Chapter – 3

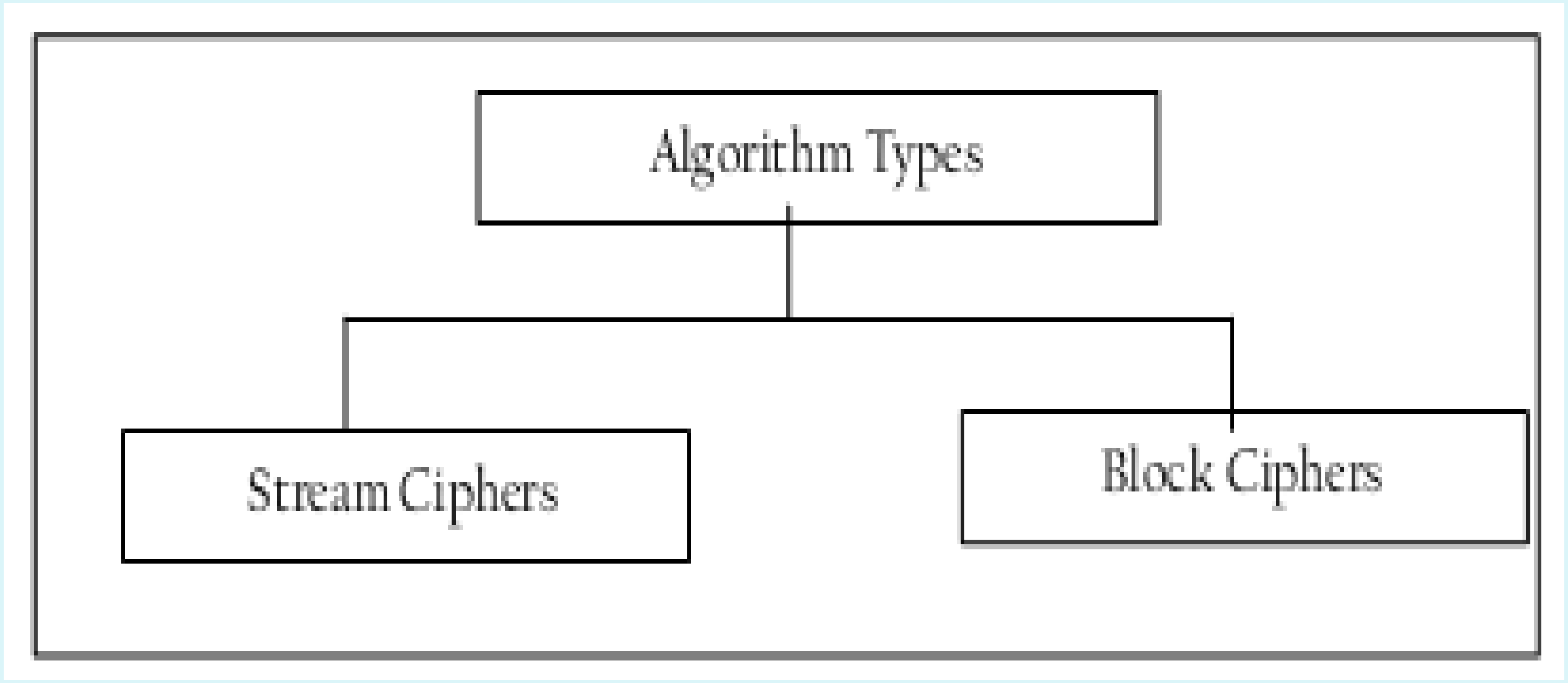
Block Cipher Principles and Operation

Outline

- Block cipher principles
  - Block cipher Vs Stream cipher
- Block cipher Algorithm modes
  - Electronic Code Book (ECB)
  - Cipher Block Chaining (CBC)
  - Cipher Feedback (CFB)
  - Output Feedback (OFB)
  - Counter Mode (CTR)

1. Block vs Stream Ciphers

- Regardless of the techniques used, at a broad level, the generation of cipher text from plain text itself can be done in two basic ways, *stream ciphers and block ciphers.*



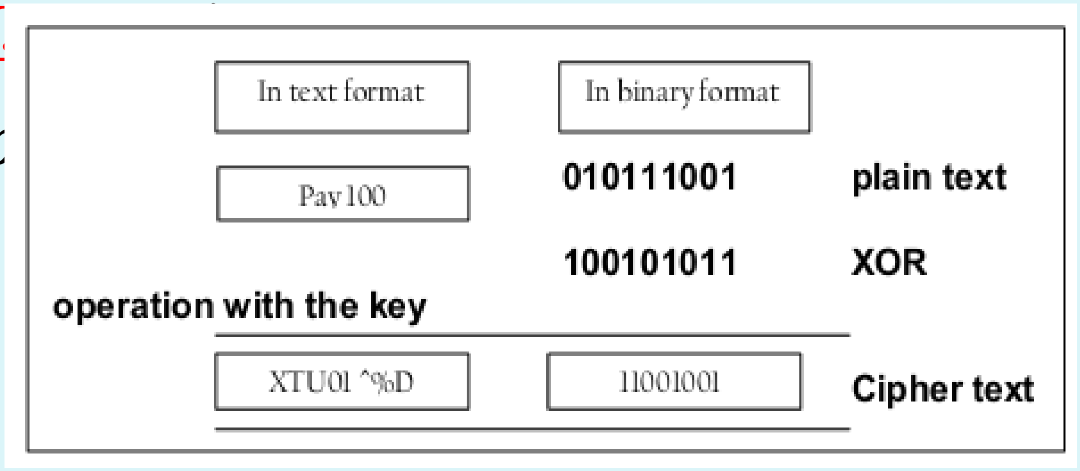
Stream Ciphers

- Process messages a *bit or byte* at a time of en/decrypting.
- *Suppose the original message (plain text) is pay 100 in ASCII (i.e. text format).*
- *When we convert these ASCII characters to their binary values, let us assume that it translate to 01011100 (hypothetically, just for simplicity, in reality, the binary text would be much larger as each text character takes seven bits).*

Cont . . .

- *Suppose the key to be applied is 10010101in binary.*
- *And also assume that we apply XOR logic as encryption algorithm.*
  - *XOR produces an output of 1 only if one input is 0 and the other is 1*
- *As a result of applying one bit of key for every respective bit of the original message, the cipher text is generated as 11001001 in binary (ZTU*
- *The decryption also happ*

very time consuming and actually unnecessary in real



In text format	In binary format	
Pay 100	010111001	plain text
	100101011	XOR
operation with the key		
XTU01 ^%D	11001001	Cipher text

