NIST Block Cipher Modes of Operation for Confidentiality

WILLIAM STALLINGS

Abstract In this article, we describe the five block cipher modes of operation that have been approved by the National Institute of Standards and Technology (NIST) for confidentiality. Each mode specifies an algorithm for encrypting/decrypting data sequences that are longer than a single block.

Keywords block cipher, modes of operation, symmetric cipher

Introduction

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. When multiple blocks of plaintext are encrypted using the same key, a number of security issues arise. To apply a block cipher in a variety of applications, five “modes of operation” have been defined by the National Institute of Standards and Technology (NIST) in SP 800-38A [4]. In essence, a mode of operation is a technique for enhancing the effect of a cryptographic algorithm or adapting the algorithm for an application, such as applying a block cipher to a sequence of data blocks or a data stream. The five modes are intended to cover a wide variety of applications of encryption for which a block cipher could be used. These modes are intended for use with any symmetric block cipher, including triple DES and AES.

Electronic Code Book

The simplest mode of operation is the electronic codebook (ECB) mode, in which plaintext is handled one block at a time and each block of plaintext is encrypted using the same key (Figure 1). The term codebook is used because, for a given key, there is a unique ciphertext for every $b$-bit block of plaintext. Therefore, we can imagine a gigantic codebook in which there is an entry for every possible $b$-bit plaintext pattern showing its corresponding ciphertext.

For a message longer than $b$ bits, the procedure is simply to break the message into $b$-bit blocks, padding the last block if necessary. Decryption is performed one block at a time, always using the same key. In Figure 1, the plaintext (padded as necessary) consists of a sequence of $b$-bit blocks, $P_1, P_2, \ldots, P_N$; the corresponding
sequence of ciphertext blocks is $C_1, C_2, \ldots, C_N$. We can define ECB mode as follows:

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

The ECB method is ideal for a short amount of data, such as an encryption key. Thus, if you want to transmit a DES or AES key securely, ECB is the appropriate mode to use.

The most significant characteristic of ECB is that the same $b$-bit block of plaintext, if it appears more than once in the message, always produces the same ciphertext.

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. For example, if it is known that the message always starts out with certain predefined fields, then the cryptanalyst may have a number of known plaintext-ciphertext pairs to work with. If the message has repetitive elements, with a period of repetition a multiple of $b$ bits, then these elements can be identified by the analyst. This may help in the analysis or may provide an opportunity for substituting or rearranging blocks.
Knudson [2] lists the following criteria and properties for evaluating and constructing block cipher modes of operation that are superior to ECB:

- **Overhead**: The additional operations for the encryption and decryption operation when compared to encrypting and decrypting in the ECB mode.
- **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. What is meant here is a bit error that occurs in the transmission of a ciphertext block, not a computational error in the encryption of a plaintext block.
- **Diffusion**: How the plaintext statistics are reflected in the ciphertext. Low entropy plaintext blocks should not be reflected in the ciphertext blocks.
- **Security**: Do ciphertext blocks leak information about the plaintext blocks?

**Cipher Block Chaining Mode**

To overcome the security deficiencies of ECB, we would like a technique in which the same plaintext block, if repeated, produces different ciphertext blocks. A simple way to satisfy this requirement is the cipher block chaining (CBC) mode (Figure 2). In

![Cipher Block Chaining Mode Diagram](image)

**Figure 2.** Cipher block chaining (CBC) mode.
this scheme, the 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. In effect, we have chained together the processing of the sequence of plaintext blocks. The input to the encryption function for each plaintext block bears no fixed relationship to the plaintext block. Therefore, repeating patterns of \( b \) bits are not exposed.

For decryption, each cipher block is passed through the decryption algorithm. The result is XORed with the preceding ciphertext block to produce the plaintext block. To see that this works, we can write

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

Then

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

\[
D(K, C_j) = C_{j-1} \oplus P_j
\]

\[
C_{j-1} \oplus D(K, C_j) = C_{j-1} \oplus C_{j-1} \oplus P_j = P_j.
\]

To produce the first block of ciphertext, an initialization vector (IV) is XORed with the first block of plaintext. On decryption, the IV is XORed with the output of the decryption algorithm to recover the first block of plaintext. The IV is a data block that is of the same size as the cipher block. We can define CBC mode as follows:

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

The IV must be known to both the sender and receiver but be unpredictable by a third party. 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. For maximum security, the IV should be protected against unauthorized changes. This could be done by sending the IV using ECB encryption. One reason for protecting the IV is as follows: If an opponent is able to fool the receiver into using a different value for IV, then the opponent is able to invert selected bits in the first block of plaintext. To see this, consider the following:

\[
C_1 = E(K, [IV \oplus P_1])
\]

\[
P_1 = IV \oplus D(K, C_1).
\]

Now use the notation that \( X[i] \) denotes the \( i \)th bit of the \( b \)-bit quantity \( X \). Then

\[
P_1[i] = IV[i] \oplus D(K, C_1)[i].
\]

Then, using the properties of XOR, we can state

\[
P_1[i]'' = IV[i]'' \oplus D(K, C_1)[i],
\]
where the prime notation denotes bit complementation. This means that if an opponent can predictably change bits in IV, the corresponding bits of the received value of $P_1$ can be changed.

