# Block ciphers and modes of operations

## *Basic concepts*

Modern ciphers are block ciphers. In block ciphers, the plaintext is divided into blocks of fixed length (in bits, usually), which will be inputted separately to the encryption function. To produce the corresponding cryptogram, the encryption process will have to pickup and combine the output blocks in a specific way, which is called an *operation mode*. There are many operation modes as we will see later in this chapter.

Note that the classic ciphers we have studied so far are *stream ciphers*: as appose to block ciphers, the encryption function takes each letter from the plaintext as input.

Example 1. A simple block cipher with block size =3

| key | 000 | 001 | 010 | 011 | 100 | 101 | 110 | 111 |
|---|---|---|---|---|---|---|---|---|
| 0 | 001 | 111 | 110 | 000 | 100 | 010 | 101 | 011 |
| 1 | 001 | 110 | 111 | 100 | 011 | 010 | 000 | 101 |
| 2 | 001 | 000 | 100 | 101 | 110 | 111 | 010 | 011 |
| 3 | 100 | 101 | 110 | 111 | 000 | 001 | 010 | 011 |
| 4 | 101 | 110 | 100 | 010 | 011 | 001 | 011 | 111 |

Consider plaintext 010100110111
010 100 110 111 ➔ 111 011 000 101 using key=1
010 100 110 111 ➔ 100 011 011 111 using key=4
Note that there are 5 keys, $2^2 < 5 < 2^3$ , so we need 3 bits to present a key ➔ key size= block size= 3. Small sizes are dangerous, however: if Eve catches C=001 then she can infer P= 000 or 101.

Be definition, a block cipher is an invertible map:

- Map n-bit plaintext blocks to n-bit ciphertext blocks (n: block size/length).
- For n-bit plaintext and ciphertext blocks and a fixed key, the encryption function is a bijection:
  $E : P_n \times K \rightarrow C_n$ s.t. for all key $k \in K$, $E(x, k)$ is an invertible mapping written $E_k(x)$.
- The inverse mapping is the decryption function, $y = D_k(x)$ denotes the decryption of plaintext x under key k.

## General condition in creating secure block ciphers

- The block size has to be large enough to prevent against statistical analysis
  (Note, however, larger block size means slower processing)

- The key space (then key length) must be large enough to prevent against exhaustive key search
  (However, key length shouldn't be too big that makes key distribution and management more difficult)

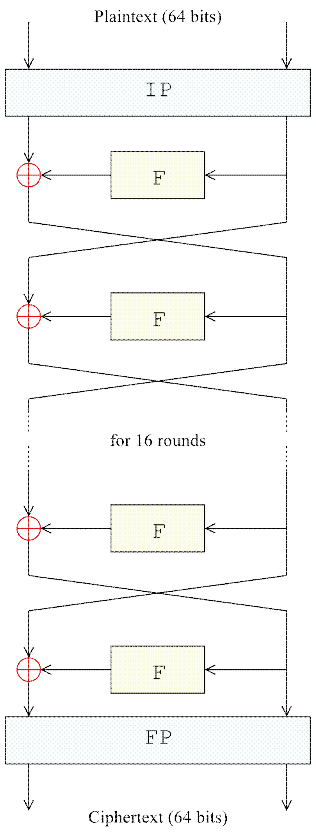
## General principles in designing secure block ciphers

- *Confusion:* As a function, the dependence of the ciphertext on the plaintext should be complex enough so that enemy can't find the rules. Typically, the function should be non-linear.
- *Diffusion:* The goal is to spread the information from the plaintext over the entire ciphertext so that changes in plaintext affect many parts in ciphertext. This makes it difficult for an enemy to break the cipher by using statistical analysis

Usually, confusion is made by using substitutions while *diffusion* by transpositions and/or permutations.

## The Feistel structure: processing in rounds

Block ciphers are usually designed with many rounds where basic round accomplishes the core function *f* for basic confusion and diffusion. The input of a round is the output of the previous round and a subkey which is generated by a key-schedule algorithm. The decryption is a reverse process where the sub-keys are handled in the reverse order. The Data Encryption Standard (DES) use a 16-round structure, depicted in the following picture.