Block Ciphers

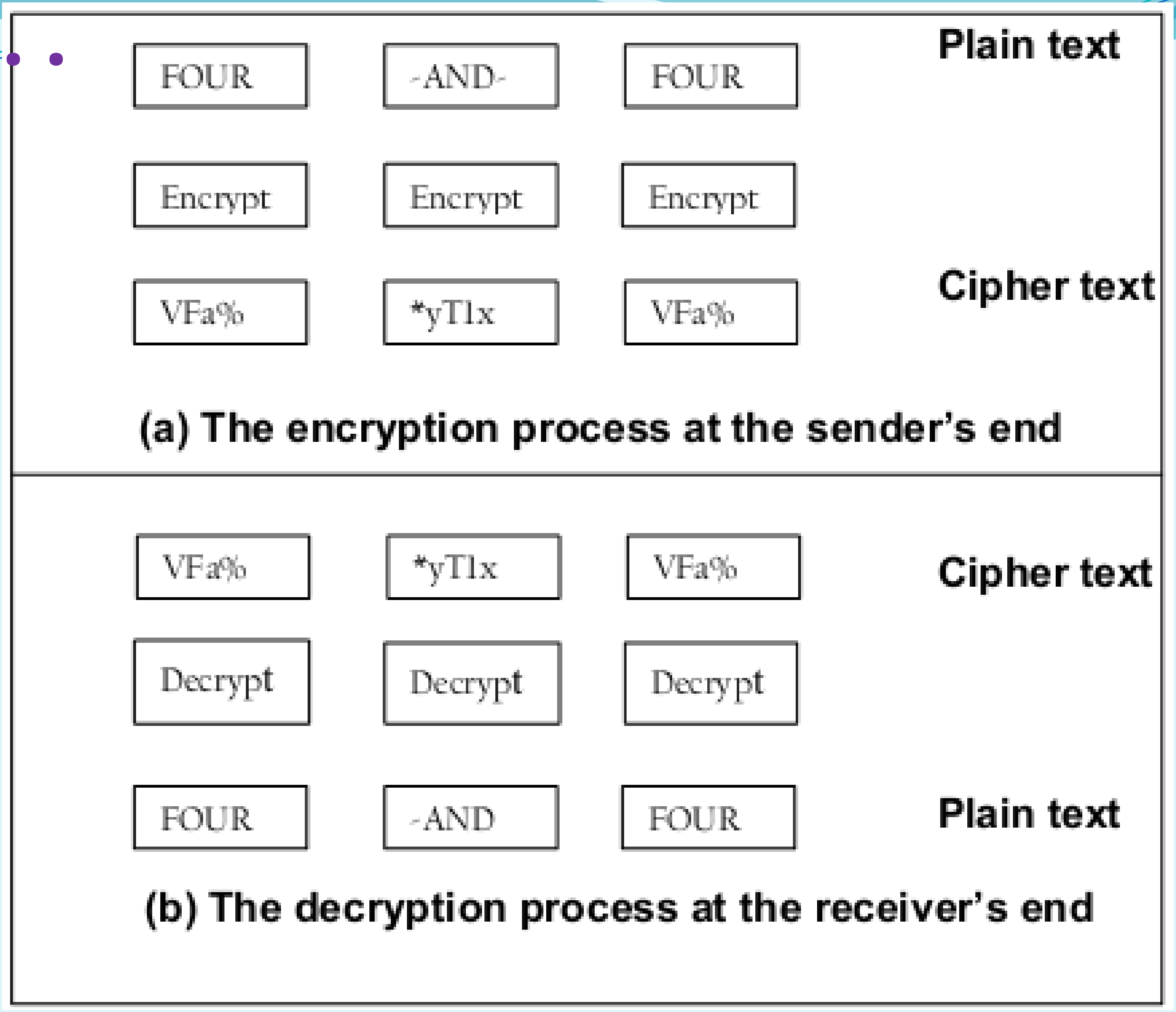
- Rather than encrypting one bit at a time, *a block of bits* is encrypted at one go.
  - *Suppose we have a plain text FOUR_AND_FOUR that needs to be encrypted.*
  - *Using block cipher, FOUR could be encrypted first, followed by _AND_ and finally FOUR.*
- Thus, one block of characters gets encrypted at a time.
- During decryption, each block would be translated back to the original form.
- An obvious problem with block ciphers is repeating text.
  - *For repeating text patterns, the same cipher is generated.*
  - *Therefore, it could give a clue to the cryptanalyst regarding the original plain text.*

Cont . . .

- *The cryptanalyst can look for repeating strings and try to break them.*
- *If succeeds in doing so, there is a danger that a relatively large portion of the original message is broken into, and therefore, the entire message can then be revealed with more effort.*
- *Even if the cryptanalyst cannot guess the remaining words, suppose she changes all debit to credit and vice versa in a funds transfer message, it could cause havoc!*

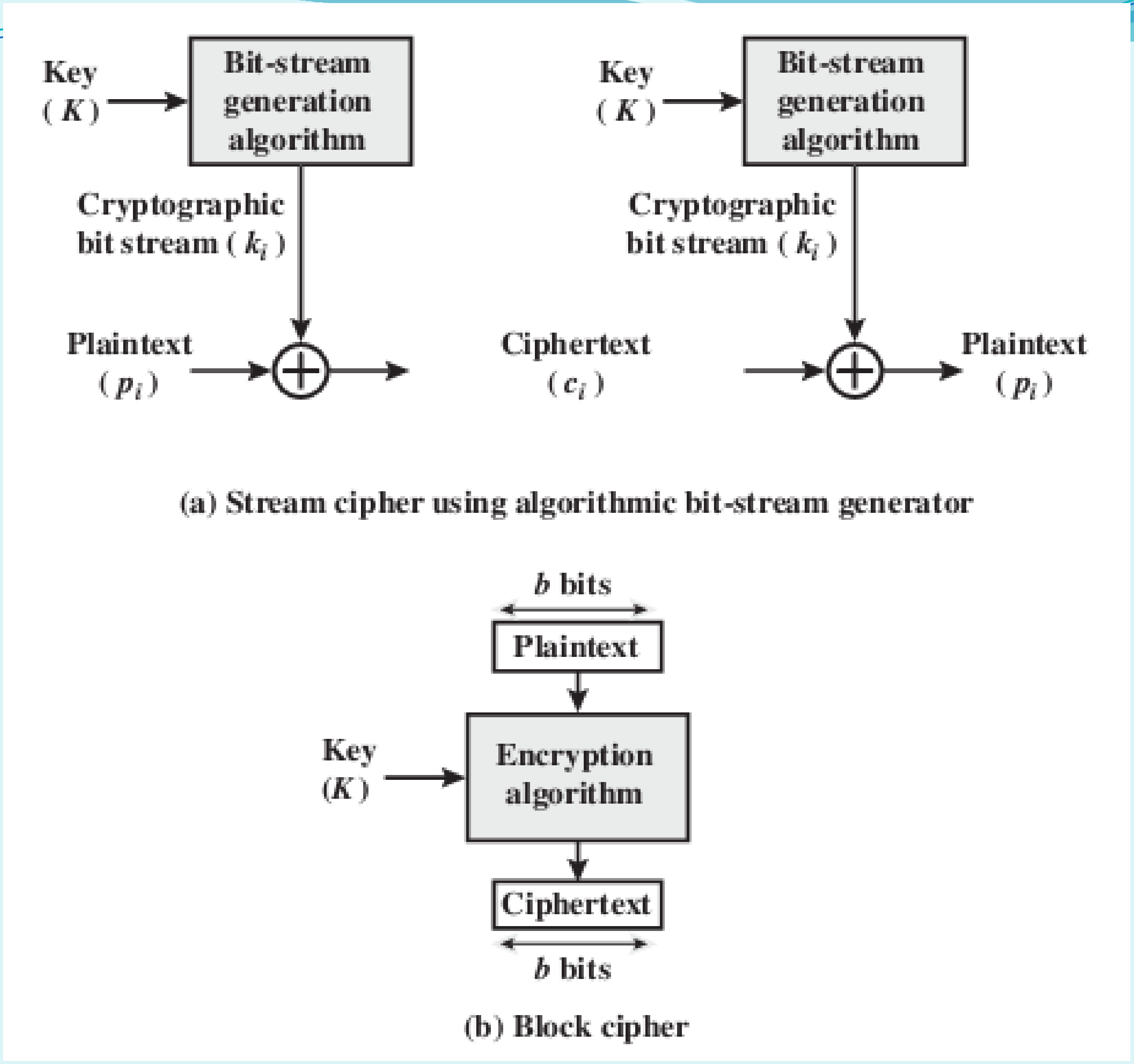
- Practically the blocks use in the block cipher generally contains *64 bits or more*
- Block ciphers are ***used more often*** in computer based cryptographic algorithms as compared to stream ciphers.

Cont . . .



