Block cipher mode 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.[1] 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.[2] 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.[3][4][5]

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.[6] 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.[2] There are, however, modes that do not require padding because they effectively use a block cipher as a stream cipher.

Historically, encryption modes have been studied extensively in regard to their error propagation properties under various scenarios of data modification. Later development regarded integrity protection as an entirely separate cryptographic goal. Some modern modes of operation combine confidentiality and authenticity in an efficient way, and are known as authenticated encryption modes.[7]

History and standardization

The earliest modes of operation, ECB, CBC, OFB, and CFB (see below for all), date back to 1981 and were specified in FIPS 81 (http://csrc.nist.gov/publications/fips/fips81/fips81.htm), *DES Modes of Operation*. In 2001, the US National Institute of Standards and Technology (NIST) revised its list of approved modes of operation by including AES as a block cipher and adding CTR mode in SP800-38A (http://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-38a.pdf), *Recommendation for Block Cipher Modes of Operation*. Finally, in January, 2010, NIST added XTS-AES in SP800-38E (http://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-38e.pdf), *Recommendation for Block Cipher Modes of Operation: The XTS-AES Mode for Confidentiality on Storage Devices*. Other confidentiality modes exist which have not been approved by NIST. For example, CTS is ciphertext stealing mode and available in many popular cryptographic libraries.

The block cipher modes ECB, CBC, OFB, CFB, CTR, and XTS provide confidentiality, but they do not protect against accidental modification or malicious tampering. Modification or tampering can be detected with a separate message authentication code such as CBC-MAC, or a digital signature. The cryptographic community recognized the need for dedicated integrity assurances and NIST responded with HMAC, CMAC, and GMAC. HMAC was approved in 2002 as FIPS 198 (http://csrc.nist.gov/publications/fips/fips198/fips-198a.pdf), *The Keyed-Hash Message Authentication Code (HMAC)*, CMAC was released in 2005 under SP800-38B (http://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-38b.pdf), *Recommendation for Block Cipher Modes of Operation: The CMAC Mode for Authentication*, and GMAC was formalized in 2007 under SP800-38D (http://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-38d.pdf), *Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC*.

The cryptographic community observed that compositing (combining) a confidentiality mode with an authenticity mode could be difficult and error prone. They therefore began to supply modes which combined confidentiality and data integrity into a single cryptographic primitive (an encryption algorithm). These combined modes are referred to as authenticated encryption, AE or "authenc". Examples of AE modes are CCM (SP800-38C (http://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-38c.pdf)), GCM (SP800-38D (http://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-38d.pdf)), CWC, EAX, IAPM, and OCB.

Modes of operation are defined by a number of national and internationally recognized standards bodies. Notable standards organizations include NIST, ISO (with ISO/IEC 10116[5]), the IEC, the IEEE, ANSI, and the IETF.

Initialization vector (IV)

An initialization vector (IV) or starting variable (SV)[5] is a block of bits that is used by several modes to randomize the encryption and hence to produce distinct ciphertexts even if the same plaintext is encrypted multiple times, without the need for a slower re-keying process.

An initialization vector has different security requirements than a key, so the IV usually does not need to be secret. For most block cipher modes it is important that an initialization vector is never reused under the same key, i.e. it must be a cryptographic nonce. Many block cipher modes have stronger requirements, such as the IV must be random or pseudorandom. Some block ciphers have particular problems with certain initialization vectors, such as all zero IV generating no encryption (for some keys).

It is recommended to review relevant IV requirements for the particular block cipher mode in relevant specification, for example SP800-38A (http://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-38a.pdf).

For CBC and CFB, reusing an IV leaks some information about the first block of plaintext, and about any common prefix shared by the two messages.

For OFB and CTR, reusing an IV causes key bitstream re-use, which breaks security.[8] This can be seen because both modes effectively create a bitstream that is XORed with the plaintext, and this bitstream is dependent on the key and IV only.

In CBC mode, the IV must be unpredictable (random or pseudorandom) at encryption time; in particular, the (previously) common practice of re-using the last ciphertext block of a message as the IV for the next message is insecure (for example, this method was used by SSL 2.0). If an attacker knows the IV (or the previous block of ciphertext) before the next plaintext is specified, they can check their guess about plaintext of some block that was encrypted with the same key before (this is known as the TLS CBC IV attack).[9]

For some keys, an all-zero initialization vector may generate some block cipher modes (CFB-8, OFB-8) to get the internal state stuck at all-zero. For CFB-8, an all-zero IV and an all-zero plaintext, causes 1/256 of keys to generate no encryption, plaintext is returned as ciphertext.[10] For OFB-8, using all zero initialization vector will generate no encryption for 1/256 of keys.[11] OFB-8 encryption returns the plaintext unencrypted for affected keys.

Some modes (such as AES-SIV and AES-GCM-SIV) are built to be more nonce-misuse resistant, i.e. resilient to scenarios in which the randomness generation is faulty or under the control of the attacker.

- Synthetic initialization vectors (SIV) synthesize an internal IV by running a pseudo-random function (PRF) construction called S2V on the input (additional data and plaintext), preventing any external data from directly controlling the IV. External nonces / IV may be fed into S2V as an additional data field.
- AES-GCM-SIVs synthesize an internal IV by running POLYVAL Galois mode of authentication on input (additional data and plaintext), followed by an AES operation.

Padding

A block cipher works on units of a fixed size (known as a *block size*), but messages come in a variety of lengths. So some modes (namely ECB and CBC) require that the final block be padded before encryption. Several padding schemes exist. The simplest is to add null bytes to the plaintext to bring its length up to a multiple of the block size, but care must be taken that the original length of the plaintext can be recovered; this is trivial, for example, if the plaintext is a C style string which contains no null bytes except at the end. Slightly more complex is the original DES method, which is to add a single one bit, followed by enough zero bits to fill out the block; if the message ends on a block boundary, a whole padding block will be added. Most sophisticated are CBC-specific schemes such as ciphertext stealing or residual block termination, which do not cause any extra ciphertext, at the expense of some additional complexity. Schneier and Ferguson suggest two possibilities, both simple: append a byte with value 128 (hex 80), followed by as many zero bytes as needed to fill the last block, or pad the last block with *n* bytes all with value *n*.

CFB, OFB and CTR modes do not require any special measures to handle messages whose lengths are not multiples of the block size, since the modes work by XORing the plaintext with the output of the block cipher. The last partial block of plaintext is XORed with the first few bytes of the last keystream block, producing a final ciphertext block that is the same size as the final partial plaintext block. This characteristic of stream ciphers makes them suitable for applications that require the encrypted ciphertext data to be the same size as the original plaintext data, and for applications that transmit data in streaming form where it is inconvenient to add padding bytes.

Common modes

Authenticated encryption with additional data (AEAD) modes

A number of modes of operation have been designed to combine secrecy and authentication in a single cryptographic primitive. Examples of such modes are ,[12] integrity-aware cipher block chaining (IACBC), integrity-aware parallelizable mode (IAPM),[13] OCB, EAX, CWC, CCM, and GCM. Authenticated encryption modes are classified as single-pass modes or double-pass modes. Some single-pass authenticated encryption algorithms, such as OCB mode, are encumbered by patents, while others were specifically designed and released in a way to avoid such encumberment.

In addition, some modes also allow for the authentication of unencrypted associated data, and these are called AEAD (authenticated encryption with associated data) schemes. For example, EAX mode is a double-pass AEAD scheme while OCB mode is single-pass.

