# DEVELOPMENT OF BLOCK CIPHER MODES OF OPERATIONS

Dr. Ali Makki Sagheer

**Abstract**

There are two basic types of symmetric cipher algorithms: block ciphers and stream ciphers. Block ciphers operate on blocks of plaintext and ciphertext—usually of 64 bits but sometimes longer. Stream ciphers operate on streams of plaintext and ciphertext one bit or byte (sometimes even one 32-bit word) at a time. With a block cipher, the same plaintext block will always encrypt to the same ciphertext block, using the same key. With a stream cipher, the same plaintext bit or byte will encrypt to a different bit or byte every time it is encrypted. Some of block cipher modes are used as a stream cipher mechanism.

In this paper we introduce three developments of Block Cipher Modes of Operation. of Block cipher. These developments merge between advantages of Counter Mode with other Modes. This gives good secure Modes for General purpose block-oriented transmission Authentication, Random access, Stream-oriented transmission over noisy channel and real time system.

**Keywords:** Block Cipher, Modes of Operations

## 1. INTRODUCTION

A cryptographic mode usually combines the basic cipher, some sort of feedback, and some simple operations. The operations are simple because the security is a function of the underlying cipher and not the mode. Even more strongly, the cipher mode should not compromise the security of the underlying algorithm. There are other security considerations: Patterns in the plaintext should be concealed, input to the cipher should be randomized, manipulation of the plaintext by introducing errors in the ciphertext should be difficult, and encryption of more than one message with the same key should be possible. Efficiency is another consideration. The mode should not be significantly less efficient than the underlying cipher. In some circumstances it is important that the ciphertext be the same size as the plaintext. A third consideration is fault-tolerance. Some applications need to parallelize encryption or decryption, while others need to be able to preprocess as much as possible. It is important that the decrypting process be able to recover from bit errors in the ciphertext stream, or dropped or added bits. As we will see, different modes have different subsets of these characteristics [1].

## 2. BLOCK CIPHER MODES OF OPERATION

A block cipher algorithm is a basic building block for providing data security. To apply a block cipher in a variety of applications, four "modes of operation" have been defined by NIST. 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 four modes are intended to cover virtually all the possible 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. The modes are summarized in Table 1 and described briefly in the next section [2].

**2.1. Electronic Codebook Mode**

The simplest mode 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 [2]. The term codebook is used because, for a given key, there is a unique ciphertext for every b-bit block of plaintext always using the same key. Consider the plaintext (padded as necessary) consists of a sequence of b-bit blocks, $P_1$, $P_2$,..., $P_N$; the corresponding sequence of ciphertext blocks is $C_1$, $C_2$,..., $C_N$.

The ECB method is ideal for a short amount of data, such as an encryption key. Thus, if you want to transmit a DES 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.

| Table 1: Block Cipher Modes of Operation | | |
|---|---|---|
| **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<br>Authentication |
| Cipher Feedback (CFB) | Input is processed j 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<br>Authentication |
| Output Feedback (OFB) | Similar to CFB, except that the input to the encryption algorithm is the preceding DES output. | 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<br>Useful for high-speed requirements |

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, 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 [1].

**2.2. Cipher Block Chaining Mode**

Chaining adds a feedback mechanism to a block cipher: The results of the encryption of previous blocks are fed back into the encryption of the current block. In other words, each block is used to

modify the encryption of the next block. Each ciphertext block is dependent not just on the plaintext block that generated it but on all the previous plaintext blocks and previous ciphertext block before it is encrypted. In cipher block chaining (CBC) mode, the plaintext is XORed with the CBC encryption in action [2]. After a plaintext block is encrypted, the resulting ciphertext is also stored in a feedback register. Before the next plaintext block is encrypted, it is XORed with the feedback register to become the next input to the encrypting routine. The resulting ciphertext is again stored in the feedback register, to be XORed with the next plaintext block, and so on until the end of the message. The encryption of each block depends on all the previous blocks.