- Block Cipher example

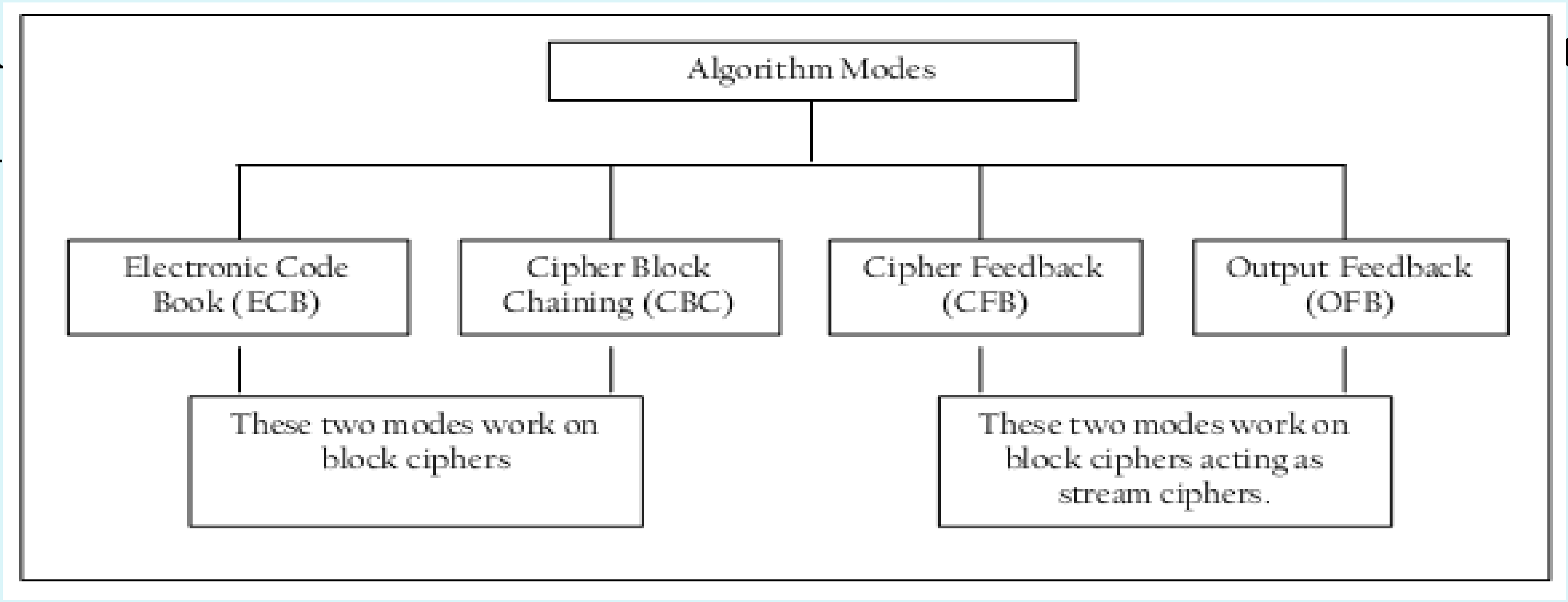
Cont . . .


Key (K)
Bit-stream generation algorithm
Cryptographic bit stream (k_i)
Plaintext (p_i)
Key (K)
Bit-stream generation algorithm
Cryptographic bit stream (k_i)
Ciphertext (c_i)
Plaintext (p_i)
(a) Stream cipher using algorithmic bit-stream generator
b bits
Plaintext
Key (K)
Encryption algorithm
Ciphertext
b bits
(b) Block cipher

- Stream Cipher and Block Cipher

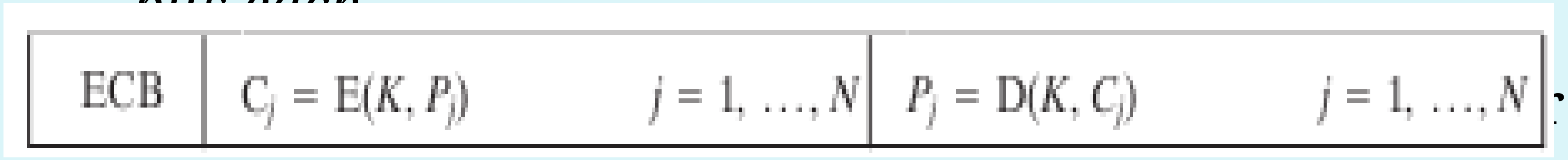
2. Block cipher Algorithm Modes

- An algorithm mode is a combination of a series of basic algorithm steps on block cipher, and some kind of feedback from previous step.
- Defines the details of the cryptographic algorithm
- There are five important algorithm modes, namely, Electronic Code Book (ECB), Cipher Block Chaining (CBC), Ciph er Mod


Algorithm Modes
Electronic Code Book (ECB)
Cipher Block Chaining (CBC)
Cipher Feedback (CFB)
Output Feedback (OFB)
These two modes work on block ciphers
These two modes work on block ciphers acting as stream ciphers.

A) Electronic Code Book (ECB)

- Simplest mode of operation.
- A block cipher takes a fixed-length block of text of length ***b*** bits and a key as input and produces a ***b-bit*** block of ciphertext.
- If the amount of plaintext to be encrypted is greater than ***b*** bits, then the block cipher can still be used by breaking the plaintext up into ***b*** -bit blocks, *padding* the last block if necessary.
    - *Mostly plain text message is divided into blocks of 64 bits each*

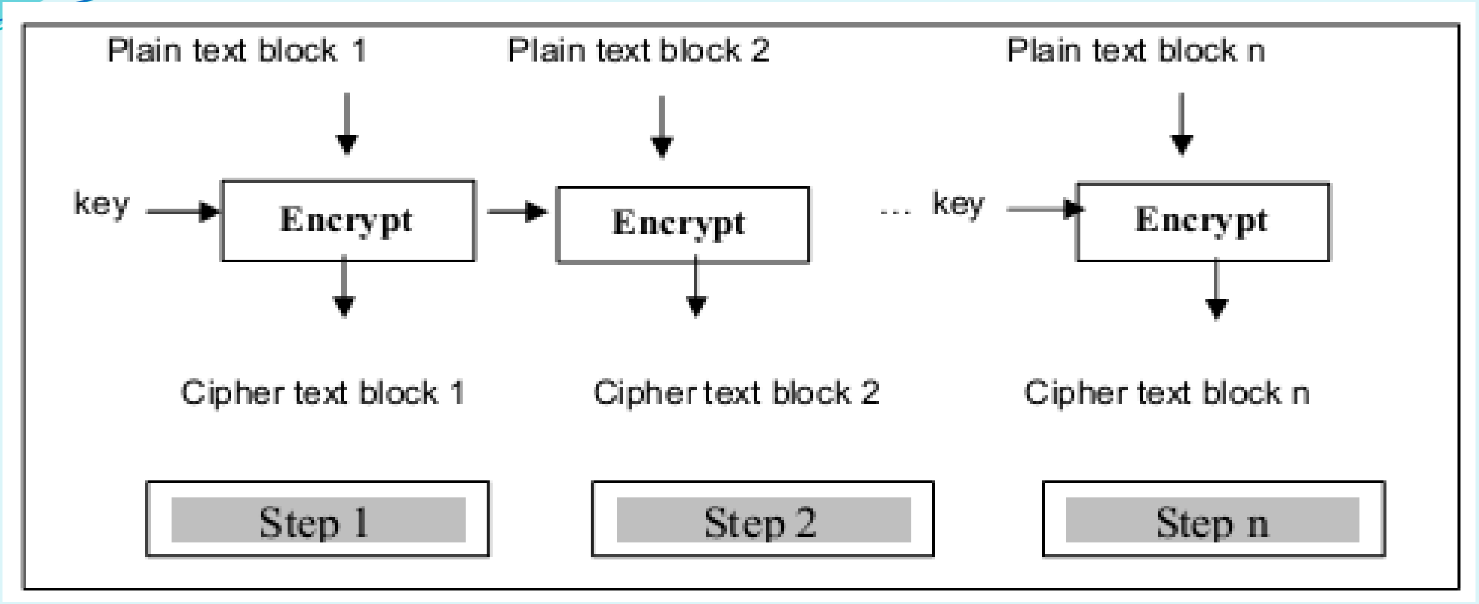


ECB	$C_j = E(K, P_j) \quad j = 1, \ldots, N$	$P_j = D(K, C_j) \quad j = 1, \ldots, N$

