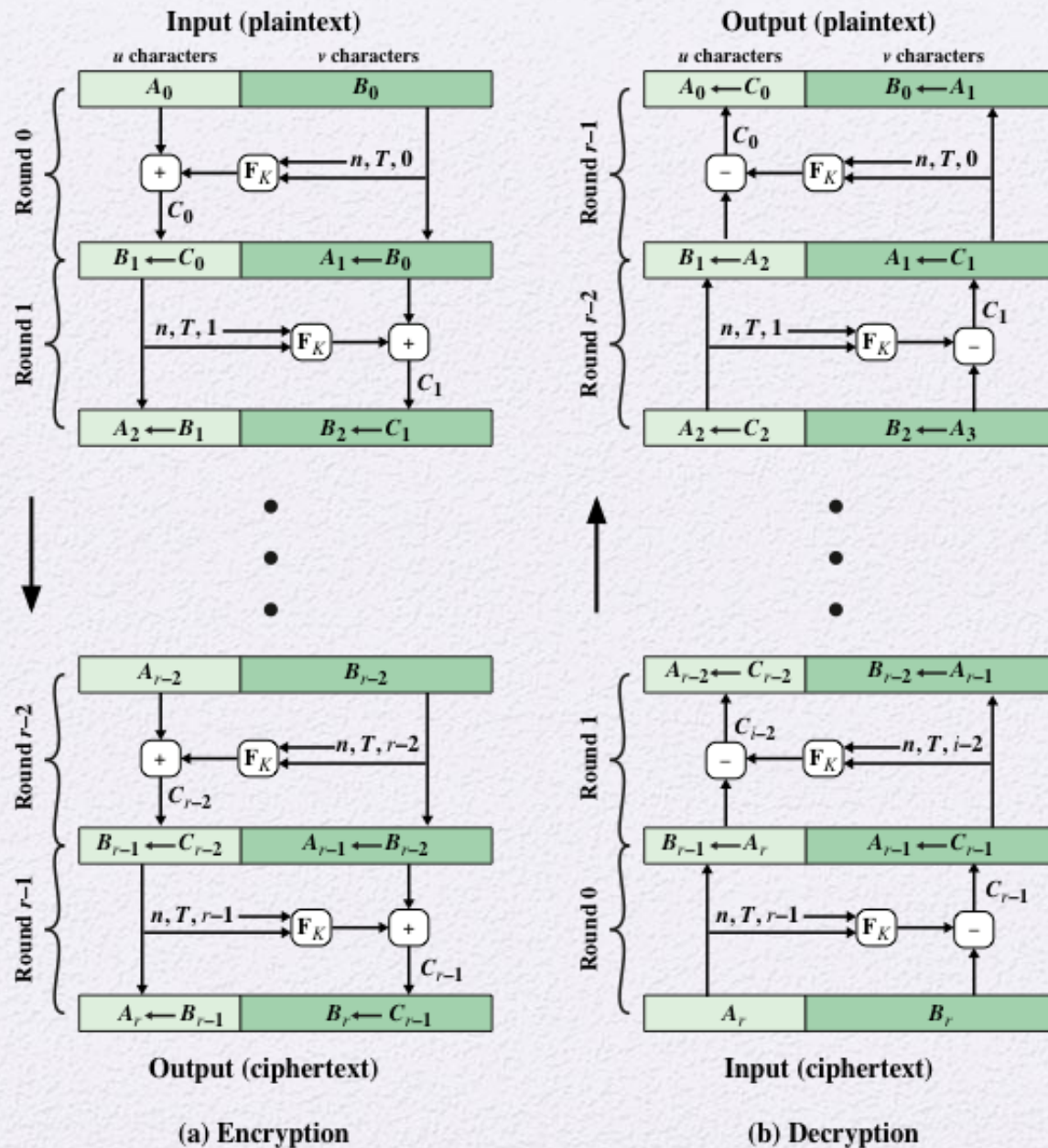


Figure 7.12 Feistel Structure for Format-Preserving Encryption

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

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 $0 \leq x < 2^{8s}$. The equivalent notation is $\text{STR}_2^{8s}(x)$.
$\text{LEN}(X)$	Length of the character string X .
$\text{NUM}_{\text{radix}}(X)$	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 $\text{radix}^{\text{LEN}(X)}$ 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 .
$\text{STR}_{\text{radix}}^m(x)$	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.
$[i \dots j]$	The set of integers between two integers i and j , including i and j .
$X[i \dots j]$	The substring of characters of a string X from $X[i]$ to $X[j]$, including $X[i]$ and $X[j]$.
$\text{REV}(X)$	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