Decryption is just as straightforward [2]. A ciphertext block is decrypted normally and also saved in a feedback register. After the next block is decrypted, it is XORed with the results of the feedback register. Then the next ciphertext block is stored in the feedback register, and so on, until the end of the message [1].

Mathematically, this looks like:

$$C_i = E_K(P_i \oplus C_{i-1})$$

$$P_i = C_{i-1} \oplus D_K(C_i)$$

**2.3. Cipher-Feedback Mode**

Block ciphers can also be implemented as a self-synchronizing stream cipher; this is called cipher-feedback (CFB) mode. With CBC mode, encryption cannot begin until a complete block of data is received. This is a problem in some network applications. In a secure network environment, for example, a terminal must be able to transmit each character to the host as it is

entered. When data has to be processed in byte-sized chunks, CBC mode just won't do.

In CFB mode, data can be encrypted in units smaller than the block size. The following example will encrypt one ASCII character at a time (this is called 8-bit CFB), but nothing is sacred about the number eight. You can encrypt data one bit at a time using 1-bit CFB, although using one complete of the block cipher to speed things up is not recommended [3].

The 8-bit CFB mode is working with a 64-bit block algorithm [2]. A block algorithm in CFB mode and can also use 64-bit CFB, or any n-bit CFB where n is less than or equal to the block size operates on a queue the size of the input block. Initially, the queue is filled with an IV, as in CBC mode. The queue is encrypted and the left-most eight bits of the result are XORed with the first 8-bit character of the plaintext to become the first 8-bit character of the ciphertext. This character can now be transmitted. The same eight bits are also moved to the right-most eight bit positions of the queue, and all the other bits move eight to the left. The eight left-most bits are discarded. Then the next plaintext character is encrypted in the same manner. Decryption is the reverse of this process. On both the encryption and the decryption side, the block algorithm is used in its encryption mode [1].

**2.4. Output-Feedback Mode**

Output-feedback (OFB) mode is a method of running a block cipher as a synchronous stream cipher. It is similar to CFB mode, except that n bits of the previous output block are moved into the right-most positions of the queue. Decryption is the reverse of this

process. This is called n-bit OFB. On both the encryption and the decryption sides, the block algorithm is used in its encryption mode. This is sometimes called internal feedback, because the feedback mechanism is independent of both the plaintext and the ciphertext streams [4].

If n is the block size of the algorithm, then n-bit OFB looks like:

$C_i = P_i \oplus S_i$ ; $S_i = E_K(S_{i-1})$

$P_i = C_i \oplus S_i$ ; $S_i = E_K(S_{i-1})$

**2.5. Counter Mode**

Block ciphers in as counter mode use sequence numbers as the input to the algorithm [5, 6, 7]. Instead of using the output of the encryption algorithm to fill the register, the input to the register is a counter. After each block encryption, the counter increments by some constant, typically one. The synchronization and error propagation characteristics of this mode are identical to those of OFB. Counter mode solves the OFB mode problem of n-bit output where n is less than the block length.

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., [6]).

Figure.1 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 2b 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.

This limits the maximum throughput of the algorithm to the reciprocal of the time for one execution of block encryption or decryption. In CTR used, with each encrypted counter XORed with a ciphertext block to recover the corresponding plaintext block mode, the throughput is only limited by the amount of parallelism that is achieved.

Software efficiency: Similarly, because of the opportunities for Lists the following advantages of CTR mode: [8]

- 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.
- 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 effectively utilized.

