2021-03-30

March 2021

1 Block Ciphers: Operations Modes.

A block cipher is much more than just an encryption algorithm. It can be used as a versatile building block with which a diverse set of cryptographic mechanisms can be realized.

1.1 ECB, Electronic Codebook

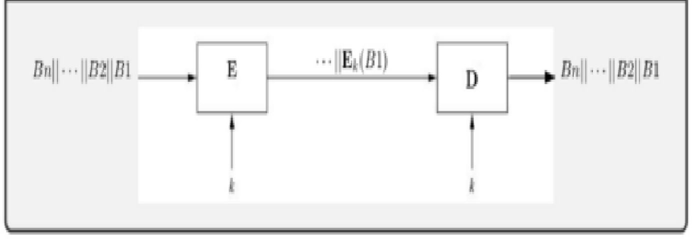
Assume that the block cipher encrypts (decrypts) blocks of size b bits. Messages which exceed b bits are partitioned into b-bit blocks. If the length of the message is not a multiple of b bits, it must be padded to a multiple of b bits prior to encryption.

The message P is divided into blocks:

$$P = B_n ... || B2 || B1$$

and each block is encrypted separately with the same key k.

Figure 5.1.1: ECB Mode



Advantages:

- Synchronization is not necessary(blocks can be decipher independently of each other):Block synchronization between the encryption and decryption parties Alice and Bob is not necessary, i.e., if the receiver does not receive all encrypted blocks due to transmission problems, it is still possible to decrypt the received blocks.
- ECB encryption can be do in parallel.
- Errors remains localized e.g. transmission errors.

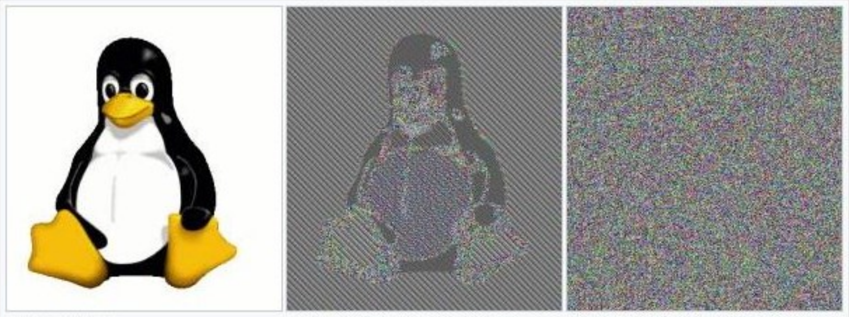
Disadvantages:

- The main problem of the ECB mode is that it encrypts highly deterministically. This means that identical plaintext blocks result in identical ciphertext blocks, as long as the key does not change. The ECB mode can be viewed as a gigantic code book — hence the mode's name—which maps every input to a certain output. Of course, if the key is changed the entire code book changes, but as long as the key is static the book is fixed. This has several undesirable consequences. First, an attacker recognizes if the same message has been sent twice simply by looking at the ciphertext. Deducing information from the ciphertext in this way is called traffic analysis.
  this means there are not integrity protection: the adversary can invert or

substitute blocks. If the key is not replaced it is possible to detect blocks sent twice by just watching the cipher block: Traffic analysis.

In particular: The disadvantage of this method is a lack of diffusion. Because ECB encrypts identical plaintext blocks into identical ciphertext blocks, it does not hide data patterns well. ECB is not recommended for use in cryptographic protocols. A striking example of the degree to which ECB can leave plaintext data patterns in the ciphertext can be seen when ECB mode is used to encrypt a bitmap image which uses large areas of uniform color. While the color of each individual pixel is encrypted, the overall image may still be discerned, as the pattern of identically colored pixels in the original remains in the encrypted version.

nal remains in the encrypted version.



Original image

Encrypted using ECB mode

Modes other than ECB result in pseudo-randomness

Figure 1: The third image is how the image might appear encrypted with CBC, CTR or any of the other more secure modes—indistinguishable from random noise. Note that the random appearance of the third image does not ensure that the image has been securely encrypted; many kinds of insecure encryption have been developed which would produce output just as "random-looking".

Note: The name Electronic Codebook comes from the fact that given a key k each clear block is cipher into a unique cipher block. So we can imagine a huge book in which each line contains the pair clear block / cipher block. Such a book can be regarded as a ”code”.