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

Prerequisites:

Approved, 128-bit block cipher, CIPH;

Key, K , for the block cipher;

Input:

Nonempty bit string, X , such that $\text{LEN}(X)$ is a multiple of 128.

Output:

128-bit block, Y

Steps:

1. Let $m = \text{LEN}(X)/128$.

2. Partition X into m 128-bit blocks X_1, \dots, X_m , so that $X = X_1 \parallel \dots \parallel X_m$

3. Let $Y_0 = [0]^{16}$

4. For j from 1 to m :

4.i 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 $b = \lceil \lceil v \log_2(radix) \rceil / 8 \rceil$; $d = 4 \lceil b/4 \rceil + 4$

4. Let $P = [1]^{16} \parallel [2]^{16} \parallel [1]^{16} \parallel [radix]^3 \parallel [10]^{16} \parallel [u \bmod 256]^{16} \parallel [n]^4 \parallel [t]^4$.

5. For i from 0 to 9:

i. Let $Q = T \parallel [0]^{(-t-b-1) \bmod 16} \parallel [i]^{16} \parallel [NUM_{radix}(B)]^b$.

ii. Let $R = PRF_K(P \parallel Q)$.

iii. Let S be the first d bytes of the following string of $\lceil d/16 \rceil$ 128-bit blocks:
 $R \parallel CIPH_K(R \oplus [1]^{16}) \parallel CIPH_K(R \oplus [2]^{16}) \parallel \dots \parallel CIPH_K(R \oplus [\lceil d/16 \rceil - 1]^{16})$.

iv. Let $y = NUM_2(S)$.

v. If i is even, let $m = u$; else, let $m = v$.

vi. Let $c = (NUM_{radix}(A) + y) \bmod radix^m$.

vii. Let $C = STR_{radix}^m(c)$.

viii. Let $A = B$.

ix. Let $B = C$.

6. Return $Y = A \parallel B$.

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 $t > 0$, $P = [radix]^1 \parallel [t]^1 \parallel [n]^1 \parallel [NUM_{tweakradix}(T)]^{13}$;
 else $P = [radix]^1 \parallel [0]^1 \parallel [n]^1 \parallel [0]^{13}$.
4. Let $J = CIPH_K(P)$
5. For i from 0 to 9:
 - i. Let $Q \leftarrow [i]^1 \parallel [NUM_{radix}(B)]^{15}$
 - ii. Let $Y \leftarrow CIPH_J(Q)$.
 - iii. Let $y \leftarrow NUM_2(Y)$.
 - iv. If i is even, let $m = u$; else, let $m = v$.
 - v. Let $c = (NUM_{radix}(A) + y) \bmod radix^m$.
 - vi. Let $C = STR_{radix}^m(c)$.
 - vii. Let $A = B$.
 - viii. Let $B = C$.
6. Return $Y = A \parallel B$.

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 $radix \in [2 \dots 2^{16}]$;
 Range of supported message lengths, $[minlen \dots maxlen]$,
 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 $n \in [minlen \dots maxlen]$;
 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 $A = X[1 \dots u]$; $B = X[u + 1 \dots n]$.
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 $P = \text{REV}([\text{NUM}_{radix}(\text{REV}(B))]^{12}) \parallel [W \oplus \text{REV}([1^4])]$.
 - 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 $C = \text{REV}(\text{STR}_{radix}^m(c))$.
 - vii. Let $A = B$.
 - viii. Let $B = C$.
5. Return $A \parallel B$.

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