Galois/counter (GCM)

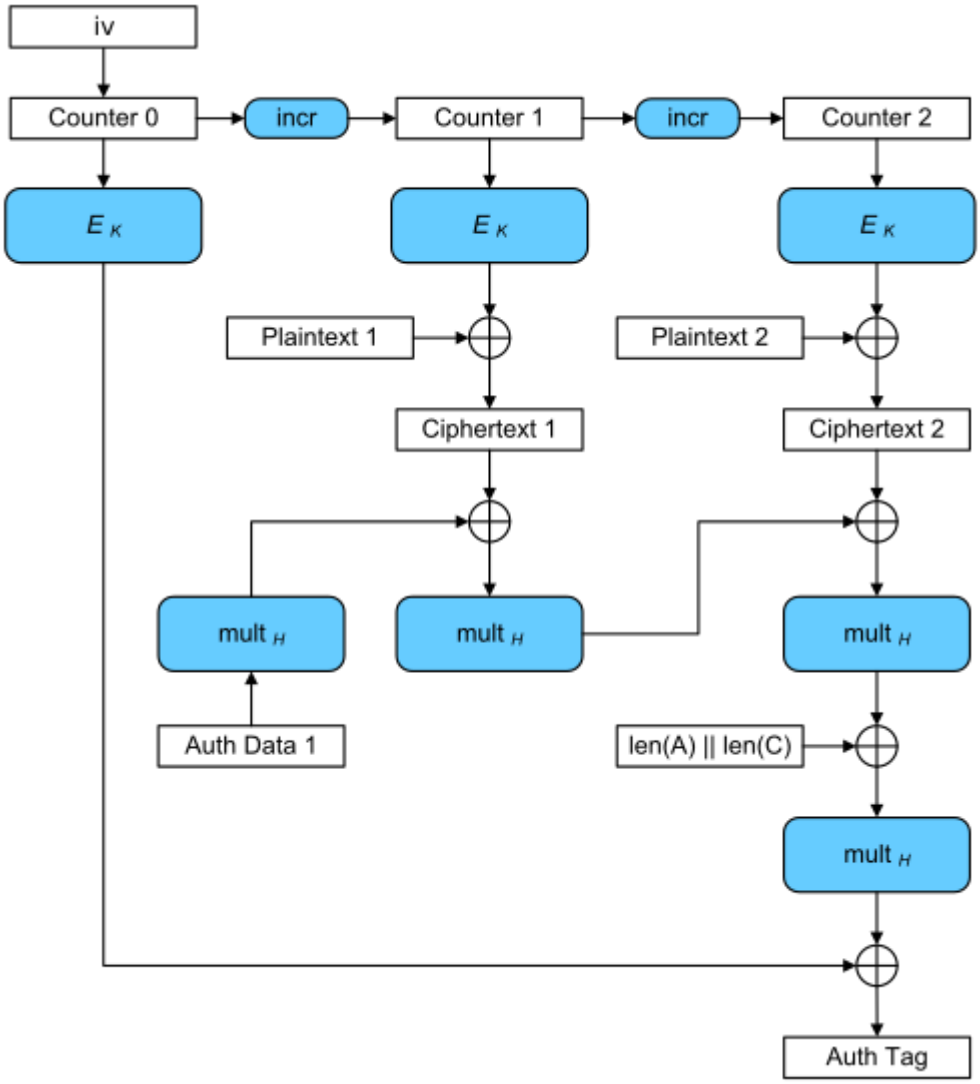
GCM
Galois/counter

Galois/counter mode (GCM) combines the well-known counter mode of encryption with the new Galois mode of authentication. The key feature is the ease of parallel computation of the Galois field multiplication used for authentication. This feature permits higher throughput than encryption algorithms.

Encryption parallelizable	Yes
Decryption parallelizable	Yes
Random read access	Yes

GCM is defined for block ciphers with a block size of 128 bits. Galois message authentication code (GMAC) is an authentication-only variant of the GCM which can form an incremental message authentication code. Both GCM and GMAC can accept initialization vectors of arbitrary length. GCM can take full advantage of parallel processing and implementing GCM can make efficient use of an instruction pipeline or a hardware pipeline. The CBC mode of operation incurs pipeline stalls that hamper its efficiency and performance.

Like in CTR, blocks are numbered sequentially, and then this block number is combined with an IV and encrypted with a block cipher E, usually AES. The result of this encryption is then XORed with the plaintext to produce the ciphertext. Like all counter modes, this is essentially a stream cipher, and so it is essential that a different IV is used for each stream that is encrypted.



The ciphertext blocks are considered coefficients of a polynomial which is then evaluated at a key-dependent point H, using finite field arithmetic. The result is then encrypted, producing an authentication tag that can be used to verify the integrity of the data. The encrypted text then contains the IV, ciphertext, and authentication tag.

Counter with cipher block chaining message authentication code (CCM)

Counter with cipher block chaining message authentication code (counter with CBC-MAC; CCM) is an authenticated encryption algorithm designed to provide both authentication and confidentiality. CCM mode is only defined for block ciphers with a block length of 128 bits.[14][15]

Synthetic initialization vector (SIV)

Synthetic initialization vector (SIV) is a nonce-misuse resistant block cipher mode.

SIV synthesizes an internal IV using the pseudorandom function S2V. S2V is a keyed hash is based on CMAC, and the input to the function is:

- Additional authenticated data (zero, one or many AAD fields are supported)
- Plaintext
- Authentication key (K_1).

SIV encrypts the S2V output and the plaintext using AES-CTR, keyed with the encryption key (K_2).

SIV can support external nonce-based authenticated encryption, in which case one of the authenticated data fields is utilized for this purpose. RFC5297[16] specifies that for interoperability purposes the last authenticated data field should be used external nonce.

Owing to the use of two keys, the authentication key K_1 and encryption key K_2, naming schemes for SIV AEAD-variants may lead to some confusion; for example AEAD_AES_SIV_CMAC_256 refers to AES-SIV with two AES-128 keys and **not** AES-256.

AES-GCM-SIV

AES-GCM-SIV is a mode of operation for the Advanced Encryption Standard which provides similar performance to Galois/counter mode as well as misuse resistance in the event of the reuse of a cryptographic nonce. The construction is defined in RFC 8452.[17]

AES-GCM-SIV synthesizes the internal IV. It derives a hash of the additional authenticated data and plaintext using the POLYVAL Galois hash function. The hash is then encrypted an AES-key, and used as authentication tag and AES-CTR initialization vector.

AES-GCM-SIV is an improvement over the very similarly named algorithm **GCM-SIV**, with a few very small changes (e.g. how AES-CTR is initialized), but which yields practical benefits to its security "This addition allows for encrypting up to 2^{50} messages with the same key, compared to the significant limitation of only 2^{32} messages that were allowed with GCM-SIV."[18]

Confidentiality only modes

Many modes of operation have been defined. Some of these are described below. The purpose of cipher modes is to mask patterns which exist in encrypted data, as illustrated in the description of the weakness of ECB.