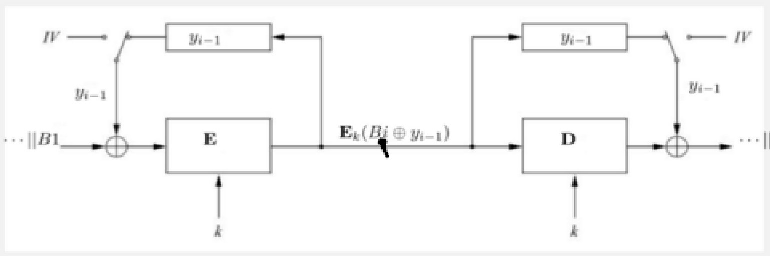
1.2 CBC, Cipher Block Chaining

The message P is divided into blocks:

$$P = B_n...||B_2||B_1$$

but blocks are chiper according the following figure:

Figure 5.1.4: CBC Mode

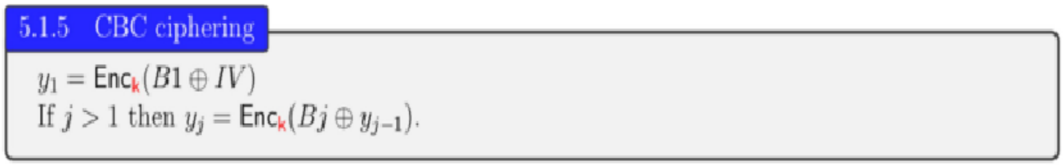


IV is called initializating vector it is usually a nonce. If IV is random the CBC is not deterministic,i.e. probabilistic encryption.

In CBC mode, each block of plaintext is XORed with the previous ciphertext block before being encrypted. This way, each ciphertext block depends on all plaintext blocks processed up to that point. To make each message unique, an initialization vector must be used in the first block.
Example:
we have different blocks: $B_3||B_2||B_1$ and we use the IV in the buffer to do $IV \bigoplus B_1$ and encrypt this with the key — $Enc_{pb}(IV \bigoplus B_1) = y_1$

y_1 goes in input to the the second block and it is stored in the buffer.

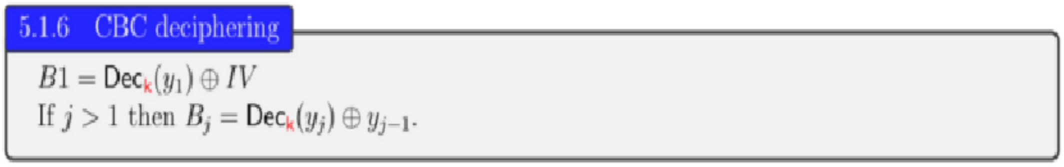
we do the same with the second block but in this case the B_1 is xored with the result of the previous encryption, i.e.
$B_2 \bigoplus y_1$ —$Enc_{pb}(B_2 \bigoplus y_1) = y_2$
$B_3 \bigoplus y_2$ — $Enc_{pb}(B_3 \bigoplus y_2) = y_3$
and so on.



5.1.5 CBC ciphering

$y_1 = \mathsf{Enc}_k(B1 \oplus IV)$

If $j > 1$ then $y_j = \mathsf{Enc}_k(Bj \oplus y_{j-1})$.



5.1.6 CBC deciphering

$B1 = \mathsf{Dec}_k(y_1) \oplus IV$

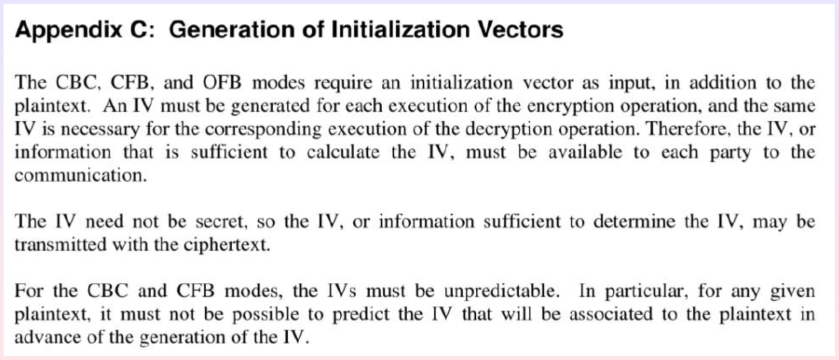
If $j > 1$ then $B_j = \mathsf{Dec}_k(y_j) \oplus y_{j-1}$.