For other possible attacks based on prior knowledge of IV, see [5].

So long as it is unpredictable, the specific choice of IV is unimportant. SP800-38a recommends two possible methods: The first method is to apply the encryption function, under the same key that is used for the encryption of the plaintext, to a nonce. The nonce must be a data block that is unique to each execution of the encryption operation. For example, the nonce may be a counter, a timestamp, or a message number. The second method is to generate a random data block using a random number generator.

In CBC encryption, the input block to each forward cipher operation (except the first) depends on the result of the previous forward cipher operation, so the forward cipher operations cannot be performed in parallel. In CBC decryption, however, the input blocks for the inverse cipher function, i.e., the ciphertext blocks, are immediately available, so that multiple inverse cipher operations can be performed in parallel.

With the ECB mode, if there is an error in a block of the transmitted ciphertext, only the corresponding plaintext block is affected. However, in the CBC mode, this error propagates. For example, an error in the transmitted $C_1$ (Figure 2) obviously corrupts $P_1$ and $P_2$. However, no blocks beyond $P_2$ are affected. For example, suppose $C_1$ is corrupted. The output block $P_3$ depends only on the input blocks $C_2$ and $C_3$.

Now suppose that there is a bit error in the source version of $P_1$. This affects $C_1$. But since $C_1$ is input to the calculation of $C_2$, $C_2$ is affected. This effect carries through indefinitely, so that all ciphertext blocks are affected. However, at the receiving end, the decryption algorithm restores the correct plaintext for blocks except the one in error. You can show this by writing out the equations for the decryption. Therefore, the error only affects the corresponding decrypted plaintext block.

**Cipher Feedback Mode**

For AES, DES, or any block cipher, encryption is performed on a block of $b$ bits. In the case of DES, $b = 64$, and in the case of AES, $b = 128$. However, it is possible to convert a block cipher into a stream cipher using one of the three modes to be discussed in this and the next two sections: cipher feedback (CFB) mode, output feedback (OFB) mode, and counter (CTR) mode. A stream cipher eliminates the need to pad a message to be an integral number of blocks. It also can operate in real time. Thus, if a character stream is being transmitted, each character can be encrypted and transmitted immediately using a character-oriented stream cipher.

One desirable property of a stream cipher is that the ciphertext be of the same length as the plaintext. Thus, if 8-bit characters are being transmitted, each character should be encrypted to produce a ciphertext output of 8 bits. If more than 8 bits are produced, transmission capacity is wasted.

Figure 3 depicts the CFB scheme. In the figure, it is assumed that the unit of transmission is $s$ bits; a common value is $s = 8$. 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. In this case, rather than units of $b$ bits, the plaintext is divided into segments of $s$ bits.
First, consider encryption. The input to the encryption function is a $b$-bit shift register that is initially set to some initialization vector (IV). The leftmost (most significant) $s$ bits of the output of the encryption function are XORed with the first segment of plaintext $P_1$ to produce the first unit of ciphertext $C_1$, which is then transmitted. In addition, the contents of the shift register are shifted left by $s$ bits and $C_1$ is placed in the rightmost (least significant) $s$ bits of the shift register. This process continues until all plaintext units have been encrypted.

For decryption, the same scheme is used, except that the received ciphertext unit is XORed with the output of the encryption function to produce the plaintext unit. Note that it is the encryption function that is used, not the decryption function.

**Figure 3.** $s$-bit cipher feedback (CFB) mode.
function. This is easily explained. Let $\text{MSB}_s(X)$ be defined as the most significant $s$ bits of $X$. Then

$$C_1 = P_1 \oplus \text{MSB}_s[E(K, IV)].$$

Therefore, by rearranging terms:

$$P_1 = C_1 \oplus \text{MSB}_s[E(K, IV)].$$

The same reasoning holds for subsequent steps in the process.

We can define CFB mode as follows:

| CFB | $I_1 = IV$ | $I_j = \text{LSB}_s(I_{j-1} | C_{j-1})$ if $s = 2, \ldots, N$ | $O_j = E(K, I_j)$ if $s = 1, \ldots, N$ | $C_j = P_j \oplus \text{MSB}_s(O_j)$ if $s = 1, \ldots, N$ |
|-----|-------------|---------------------------------|-----------------------|---------------------|

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 plaintext also depends on the plaintext.