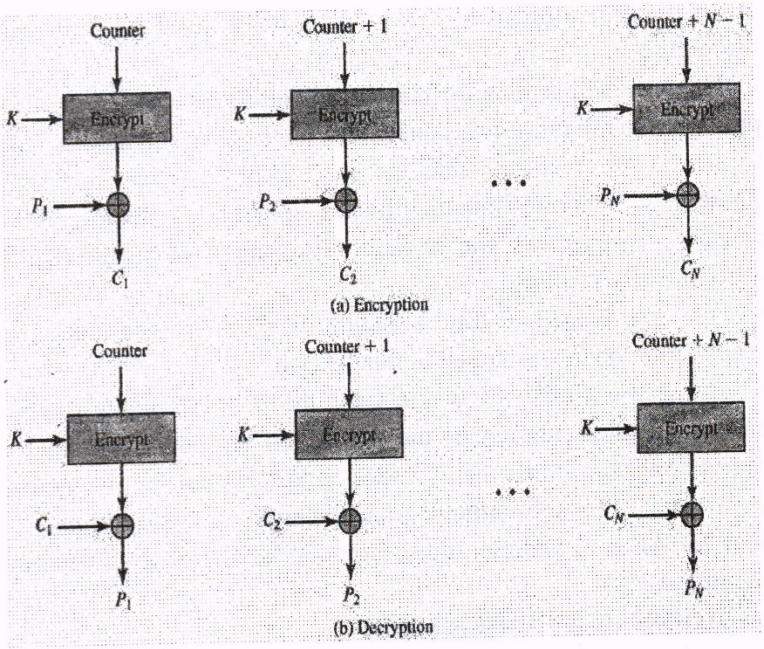


**Figure.1: Counter**

• 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.1. 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 ith 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 block 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 section.
- 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

## 3. CHOOSING A CIPHER MODE

If simplicity and speed are your main concerns, ECB is the easiest and fastest mode to use a block cipher. It is also the weakest. Besides being vulnerable to replay attacks, an algorithm in ECB mode is the easiest to cryptanalyze. For encrypting random data, such as other keys, ECB is a good mode to use. Since the data is short and random, none of the shortcomings of ECB matter for this application.

For normal plaintext, use CBC, CFB, or OFB. Which mode you choose depends on your specific requirements.

CBC is generally best for encrypting files. The increase in security is significant; and while there are sometimes bit errors in stored data, there are almost never synchronization errors.

If your application is software-based, CBC is almost always the best choice.

CFB—specifically 8-bit CFB—is generally the mode of choice for encrypting streams of characters when each character has to be treated individually, as in a link between a terminal and a host. OFB is most often used in high-speed synchronous systems where error propagation is intolerable. OFB is also the mode of choice if preprocessing is required.

OFB is the mode of choice in a error-prone environment, because it has no error extension. Stay away from the weird modes. One of the four basic modes—ECB, CBC, OFB, and CFB—is suitable for almost any application. These modes are not overly complex and probably do not reduce the security of the system. While it is possible that a complicated mode might increase the security of a system, most likely it just increases the complexity. None of the weird modes has any better error propagation or error recovery characteristics [1].

## 4. DEVELOPMENT OF MODES

In this section we introduce three developed variation methods of the Counter Mode. We exploit some of advantages and characteristics of Counter Mode. These advantages may be added into previous other modes to introduce the following developed methods.

**4.1. Counter Output Block Chaining (COBC) Mode**

Counter adds a Block Chaining mechanism to a block cipher. The results of the encryption of previous blocks are fed back into the encryption of the current block. In other words, block is used to modify the encryption of the next block. Each ciphertext block is dependent not just on the plaintext block that generated it but on all the previous plaintext blocks and Counter. The COBC mode is illustrated in Figure.2. In this scheme, the input to the encryption algorithm is the XOR of the current Counter block and the preceding output 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. The output of the encryption function is XORed with current plaintext to produce the current ciphertext. Therefore, repeating patterns of bits are not exposed.

For decryption, each cipher block is passed through the decryption algorithm. The result is XORed with the preceding output block to produce the plaintext block.

Note that it is the *encryption* function that is used, not the decryption function.