Advantages:

- Allows Probabilistic Encryption:If we encrypt a string of blocks x1, . . . ,xt once with a first IV and a second time with a different IV, the two resulting ciphertext sequences look completely unrelated to each other for an attacker. Note that we do not have to keep the IV secret. However, in most cases, we want the IV to be a nonce, i.e., a number used only once. There are many different ways of generating and agreeing on initialization values. In the simplest case, a randomly chosen number is transmitted in the clear between the two communication parties prior to the encrypted session. Alternatively it is a counter value that is known to Alice and Bob, and it is incremented every time a new session starts (which requires that the counter value must be stored between sessions)
- Blocks depends upon each other as in a chain. So changing a bit from IV or from a block affects all the following cipher blocks.
- Deciphering in CBC mode can be done in parallel: $B_j = Dec(y_j) \bigoplus y_{j-1}$ so Bj can be recover from two consecutive enciphered blocks.

Note: how the deciphering works in parallel? Image you have to decrypt $y_1 y_2 y_3 y_4, y_5$ Dechipering by parallelization means you have to use different computer (e.g. pc1, pc2, pc3, pc4, pc5). each computer is feeded with 2 blocks consecutive: to feed computer1 you send y1 and y2, to feed PC2 you send y2 and y3 e so on.
each one of them works independently and in this way it is possibile recover B1, b2, b3,b4,b5 from each computer.

Decrypting with the incorrect IV causes the first block of plaintext to be corrupt but subsequent plaintext blocks will be correct. This is because each block is XORed with the ciphertext of the previous block, not the plaintext, so one does not need to decrypt the previous block before using it as the IV for the decryption of the current one. This means that a plaintext block can be recovered from two adjacent blocks of ciphertext. As a consequence, decryption can be parallelized. Note that a one-bit change to the ciphertext causes complete corruption of the corresponding block of plaintext, and inverts the corresponding bit in the following block of plaintext, but the rest of the blocks remain intact.



Appendix C: Generation of Initialization Vectors

The CBC, CFB, and OFB modes require an initialization vector as input, in addition to the plaintext. An IV must be generated for each execution of the encryption operation, and the same IV is necessary for the corresponding execution of the decryption operation. Therefore, the IV, or information that is sufficient to calculate the IV, must be available to each party to the communication.

The IV need not be secret, so the IV, or information sufficient to determine the IV, may be transmitted with the ciphertext.

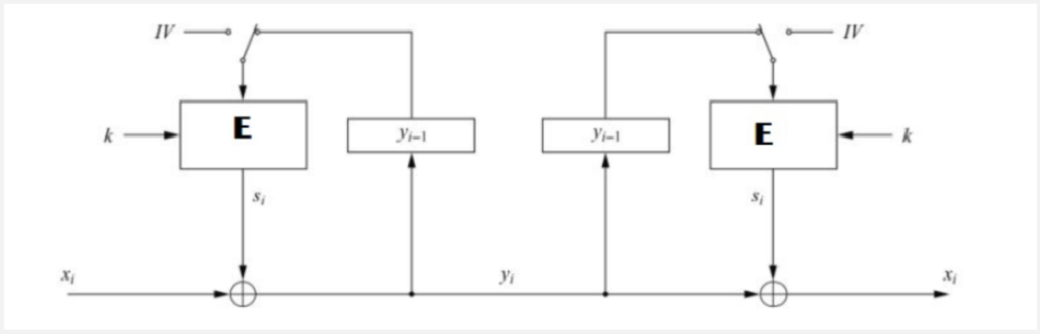
For the CBC and CFB modes, the IVs must be unpredictable. In particular, for any given plaintext, it must not be possible to predict the IV that will be associated to the plaintext in advance of the generation of the IV.

