CRYPTO M2 (p-1)

techworldthink • March 19, 2022

Conventional Symmetric Key Encryption

1.Block ciphers

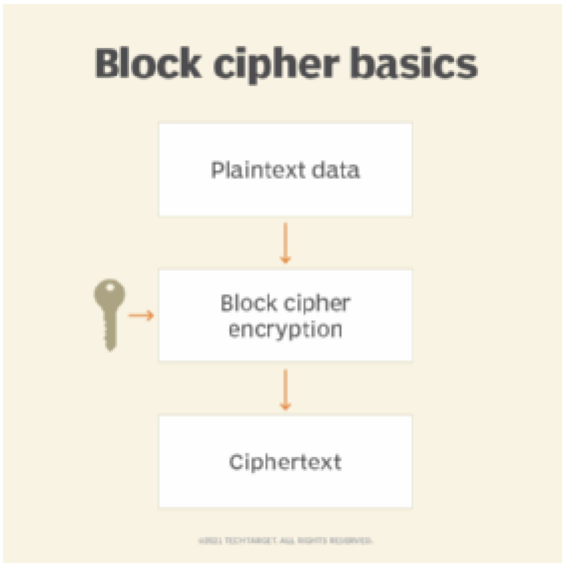
What is a block cipher?

A block cipher is a method of encrypting data in blocks to produce ciphertext using a cryptographic key and algorithm. The block cipher processes fixed-size blocks simultaneously, as opposed to a stream cipher, which encrypts data one bit at a time. Most modern block ciphers are designed to encrypt data in fixed-size blocks of either 64 or 128 bits.

Block ciphers have the advantage of high diffusion and strong tamper resistance without detection. They have the disadvantage of slower encryption speed since the entire block must be captured for encryption/decryption. Block ciphers also breed errors since a mistake in just one symbol could alter the whole block.

How does a block cipher work?

A block cipher uses a symmetric key and algorithm to encrypt and decrypt a block of data. A block cipher requires an initialization vector (IV) that is added to the input plaintext in order to increase the keyspace of the cipher and make it more difficult to use brute force to break the key. The IV is derived from a random number generator, which is combined with text in the first block and the key to ensure all subsequent blocks result in ciphertext that does not match that of the first encryption block.



The steps in a block cipher

The *block size* of a block cipher refers to the number of bits that are processed together. Data Encryption Standard (DES) and Advanced Encryption Standard (AES) are both symmetric block ciphers.

The DES block cipher was originally designed by IBM in 1975 and consisted of 64-bit blocks and a 56-bit key. This cipher is not considered secure anymore, due to the short key size, and was replaced in 1998 by AES. AES uses a 128-bit block size and a 128-, 192- or 256-bit key size.

Block Cipher Principles

A block cipher is designed by considering its three critical aspects which are listed as below:

1. Number of Rounds
2. Design of Function F
3. Key Schedule Algorithm

1. Number of Rounds

The number of rounds judges the strength of the block cipher algorithm. It is considered that more is the number of rounds, difficult is for cryptanalysis to break the algorithm.

It is considered that even if the function F is relatively weak, the number of rounds would make the algorithm tough to break.

2. Design of Function F

The function F of the block cipher must be designed such that it must be impossible for any cryptanalysis to unscramble the substitution. The criterion that strengthens the function F is it non-linearity.

More the function F is nonlinear, more it would be difficult to crack it. Well, while designing the function F it should be confirmed that it has a good avalanche property which states that a change in one-bit of input must reflect the change in many bits of output.

The Function F should be designed such that it possesses a bit independence criterion which states that the output bits must change independently if there is any change in the input bit.

3. Key Schedule Algorithm

It is suggested that the key schedule should confirm the strict avalanche effect and bit independence criterion.

What are the different modes of operation in block cipher?

Block ciphers only encrypt messages that are the same size as their block length, so each block of plaintext with more or less blocks needs to be encrypted separately.

- Electronic codebook (ECB) mode. ECB mode is used to electronically code messages as their plaintext form. It is the simplest of all block cipher modes of operation. It does not add any randomness to the key stream, and it is the only mode that can be used to encrypt a single-bit stream. This means that each plaintext symbol, such as a character from the plaintext alphabet, is converted into a ciphertext symbol using the cipher's key and a substitution alphabet. Each plaintext block is encrypted independently of all the other blocks. If a plaintext block is only 8 bytes, only 8 bytes of the key are used; if a plaintext block is 100 bytes, all 100 bytes of the key are used.
- Cipher block chaining (CBC) mode. CBC mode is a method of encrypting data that ensures that each block of plaintext is combined with the previous ciphertext block before being encrypted. The symmetric key algorithm creates a ciphertext that depends on all plaintext blocks processed before it in a data stream. This is done to ensure that each block of the ciphertext is dependent on all of the

previous blocks. Each plaintext block is XORed (exclusive OR) with the previous ciphertext block before being encrypted with the cipher algorithm. CBC mode is used in a variety of security applications. For example, Secure Sockets Layer/Transport Layer Security uses CBC mode to encrypt data that is transferred over the internet.