Different cipher modes mask patterns by cascading outputs from the cipher block or other globally deterministic variables into the subsequent cipher block. The inputs of the listed modes are summarized in the following table:

Summary of modes

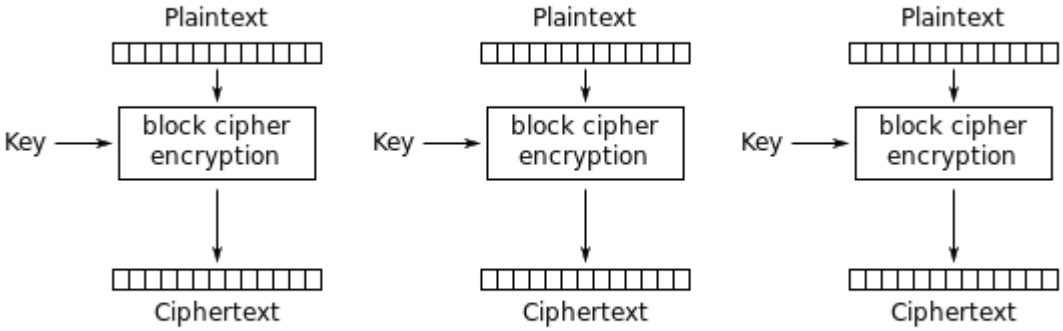
Mode		Formulas	Ciphertext
Electronic codebook	(ECB)	$Y_i = F(\text{PlainText}_i, \text{Key})$	Y_i
Cipher block chaining	(CBC)	$Y_i = \text{PlainText}_i \text{ XOR } \text{Ciphertext}_{i-1}$	$F(Y, \text{Key})$; $\text{Ciphertext}_0 = \text{IV}$
Propagating CBC	(PCBC)	$Y_i = \text{PlainText}_i \text{ XOR } (\text{Ciphertext}_{i-1} \text{ XOR } \text{PlainText}_{i-1})$	$F(Y, \text{Key})$; $\text{Ciphertext}_0 = \text{IV}$
Cipher feedback	(CFB)	$Y_i = \text{Ciphertext}_{i-1}$	Plaintext XOR $F(Y, \text{Key})$; $\text{Ciphertext}_0 = \text{IV}$
Output feedback	(OFB)	$Y_i = F(Y_{i-1}, \text{Key})$; $Y_0 = F(\text{IV}, \text{Key})$	Plaintext XOR Y_i
Counter	(CTR)	$Y_i = F(\text{IV} + g(i), \text{Key})$; IV = token()	Plaintext XOR Y_i

Note: $g(i)$ is any deterministic function, often the identity function.

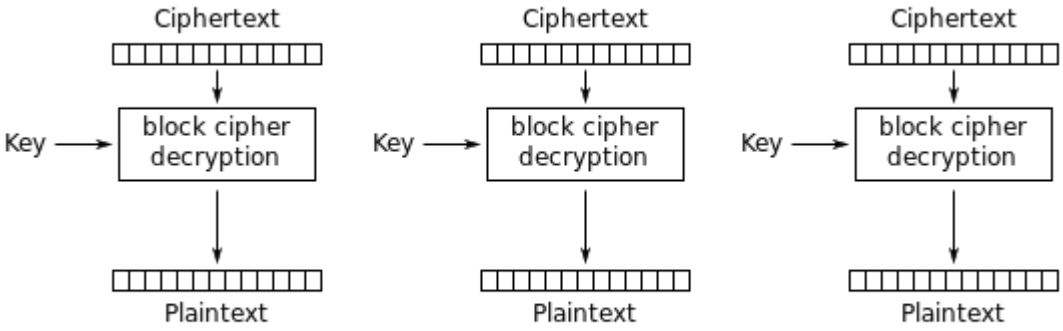
Electronic codebook (ECB)

The simplest (and not to be used anymore) of the encryption modes is the **electronic codebook** (ECB) mode (named after conventional physical codebooks[19]). The message is divided into blocks, and each block is encrypted separately.

ECB	
Electronic codebook	
Encryption parallelizable	Yes
Decryption parallelizable	Yes
Random read access	Yes



Electronic Codebook (ECB) mode encryption

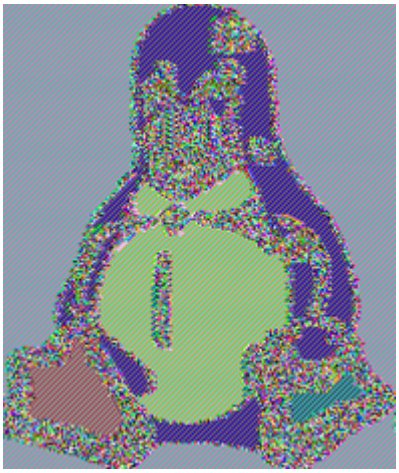


Electronic Codebook (ECB) mode decryption

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.[20][21][22]

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.

Original image



Using ECB allows patterns to be easily discerned

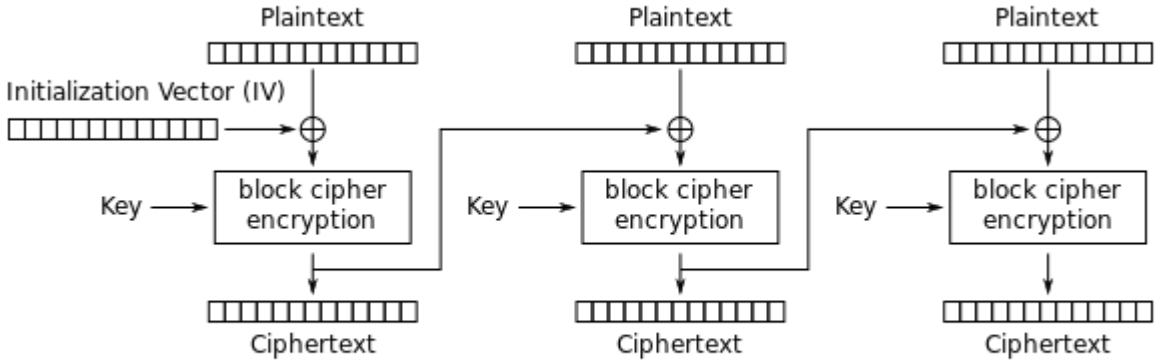
Modes other than ECB result in pseudo-randomness

ECB mode can also make protocols without integrity protection even more susceptible to replay attacks, since each block gets decrypted in exactly the same way.

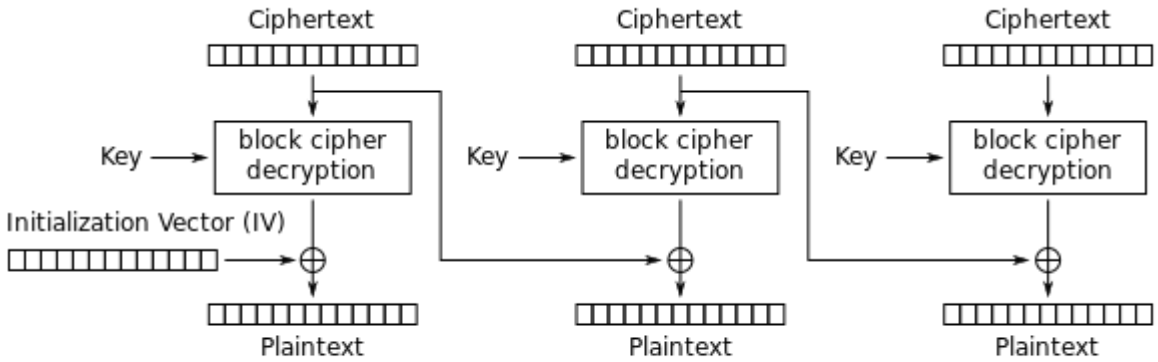
Cipher block chaining (CBC)

CBC	
Cipher block chaining	
Encryption parallelizable	No
Decryption parallelizable	Yes
Random read access	Yes

Ehrsam, Meyer, Smith and Tuchman invented the cipher block chaining (CBC) mode of operation in 1976.[23] 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.