This is easily explained.

$$C_1 = P_1 \oplus E_K(Counter_i \oplus IV)$$

$$P_1 = C_1 \oplus E_K(Counter_i \oplus IV)$$

Also

$$C_i = P_i \oplus E_K(Counter_i \oplus O_{i-1})$$

$$P_i = C_i \oplus E_K(Counter_i \oplus O_{i-1})$$

Then

$C_i \oplus E_K(Counter_i \oplus O_{i-1}) = [P_i \oplus E_K(Counter_i \oplus O_{i-1})] \oplus E_K(Counter_i \oplus O_{i-1})$

$= P_i$

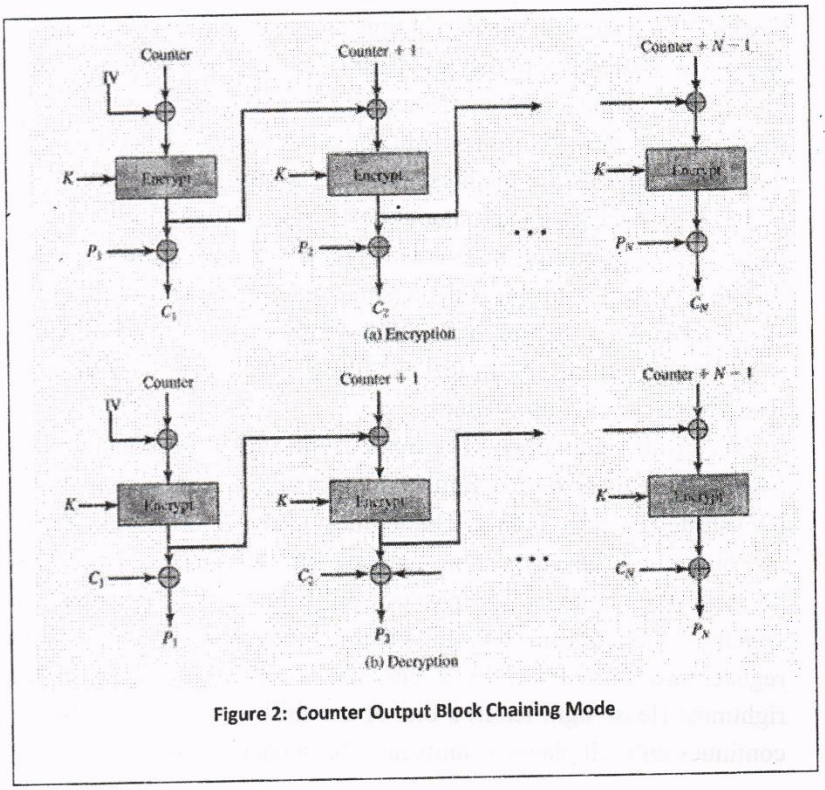


**Figure 2: Counter Output Block Chaining Mode**

### 4.2. S-Bits Counter Stream (SBCS) Mode

The Block Cipher scheme is essentially a block cipher technique that uses $s$-bit blocks. However, it is possible to convert into a stream cipher, using the S-bits Counter Stream 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 cipher text output of 8 bits. If more than 8 bits are produced, transmission capacity is wasted. Figure.3 depicts the SBCS scheme. In the figure, it is assumed that the unit of transmission is $s$ bits; a common value is $s = 8$. As with SBCS, 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) shifted by the Counter. 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 $Counter_i$ is placed in the rightmost (least significant) $s$ bits of the shift register. This process continues until all plaintext units have been encrypted.