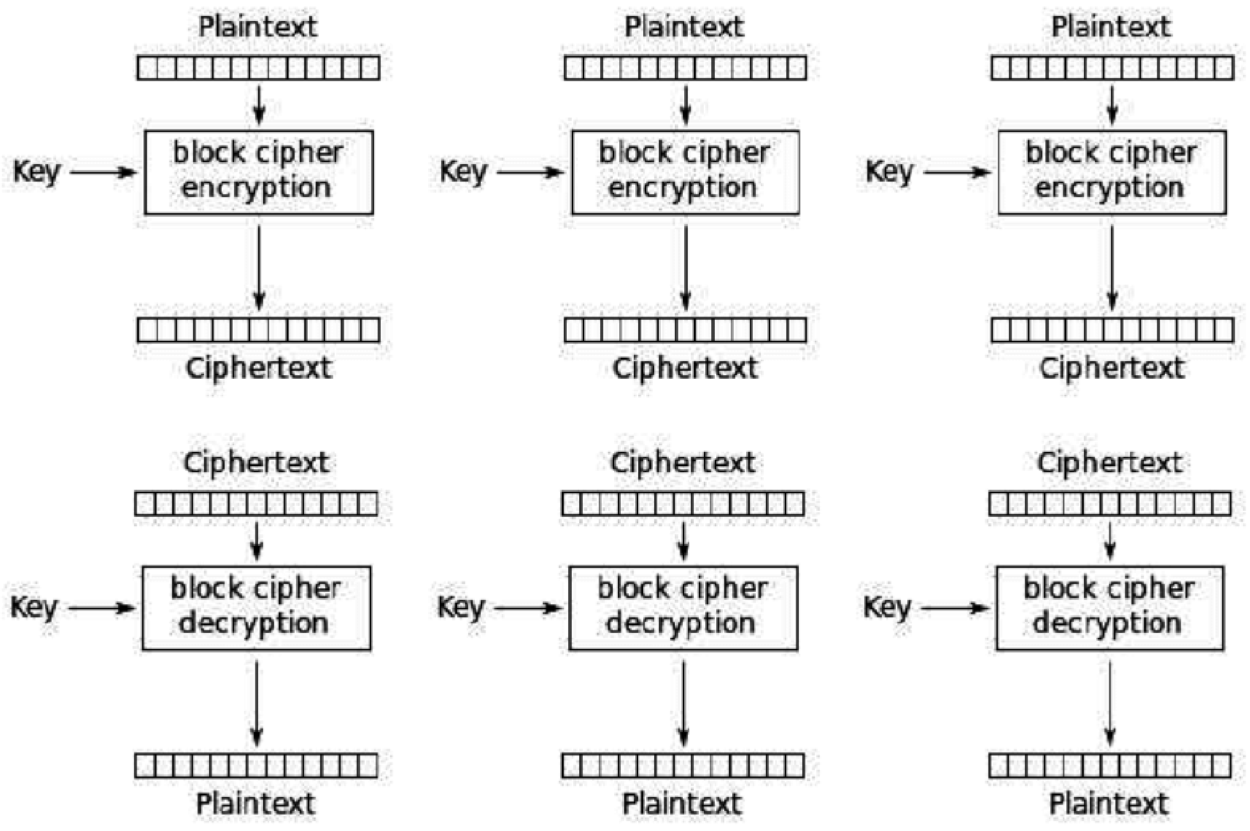
- Ciphertext feedback (CFB) mode. In contrast to CBC mode, which encrypts a set number of bits of plaintext at a time, it is sometimes necessary to encrypt and transfer plaintext values instantly, one at a time. Like CBC, CFB also uses an IV. CFB uses a block cipher as a component of a random number generator. In CFB mode, the previous ciphertext block is encrypted, and the output is XORed with the current plaintext block to create the current ciphertext block. The XOR operation conceals plaintext patterns.
- Output feedback (OFB) mode. OFB mode can be used with any block cipher and is similar in some respects to CBC mode. It uses a feedback mechanism, but instead of XORing the previous block of ciphertext with the plaintext before encryption, in OFB mode, the previous block of ciphertext is XORed with the plaintext after it is encrypted.
- Counter (CTR) mode. CTR mode uses a block chaining mode of encryption as a building block. The process of encrypting data is performed by XORing the plaintext with a sequence of pseudorandom values, each of which is generated from the ciphertext using a feedback function. The CTR encryption process can be visualized as a series of XORs between blocks of plaintext and corresponding blocks of ciphertext.

Electronic Code Book (ECB)

Electronic Code Book (ECB) is a simple mode of operation with a block cipher that's mostly used with symmetric key encryption. It is a straightforward way of processing a series of sequentially listed message blocks.

The input plaintext is broken into numerous blocks. The blocks are individually and independently encrypted (ciphertext) using the encryption key. As a result, each encrypted block can also be decrypted individually. ECB can support a separate encryption key for each block type.

In ECB, each block of plaintext has a defined corresponding ciphertext value, and vice versa. So, identical plaintexts with identical keys always encrypt to identical ciphertexts. This means that if plaintext blocks P1, P2 and so on are encrypted multiple times under the same key, the output ciphertext blocks will always be the same.



In other words, the same plaintext value will always result in the same ciphertext value. This also applies to plaintexts with partial identical portions. For instance, plaintexts containing identical headers of a letter and encrypted with the same key will have partially identical ciphertext portions.

For any given key, a codebook of ciphertexts can be created for all possible plaintext blocks. With the ECB mode, encryption entails only looking up the plaintext(s) and selecting the corresponding ciphertext(s). This operation is like assigning code words in a codebook. In fact, the term "code book" derives from the cryptographic codebooks used during the United States Civil War (1861-1865).

In terms of error correction, any bit errors in a ciphertext block will only affect decryption of that block. Chaining dependency is not an issue. Any reordering of the ciphertext blocks will only reorder the corresponding plaintext blocks. It won't affect decryption.