Cipher Block Chaining (CBC) mode encryption



Cipher Block Chaining (CBC) mode decryption

If the first block has index 1, the mathematical formula for CBC encryption is

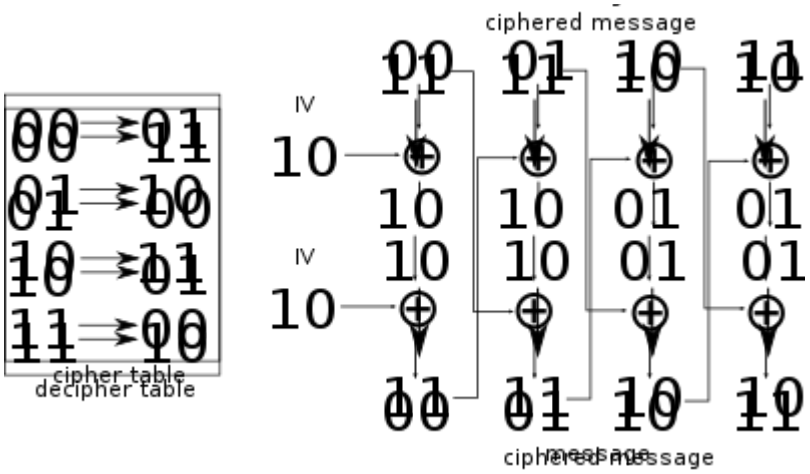
$$C_i = E_K(P_i \oplus C_{i-1}),$$
$$C_0 = IV,$$

while the mathematical formula for CBC decryption is

$$P_i = D_K(C_i) \oplus C_{i-1},$$
$$C_0 = IV.$$

Example

message



CBC has been the most commonly used mode of operation. Its main drawbacks are that encryption is sequential (i.e., it cannot be parallelized), and that the message must be padded to a multiple of the cipher block size. One way to handle this last issue is through the method known as ciphertext stealing. Note that a one-bit change in a plaintext or initialization vector (IV) affects all following ciphertext blocks.

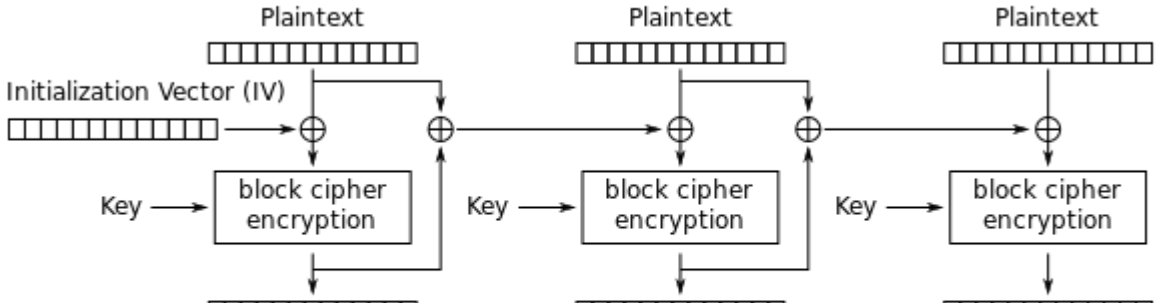
Decrypting with the incorrect IV causes the first block of plaintext to be corrupt but subsequent plaintext blocks will be correct. This is because each block is XORed with the ciphertext of the previous block, not the plaintext, so one does not need to decrypt the previous block before using it as the 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. This peculiarity is exploited in different padding oracle attacks, such as POODLE.

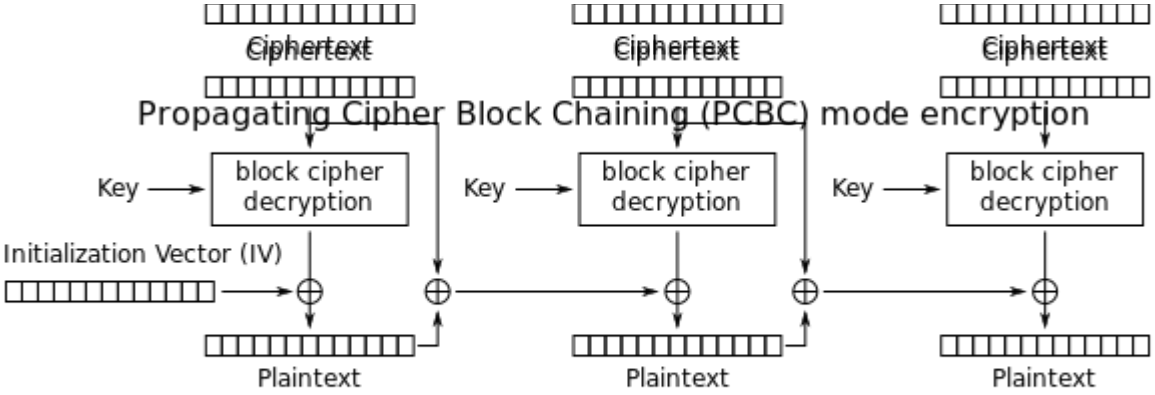
Explicit initialization vectors[24] takes advantage of this property by prepending a single random block to the plaintext. Encryption is done as normal, except the IV does not need to be communicated to the decryption routine. Whatever IV decryption uses, only the random block is "corrupted". It can be safely discarded and the rest of the decryption is the original plaintext.

Propagating cipher block chaining (PCBC)

PCBC	
Propagating cipher block chaining	
Encryption parallelizable	No
Decryption parallelizable	No
Random read access	No

The *propagating cipher block chaining*[25] or *plaintext cipher-block chaining*[26] mode was designed to cause small changes in the ciphertext to propagate indefinitely when decrypting, as well as when encrypting. In PCBC mode, each block of plaintext is XORed with both the previous plaintext block and the previous ciphertext block before being encrypted. Like with CBC mode, an initialization vector is used in the first block. Unlike CBC, decrypting PCBC with the incorrect IV (initialization vector) causes all blocks of plaintext to be corrupt.





Propagating Cipher Block Chaining (PCBC) mode decryption

Encryption and decryption algorithms are as follows:

$$C_i = E_K(P_i \oplus P_{i-1} \oplus C_{i-1}), P_0 \oplus C_0 = IV,$$
$$P_i = D_K(C_i) \oplus P_{i-1} \oplus C_{i-1}, P_0 \oplus C_0 = IV.$$

PCBC is used in Kerberos v4 and WASTE, most notably, but otherwise is not common. On a message encrypted in PCBC mode, if two adjacent ciphertext blocks are exchanged, this does not affect the decryption of subsequent blocks.[27] For this reason, PCBC is not used in Kerberos v5.

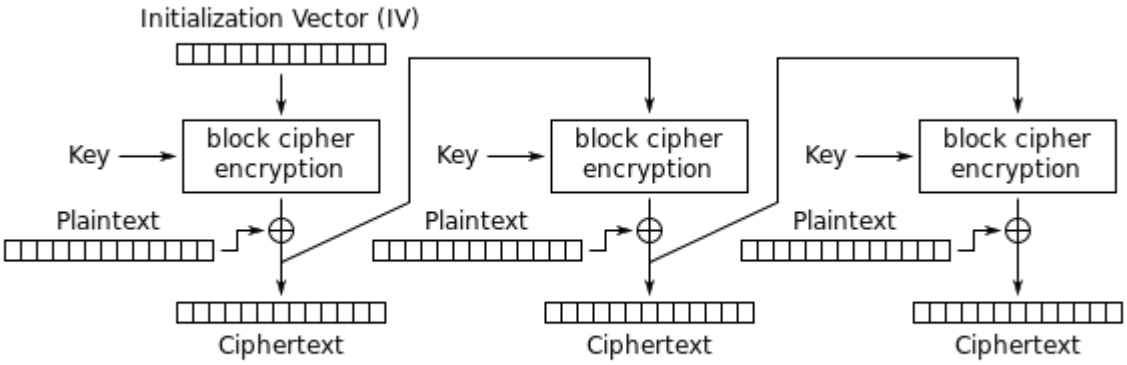
Cipher feedback (CFB)