The core function *f* has a special property, called *involution*, that is $f = f^{-1}$ or $f(f(x)) = x$ for all values *x*.

### Block cipher features

Each block cipher is characterized by the followings:

- Block size**:** in general larger block sizes mean greater security.
- Key size**:** larger key size means greater security (larger key space).
- Number of rounds**:** multiple rounds offer increasing security.
- Operation (Encryption) modes: define how messages larger than the block size are encrypted, very important for the security of the encrypted message.

## *Data Encryption Standard (DES)*

### Historic facts

The **Data Encryption Standard** (**DES**) was selected by the National Bureau of Standards as an official Federal Information Processing Standard for the United States in 1976 and hence, was in widespread use over the world. However, the DES algorithm with classified design elements was a controversial story for many years because of a relatively short key length (56-bit) and suspicions about a National Security Agency (NSA) backdoor. These suspicions caused intense academic scrutiny, a strong motivation due to which scientists have learned a lot about block ciphers and their cryptanalysis. Below is list of histori facts.

- 1967: Feistel did his work at IBM and came up with Lucifer cipher: block size 128 bits; key size 128 bits.
- 1972: NBS asked for an encryption standard
- 1975: IBM developed DES (modification of Lucifer) but with 64-bit block size and 56-bit key size.
- 1975: NSA suggested modification
- 1977: NBS adopted DES as encryption standard in (FIPS 46-1, 46-2).
- 2001: NIST adopted Rijndael as replacement to DES

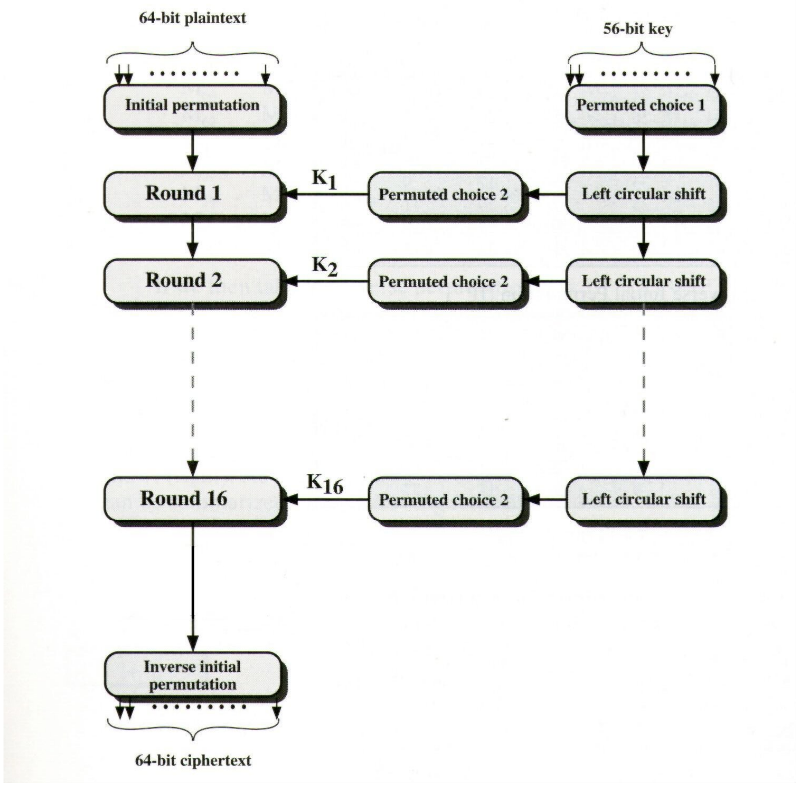
***Thus, DES main features***:

- Block size = 64 bits
- Key size = 56 bits
- Number of rounds = 16
- 16 intermediary keys, each 48 bits

### Subkeys and the key scheduler

The key scheduler algorithm is to generate the 16 sub-keys to be used with each round in the Feistel's structure. Initially, 56 bits of the key are selected from the initial 64 by an operation called *Permuted Choice 1* whereby the remaining eight bits are either discarded or used as parity check bits. The 56 selected bits are then divided into two 28-bit halves each of which is treated separately. During each round, both halves are rotated left (left circular shifted) by one or two bits, and then 48 subkey bits are selected by