There are some drawbacks to using ECB, including:

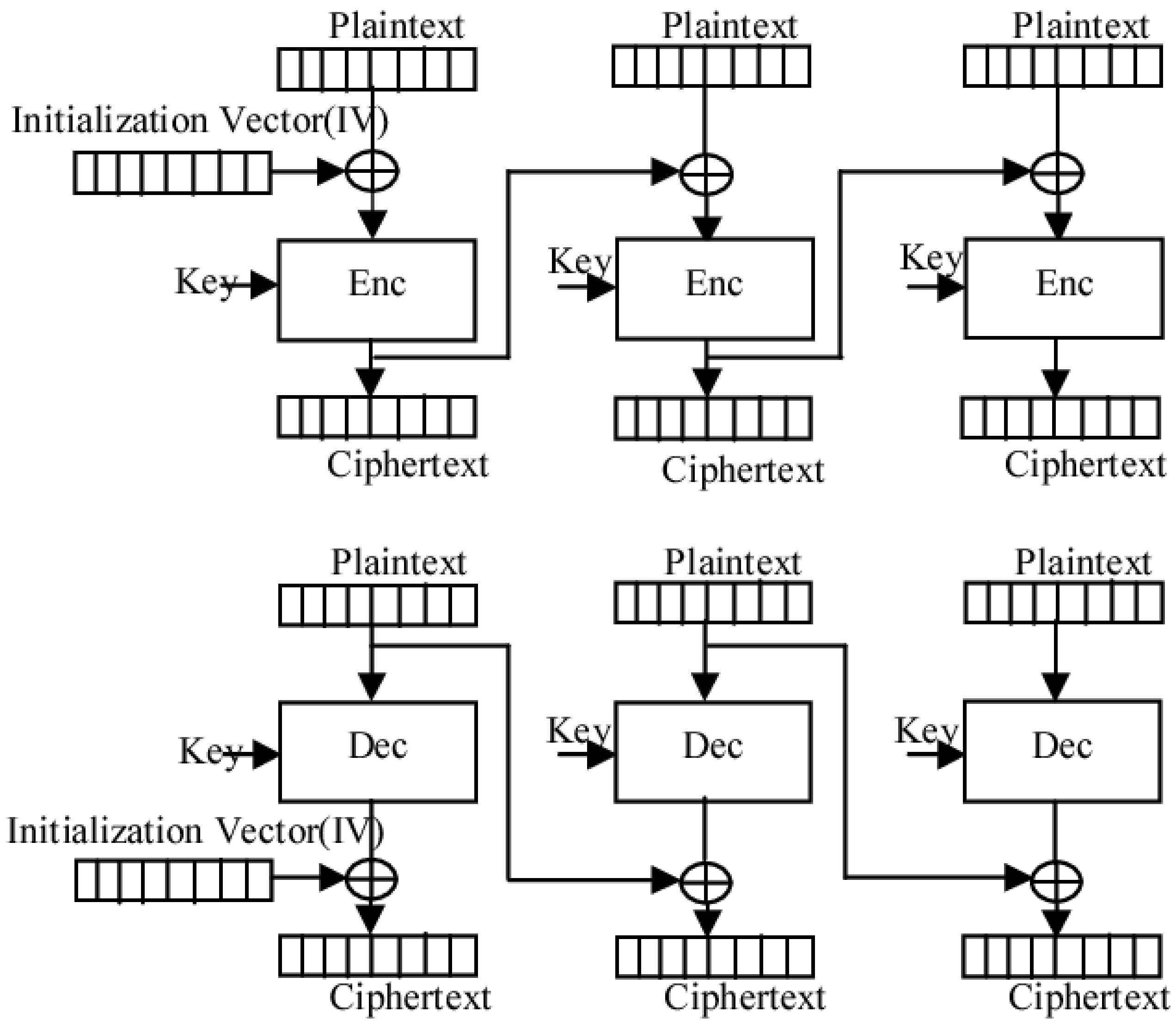
- ECB uses simple substitution rather than an initialization vector or chaining. These qualities make it easy to implement. However, this is also its biggest drawback. Two identical blocks of plaintext result in two correspondingly identical blocks of ciphertext, making it cryptologically weak.
- ECB is not good to use with small block sizes -- say, for blocks smaller than 40 bits -- and identical encryption modes. In small block sizes some words and phrases may be reused often in the plaintext. This means that the ciphertext may carry (and betray) patterns from the same plaintext, and the same repetitive part-blocks of ciphertext can emerge. When the plaintext patterns are obvious, it creates opportunities for bad actors to guess the patterns and perpetrate a codebook attack.
- ECB security is weak but may be improved by adding random pad bits to each block. Larger blocks (64-bit or more) would likely contain enough unique characteristics (entropy) to make a codebook attack unlikely.

cipher block chaining (CBC)

Cipher block chaining (CBC) is a mode of operation for a block cipher -- one in which a sequence of bits are encrypted as a single unit, or block, with a cipher key applied to the entire block. Cipher block chaining uses what is known as an initialization vector (IV) of a certain length. By using this along with a single encryption key, organizations and individuals can safely encrypt and decrypt large amounts of plaintext.

One of CBC's key characteristics is that it uses a chaining process that causes the decryption of a block of ciphertext to depend on all the preceding ciphertext blocks. As a result, the entire validity of all preceding blocks is contained in the previous, adjacent ciphertext block. A single bit error in a ciphertext block affects the decryption of all subsequent blocks. Rearrangement of the order of the ciphertext blocks, for example, can cause the decryption process to become corrupted.

Essentially, in cipher block chaining, each plaintext block is XORed (numerically combined) with the previous ciphertext block and then encrypted. An XOR is a coding mechanism used to combine different inputs. It is used in this case to facilitate the combination of plaintext blocks and encryption keys. The process repeats itself until all plaintext blocks have been successfully turned into ciphertext blocks.