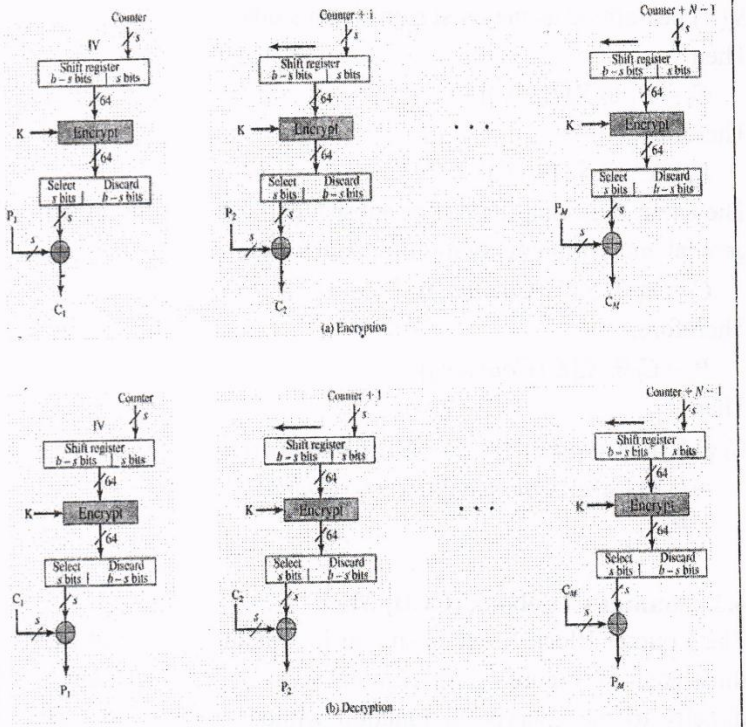


**Figure 3: *S*-Bit Counter Stream Mode**

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. This is easily explained. Let $S_s(X)$ be defined as the most significant $s$ bits of $X$.

Then

$$C_1 = P_1 \oplus S_s[E_k(\mathrm{IV})]$$

Therefore,

$$P_1 = C_1 \oplus S_s[E_k(\mathrm{IV})]$$

The same reasoning holds for subsequent steps in the process. The general encryption/decryption functions are:

$$C_i = P_i \oplus S_s[E_k(Counter_i)]$$

Therefore,

$$P_i = C_i \oplus S_s[E_k(Counter_i)]$$

Then

$$C_i \oplus S_s[E_k(Counter_i] = [P_i \oplus S_s[E_k(Counter_i)]] \oplus S_s[E_k(Counter_i]$$
$$= P_i \oplus [S_s[E_k(Counter_i)] \oplus S_s[E_k(Counter_i]]$$
$$= P_i$$

### 4.3. Counter Codebook (CCB) Mode

The Counter Block Mode is similar in structure to that of SBCS, as illustrated in Figure.4. As can be seen, it is the Counter input directly to the encryption function without shifting. 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. This process continues until all plaintext units have been encrypted.

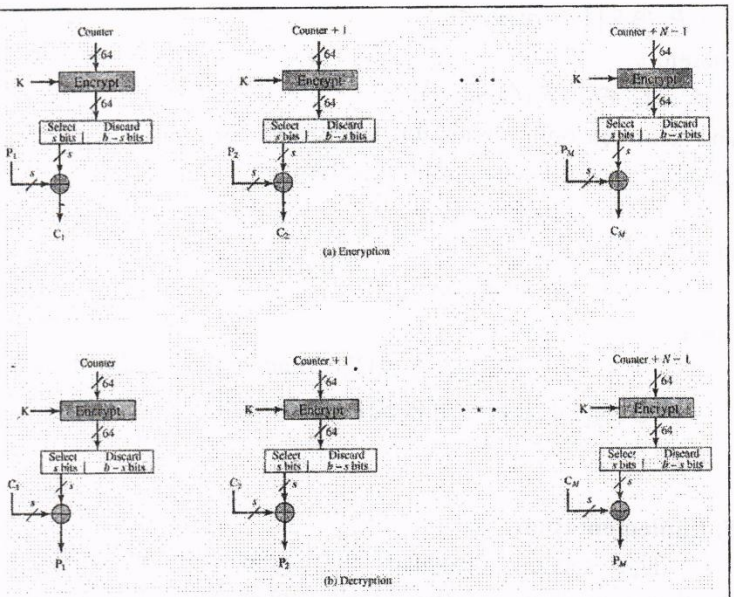


**Figure.4: Counter Codebook Block Mode**

One advantage of the CCB 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 is $P_1$ affected; subsequent plaintext units are not corrupted.

Note that it is the *encryption* function that is used, not the decryption function. This is easily explained. Let $S_s(X)$ be defined as the most significant $s$ bits of $X$.