NIST 800-38a

1.3 CFB, Cipher FeedBack mode (PRNG)

In this mode the block cipher is used to construct a stream cipher:

Figure 5.1.7: CFB Mode



The idea behind the CFB mode is as follows: To generate the first key stream block s1, we encrypt an IV. For all subsequent key stream blocks s2, s3, . . ., we encrypt the previous ciphertext. Since the CFB mode forms a stream cipher, encryption and decryption are exactly the same operation. The CFB mode is an example of an asynchronous stream cipher since the stream cipher output is also a function of the ciphertext. As a result of the use of an IV, the

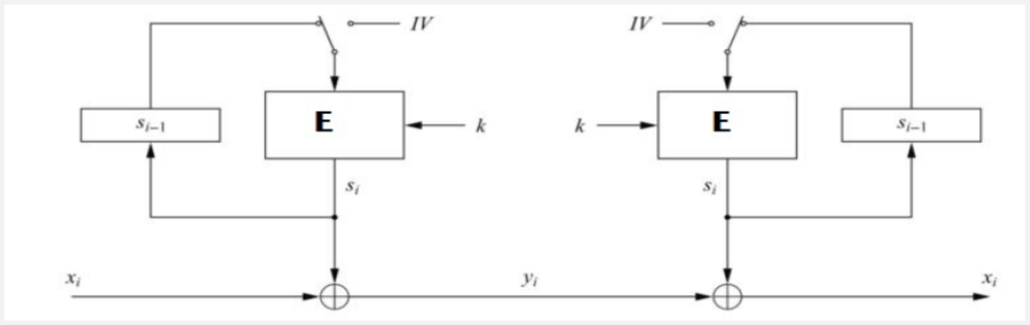
OFB encryption is also nondeterministic, hence, encrypting the same plaintext twice results in different ciphertexts. As in the case for the CBC mode, the IV should be a nonce. One advantage of the OFB mode is that the block cipher computations are independent of the plaintext. Hence, one can precompute one or several blocks si of key stream material.

Exercise 5.1-8: Take a look to the explanation of FED in CFB mode at page 5 of the document FIPS81
The Cipher Feedback (CFB) mode is defined as follows. A message to be encrypted is divided into data units each containing K bits (K = 1,2,... ,64). In both the CFB encrypt and decrypt operations, an initialization vector (IV) of length L is used. The IV is placed in the least significant bits of the DES input block with the unused bits set to "0's," i.e., (I1,I2,...,I64) = (0,0,...,0,IV1,IV2,IVL). This input block is processed through the DES device in the encrypt state to produce an output block. During encryption, cipher text is produced by exclusiveORing a K-bit plain text data unit with the most significant K bits of the output block, i.e., $(C1, C2, ..., CK) = (D1^O 1, D2^O 2, ..., DK^O K)$. Similarly, during decryption, plain text is produced by exclusive-ORing a K-bit unit of cipher text with the most significant K bits of the output block, i.e., $(D1, D2, ..., DK) = (C1^O 1, C2^O 2, ..., CK^O K)$. In both cases the unused bits of the DES output block are discarded. In both cases the next input block is created by discarding the most signif icant K bits of the previous input block, shifting the remaining bits K positions to the left and then inserting the K bits of cipher text just produced in the encryption operation or just used in the decrypt operation into the least significant bit positions, i.e., $(I1, I2, ..., I64) = (I[K + 1], I[K + 2], ..., 164, C1, C2, ..., CK)$. This input block is then processed through the DES device in the encrypt state to produce the next output block. This process continues until the entire plain text message has been encrypted or until the entire cipher text message has been decrypted. The CFB mode may operate on data units of length 1 through 64 inclusive. K-bit CFB is defined to be the CFB mode operating on data units of length K for K = 1,2,... ,64. For each operation of the DES device one K-bit unit of plain text produces one K-bit unit of cipher text or one K-bit unit of cipher text produces one K-bit unit of plain text. An acceptable alternative for 8-bit CFB when enciphering 7-bit entities using an 8-bit feedback path is to insert a "1" bit in bit position one of the 8-bit feedback path, i.e., ("1",C1,C2,... ,C7). This results in a "1" always being placed in bit location 57 of the DES input block. This alternative is called the 7-bit CFB(a) mode of operation

1.4 OFB, Output FeedBack mode (PRNG)

In this mode the block cipher is used to construct a stream cipher:

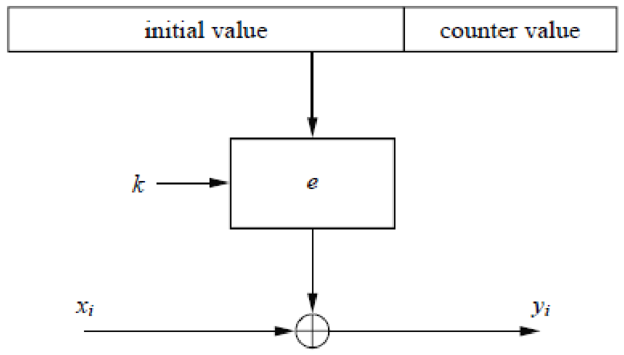
Figure 5.1.9: OFB Mode



In the Output Feedback (OFB) mode a block cipher is used to build a stream cipher encryption scheme. Note that in OFB mode the key stream is not generated bitwise but instead in a blockwise fashion. The output of the cipher gives us b key stream bits, where b is the width of the block cipher used, with which we can encrypt b plaintext bits using the XOR operation. The idea behind the OFB mode is quite simple. We start with encrypting an IV with a block cipher. The cipher output gives us the first set of b key stream bits. The next block of key stream bits is computed by feeding the previous cipher output back into the block cipher and encrypting it. The OFB mode forms a synchronous stream cipher as the key stream does not depend on the plain or ciphertext. In fact, using the OFB mode is quite similar to using a standard stream cipher such as RC4 or Trivium. Since the OFB mode forms a stream cipher, encryption and decryption are exactly the same operation. As can be seen in the right-hand part of the figure, the receiver does not use the block cipher in decryption mode to decrypt the ciphertext. This is because the actual encryption is performed by the XOR function, and in order to reverse it, i.e., to decrypt it, we simply have to perform another XOR function on the receiver side. This is in contrast to ECB and CBC mode, where the data is actually being encrypted and decrypted by the block cipher.

As a result of the use of an IV, the OFB encryption is also nondeterministic, hence, encrypting the same plaintext twice results in different ciphertexts. As in the case for the CBC mode, the IV should be a nonce. One advantage of the OFB mode is that the block cipher computations are independent of the plaintext. Hence, one can precompute one or several blocks si of key stream material.

1.5 CTR, Counter Mode



Another mode which uses a block cipher as a stream cipher is the Counter (CTR) mode. As in the OFB and CFB modes, the key stream is computed in a blockwise fashion. The input to the block cipher is a counter which assumes a different value every time the block cipher computes a new key stream block. We have to be careful how to initialize the input to the block cipher. We must prevent using the same input value twice. Otherwise, if an attacker knows one ofthe two plaintexts that were encrypted with the same input, he can compute the key stream block and thus immediately decrypt the other ciphertext. In order to achieve this uniqueness, often the following approach is taken in practice. Let's assume a block cipher with an input width of 128 bits, such as an AES. First we choose an IV that is a nonce with a length smaller than the block length, e.g., 96 bits. The remaining 32 bits are then used by a counter with the value CTR 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 example, the number of blocks we can encrypt without choosing a new IV is 232. Since every block consists of 8 bytes, a maximum of 8×232 = 235 bytes, or about 32 Gigabytes, can be encrypted before a new IV must be generated.

So the plaintext P is divided into N blocks P = BN|| · · · ||B1. Then N blocks TN, · · · , T1 called ”counters” are created and here is how the block cipher is used;

Usually the counters TN, · · · , T1 are obtained from the initial block T1 by increment or partial increment. For example, $T1 = IV||Q$, where $Q \in Z_{2^m}$ and $Tj := IV||(Q + j - 1)$ and IV is a nonce.

Counters can be created by a LFSR form an initial state T1.

5.1.10 CTR ciphering

$\mathrm{O}_j = \mathsf{Enc}_{\mathsf{k}}(Tj)$ for $j = 1, \cdots, N$
$\mathrm{C}_j = Bj \oplus \mathrm{O}_j$ for $j = 1, \cdots, N$

5.1.11 CTR deciphering

$\mathrm{O}_j = \mathsf{Enc}_{\mathsf{k}}(Tj)$ for $j = 1, \cdots, N$
$Bj = \mathrm{C}_j \oplus \mathrm{O}_j$ for $j = 1, \cdots, N$

NOTE 5.1.12

It is important to prevent two counters from being equal. Otherwise you risk a KPA attack: if $T1 = Tj$ and $(Ct_1, B1)$ known to the cryptanalyst then he also easily gets Bj.