How does cipher block chaining work?

Cipher block chaining is a process used to encrypt and decrypt large plaintext inputs by creating a cryptographic chain wherein each ciphertext block is dependent on the last.

The first step to initiating a cipher block chain is to XOR the first of many plaintext blocks with an IV -- a unique, fixed-length conversion function -- to create a random, or pseudorandom, output. This XOR output is then encrypted using a cipher key to produce a ciphertext block, an encrypted text format that can be decrypted with the correct key.

For example, after the first plaintext block has been transformed into a ciphertext block, the subsequent plaintext block must be encrypted using a similar process. The only difference, however, is that the ciphertext block replaces the IV as one of the XOR inputs. This means that the encryption of the plaintext block after the first one is dependent on the encryption of the first plaintext block. With each plaintext block encryption, the adjacent ciphertext block must be used -- like a chain. Therefore, the second ciphertext block is produced by XORing the first ciphertext block with the second plaintext block and using the same encryption key. This process would repeat itself until there is no more plaintext left to encrypt.

The CBC decryption process works in a similar but distinct way. Contrary to similar decryption methods, the process does not start with the final ciphertext block. In fact, it can all happen simultaneously because all inputs are present.

To invert the cipher block chaining procedure, one must essentially reverse the encryption process. To do that, one must first feed the first ciphertext block through the decryption process. This involves using the same encryption key as before but on the ciphertext block. The product of this interaction is then XORed with the original IV to extract the original plaintext block. While similar, decrypting the second ciphertext block is different from decrypting the first one because an IV cannot be used.

After combining the second ciphertext block with the cipher key, the output is XORed with the first ciphertext block to produce the second plaintext block. In this case, the previous ciphertext block replaces the IV during the decryption process. Remember, this is how the second ciphertext block was originally created; the second plaintext block and the first ciphertext block were XORed together. The process is complete once all ciphertexts have been successfully decrypted into plaintext.

Identical ciphertext blocks can only be produced if the same plaintext block is encrypted using the same key, IV and ciphertext block order. Ideally, the IV should be different for any two messages encrypted with the same key. Patterns like this can make it that much easier for malicious hackers or cybercriminals to decrypt a series of responses because the decryption is more predictable. Though the IV doesn't need to be a secret, some applications, such as security consultancy, may find this desirable.

The advantages of cipher block chaining

Cipher block chaining is one of the most used methods for encrypting large messages. As the more secure successor of electronic codebook (ECB) -- the easiest block cipher mode of functioning -- CBC can reliably encrypt large plaintext inputs but at a slower pace than some parallel encryption algorithms.

Here are some advantages to cipher block chaining.

1.Identical blocks do not share the same cipher

CBC has the advantage over the ECB mode in that the XORing process hides plaintext patterns. Even if the first plaintext block and third plaintext block were the exact same segment of plaintext, it is highly unlikely that the first ciphertext block and third ciphertext block would be the same. Essentially, this means that two identical pieces of plaintext, when encrypted, should not produce identical, or even similar, results. The only reason why it is not impossible is that there is a minuscule chance that XORing the second ciphertext block and the third plaintext block produces the same product as XORing the first plaintext block with the IV.

2.Better security

Because cipher block chaining relies on using previous ciphertext blocks to encrypt subsequent plaintext blocks, hackers and decryptors must have all ciphertext blocks available in order to successfully decrypt entire CBC outputs. This multistep encryption mechanism makes it difficult to deconstruct, thereby increasing the security of the messages it is trying to encrypt.

In a way, CBC is an example of effective challenge-response authentication. A user or group that requires access to a certain set of documents must be able to present the necessary ciphertext blocks to successfully decrypt the entire message or text.

The disadvantages of cipher block chaining

As effective as cipher block chaining is in securing large amounts of plaintext, it does have its challenges, especially when it comes to speed and convenience.

1.Not tolerant of block losses

While distinct ciphertext blocks are extremely useful in terms of encryption strength, they can be a detriment in terms of decryption reliability. If one or more of the ciphertext blocks becomes lost, damaged or corrupted, a user won't be able to perform a complete decryption. While this can be a minimal inconvenience and rarely happens, it does force agencies to employ secure storage systems to retain all ciphertext data.

2.Parallel CBC encryption is not possible

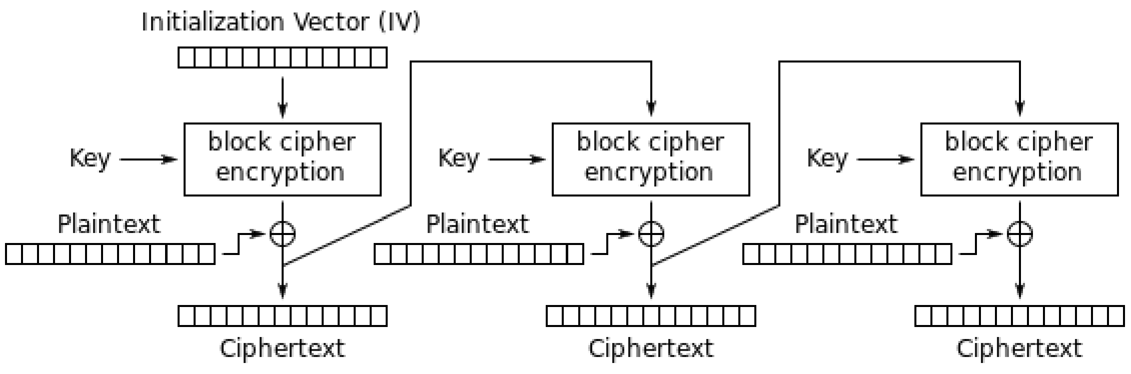
Due to the recursive nature of CBC's encryption process, it is impossible to simultaneously encrypt all plaintext inputs using cipher block chaining. Without having collected all previous ciphertext blocks, a user cannot achieve parallel encryption because each plaintext block encryption is dependent on the last. The order is so important that, if it were switched during the CBC encryption process, it would result in a completely different set of ciphertext blocks.