Full-block CFB

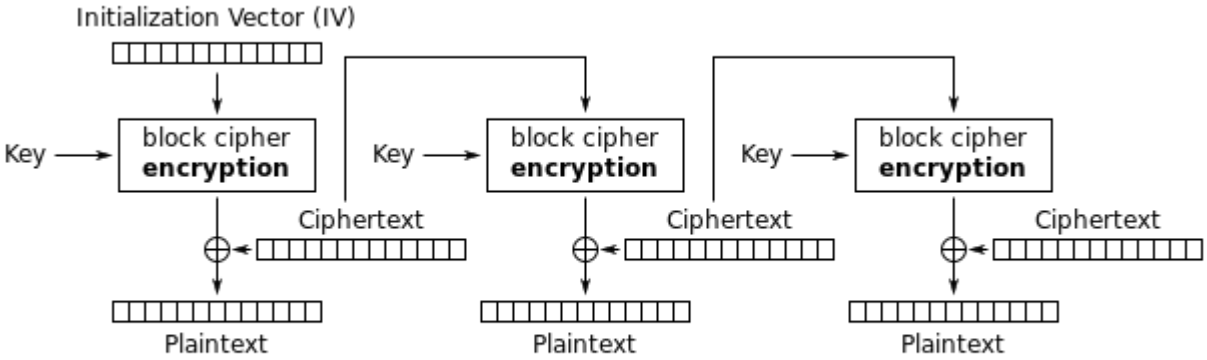
CFB	
Cipher feedback	
Encryption parallelizable	No
Decryption parallelizable	Yes
Random read access	Yes

The *cipher feedback* (CFB) mode, in its simplest form uses the entire output of the block cipher. In this variation, it is very similar to CBC, makes a block cipher into a self-synchronizing stream cipher. CFB decryption in this variation is almost identical to CBC encryption performed in reverse:

$$C_i = \begin{cases} \mathrm{IV}, & i = 0 \\ E_K(C_{i-1}) \oplus P_i, & \text{otherwise} \end{cases}$$
$$P_i = E_K(C_{i-1}) \oplus C_i,$$



Cipher Feedback (CFB) mode encryption



Cipher Feedback (CFB) mode decryption

CFB-1, CFB-8, CFB-64, CFB-128, etc.

NIST SP800-38A defines CFB with a bit-width.[28] *The CFB mode also requires an integer parameter, denoted s, such that 1 ≤ s ≤ b. In the specification of the CFB mode below, each plaintext segment (Pj) and ciphertext segment (Cj) consists of s bits. The value of s is sometimes incorporated into the name of the mode, e.g., the 1-bit CFB mode, the 8-bit CFB mode, the 64-bit CFB mode, or the 128-bit CFB mode.*

These modes will truncate the output of the underlying block cipher.

$$
\begin{aligned}
&I_0 = \mathrm{IV}.\\
&I_i = \left((I_{i-1} \ll s) + C_i\right) \bmod 2^b,\\
&C_i = \mathrm{MSB}_s\left(E_K(I_{i-1})\right) \oplus P_i,\\
&P_i = \mathrm{MSB}_s\left(E_K(I_{i-1})\right) \oplus C_i,
\end{aligned}
$$

CFB-1 is considered self synchronizing and resilient to loss of ciphertext; "When the 1-bit CFB mode is used, then the synchronization is automatically restored b+1 positions after the inserted or deleted bit. For other values of s in the CFB mode, and for the other confidentiality modes in this recommendation, the synchronization must be restored externally." (NIST SP800-38A). I.e. 1-bit loss in a 128-bit-wide block cipher like AES will render 129 invalid bits before emitting valid bits.

CFB may also self synchronize in some special cases other than those specified. For example, a one bit change in CFB-128 with an underlying 128 bit block cipher, will re-synchronize after two blocks. (However, CFB-128 etc. will not handle bit loss gracefully; a one-bit loss will cause the decryptor to lose alignment with the encryptor)

CFB compared to other modes

Like CBC mode, changes in the plaintext propagate forever in the ciphertext, and encryption cannot be parallelized. Also like CBC, decryption can be parallelized.

CFB, OFB and CTR share two advantages over CBC mode: the block cipher is only ever used in the encrypting direction, and the message does not need to be padded to a multiple of the cipher block size (though ciphertext stealing can also be used for CBC mode to make padding unnecessary).

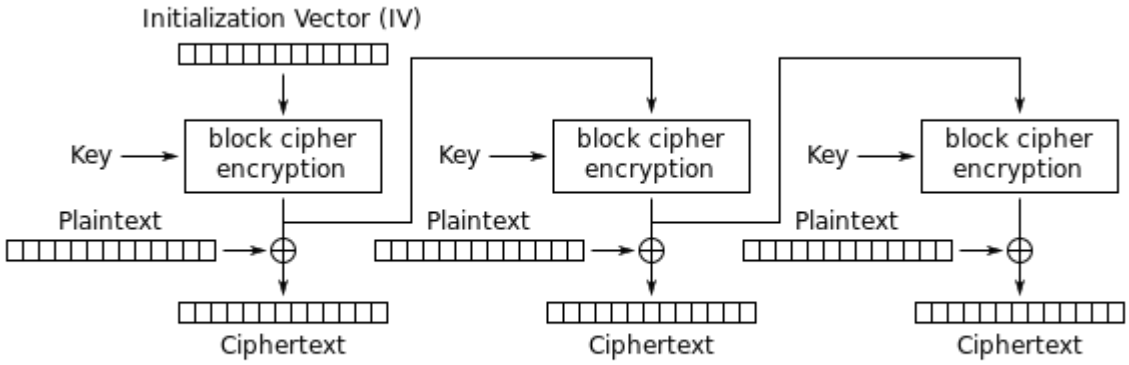
Output feedback (OFB)

The *output feedback* (OFB) mode makes a block cipher into a synchronous stream cipher. It generates keystream blocks, which are then XORed with the plaintext blocks to get the ciphertext. Just as with other stream ciphers, flipping a bit in the ciphertext produces a flipped bit in the plaintext at the same location. This property allows many error-correcting codes to function normally even when applied before encryption.

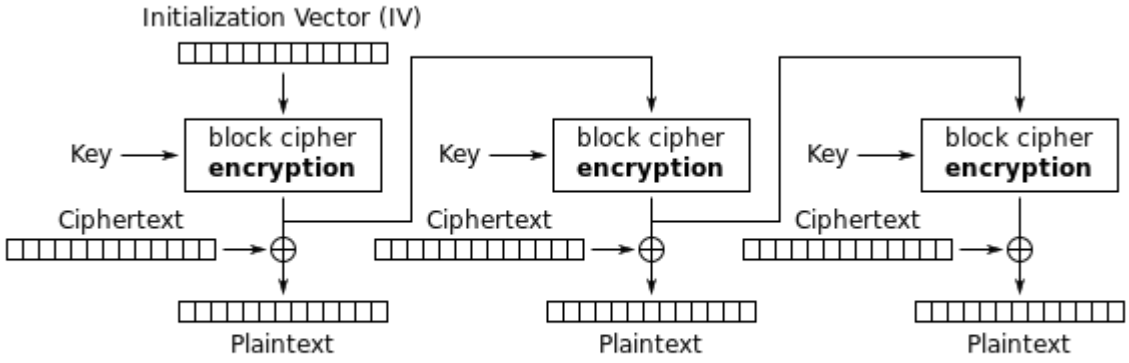
OFB	
Output feedback	
Encryption parallelizable	No
Decryption parallelizable	No
Random read access	No

Because of the symmetry of the XOR operation, encryption and decryption are exactly the same:

$$
\begin{aligned}
C_j &= P_j \oplus O_j, \\
P_j &= C_j \oplus O_j, \\
O_j &= E_K(I_j), \\
I_j &= O_{j-1}, \\
I_0 &= \mathrm{IV}.
\end{aligned}
$$



Output Feedback (OFB) mode encryption



Output Feedback (OFB) mode decryption

Each output feedback block cipher operation depends on all previous ones, and so cannot be performed in parallel. However, because the plaintext or ciphertext is only used for the final XOR, the block cipher operations may be performed in advance, allowing the final step to be performed in parallel once the plaintext or ciphertext is available.