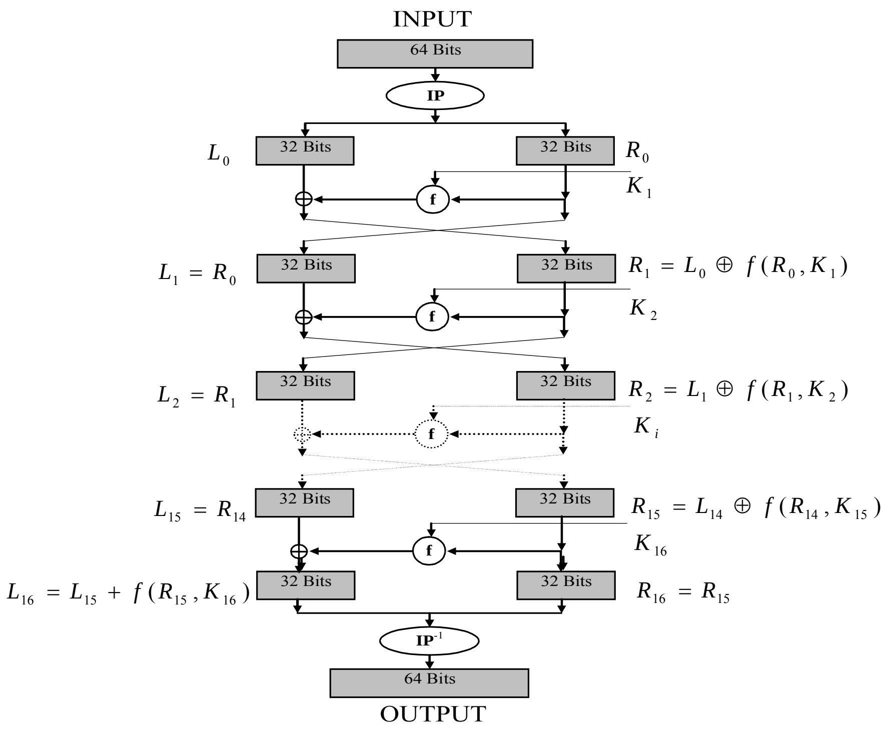
*Permuted Choice 2* from both halves (24 bits from each). The key scheduler for decryption works similarly but in the reverse order.



## DES encryption: A closer look

Each round of the DES algorithm perform such a basic conversion, depicted by:

$$(L_i,R_i) = (R_{i-1}, L_{i-1} ( F(R_{i-1},K_i))$$



## Cryptanalysis of DES

### Brute Force attack.

For any cipher, the most basic method of attack is brute force — trying every possible key until finding one which show the plaintext (under *Known-Plaintext Attack*). Clearly, the length of the key determines the size of the key space, and hence the feasibility of this approach. Historically, NSA with an external consult decided to reduced the key size from 128 bits (in Lucipher) to 56 bits to fit on a single chip. This, however created an controversy about the possibility of brute-force attack against such a short key.

For several years after DES birth, various proposals for a DES-cracking machine were discussed:

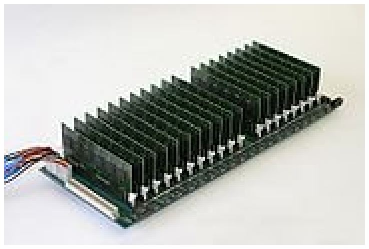
- 1977, Diffie and Hellman; machine costing ~ US$20 million; could find a DES key in a single day
- 1993, Wiener; machine costing US$1 million; within 7 hours

(*However, these early proposals were never implemented*)

Breaking DES was practically demonstrated in the late 1990s:

- 1997, RSA Security's contest ($10,000 prize to the first that broke a message encrypted with DES) was won by the DESCHALL Project, using idle cycles of thousands of computers across the Internet.
- 1998, a custom DES-cracker, built by the Electronic Frontier Foundation (EFF), costed of ~ US$250,000 brute-forced a key in a little more than 2 days search.