In CFB encryption, like CBC encryption, the input block to each forward cipher function (except the first) depends on the result of the previous forward cipher function; therefore, multiple forward cipher operations cannot be performed in parallel. In CFB decryption, the required forward cipher operations can be performed in parallel if the input blocks are first constructed (in series) from the IV and the ciphertext.

For the CFB mode, the effect of error propagation is somewhat different from the other modes. Suppose a bit error occurs in the transmission of a ciphertext byte in 8-bit CFB mode. Nine plaintext characters are affected. The plaintext character corresponding to the ciphertext character is obviously altered. In addition, the altered ciphertext character enters the shift register and is not removed until the next eight characters are processed.

### Output Feedback Mode

The output feedback (OFB) mode is similar in structure to that of CFB. As can be seen in Figure 4, it is the output of the encryption function that is fed back to the shift register in OFB, whereas in CFB the ciphertext unit is fed back to the shift register. The other difference is that the OFB mode operates on full blocks of plaintext and ciphertext, not on an $s$-bit subset. We can show that decryption is performed properly as follows:

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

Therefore, by rearranging terms:

$$P_j = C_j \oplus E(K, [C_{j-1} \oplus P_{j-1}]).$$
We can define OFB mode as follows:

Let the size of a block be $b$. If the last block of plaintext contains $u$ bits, 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.

As with CBC and CFB, the OFB mode requires an initialization vector. In the case of OFB, the IV must be a nonce; that is, the IV must be unique to each execution of the encryption operation. The reason for this is that the sequence of encryption output blocks, $O_i$, depends only on the key and the IV, and does not depend on the plaintext. Therefore, for a given key and IV, the stream of output bits

Figure 4. Output feedback (OFB) mode.
used to XOR with the stream of plaintext bits is fixed. If two different messages had an identical block of plaintext in the identical position, then an attacker would be able to determine that portion of the $O_i$ stream.

One advantage of the OFB method is that bit errors in transmission do not propagate. For example, if a bit error occurs in $C_1$, only the recovered value of $P_1$ is affected; subsequent plaintext units are not corrupted. With CFB, $C_1$ also serves as input to the shift register and therefore causes additional corruption downstream.

The disadvantage of OFB is that it is more vulnerable to a message stream modification attack than is CFB. Consider that complementing a bit in the ciphertext complements the corresponding bit in the recovered plaintext. Thus, controlled changes to the recovered plaintext can be made. This may make it possible for an opponent, by making the necessary changes to the checksum portion of the message as well as to the data portion, to alter the ciphertext in such a way that it is not detected by an error-correcting code. For a further discussion, see [5].

OFB has the structure of a typical stream cipher, in that the cipher generates a stream of bits as a function of an initial value and a key, and that stream of bits is XORed with the plaintext bits. The generated stream that is XORed with the plaintext is itself independent of the plaintext; this is highlighted by dashed boxes in Figure 4. One distinction from purpose-built stream ciphers is that OFB encrypts plaintext a full block at a time, where typically a block is 64 or 128 bits. Many stream ciphers encrypt one byte at a time.

**Counter Mode**

Although interest in the counter mode (CTR) has increased recently, with applications to ATM (asynchronous transfer mode) network security and IPSec (IP security), this mode was proposed early on (e.g., [1]).

Figure 5 depicts the CTR mode. A counter, equal to the plaintext block size is used. The only requirement stated in SP 800-38A is that the counter value must be different for each plaintext block that is encrypted. 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 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. Given a sequence of counters $T_1, T_2, \ldots, T_N$ can define CBC mode as follows:

<table>
<thead>
<tr>
<th>CTR</th>
<th>$C_j = P_j \oplus E(K, T_j)$ for $j = 1, \ldots, N-1$</th>
<th>$P_j = C_j \oplus E(K, T_j)$ for $j = 1, \ldots, N-1$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$C_N = P_N \oplus \text{MSB}_b[E(K, T_N)]$</td>
<td>$P_N = C_N \oplus \text{MSB}_b[E(K, T_N)]$</td>
</tr>
</tbody>
</table>

As with the OFB mode, the initial counter value must be a nonce; that is, $T_1$ must be different for all of the messages encrypted using the same key. Further, all $T_j$ 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. In particular, if any plaintext block that is encrypted using a given counter value is known, then
the output of the encryption function can be determined easily from the associated ciphertext block. This output allows any other plaintext blocks that are encrypted using the same counter value to be easily recovered from their associated ciphertext blocks.

One way to ensure the uniqueness of counter values is to continue to increment the counter value by 1 across messages. That is, the first counter value of each message is one more than the last counter value of the preceding message.

The following advantages of CTR mode are listed in Lipmaa et al. [3]:

- **Hardware efficiency:** Unlike the three chaining modes, encryption (or decryption) in CTR mode can be done in parallel 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. This limits the maximum throughput of the algorithm to the reciprocal of the time for one execution of block encryption or decryption. In CTR mode, the throughput is only limited by the amount of parallelism that is achieved.

