Substitution Techniques

Substitution technique is one that the letters in the plaintext will be replaced by other letters or by numbers or symbols.

[Caesar Cipher]

The earliest use of substitution cipher is also the simplest one that is proposed by Julius Caesar, called Caesar Cipher. The Caesar Cipher works with replacing each letter with the letter standing three places further down of the alphabet order.

For example:

plaintext: **a b c d e f g h w x y z**

ciphertext: **e f g h i j k l z a b c**

So if the plaintext is "meet me after the party". The ciphertext would be "phhw ph diwhu wkh sduwb".

plaintext: **meet me after the party**

ciphertext: **phhw ph diwhu wkh sduwb**

If we assign each letter a number from 0 to 25(from A to Z). Take the Ciphertext as C, Encryption as E, and plaintext as P. Then we can describe the Caesar Cipher as below:

$$C=E(p)=(p+3)\mathrm{mod}(26) \quad (1)$$

A shift could be any amount, so the general Caesar algorithm is

$$C=E(p)=(p+k)\mathrm{mod}(26) \quad (2)$$

where k takes on a value in the range from 1 to 25. And the decryption algorithm is simply

$$p=D(C)=(C-k)\mathrm{mod}(26) \quad (3)$$

If it is known that a given ciphertext is a Caesar cipher, then a brute-force cryptanalysis will be easily performed. Just try all the 25 for the possible value of k.

In this example, there are three reasons for us to use the brute-force cryptanalysis. First is that the encryption and the decryption algorithms are known. Second is that there are only 25 keys to try. Third is that the language of the plaintext is known and easily recognizable.

Transposition Techniques

Transposition technique is achieved by performing some kind of permutation on the plaintext letters. It is very simple to realize this kind of cipher. We can do it by the example. If the plaintext is "meet me after the party", we can rearrange it by this way:

m e m a t r h p r y

e t e f e t e a t

So we get the plaintext and the ciphertext like this:

plaintext: **meet me after the party**

ciphertext: **mematrhpryetefeteat**

[Columnar transposition]

Another simple transposition cipher is called Columnar transposition. If the plaintext is "data encryption", we will compose the sentence into a 3*5 matrix. For example:

key: **4 1 2 3 5**

plaintext : **d a t a**

e n c r y

p t I o n

ciphertext: anttciarodep yn

Of course, the transposition cipher can be made more secure by performing more than one stage of transposition. For example, doing the Columnar transposition 2 or 3 times and it will efficiently to increase the security of this cipher.

Block Cipher Modes of Operation

In cryptography, a **block cipher mode of operation** is an algorithm that uses a block cipher to provide information security such as confidentiality or authenticity. A block cipher by itself is only suitable for the secure cryptographic transformation (encryption or decryption) of one fixed-length group of bits called a block. A mode of operation describes how to repeatedly apply a cipher's single-block operation to securely transform amounts of data larger than a block.

Most modes require a unique binary sequence, often called an initialization vector (IV), for each encryption operation. The IV has to be non-repeating and, for some modes, random as well. The initialization vector is used to ensure distinct ciphertexts are produced even when the same plaintext is encrypted multiple times independently with the same key. Block ciphers may be capable of operating on more than one block size, but during transformation the block size is always fixed. Block cipher modes operate on whole blocks and require that the last part of the data be padded to a full block if it is smaller than the current block size. There are, however, modes that do not require padding because they effectively use a block cipher as a stream cipher.

For different applications and uses, there are several modes of operations for a block cipher.