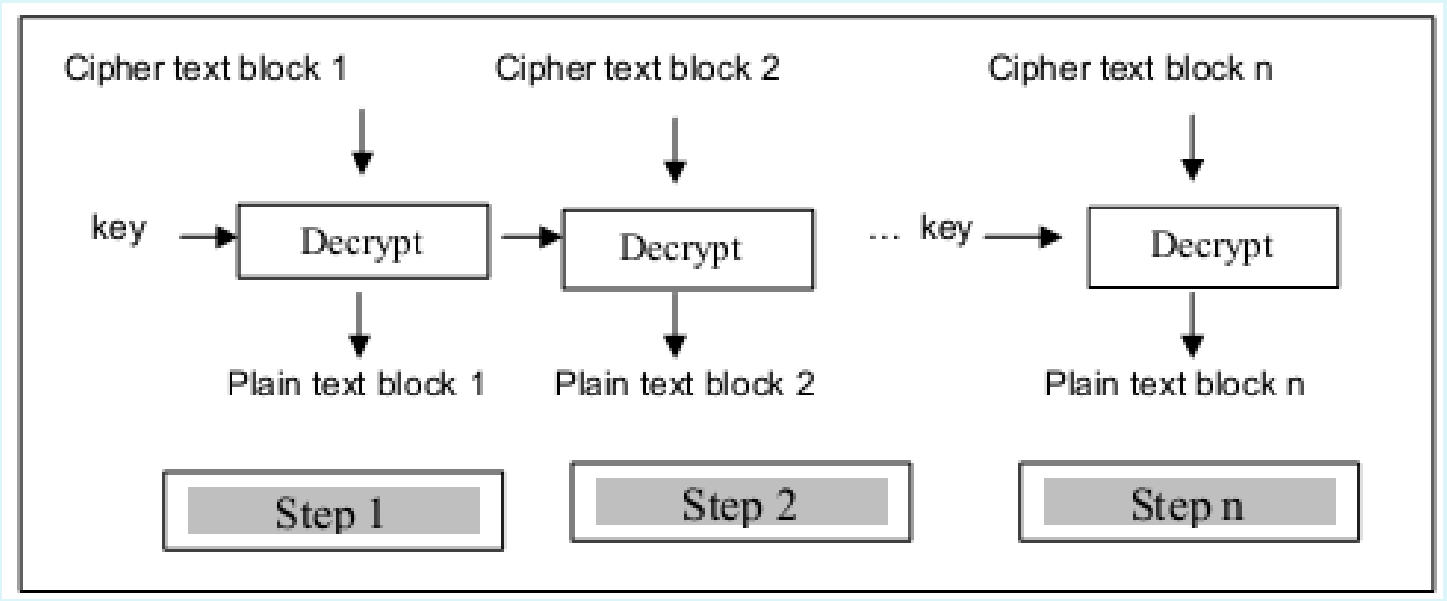
-

blocks.

Cont . . .



- Encryption



- Decryption

Cont . . .

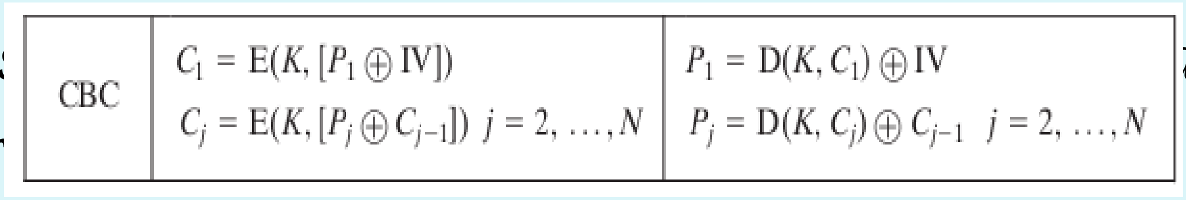
- At the receiver's end, the incoming data is divided into 64-bit blocks, and by using the same key as was used for encryption, each block is decrypted to produce the corresponding plain text block.

Problem

- Since a single key is used for encrypting all the blocks of a message, if a plain text block repeats in the original message, corresponding cipher text block will also repeat in the encrypted message.
- Therefore, ECB is suitable only for encrypting small messages, where the scope for repeating the same plain

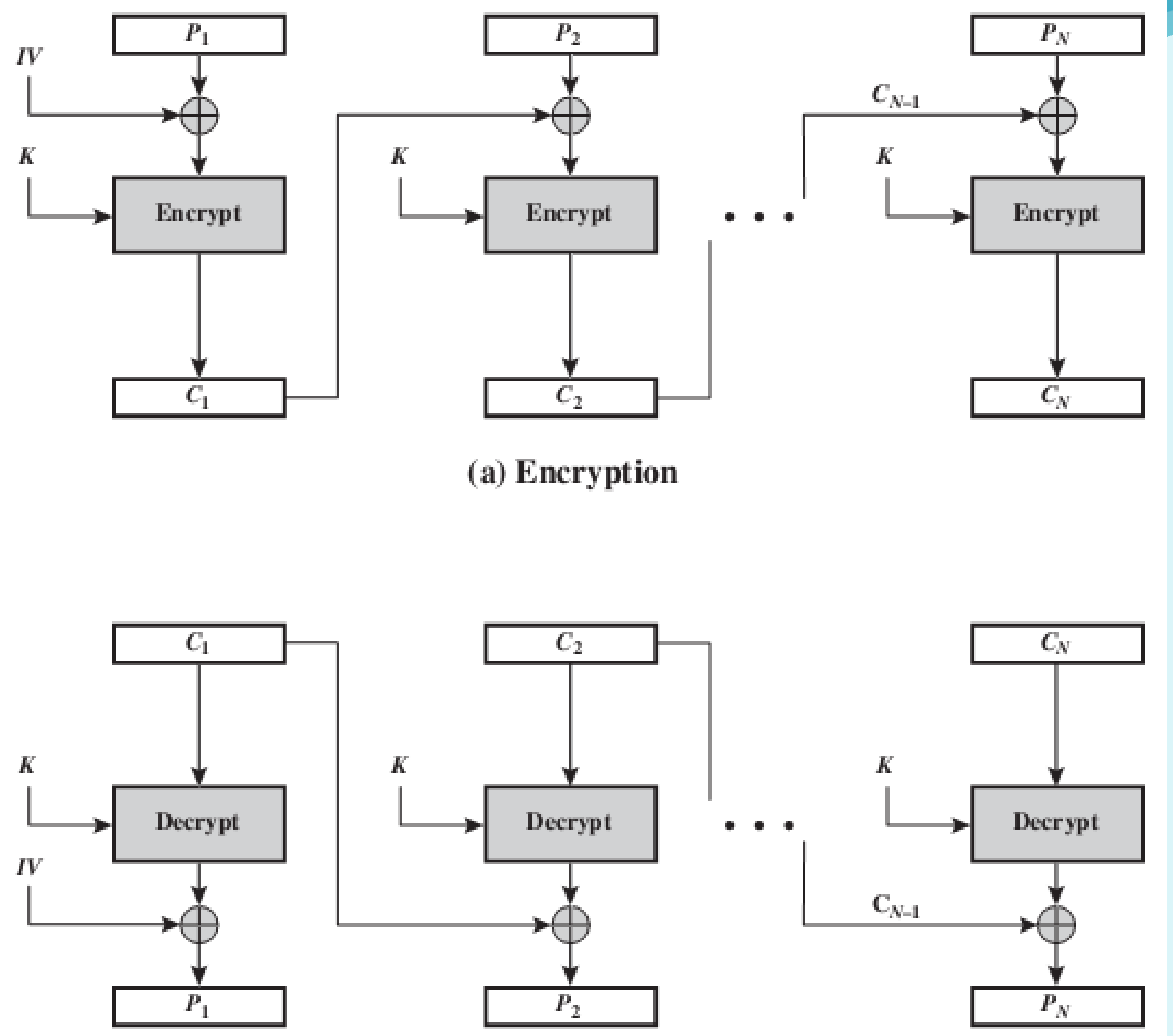
B) Cipher Block Chaining (CBC)