Figure 5. Counter (CTR) mode.
Software efficiency: Similarly, because of the opportunities for parallel execution in CTR mode, processors that support parallel features, such as aggressive pipelining, multiple instruction dispatch per clock cycle, a large number of registers, and SIMD instructions, can be utilized effectively.

Preprocessing: The execution of the underlying encryption algorithm does not depend on input of the plaintext or ciphertext. Therefore, if sufficient memory is available and security is maintained, preprocessing can be used to prepare the output of the encryption boxes that feed into the XOR functions in Figure 5. When the plaintext or ciphertext input is presented, then the only computation is a series of XORs. Such a strategy greatly enhances throughput.

Random access: The $i$th block of plaintext or ciphertext can be processed in random-access fashion. With the chaining modes, block $C_i$ cannot be computed until the $i-1$ prior blocks are computed. There may be applications in which a ciphertext is stored and it is desired to decrypt just one block; for such applications, the random access feature is attractive.

Provable security: It can be shown that CTR is at least as secure as the other modes discussed in this paper.

Simplicity: Unlike ECB and CBC modes, CTR mode requires only the implementation of the encryption algorithm and not the decryption algorithm. This matters most when the decryption algorithm differs substantially from the encryption algorithm, as it does for AES. In addition, the decryption key scheduling need not be implemented.

Note that, with the exception of ECB, all of the NIST-approved block cipher modes of operation involve feedback. This is clearly seen in Figure 6. To highlight the feedback mechanism, it is useful to think of the encryption function as taking input from a input register whose length equals the encryption block length and with output stored in an output register. The input register is updated one block at a time by the feedback mechanism. After each update, the encryption algorithm is executed, producing a result in the output register. Meanwhile, a block of plaintext is accessed. Note that both OFB and CTR produce output that is independent of both the plaintext and the ciphertext. Thus, they are natural candidates for stream ciphers that encrypt plaintext by XOR one full block at a time.

Other Modes

The five modes of operation approved by NIST that are discussed in this paper are referred to as confidentiality modes because they implement the encryption function. A confidentiality mode takes as input a sequence of plaintext blocks and a secret key and produces a sequence of ciphertext blocks. Another type of block cipher mode of operation is known as the authentication mode. An authentication mode takes as input a sequence of plaintext blocks and a secret key and produces a single output block, known as a message digest. A sender uses this mode and transmits the plaintext message plus the message digest. A recipient who shares a secret key with the sender uses this mode to calculate the message digest and compares the calculated message digest with the received message digest. If the two match, the recipient is assured that (1) the message is from the alleged sender, because only the sender shares the secret key with the recipient; and (2) that the message has not been altered in transit, because this would produce a different message digest.
NIST has approved one authentication mode of operation, the Cipher-Based Message Authentication Code (CMAC). NIST has also approved two modes of operation that combine authentication and encryption; the combination is referred to as authenticated encryption. CCM (Counter with Cipher Block Chaining-Message Authentication Code) combines the counter mode for confidentiality with the cipher block chaining technique for authentication. CCM was designed for use with the IEEE 802.11 standard for wireless local area networks. GCM (Galois/Counter Mode) combines the counter mode for confidentiality with an authentication mechanism that is based on a universal hash function. GCM was designed to facilitate high-throughput hardware implementations.
NIST is also considering a number of other modes of operation for approval. The two most noteworthy are the AES Key Wrap (AESKW) and XTS-AES. AESKW is intended for the authenticated encryption (“wrapping”) of specialized data, such as cryptographic keys, for distribution or storage. AESKW invokes the block cipher about twelve times per block of data. The design provides security properties that may be desired for high assurance applications; the tradeoff is relatively inefficient performance compared to other modes.

NIST is currently in the process of approving XTS-AES as a confidentiality mode of operation. This mode is also an IEEE standard, IEEE Std 1619-2007, which was developed by the IEEE Security in Storage Working Group (P1619). The 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.

For more information on these and other possible block cipher modes of operation, visit the NIST Web site devoted to this subject at http://csrc.nist.gov/groups/ST/toolkit/BCM/index.html.

About the Author

William Stallings holds a Ph.D. from M.I.T. in Computer Science. He has authored numerous books on security, computer networking, and computer architecture. He has twelve times received the award for the Best Computer Science and Engineering Textbook of the Year from the Textbook and Academic Authors Association. His most recent book is Cryptography and Network Security, Principles and Practice, Fifth Edition (Prentice Hall, 2010). He is also co-author, with Lawrie Brown, of Computer Security, Principles and Practice (Prentice Hall, 2008). He created and maintains the Computer Science Student Resource Site at WilliamStallings.com/StudentSupport.html. This site provides documents and links on a variety of subjects of general interest to computer science students (and professionals).

References