Then

$C_1 = P_1 \oplus S_s[E_k(Counter)]$

Therefore,

$P_1 = C_1 \oplus S_s[E_k(Counter)]$

The same reasoning holds for subsequent steps in the process. The general encryption/decryption functions are:

$C_i = P_i \oplus S_s[E_k(Counter_i)]$

Therefore,

$P_i = C_i \oplus S_s[E_k(Counter_i)]$

Then

$C_i \oplus S_s[E_k(Counter_i] = [P_i \oplus S_s[E_k(Counter_i)]] \oplus S_s[E_k(Counter_i]$

$= P_i \oplus [S_s[E_k(Counter_i)] \oplus S_s[E_k(Counter_i]]$

$= P_i$

## 5. Implementation

This section introduces the implementation of the three developed methods of the Block Cipher Modes. The proposed techniques are implemented using Visual Basic V6, the plaintext can be input as a printed text or as text file, where the key is inserted as an eight bytes. The execution time is approximately identical between equivalent modes, but the developed modes may give more security due to the chaining and feedback.

## 6. Conclusions and Discussion

One of the four basic modes—ECB, CBC, OFB, and CFB—is suitable for almost any application. These modes are not overly complex and probably do not reduce the security of the system. While it is possible that a complicated mode might increase the security of a system, most likely it just increases the complexity.

**Counter Output Block Chaining Mode:**

The input to the encryption algorithm is the Counter is XORed with the preceding 64 bits of output of encryption function or IV for first step. The output the encryption algorithm is XORed of plaintext to produce the next 64 bits ciphertext. It is applied to General purpose block oriented transmission, Authentication and Random Access. This mode gives security more than CTR due to the chaining, each ciphertext is produces from current plaintext and all previous blocks.

***S*-Bit Counter Stream Mode:**

Input is processed *s*-bits at a time. Counter is used as input to the encryption algorithm to produce pseudorandom output, which is XORed with plaintext to produce next unit of ciphertext. It is applied to General purpose stream oriented transmission over noisy channel e.g., satellite communication, Authentication, Random Access and real time system.

**Counter Codebook Mode:**

Similar to SBCS, except that the input to the encryption algorithm has the block size of 64-bit encryption output. i.e. the maximum length of the counter round is longer than SBCS. It is applied to Stream-oriented transmission over noisy channel e.g., satellite communication.

Finally, all developed methods are more secure than Original Counter Mode due to the Chaining and shifting trade-off with speed. Despite, the error in each block do not affect the other blocks. However, all advantages of the CTR mode are achieved.

**References**

[1] B. Schneier, *Applied Cryptography: Protocols, Algorithms, and Source Code in C*, John Wiley & Sons, New York, 2nd edition, 1996.

[2] W. Stallings, *Cryptography and Network Security, Principle and Practice*, Addison Wesley, 1999.

[3] B. Preneel, M. Nuttin, V. Rijmen, and J. Buelens, "Cryptanalysis of the CFB mode of the DES with a Reduced Number of Rounds," Advances in Cryptology—CRYPTO '93 Proceedings, Springer–Verlag, 1994, pp. 212–223.

[4] C. M. Campbell, "Design and Specification of Cryptographic Capabilities," IEEE Computer Society Magazine, v. 16, n. 6, Nov 1978, pp. 15–19.

[5] S.T. Kent, "Encryption–Based Protection Protocols for Interactive User–Computer Communications," MIT/LCS/TR–162, MIT Laboratory for Computer Science, May 1976.

[6] W. Diffie and M.E. Hellman, "Privacy and Authentication: An Introduction to Cryptography," Proceedings of the IEEE, v. 67, n. 3, Mar 1979, pp. 397–427.

[7] T. Herlestam, "Critical Remarks on Some Public–Key Cryptosystems," BIT, v. 18, 1978, pp. 493–496.

[8] Lipmaa, H.; Rogaway, P.; and Wagner, D. "CTR Mode Encryption." NIST First Modes of Operation Workshop, October 2000. http://csrc.nist.gov/encryption/modes

[9] Voydock, V., and Kent., S. "Security Mechanisms in High-Level Network Protocols." Computing Surveys, June 1983.

[10] M. Stamp, *Information Security Principles and practice*, JohnWiley & Sons, Inc., 2006.