It is possible to obtain an OFB mode keystream by using CBC mode with a constant string of zeroes as input. This can be useful, because it allows the usage of fast hardware implementations of CBC mode for OFB mode encryption.

Using OFB mode with a partial block as feedback like CFB mode reduces the average cycle length by a factor of 2^{32} or more. A mathematical model proposed by Davies and Parkin and substantiated by experimental results showed that only with full feedback an average cycle length near to the obtainable maximum can be achieved. For this reason, support for truncated feedback was removed from the specification of OFB.[29]

Counter (CTR)

> Note: CTR mode (CM) is also known as *integer counter mode* (ICM) and *segmented integer counter* (SIC) mode.

CTR	
Counter	
Encryption parallelizable	Yes
Decryption parallelizable	Yes
Random read access	Yes

Like OFB, counter mode turns a block cipher into a stream cipher. It generates the next keystream block by encrypting successive values of a "counter". The counter can be any function which produces a sequence which is guaranteed not to repeat for a long time, although an actual increment-by-one counter is the simplest and most popular. The usage of a simple

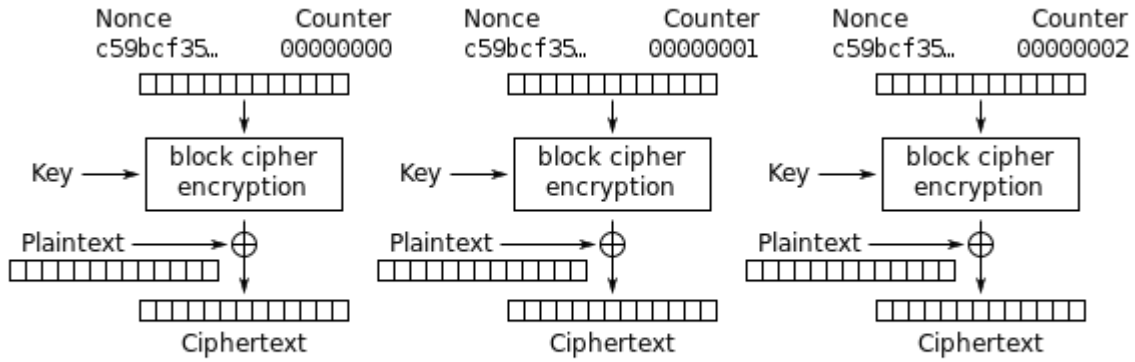
deterministic input function used to be controversial; critics argued that "deliberately exposing a cryptosystem to a known systematic input represents an unnecessary risk".[30] However, today CTR mode is widely accepted, and any problems are considered a weakness of the underlying block cipher, which is expected to be secure regardless of systemic bias in its input.[31] Along with CBC, CTR mode is one of two block cipher modes recommended by Niels Ferguson and Bruce Schneier.[32]

CTR mode was introduced by Whitfield Diffie and Martin Hellman in 1979.[31]

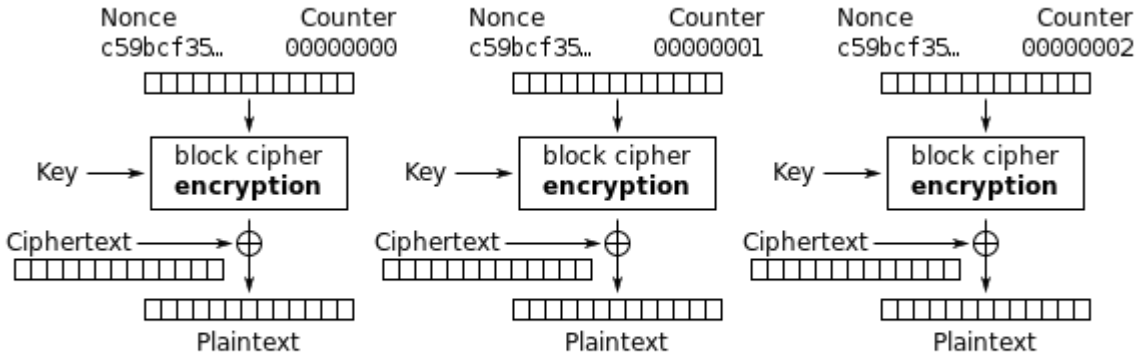
CTR mode has similar characteristics to OFB, but also allows a random-access property during decryption. CTR mode is well suited to operate on a multi-processor machine, where blocks can be encrypted in parallel. Furthermore, it does not suffer from the short-cycle problem that can affect OFB.[33]

If the IV/nonce is random, then they can be combined with the counter using any invertible operation (concatenation, addition, or XOR) to produce the actual unique counter block for encryption. In case of a non-random nonce (such as a packet counter), the nonce and counter should be concatenated (e.g., storing the nonce in the upper 64 bits and the counter in the lower 64 bits of a 128-bit counter block). Simply adding or XORing the nonce and counter into a single value would break the security under a chosen-plaintext attack in many cases, since the attacker may be able to manipulate the entire IV–counter pair to cause a collision. Once an attacker controls the IV–counter pair and plaintext, XOR of the ciphertext with the known plaintext would yield a value that, when XORed with the ciphertext of the other block sharing the same IV–counter pair, would decrypt that block.[34]

Note that the nonce in this diagram is equivalent to the initialization vector (IV) in the other diagrams. However, if the offset/location information is corrupt, it will be impossible to partially recover such data due to the dependence on byte offset.



Counter (CTR) mode encryption



Counter (CTR) mode decryption

Error propagation

"Error propagation" properties describe how a decryption behaves during bit errors, i.e. how error in one bit cascades to different decrypted bits.

Bit errors may occur intentionally in attacks or randomly due to transmission errors.

- Random bit errors occur independently in any bit position with an expected probability of ½.
- Specific bit errors occur in the same bit position(s) as the original bit error(s).
- Specific bit errors in stream cipher modes (OFB, CTR, etc.) are trivial. They affect only the specific bit intended.
- Specific bit errors in more complex modes such (e.g. CBC): adaptive chosen-ciphertext attack may intelligently combine many different specific bit errors to break the cipher mode. In Padding oracle attack, CBC can be decrypted in the attack by guessing encryption secrets based on error responses. The Padding Oracle attack variant "CBC-R" (CBC Reverse) lets the attacker construct any valid message.

For modern authenticated encryption (AEAD) or protocols with message authentication codes chained in MAC-Then-Encrypt order, any bit error should completely abort decryption and must not generate any specific bit errors to decryptor. I.e. if decryption succeeded, there should not be any bit error. As such error propagation is less important subject in modern cipher modes than in traditional confidentiality-only modes.