ciphertext feedback (CFB)

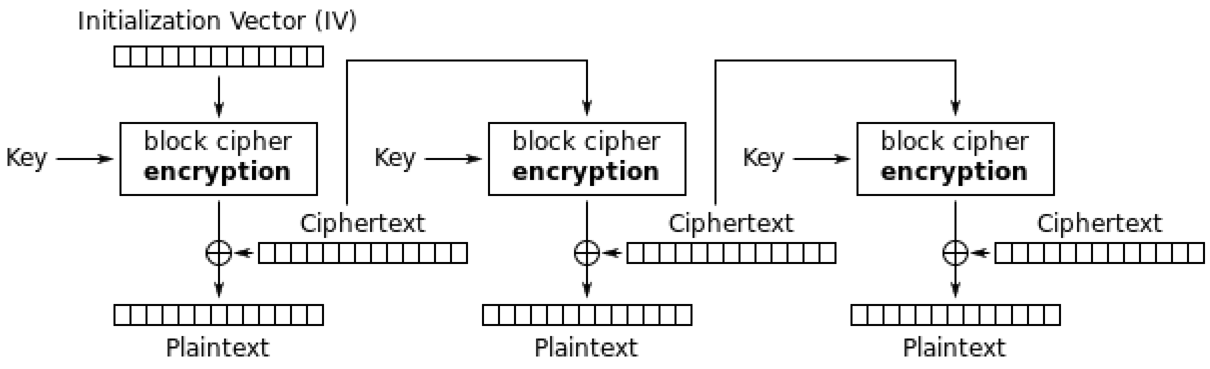
In cryptography, ciphertext feedback (CFB), also known as cipher feedback, is a mode of operation for a block cipher. Ciphertext refers to encrypted text transferred from plaintext using an encryption algorithm, or cipher. A block cipher is a method of encrypting data in blocks to produce ciphertext using a cryptographic key and algorithm.

Like cipher block chaining (CBC), ciphertext feedback uses an initialization vector (IV). CFB uses a block cipher as a component of a random number generator. In CFB mode, the previous ciphertext block is encrypted and the output is XORed with the current plaintext block to create the current ciphertext block. The XOR operation conceals plaintext patterns. Plaintext cannot be directly worked on unless the blocks from either the beginning or end of the ciphertext are retrieved.

The entropy that results can be implemented as a stream cipher, a method of encrypting text whereby a cryptographic key and algorithm are applied one bit at a time to each binary digit in a data stream. CFB is primarily a mode to derive some characteristics of a stream cipher from a block cipher. Similar to CBC mode, changing the IV to the same plaintext block in CFB results in different output. Though the IV need not be secret, some applications would consider this desirable. Chaining dependencies are similar to CBC in that reordering ciphertext block sequences alters decryption output, as decryption of one block depends on the decryption of the preceding blocks. This means that a corrupted bit that has been manifested during transmission will propagate that error, affecting transmitted bits in the block -- usually one to two bytes that follow.



Cipher Feedback (CFB) mode encryption



Cipher Feedback (CFB) mode decryption

How is cipher feedback used?

CFB is used in encryption algorithms such as Data Encryption Standard (DES), Triple DES and Advanced Encryption Standard (AES). If CFB mode is used within the encryption algorithm, it's often used to encrypt the following services:

- Wi-Fi communications
- secure websites
- chip-based security
- file encryption

- Secure Sockets Layer or Transport Layer Security encrypted virtual private network tunnels
- Simple Mail Transfer Protocol email
- messaging services based on the Extensible Messaging and Presence Protocol
- secure FTP
- Secure Shell
- Voice over IP

The AES algorithm below shows where the cipher modes, including CFB, fit into the encryption process. The unencrypted plaintext is combined with a secret key and encrypted using one of five supported cipher modes, including CFB. The output is an encrypted ciphertext that can be securely transported across unsecure networks.

What is the difference between CBC and CFB?

CBC and CFB perform the same general data encryption duties, with a few notable exceptions. First, CFB mode data is encrypted in units that are smaller than a predefined block size cipher unit, which is usually 64 bits, as is the case with CBC. Instead, CFB encrypts units in 1- or 2-byte (8- or 16-bit) block sizes and processes each bit at a time as opposed to the entire 64 bits. Therefore, using CFB mode, no padding process is needed when the data size is less than 64 bits and needs appending to meet the 64-bit length requirement.

Another major differentiator between CBC and CFB is how they encrypt plaintext data. Unlike CBC, which directly encrypts plaintext blocks, CFB encrypts the previously encrypted plaintext block and then adds this to the next plaintext block. This means that the same algorithm used to encrypt the data can be used to decrypt it, which simplifies the decryption process of CFB.

output feedback (OFB)

OFB (short for output feedback) is an AES block cipher mode similar to the CFB mode. What mainly differs from CFB is that the OFB mode relies on XOR-ing plaintext and ciphertext blocks with expanded versions of the initialization vector.