- Overcome ECB problem,
- CBC mode ensures that even if a block of plain text repeats in the input, these two (or more) identical plain text blocks yield totally different cipher text blocks in the output.
- For this, a feedback mechanism is used.
  - *The results of the encryption of the previous block are fed back into the encryption of the current block.*
  - *That is, each cipher block is used to modify encryption of next block.*
  - *Thus* *dent on the corr* *and all the*



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

Cont . . .



(a) Encryption

(b) Decryption

Cont . . .

(1) As shown in figure, first step receives 2 inputs: first block of plain text and random block of text, called as *Initialization Vector (IV).*

- *IV has no special meaning: it is simply used to make* each message unique.
- *Since the value of IV is randomly generated, the likelihood of it repeating in two different messages is quite rare.*
- *Consequently, IV helps in making the cipher text somewhat unique or at least quite different from all the other cipher texts in a different message.*
- *No need to keep IV secret, it can be known to*

Cont . . .

- *The output of step 1 is cipher text block 1, which is also one of the two inputs to the second encryption (i.e., Cipher text block 1 is also an IV for step 2 Similarly, Cipher text block 2 is also an IV for step 3, and so on.)*
- *The key used for encryption is what needs to be kept secret. However, in practice, for maximum security, both the key and the IV are kept secret.*

- The first block of plain text and IV are combined using XOR and then encrypted using a key to produce the first cipher text block.
- The first cipher text block is then provided as a feedback to the next plain text block, as explained below

Cont . . .

(2) In the second step, the second plain text block is XORed with the output of step 1, i.e. the first cipher text block.

- *It is then encrypted with the same key, as used in step 1.*
- *This produces cipher text block 2.*

(3) In the third step, the third plain text block is XORed with the output of step 2, i.e. the second cipher text block.

- *It is then encrypted with the same key, as used in step 1.*

(4) This process continues for all the remaining plain text blocks of the original message.

Notes

- *Remember that the IV is used only in the first plain text block.*

C) Cipher Feedback (CFB)

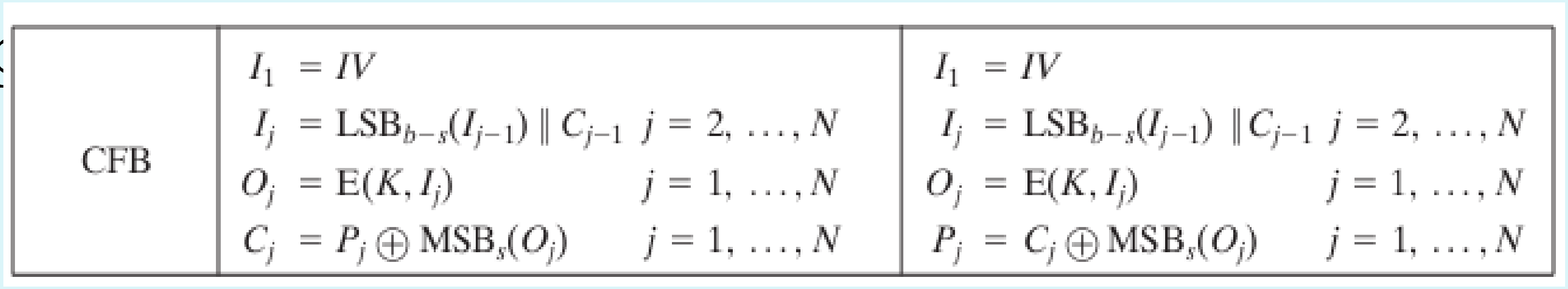
- Not all applications can work with blocks of data.
- Security is also required in applications that are character-oriented.
- For instance, an operator can be typing keystrokes at a terminal, which need to be immediately transmitted across the communications link in a Secure manner, i.e. by using encryption.
- In such situations, stream cipher must be used.
- The Cipher Feedback (CFB) mode is useful in such cases
- Let us understand how CFB mode works, assuming that we are dealing with *j bits* at a time (but not always, j=8).

Cont . . .

- **Step 1:** Like CBC, a 64-bit Initialization vector (IV) is used in the case of CFB mode. The IV is kept in a shift register. It is encrypted in the first step to produce a corresponding 64-bit IV cipher text.
- **Step 2:** Now, the leftmost (i.e. the most significant) j bits of the encrypted IV are XORed with the first j bits of the plain text. This produces the first portion of cipher text (say C) and C is transmitted to the receiver.
- **Step 3:** Now, the bits of IV (i.e. the contents of the shift register containing IV) are shifted left by j positions. Thus the rightmost j positions of the shift register now contain unpredictable data. These rightmost j positions are now

Cont . . .

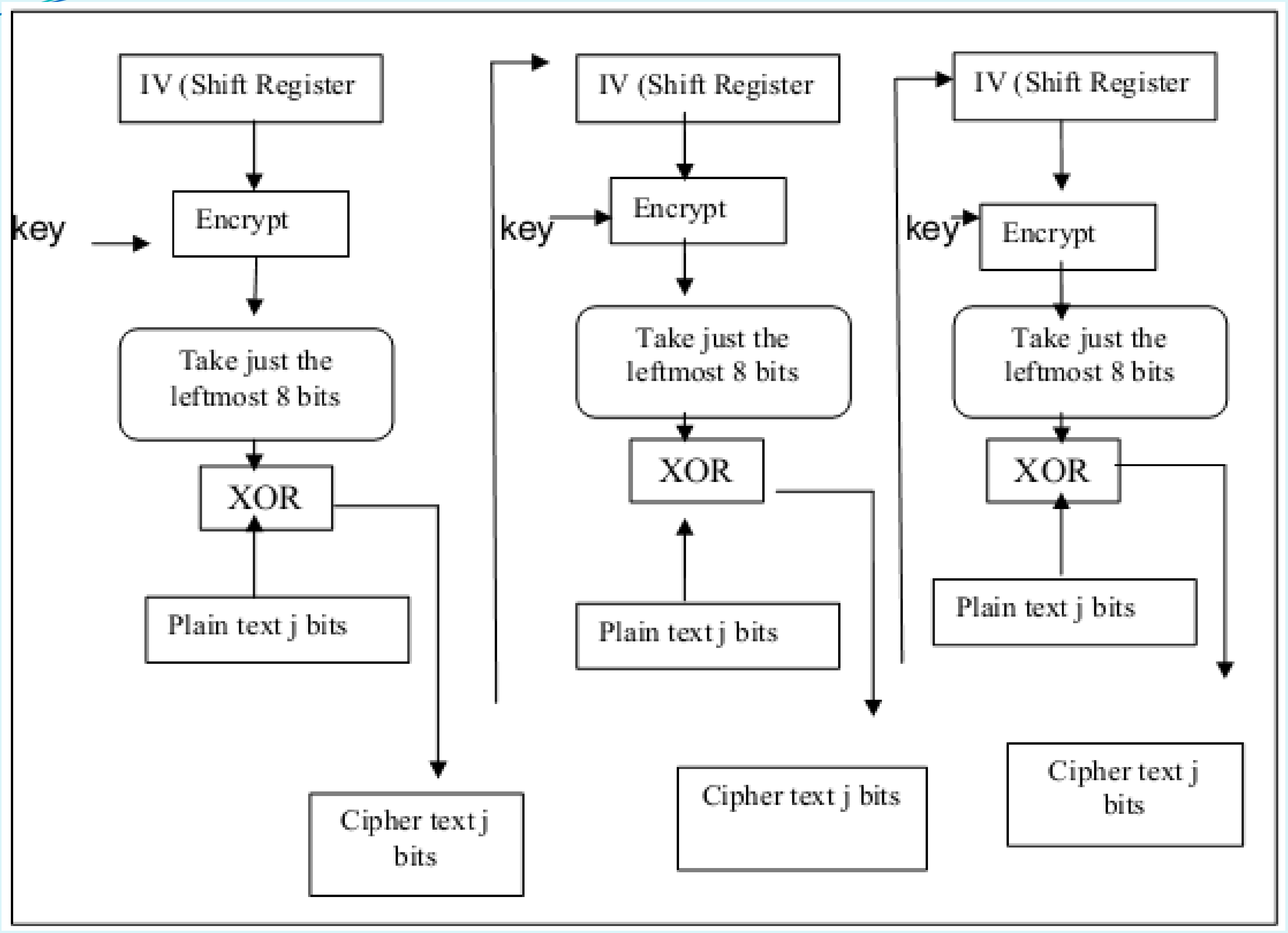
- **Step 4:** Now step 1 through 3 continue until all the plain text units are encrypted. That is, the following steps rep