Mode	Effect of bit errors in C_i	Effect of bit errors in the IV or nonce
ECB	Random bit errors in P_i	—
CBC	Random bit errors in P_i Specific bit errors in P_{i+1}	Specific bit errors in P_1
CFB	Specific bit errors in P_i Random bit errors in P_{i+1}, …, until synchronization is restored	Random bit errors in P_1, …, until synchronization is restored
OFB	Specific bit errors in P_i	Random bit errors in P_1, P_2, …, P_n
CTR	Specific bit errors in P_i	Random bit errors in P_i for bit error in counter block T_i

(Source: SP800-38A Table D.2: Summary of Effect of Bit Errors on Decryption)

It might be observed, for example, that a one-block error in the transmitted ciphertext would result in a one-block error in the reconstructed plaintext for ECB mode encryption, while in CBC mode such an error would affect two blocks. Some felt that such resilience was desirable in the face of random errors (e.g., line noise), while others argued that error correcting increased the scope for attackers to maliciously tamper with a message.

However, when proper integrity protection is used, such an error will result (with high probability) in the entire message being rejected. If resistance to random error is desirable, error-correcting codes should be applied to the ciphertext before transmission.

Other modes and other cryptographic primitives

Many more modes of operation for block ciphers have been suggested. Some have been accepted, fully described (even standardized), and are in use. Others have been found insecure, and should never be used. Still others don't categorize as confidentiality, authenticity, or authenticated encryption – for example key feedback mode and Davies–Meyer hashing.

NIST maintains a list of proposed modes for block ciphers at *Modes Development*.[28][35]

Disk encryption often uses special purpose modes specifically designed for the application. Tweakable narrow-block encryption modes (LRW, XEX, and XTS) and wide-block encryption modes (CMC and EME) are designed to securely encrypt sectors of a disk (see disk encryption theory).

Many modes use an initialization vector (IV) which, depending on the mode, may have requirements such as being only used once (a nonce) or being unpredictable ahead of its publication, etc. Reusing an IV with the same key in CTR, GCM or OFB mode results in XORing the same keystream with two or more plaintexts, a clear misuse of a stream, with a catastrophic loss of security. Deterministic authenticated encryption modes such as the NIST Key Wrap algorithm and the SIV (RFC 5297) AEAD mode do not require an IV as an input, and return the same ciphertext and authentication tag every time for a given plaintext and key. Other IV misuse-resistant modes such as AES-GCM-SIV (https://cyber.biu.ac.il/aes-gcm-siv/) benefit from an IV input, for example in the maximum amount of data that can be safely encrypted with one key, while not failing catastrophically if the same IV is used multiple times.

Block ciphers can also be used in other cryptographic protocols. They are generally used in modes of operation similar to the block modes described here. As with all protocols, to be cryptographically secure, care must be taken to design these modes of operation correctly.

There are several schemes which use a block cipher to build a cryptographic hash function. See one-way compression function for descriptions of several such methods.

Cryptographically secure pseudorandom number generators (CSPRNGs) can also be built using block ciphers.

Message authentication codes (MACs) are often built from block ciphers. CBC-MAC, OMAC and PMAC are examples.

See also

- Disk encryption
- Message authentication code
- Authenticated encryption
- One-way compression function

References

1. NIST Computer Security Division's (CSD) Security Technology Group (STG) (2013). "Block cipher modes" (http://csrc.nist.gov/groups/ST/toolkit/BCM/index.html). *Cryptographic Toolkit*. NIST. Archived (https://web.archive.org/web/20121106212417/http://csrc.nist.gov/groups/ST/toolkit/BCM/index.html) from the original on November 6, 2012. Retrieved April 12, 2013.
2. Ferguson, N.; Schneier, B.; Kohno, T. (2010). *Cryptography Engineering: Design Principles and Practical Applications*. Indianapolis: Wiley Publishing, Inc. pp. 63, 64. ISBN 978-0-470-47424-2.
3. NIST Computer Security Division's (CSD) Security Technology Group (STG) (2013). "Proposed modes" (http://csrc.nist.gov/groups/ST/toolkit/BCM/modes_development.html). *Cryptographic Toolkit*. NIST. Archived (https://web.archive.org/web/20130402203904/http://csrc.nist.gov/groups/ST/toolkit/BCM/modes_development.html) from the original on April 2, 2013. Retrieved April 14, 2013.
4. Alfred J. Menezes; Paul C. van Oorschot; Scott A. Vanstone (1996). *Handbook of Applied Cryptography* (https://archive.org/details/handbookofapplie0000mene/page/228). CRC Press. pp. 228–233 (https://archive.org/details/handbookofapplie0000mene/page/228). ISBN 0-8493-8523-7.
5. "ISO/IEC 10116:2006 – Information technology – Security techniques – Modes of operation for an *n*-bit block cipher" (http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=38761). *ISO Standards Catalogue*. 2006. Archived (https://web.archive.org/web/20120317153312/http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=38761) from the original on 2012-03-17.
6. Conrad, Eric; Misenar, Seth; Feldman, Joshua (2017-01-01), Conrad, Eric; Misenar, Seth; Feldman, Joshua (eds.), "Chapter 3 - Domain 3: Security engineering" (http://www.sciencedirect.com/science/article/pii/B9780128112489000036), *Eleventh Hour CISSP® (Third Edition)*, Syngress, pp. 47–93, doi:10.1016/b978-0-12-811248-9.00003-6 (https://doi.org/10.1016%2Fb978-0-12-811248-9.00003-6), ISBN 978-0-12-811248-9, retrieved 2020-11-01
7. NIST Computer Security Division's (CSD) Security Technology Group (STG) (2013). "Current modes" (http://csrc.nist.gov/groups/ST/toolkit/BCM/current_modes.html). *Cryptographic Toolkit*. NIST. Archived (https://web.archive.org/web/20130402203842/http://csrc.nist.gov/groups/ST/toolkit/BCM/current_modes.html) from the original on April 2, 2013. Retrieved April 12, 2013.