This process can be seen as a one-time pad and the expanded vectors as *pad vectors*. The following formula depicts how a sequence of pad vectors is created:

Vi = EK(Vi-1)

> *where EK denotes the block encryption algorithm using key K and Vi and Vi-1 are adjacent vectors.Note: In the formula above, we are assuming V0 to be the initialization vector.*

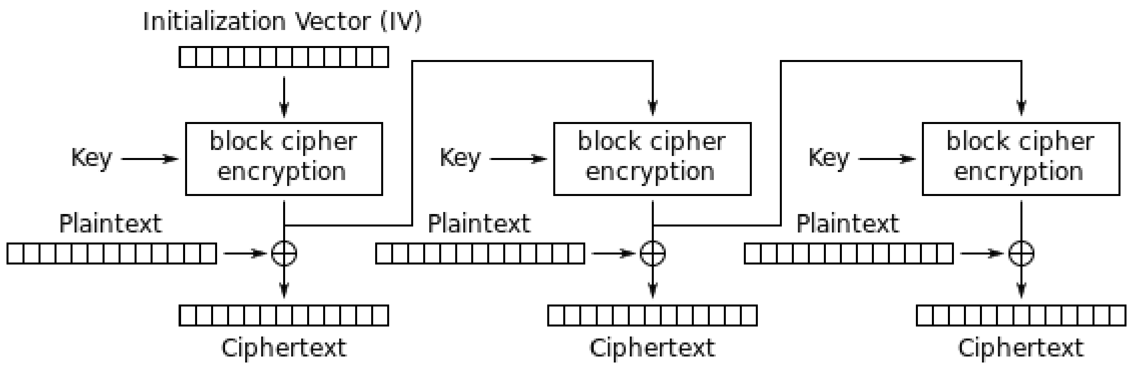
Once the sequence of pad vectors is generated, encryption with the OFB mode can be carried out using the following formula:

Ci = Vi ⊕ Bi

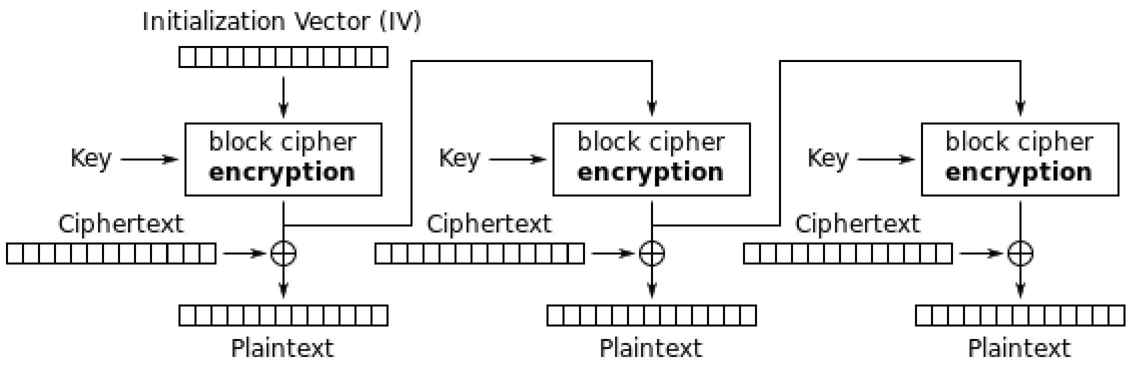
Decryption is carried out in a similar way:

Bi = Vi ⊕ Ci

> *Note: Like the CFB mode, OFB also makes use of a single encryption algorithm for both encryption and decryption.*



Output Feedback (OFB) mode encryption



Output Feedback (OFB) mode decryption

Advantages and disadvantages of using the OFB mode

Since blocks are independent of one another using the OFB mode, both encryption and decryption of blocks can be done in parallel once the pad vectors have been generated. The lack of interdependency also means that the OFB mode is tolerant to the loss in blocks.

A significant drawback of the OFB is that 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.

Counter Mode

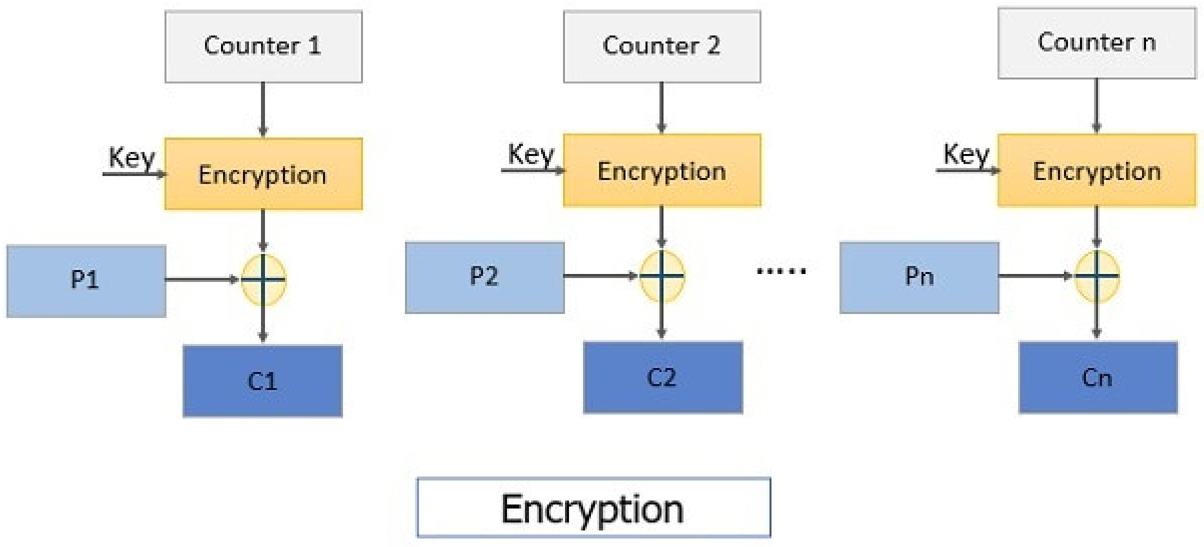
It is similar to OFB but there is no feedback mechanism in counter mode. Nothing is being fed from the previous step to the next step instead it uses a sequence of number which is termed as a counter which is input to the encryption function along with the key. After a plain text block is encrypted the counter value increments by 1.