To avoid two counters being the same, you should choose the size m of Q to be able to encrypt all N blocks of our message before changing the nonce IV, ie $N \leq 2^m$.

It is not necessary for the first counter $T1$ to be secret. The sender could send $T1$ together with the first encrypted block Ct_1 to the recipient.

Advantages 🙂

CRT mode is parallelizable.

1.6 Galois Counter Mode (GCM)

Galois mode is combination of CTR mode and a MAC i.e. Message Authentication Code. Namely, a tag t of 128 bits is produce to allow the receiver to check the integrity of the message.

The Galois Counter Mode (GCM) is an encryption mode which also computes a message authentication code (MAC). AMAC provides a cryptographic checksum that is computed by the sender, Alice, and appended to the message. Bob also computes a MAC from the message and checks whether his MAC is the same as the one computed by Alice. This way, Bob can make sure that (1) the message was really created by Alice and (2) that nobody tampered with the ciphertext during transmission. These two properties are called message authentication and integrity, respectively.
GCM protects the confidentiality of the plaintext x by using an encryption in counter mode. Additionally, GCM protects not only the authenticity of the plaintext x but also the authenticity of a string AAD called additional authenticated data. This authenticated data is, in contrast to the plaintext, left in clear in this mode of operation. In practice, the string AAD might include addresses and parameters in a network protocol. The GCM consists of an underlying block cipher and a Galois field multiplier with which the two GCM functions authenticated encryption and authenticated decryption are realized. The cipher needs to have a block size of $128 (= 2^8)$ bits such as AES.

It is called "Galois" because blocks are regarded as numbers of the finite

Galois field GF(2^8).

On the sender side, GCM encrypts data using the Counter Mode (CTR) followed by the computation of a MAC value. For encryption, first an initial counter is derived from an IV and a serial number. Then the initial counter value is incremented, and this value is encrypted and XORed with the first plaintext block. For subsequent plaintexts, the counter is incremented and then encrypted. Note that the underlying block cipher is only used in encryption mode. GCM allows for precomputation of the block cipher function if the initialization vector is known ahead of time.

For authentication, GCM performs a chained Galois field multiplication. For every plaintext xi an intermediate authentication parameter gi is derived. gi is computed as the XOR sum of the current ciphertext yi and gi, and multiplied by the constant H. The value H is a hash subkey which is generated by encryption of the all-zero input with the block cipher. All multiplications are in the 128-bit Galois field GF(2128) with the irreducible polynomial P(x) = x128+x7+x2+x+1. Since only one multiplication is required per block cipher encryption, the GCM mode adds very little computational overhead to the encryption. The receiver of the packet [(y1, . . . ,yn),T,ADD] decrypts the ciphertext by also applying the Counter mode. To check the authenticity of the data, the receiver also computes an authentication tag T' using the received ciphertext and ADD as input. He employs exactly the same steps as the sender. If T and T' match, the receiver is assured that the cipertext (and ADD) were not manipulated in transit and that only the sender could have generated the message.

In practice, the string AAD might include IP addresses and parameters in a network protocol. So the tag is used for the authentication of the AAD.

Here is how it works. The message P = BN|| $\cdots$ ||B1 consists of blocks of 128 bits. A counter is used to produce Tj and blocks are encrypted as $C_j = B_j \bigoplus T_j$ Here is how the tag is produced:

5.1.13 GCM tag t

$H = \mathsf{Enc}_{\mathsf{k}}(0)$;
$g_0 = \mathrm{AAD} \otimes H$; where $\otimes$ is the Galois multiplication in GF(2^8).
$g_j = (g_{j-1} \oplus \mathrm{C}_j) \otimes H$ for $j = 1, \cdots, N$;
$\mathsf{t} = (g_N \otimes H) \oplus \mathsf{Enc}_{\mathsf{k}}(T1 - 1)$.

The receiver gets

$$(C_N, ..., C_1, t, ADD)$$

from the sender