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

- *Although CFB can be viewed as a stream cipher, it does not conform to the typical construction of a stream cipher.*
- *In a typical stream cipher, the cipher takes as input some initial value and a key and generates a stream of bits, which is then XORed with the plaintext bits .*
- *In the case of CFB, the stream of bits that is XORed with the*

Cont . . .


IV (Shift Register
key
Encrypt
Take just the leftmost 8 bits
XOR
Plain text j bits
Cipher text j bits
IV (Shift Register
key
Encrypt
Take just the leftmost 8 bits
XOR
Plain text j bits
Cipher text j bits
IV (Shift Register
key
Encrypt
Take just the leftmost 8 bits
XOR
Plain text j bits
Cipher text j bits

D) Output Feedback (OFB)

- The Output Feedback (OFB) mode is extremely similar to the CFB.
- The only difference is that in case of CFB, the cipher text is fed into the next stage of encryption process. But in the case of OFB, the output of the IV encryption process is fed into the next stage of encryption process.
- The other difference is that the OFB mode operates on full blocks of plaintext and ciphertext, not on an *S-bit* subset.
- Encryption can be

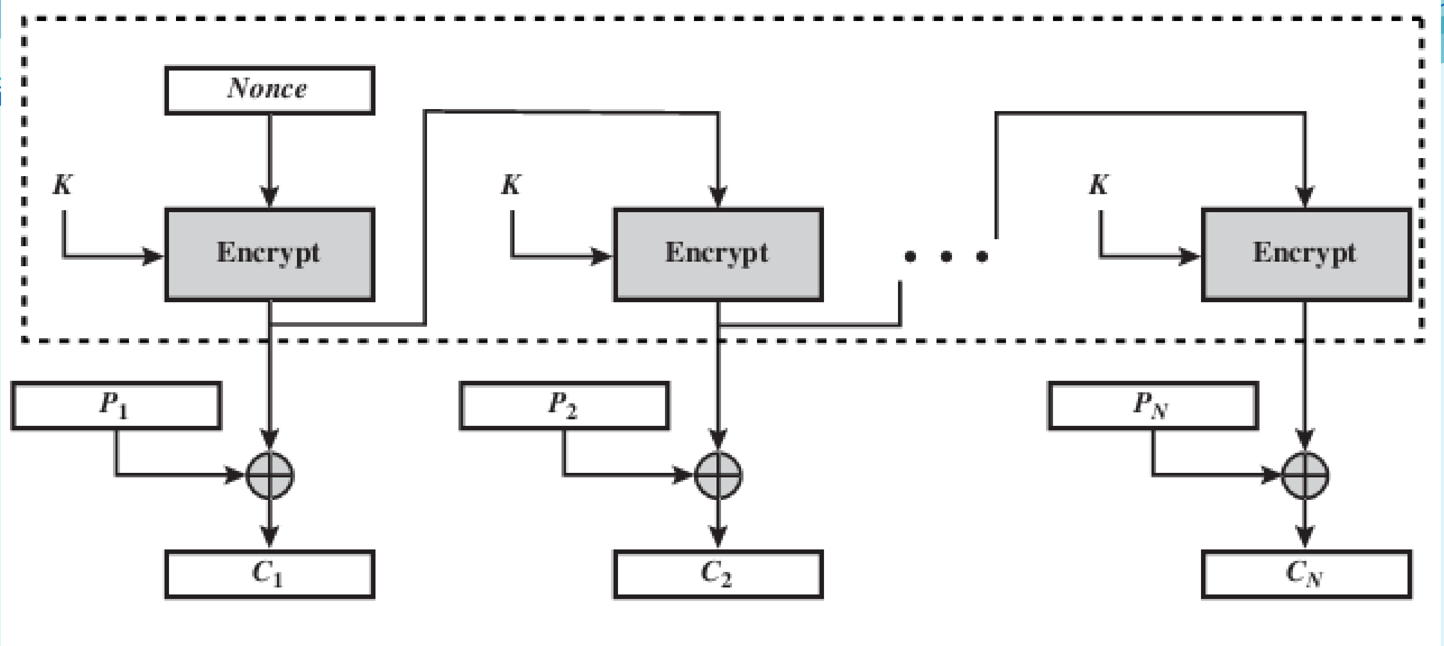
$$C_j = P_j \oplus \mathrm{E}(K, [C_{j-i} \oplus P_{j-1}])$$

- By rearrang

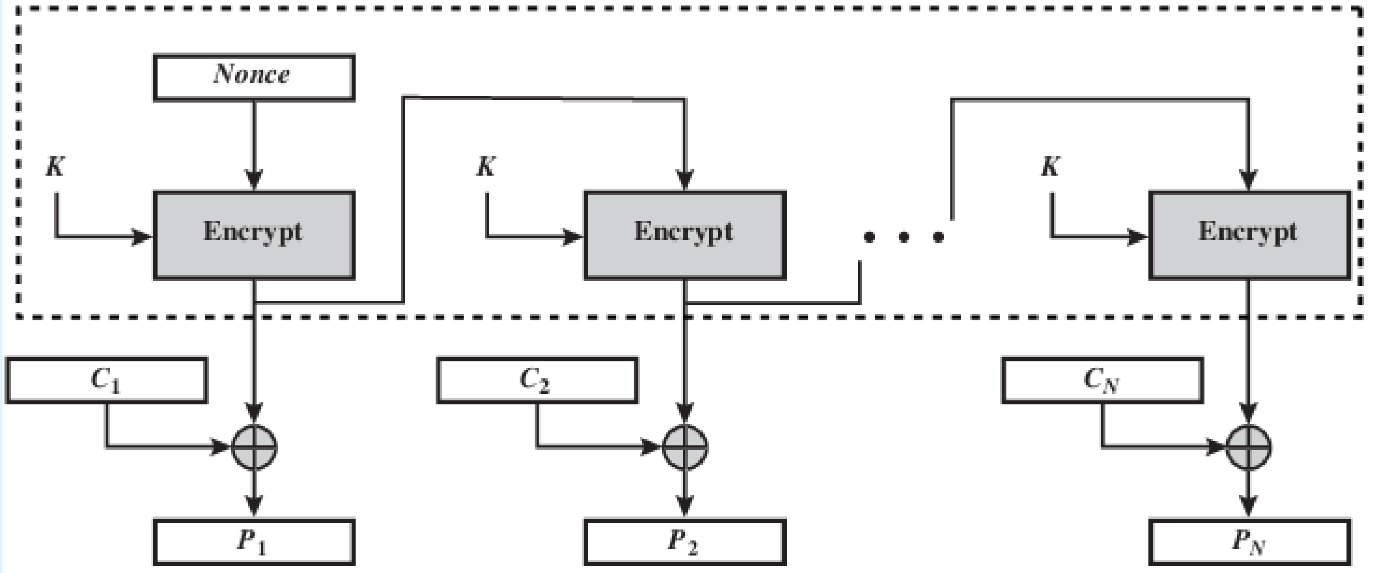
$$P_j = C_j \oplus \mathrm{E}(K, [C_{j-1} \oplus P_{j-1}])$$