Steps of encryption:

Step1: The counter value is encrypted using a key.

Step 2: The encrypted counter value is XORed with the plain text block to obtain a ciphertext block.

To encrypt the next subsequent plain text block the counter value is incremented by 1 and step 1 and 2 are repeated to obtain the corresponding ciphertext.



Counter Mode Encryption

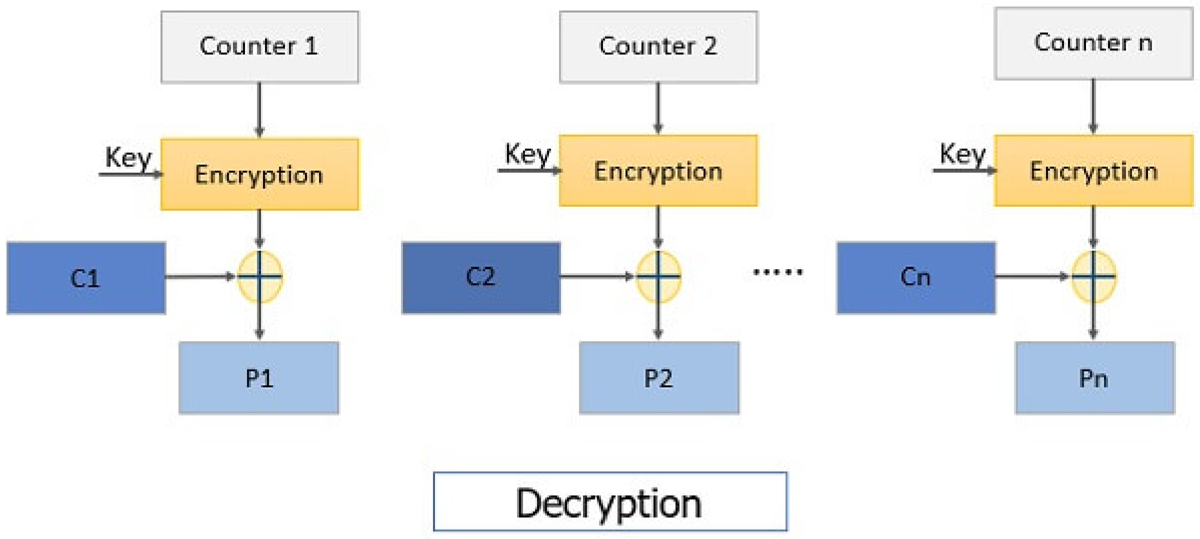
The process continues until all plain text block is encrypted.

Steps for decryption:

Step1: The counter value is encrypted using a key.

Note: Encryption function is used in the decryption process. The same counter values are used for decryption as used while encryption.

Step 2: The encrypted counter value is XORed with the ciphertext block to obtain a plain text block.



Counter Mode Decryption

To decrypt the next subsequent ciphertext block the counter value is incremented by 1 and step 1 and 2 are repeated to obtain corresponding plain text.

The process continues until all ciphertext block is decrypted.

2. Stream cipher

A stream cipher is a method of encrypting text (to produce ciphertext) in which a cryptographic key and algorithm are applied to each binary digit in a data stream, one bit at a time. The main alternative method to stream cipher is, in fact, the block cipher, where a key and algorithm are applied to blocks of data rather than individual bits in a stream.