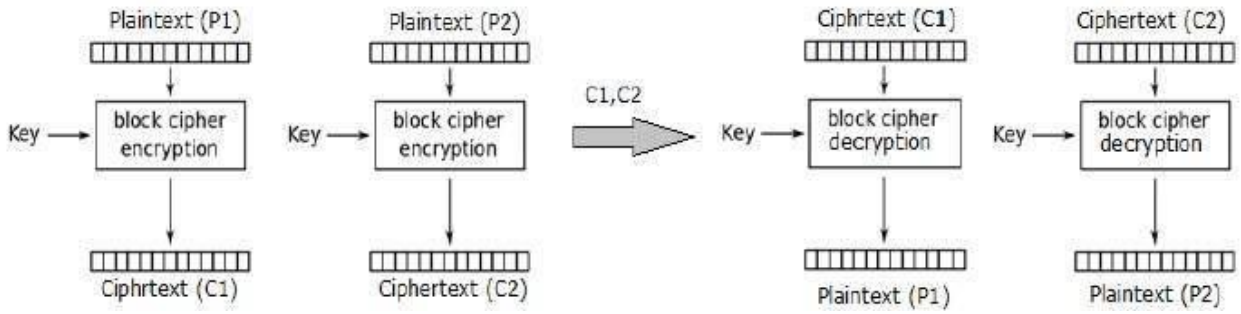
Electronic Code Book (ECB) Mode

Electronic code book is the easiest block cipher mode of functioning. It is easier because of direct encryption of each block of input plaintext and output is in form of blocks of encrypted ciphertext. Generally, if a message is larger than *b* bits in size, it can be broken down into bunch of blocks and the procedure is repeated.

Operation

- The user takes the first block of plaintext and encrypts it with the key to produce the first block of ciphertext.
- He then takes the second block of plaintext and follows the same process with same key and so on so forth.

Procedure of ECB is illustrated below:



Cipher Block Chaining (CBC) Mode

Cipher block chaining or CBC is an advancement made on ECB since ECB compromises some security requirements. In CBC, previous cipher block is given as input to next encryption algorithm after XOR with original plaintext block. In a nutshell here, a cipher block is produced by encrypting a XOR output of previous cipher block and present plaintext block.

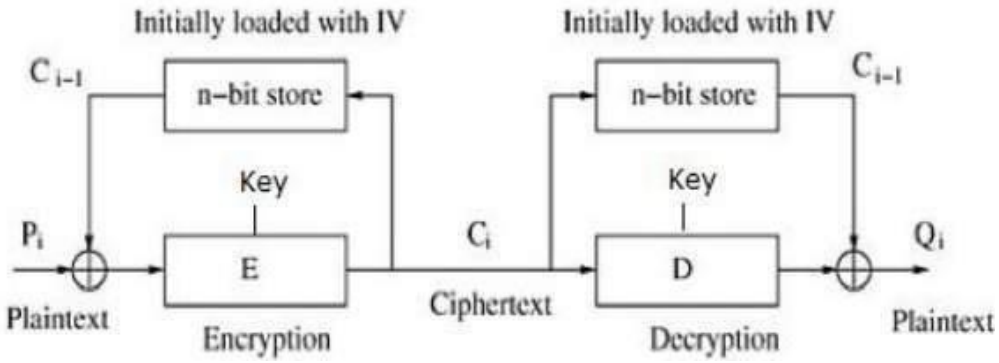
Operation

The operation of CBC mode is depicted in the following illustration. The steps are as follows −

- Load the n-bit Initialization Vector (IV) in the top register.
- XOR the n-bit plaintext block with data value in top register.
- Encrypt the result of XOR operation with underlying block cipher with key K.
- Feed ciphertext block into top register and continue the operation till all plaintext blocks are processed.

- For decryption, IV data is XORed with first ciphertext block decrypted. The first ciphertext block is also fed into to register replacing IV for decrypting next ciphertext block.

Procedure of CBC is illustrated below:



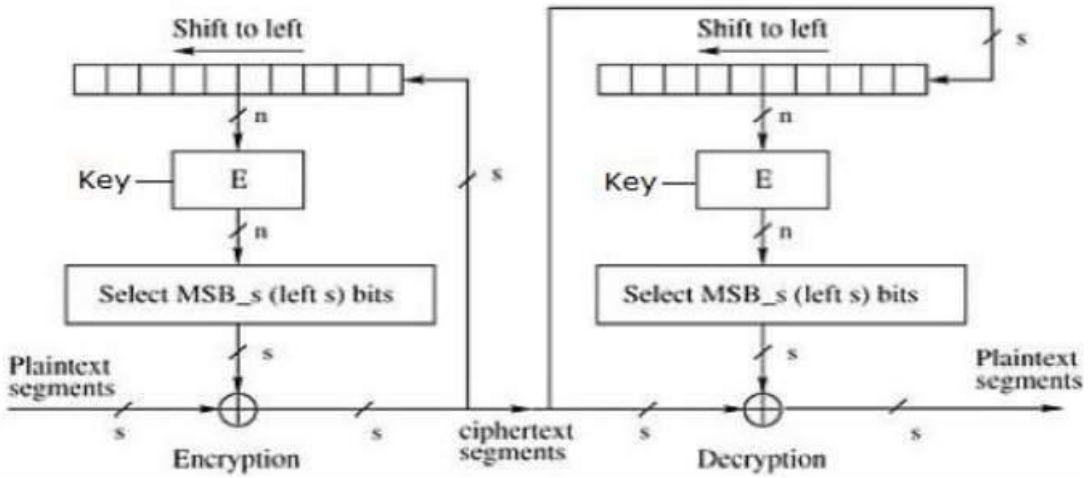
Cipher Feedback (CFB) Mode

In this mode the cipher is given as feedback to the next block of encryption with some new specifications: first an initial vector IV is used for first encryption and output bits are divided as set of *s*and*b-s* bits the left hand side *s*bits are selected and are applied an XOR operation with plaintext bits. The result given as input to a shift register and the process continues. The encryption and decryption process for the same is shown below, both of them use encryption algorithm.

Operation

The operation of CFB mode is depicted in the following illustration. For example, in the present system, a message block has a size 's' bits where 1 < s < n. The CFB mode requires an initialization vector (IV) as the initial random n-bit input block. The IV need not be secret. Steps of operation are −

- Load the IV in the top register.
- Encrypt the data value in top register with underlying block cipher with key K.
- Take only 's' number of most significant bits (left bits) of output of encryption process and XOR them with 's' bit plaintext message block to generate ciphertext block.
- Feed ciphertext block into top register by shifting already present data to the left and continue the operation till all plaintext blocks are processed.
- Essentially, the previous ciphertext block is encrypted with the key, and then the result is XORed to the current plaintext block.
- Similar steps are followed for decryption. Pre-decided IV is initially loaded at the start of decryption



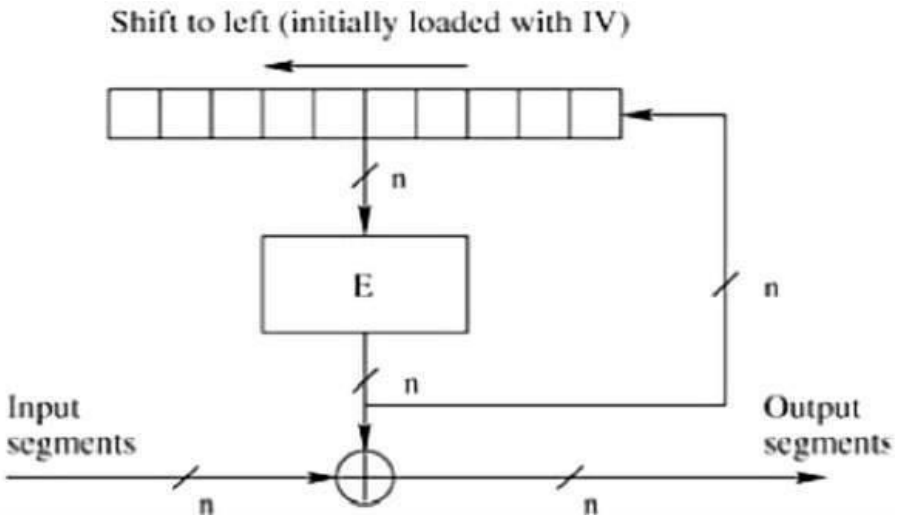
Output Feedback (OFB) Mode

The output feedback mode follows nearly same process as the Cipher Feedback mode except that it sends the encrypted output as feedback instead of the actual cipher which is XOR output. In this output feedback mode, all bits of the block are send instead of sending selected *s* bits. The Output Feedback mode of block cipher holds great resistance towards bit transmission errors. It also decreases dependency or relationship of cipher on plaintext.

It involves feeding the successive output blocks from the underlying block cipher back to it. These feedback blocks provide string of bits to feed the encryption algorithm which act as the key-stream generator as in case of CFB mode.

The key stream generated is XOR-ed with the plaintext blocks. The OFB mode requires an IV as the initial random n-bit input block. The IV need not be secret.