nstrate that decryption works.

Cont . . .

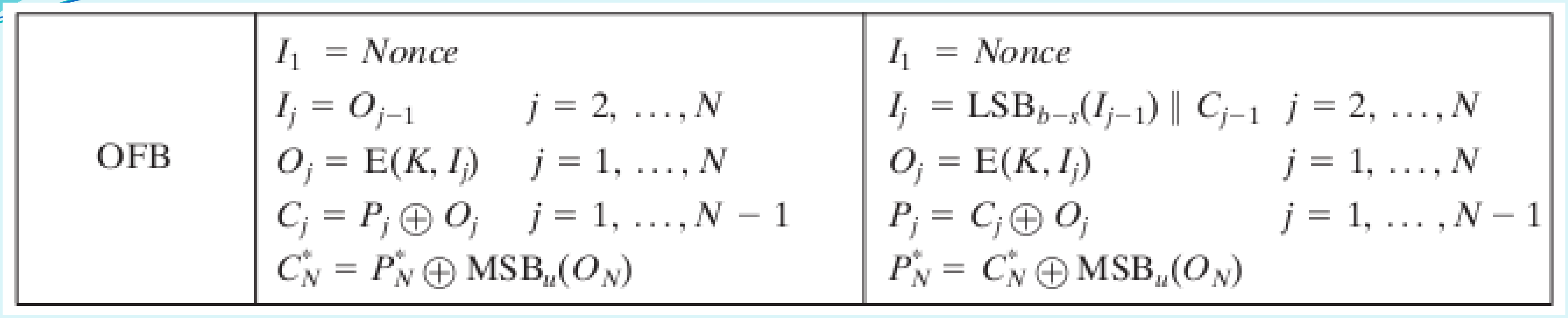


(a) Encryption



(b) Decryption

Cont . . .



OFB	$I_1 = Nonce$ $I_j = O_{j-1} \quad j = 2, \ldots, N$ $O_j = \mathrm{E}(K, I_j) \quad j = 1, \ldots, N$ $C_j = P_j \oplus O_j \quad j = 1, \ldots, N-1$ $C^*_N = P^*_N \oplus \mathrm{MSB}_u(O_N)$	$I_1 = Nonce$ $I_j = \mathrm{LSB}_{b-s}(I_{j-1}) \parallel C_{j-1} \quad j = 2, \ldots, N$ $O_j = \mathrm{E}(K, I_j) \quad j = 1, \ldots, N$ $P_j = C_j \oplus O_j \quad j = 1, \ldots, N-1$ $P^*_N = C^*_N \oplus \mathrm{MSB}_u(O_N)$

- Let the size of a block be b. If the last block of plaintext contains bits (indicated by *), with $u < b$, the most significant u bits of the last output block O_N are used for the XOR operation; the remaining $b - u$ bits of the last output block are discarded.
- The IV must be a *nonce*; that is, the IV must be unique to each execution of the encryption operation.
  - *i.e. The sequence of encryption output blocks, depends*

Cont . . .

Advantages

- Bit errors in transmission do not propagate.
  - ✓ *E.g., if a bit error occurs in C_i, only recovered value of P_i is affected; subsequent plaintext units are not corrupted like that of CFB.*

Disadvantages

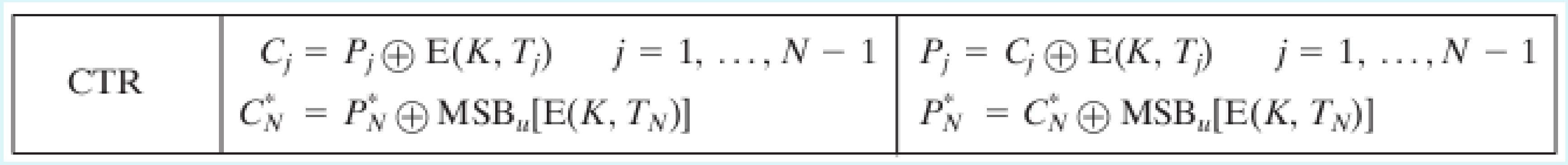
- OFB is more vulnerable to a message stream modification attack than is CFB.
  - ✓ Consider that complementing a bit in the cipher-text complements the corresponding bit in the recovered plaintext. Thus, controlled changes to the recovered plaintext can be made.

E) Counter Mode (CTR)

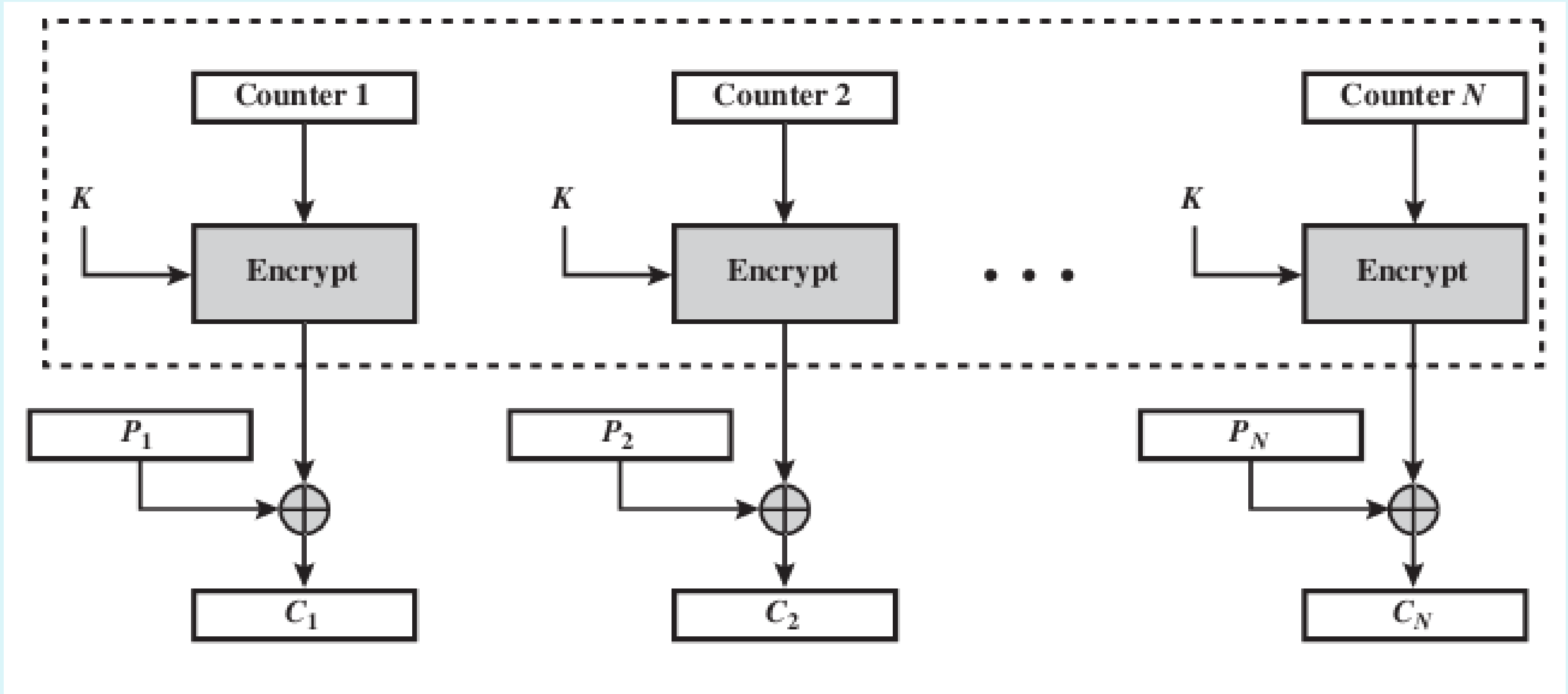
- A counter equal to the plaintext block size is used
- Counter value must be different for each plaintext block *(nonce)*
- Typically, the counter is initialized to some value and then incremented by 1 for each subsequent block.
- *For encryption,* the counter is encrypted and then XORed with the plaintext block to produce the ciphertext block; there is no chaining.
- *For decryption*, the same sequence of counter values is used, with each encrypted counter XORed with a ciphertext block to recover the corresponding plaintext block.