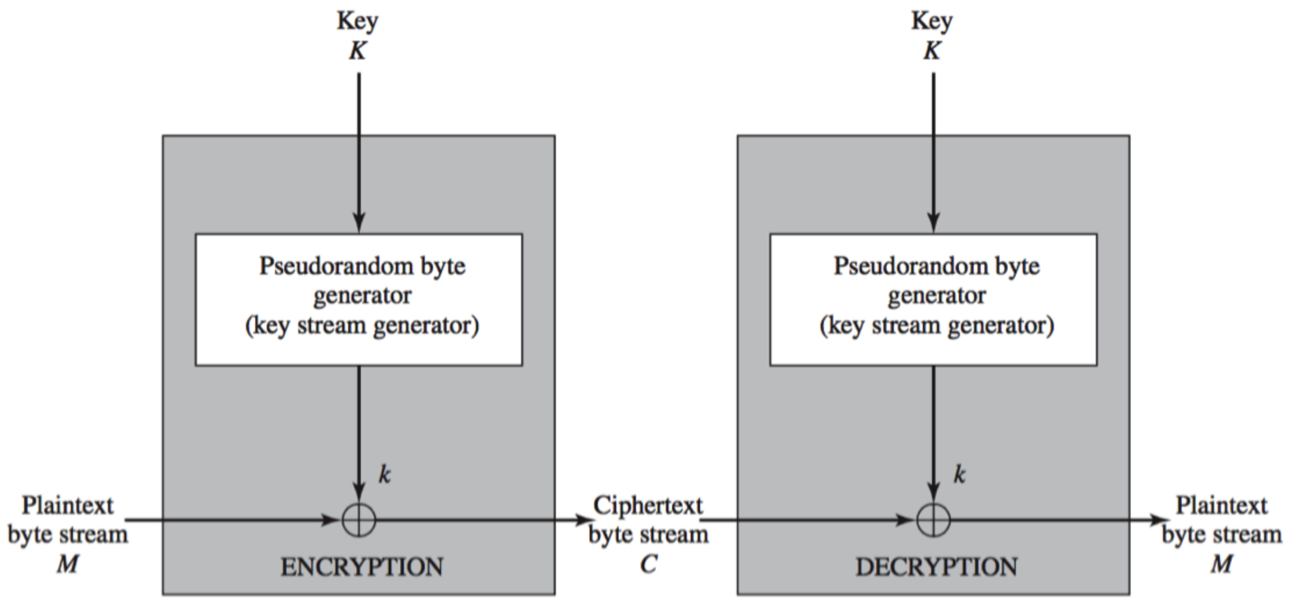

Key
K
Pseudorandom byte generator (key stream generator)
k
Plaintext byte stream M
ENCRYPTION
Ciphertext byte stream C
Key
K
Pseudorandom byte generator (key stream generator)
k
DECRYPTION
Plaintext byte stream M

Figure 7.5 Stream Cipher Diagram

How does a stream cipher work?

A stream cipher is an encryption algorithm that uses a symmetric key to encrypt and decrypt a given amount of data. A symmetric cipher key, as opposed to an asymmetric cipher key, is an encryption tool that is used in both encryption and decryption. Asymmetric keys will sometimes use one key to encrypt a message and another to decrypt the respective ciphertext.

What makes stream ciphers particularly unique is that they encrypt data one bit, or byte, at a time. This makes for a fast and relatively simple encryption process.

Basic encryption requires three main components:

1. a message, document or piece of data

2. a key
3. an encryption algorithm

The key typically used with a stream cipher is known as a *one-time pad*. Mathematically, a one-time pad is unbreakable because it's always at least the exact same size as the message it is encrypting.

Here is an example to illustrate the one-timed pad process of stream ciphering: Person A attempts to encrypt a 10-bit message using a stream cipher. The one-time pad, in this case, would also be at least 10 bits long. This can become cumbersome depending on the size of the message or document they are attempting to encrypt, however.

Cryptographers also refer to the symmetric key used in a stream cipher as a *keystream*. This is because Person A could opt to create a pseudo-random cipher digit stream, or keystream, using a key that is smaller than the size of the plaintext file. Furthermore, to avoid having to create a larger keystream, users can use a cryptographic number generator to create a larger keystream from a smaller, pseudo-random key.

Here, Person A decides to use a 4-bit key to encrypt a 10-bit message. To do that, they must first use an initialization vector (IV) to generate a random seed value. Placing this seed value into a cryptographic number generator, Person A can create a pseudo-random keystream that matches the size of their desired plaintext file.

The quality of the number generator contributes to the randomness and security of the ciphertext, however. Lower-end cryptographic number generators can sometimes have patterns that malicious users, or hackers, can identify and use to decrypt the ciphertext.

After the user has created the keystream, the stream cipher combines the keystream with the corresponding digits of the plaintext using the exclusive-or (XOR) operator. The XOR operator creates new binary values, which make up the ciphertext. It generates these values by comparing bits in the plaintext and the keystream that share the same position.

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

Once each bit of data has been XOR-ed by the stream cipher, it will produce an unreadable ciphertext message.

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

Stream ciphers users should not use the same IV more than once, however, to maximize the security of this process.

The advantages and disadvantages of using a stream cipher

Speed of encryption tops the list of advantages for stream ciphers. Once a stream cipher makes a key, the encryption and decryption process is almost instantaneous. This is largely due to the simplicity of operation, a basic XOR function using two distinct data bits. Also, for this reason, the hardware complexity of a stream cipher is quite low -- meaning a wider range of technologies can facilitate this mode of operation.

Another advantage of using stream ciphers is the ability to decrypt selected sections of ciphertext. Since each bit of data in the ciphertext corresponds with plaintext data in the same position, users can decrypt ciphertext for part of a document rather than the entire file. If Person A, for example, wanted to retrieve the original plaintext data for just the first five bits, they would only need to decrypt the first five bits of the ciphertext.

The positional alignment among the plaintext, keystream and ciphertext is a significant security vulnerability of stream ciphers. If the encryption process does not use a hashing algorithm or IV during the encryption process, a hacker who obtains a segment of plaintext and its respective ciphertext will be able to deduce the keystream used by the process.

This is why cryptographers refer to stream ciphers as having *low diffusion*, meaning the plaintext and ciphertext are not vastly different from each other.

Confusion vs. diffusion

Evaluate encryption methods using two main criteria: confusion and diffusion.

Confusion contributes to the ambiguity of the ciphertext -- i.e., the methods used to complicate the relationship between the plaintext and the ciphertext.

Diffusion, on the other hand, refers to how well the encryption hides these complex, or simple, relationships. For example, a standard for diffusion is that at least 50% of the ciphertext should change when a cipher alters a single bit of plaintext. Encryption operations with low diffusion are less secure than those with high diffusion.

Stream cipher vs. block cipher

Block ciphers and stream ciphers, while similar, are two distinct forms of symmetric key encryption. The main difference between these two modes of operations is the amount of data they encrypt at one time.

While stream ciphers perform data encryption one bit at a time, block ciphers encrypt fixed blocks of information all at once. These predetermined blocks of data will typically be either 64 or 128 bits long. If the data is not long enough -- i.e., they are smaller than the block size -- pads will be added.