The operation is depicted in the following illustration −



Counter (CTR) Mode

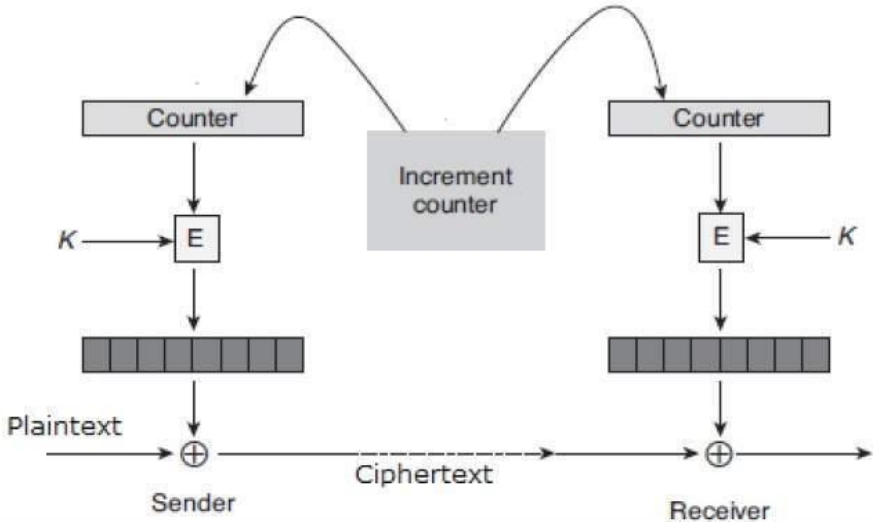
The Counter Mode or CTR is a simple counter based block cipher implementation. Every time a counter initiated value is encrypted and given as input to XOR with plaintext which results in ciphertext block. The CTR mode is independent of feedback use and thus can be implemented in parallel.

It can be considered as a counter-based version of CFB mode without the feedback. In this mode, both the sender and receiver need to access to a reliable counter, which computes a new shared value each time a ciphertext block is exchanged. This shared counter is not necessarily a secret value, but challenge is that both sides must keep the counter synchronized.

Operation

Both encryption and decryption in CTR mode are depicted in the following illustration. Steps in operation are −

- Load the initial counter value in the top register is the same for both the sender and the receiver. It plays the same role as the IV in CFB (and CBC) mode.
- Encrypt the contents of the counter with the key and place the result in the bottom register.
- Take the first plaintext block P1 and XOR this to the contents of the bottom register. The result of this is C1. Send C1 to the receiver and update the counter. The counter update replaces the ciphertext feedback in CFB mode.
- Continue in this manner until the last plaintext block has been encrypted.
- The decryption is the reverse process. The ciphertext block is XORed with the output of encrypted contents of counter value. After decryption of each ciphertext block counter is updated as in case of encryption.



Hybrid Encryption

Hybrid encryption is a mode of encryption that merges two or more encryption systems. It incorporates a combination of asymmetric and symmetric encryption to benefit from the strengths of each form of encryption. These strengths are respectively defined as speed and security.

Hybrid encryption is considered a highly secure type of encryption as long as the public and private keys are fully secure. A hybrid encryption scheme is one that blends the convenience of an asymmetric encryption scheme with the effectiveness of a symmetric encryption scheme. Hybrid encryption is achieved through data transfer using unique session keys along with symmetrical encryption. Public key encryption is implemented for random symmetric key encryption. The recipient then uses the public key encryption method to decrypt the symmetric key. Once the symmetric key is recovered, it is then used to decrypt the message.

Advantages of Hybrid Encryption

The combination of encryption methods has various advantages. One is that a connection channel is established between two users' sets of equipment. Users then have the ability to communicate through hybrid encryption. Asymmetric encryption can slow down the encryption process, but with the simultaneous use of symmetric encryption, both forms of encryption are enhanced. The result is the added security of the transmittal process along with overall improved system performance.

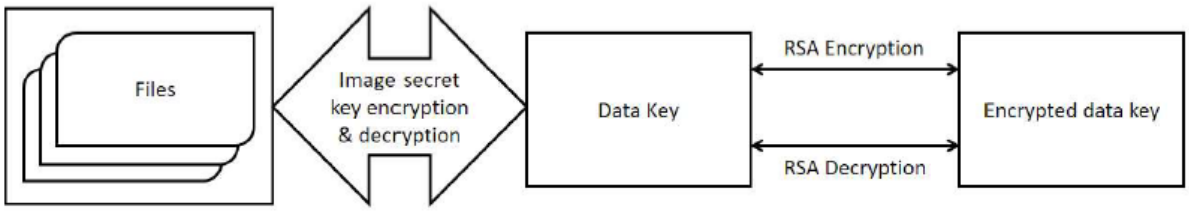


Fig. 1. Hybrid encryption mechanism.