Cont . . .

- Given a sequence of counters $T_1, T_2, T_3, \ldots T_N$ we can define CTR mode as follows



CTR	$C_j = P_j \oplus \mathrm{E}(K, T_j) \quad j = 1, \ldots, N-1$ $C^*_N = P^*_N \oplus \mathrm{MSB}_u[\mathrm{E}(K, T_N)]$	$P_j = C_j \oplus \mathrm{E}(K, T_j) \quad j = 1, \ldots, N-1$ $P^*_N = C^*_N \oplus \mathrm{MSB}_u[\mathrm{E}(K, T_N)]$



(a) Encryption

Cont . . .

- For the last plaintext block, which may be a partial block of bits, the most significant bits of the last output block are used for the XOR operation; the remaining bits are discarded.
- Unlike the ECB, CBC, and CFB modes, we do not need to use padding because of the structure of the CTR mode.
- All T_i, values across all messages must be unique. If, contrary to this requirement, a counter value is used multiple times, then the confidentiality of all of the plaintext blocks corresponding to that counter value may be compromised.
- One way to ensure the uniqueness of counter values is to

Cont . . .

Advantages

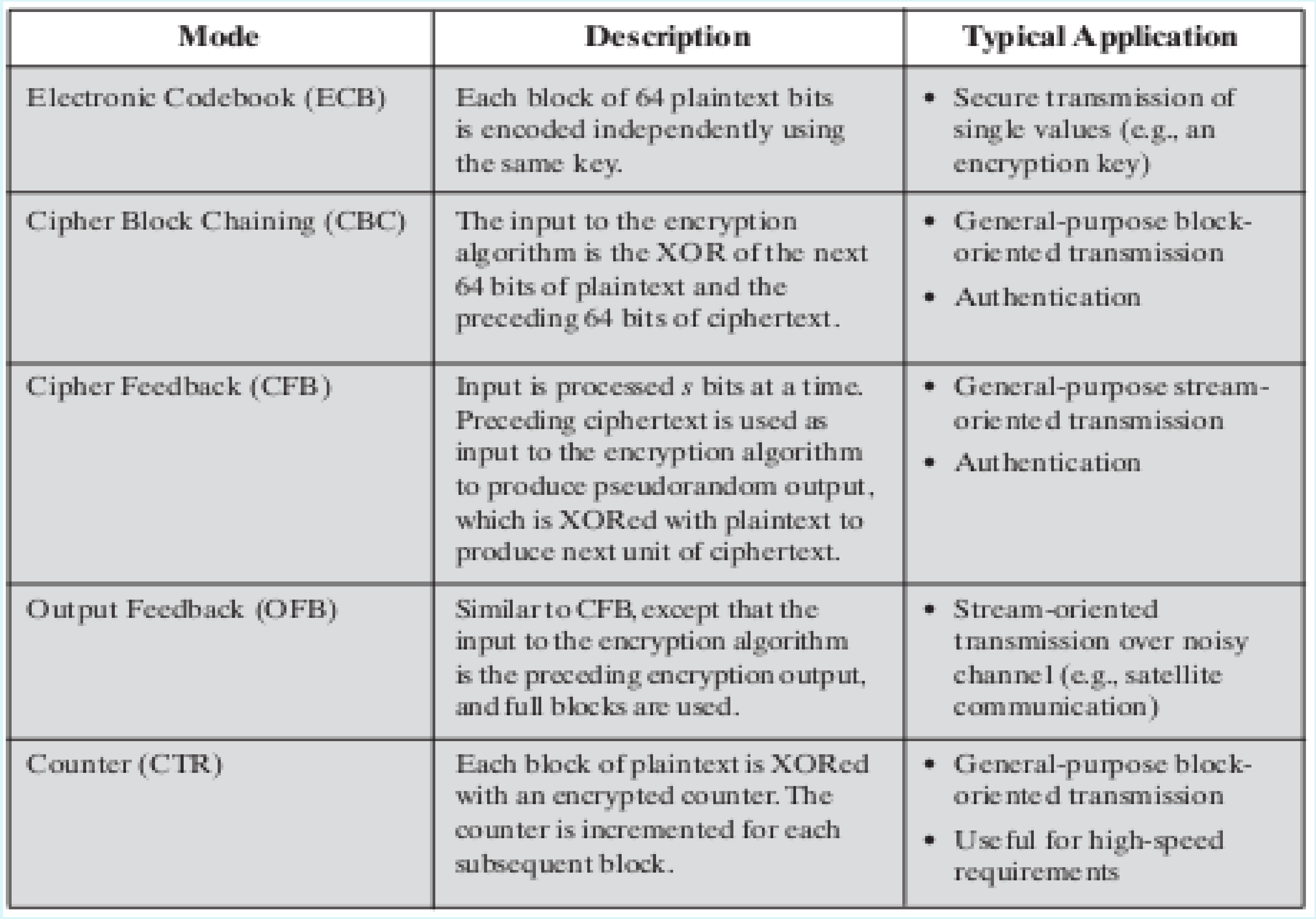
- Hardware efficiency:
  - *CTR mode can be done in parallel on multiple blocks of plaintext or ciphertext.*
  - *Throughput, only limited by amount of parallelism that is achieved*
- Software efficiency:
  - *Parallel execution*
  - *Parallel features, like aggressive pipelining, multiple instruction dispatch per clock cycle, large number of registers, and SIMD instructions, can be effectively utilized*

Cont . . .

Advantages

- Random access:
  - *Block of plaintext or ciphertext can be processed in random-access fashion*
- Preprocessing:
  - *Underlying encryption algorithm does not depend on input of the plaintext or ciphertex*
  - *Preprocessing can be used to prepare the output of the encryption boxes that feed into the XOR functions*
- Simplicity:
  - *Requires only the implementation of the encryption algorithm, not the decryption algorithms*

Algorithm modes Summary



Mode	Description	Typical Application
Electronic Codebook (ECB)	Each block of 64 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 64 bits of plaintext and the preceding 64 bits 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

THE END
???