8. "Stream Cipher Reuse: A Graphic Example" (http://cryptosmith.com/2008/05/31/stream-reuse/). Cryptosmith LLC. 31 May 2008. Archived (https://web.archive.org/web/20150125035328/http://cryptosmith.com/2008/05/31/stream-reuse/) from the original on 25 January 2015. Retrieved 7 January 2015.
9. B. Moeller (May 20, 2004), *Security of CBC Ciphersuites in SSL/TLS: Problems and Countermeasures* (https://www.openssl.org/~bodo/tls-cbc.txt), archived (https://web.archive.org/web/20120630143111/http://www.openssl.org/~bodo/tls-cbc.txt) from the original on June 30, 2012
10. Tervoort, Tom. "Zerologon: Unauthenticated domain controller compromise by subverting Netlogon cryptography (CVE-2020-1472)" (https://www.secura.com/pathtoimg.php?id=2055). *Secura*. Retrieved 14 October 2020.
11. Blaufish (14 October 2020). "Netlogon CFB8 considered harmful. OFB8 also" (https://github.com/blaufish/cfb8-vulnerable). *GitHub*. Retrieved 14 October 2020.
12. Gligor, Virgil D.; Donescu, Pompiliu (2002). Matsui, M. (ed.). *Fast Encryption and Authentication: XCBC Encryption and XECB Authentication Modes* (https://link.springer.com/content/pdf/10.1007%2F3-540-45473-X_8.pdf) (PDF). Fast Software Encryption 2001. Lecture Notes in Computer Science. Vol. 2355. Berlin: Springer. pp. 92–108. doi:10.1007/3-540-45473-X_8 (https://doi.org/10.1007%2F3-540-45473-X_8). ISBN 978-3-540-43869-4.
13. Jutla, Charanjit S. (May 2001). *Encryption Modes with Almost Free Message Integrity* (https://iacr.org/archive/eurocrypt2001/20450525.pdf) (PDF). Eurocrypt 2001. Lecture Notes in Computer Science. Vol. 2045. Springer. doi:10.1007/3-540-44987-6_32 (https://doi.org/10.1007%2F3-540-44987-6_32).
14. Dworkin, Morris (May 2004). *Recommendation for Block Cipher Modes of Operation: The CCM Mode for Authentication and Confidentiality* (http://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-38c.pdf) (PDF) (Technical report). NIST Special Publications. NIST. doi:10.6028/NIST.SP.800-38C (https://doi.org/10.6028%2FNIST.SP.800-38C). 800-38C.
15. Whiting, D.; Housley, R.; Ferguson, N. (September 2003). *Counter with CBC-MAC (CCM)* (https://datatracker.ietf.org/doc/html/rfc3610). IETF. doi:10.17487/RFC3610 (https://doi.org/10.17487%2FRFC3610). RFC 3610 (https://datatracker.ietf.org/doc/html/rfc3610).
16. Harkins, Dan (October 2008). "Synthetic Initialization Vector (SIV) Authenticated Encryption Using the Advanced Encryption Standard (AES)" (https://tools.ietf.org/html/rfc5297). Retrieved 21 October 2020.
17. Gueron, S. (April 2019). *AES-GCM-SIV: Nonce Misuse-Resistant Authenticated Encryption* (https://datatracker.ietf.org/doc/html/rfc8452). IETF. doi:10.17487/RFC8452 (https://doi.org/10.17487%2FRFC8452). RFC 8452 (https://datatracker.ietf.org/doc/html/rfc8452). Retrieved August 14, 2019.
18. Gueron, Shay; Langley, Adam; Lindell, Yehuda (14 Dec 2018). "AES-GCM-SIV: Specification and Analysis" (https://eprint.iacr.org/2017/168). *Cryptology ePrint Archive*. Report (2017/168). Retrieved 19 October 2020.
19. "Recommendation for Block Cipher Modes of Operation" (http://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-38a.pdf) (PDF). *NIST.gov*. NIST. p. 9. Archived (https://web.archive.org/web/20170329041940/http://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-38a.pdf) (PDF) from the original on 29 March 2017. Retrieved 1 April 2017.
20. Menezes, Alfred J.; van Oorschot, Paul C.; Vanstone, Scott A. (2018). *Handbook of Applied Cryptography* (https://books.google.com/books?id=YyCyDwAAQBAJ). CRC Press. p. 228. ISBN 9780429881329.
21. Dam, Kenneth W.; Lin, Herbert S. (1996). *Cryptography's Role in Securing the Information Society* (https://books.google.com/books?id=H8OaAgAAQBAJ). National Academies Press. p. 132. ISBN 9780309054751.

22. Schneier, Bruce (2015). *Applied Cryptography: Protocols, Algorithms and Source Code in C* (https://books.google.com/books?id=VjC9BgAAQBAJ). John Wiley & Sons. p. 208. ISBN 9781119096726.
23. William F. Ehrsam, Carl H. W. Meyer, John L. Smith, Walter L. Tuchman, "Message verification and transmission error detection by block chaining", US Patent 4074066, 1976.
24. "The Transport Layer Security (TLS) Protocol Version 1.1" (https://web.archive.org/web/20150107231938/http://www.ietf.org/rfc/rfc4346.txt). p. 20. Archived from the original (http://www.ietf.org/rfc/rfc4346.txt) on 2015-01-07. Retrieved 7 January 2015.
25. "Kryptographie FAQ: Frage 84: What are the Counter and PCBC Modes?" (http://www.iks-jena.de/mitarb/lutz/security/cryptfaq/q84.html). *www.iks-jena.de*. Archived (https://web.archive.org/web/20120716222121/http://www.iks-jena.de/mitarb/lutz/security/cryptfaq/q84.html) from the original on 16 July 2012. Retrieved 28 April 2018.
26. Kaufman, C.; Perlman, R.; Speciner, M. (2002). *Network Security* (2nd ed.). Upper Saddle River, NJ: Prentice Hall. p. 319. ISBN 0130460192.
27. Kohl, J. (1990). "The Use of Encryption in Kerberos for Network Authentication" (https://web.archive.org/web/20090612060426/http://dsns.csie.nctu.edu.tw/research/crypto/HTML/PDF/C89/35.PDF) (PDF). *Proceedings, Crypto '89*. Berlin: Springer. ISBN 0387973176. Archived from the original (http://dsns.csie.nctu.edu.tw/research/crypto/HTML/PDF/C89/35.PDF) (PDF) on 2009-06-12.
28. (NIST), Author: Morris Dworkin (2001). "SP 800-38A, Recommendation for Block Cipher Modes of Operation: Methods and Techniques" (http://csrc.nist.gov/publications/nistpubs/800-38a/sp800-38a.pdf) (PDF). *csrc.nist.gov*. doi:10.6028/NIST.SP.800-38A (https://doi.org/10.6028%2FNIST.SP.800-38A). Archived (https://web.archive.org/web/20170828210645/http://csrc.nist.gov/publications/nistpubs/800-38a/sp800-38a.pdf) (PDF) from the original on 28 August 2017. Retrieved 28 April 2018. {{cite journal}}: |first= has generic name (help)
29. Davies, D. W.; Parkin, G. I. P. (1983). "The average cycle size of the key stream in output feedback encipherment". *Advances in Cryptology, Proceedings of CRYPTO 82*. New York: Plenum Press. pp. 263–282. ISBN 0306413663.
30. Jueneman, Robert R. (1983). "Analysis of certain aspects of output feedback mode". *Advances in Cryptology, Proceedings of CRYPTO 82*. New York: Plenum Press. pp. 99–127. ISBN 0306413663.
31. Lipmaa, Helger; Wagner, David; Rogaway, Phillip (2000). "Comments to NIST concerning AES Modes of Operations: CTR-Mode Encryption" (http://csrc.nist.gov/groups/ST/toolkit/BCM/documents/proposedmodes/ctr/ctr-spec.pdf) (PDF). Archived (https://web.archive.org/web/20150226072817/http://csrc.nist.gov/groups/ST/toolkit/BCM/documents/proposedmodes/ctr/ctr-spec.pdf) (PDF) from the original on 2015-02-26.
32. Ferguson, Niels; Schneier, Bruce; Kohno, Tadayoshi (2010). *Cryptography Engineering*. p. 71.
33. "Basic Block Cipher Modes" (http://www.quadibloc.com/crypto/co040601.htm). *www.quadibloc.com*. Archived (https://web.archive.org/web/20171024164915/http://www.quadibloc.com/crypto/co040601.htm) from the original on 24 October 2017. Retrieved 28 April 2018.
34. "Cryptography I" (https://www.coursera.org/learn/crypto). *Coursera*. Archived (https://web.archive.org/web/20180323220803/https://www.coursera.org/learn/crypto) from the original on 23 March 2018. Retrieved 28 April 2018.

35. "Modes Development – Block Cipher Techniques – CSRC" (http://csrc.nist.gov/groups/ST/toolkit/BCM/modes_development.html). Computer Security Division, Information Technology Laboratory, National Institute of Standards and Technology, U.S. Department of Commerce. 4 January 2017. Archived (https://web.archive.org/web/20170904011624/http://csrc.nist.gov/groups/ST/toolkit/BCM/modes_development.html) from the original on 4 September 2017. Retrieved 28 April 2018.