In 2006, COPACOBANA machine, built using 120 low-cost FPGAs by the Universities of Bochum and Kiel, Germany, costed only ~ US$10,000 and could perform an exhaustive key search on DES in ~9 days.



Below, we will discuss smatter attacks which do need to perform an exhaustive search.

### Differential cryptanalysis (DC)

This attack had been actually known earlier to both IBM and the NSA during the design of DES. However, it was rediscovered (so, officially public) in the late 1980s by Biham and Shamir. To break the full 16 rounds under the chosen-plaintext model, differential cryptanalysis requires $2^{47}$ chosen plaintexts, which was a hard computation task to do at that time. Only quite a while later, the public knew about DC and that DES was designed to be resistant to DC.

**Linear cryptanalysis (LC)**

Proposed by Mitsuru Matsui in 1993, LC uses known-plaintext model and needs $2^{43}$ known plaintexts. There is no evidence that DES was designed with knowledge of this type of attack. Other refined versions or generalization of LC have been proposed later to reduce the number of computation complexity to around $2^{40}$ DES operations.

There have also been attacks proposed against i.e. lesser versions of DES (with fewer than sixteen rounds). These give an insight knowledge into how many rounds are required for safety: the full 16-round version can be reasonably explained. Initiated by Langford and Hellman in 1994, a compound attack using both differential and linear cryptanalysis (hence, Differential-linear cryptanalysis) can break 9-round DES with $2^{15.8}$ known plaintexts and has a $2^{29.2}$ time complexity (Biham et al., 2002).

## *Advanced Encryption Standard (AES)*

AES is an encryption standard adopted by the U.S. government. The standard consists of three block ciphers, AES-128, AES-192 and AES-256, selected and from the original, larger collection known as **Rijndael.** Each AES cipher has a 128-bit block size, with key sizes of 128, 192 and 256 bits, respectively. Just like DES, AES has been analyzed extensively and now enjoy a worldwide use.

The above mentioned problems with DES short keys were the reason that National Institute of Standards and Technology (NIST) attempted to find a successor by organizing a world-wide competition for new block cipher standards in the late 90's. The Rijndael cipher was developed by Daemen and Rijmen and was announced by NIST as U.S. FIPS PUB 197 (FIPS 197) in 2001 after a 5-year standardization process (wherein fifteen competing designs were proposed and evaluated).

### AES features

Below is a brief list of the cipher main features:

- Designed to be efficient in both hardware and software across a variety of platforms. AES is based on a design principle known as a Substitution permutation network. It is fast in both software and hardware. Unlike DES, AES does not use a Feistel structure.
- Rijndael can be specified with block and key sizes in any multiple of 32 bits within 128 bits and 256 bits. However, AES uses fixed block size of 128 bits and a key size of 128, 192, or 256 bits.
- 128-bit round key used for each round (Can be pre-computed and cached for future encryptions). Note that AES uses a 128-bit block size.
- Variable number of rounds (10, 12, 14):
    - 10 if block size = key size = 128 bits
    - 12 if either block size or key size is 192 and the other is $\leq 192$
    - 14 if either block size or key size is 256 bit

- AES operates on a 4×4 array of bytes, termed the *state* (versions of Rijndael with a larger block size have additional columns in the state). Most AES calculations are done in a special finite field.
- Algorithms composed of three layers: linear diffusion, non-linear diffusion and key mixing
- The decryption algorithm is not identical with the encryption algorithm, but uses the same key schedule.
- There is also a way of implementing the decryption with an algorithm that is equivalent to the encryption algorithm (each operation replaced with its inverse), however in this case, the key schedule must be changed.

## AES cryptanalysis

From wikipedia: "A related-key attack can break 256-bit AES with a complexity of $2^{119}$, which is faster than brute force but is still infeasible. 192-bit AES can also be defeated in a similar manner, but at a complexity of $2^{176}$. 128-bit AES is not affected by this attack. A chosen-plaintext attack can break 8 rounds of 192- and 256-bit AES, and 7 rounds of 128-bit AES, although the workload is impractical at $2^{128}$ - $2^{119}$. (Ferguson et al., 2000)."

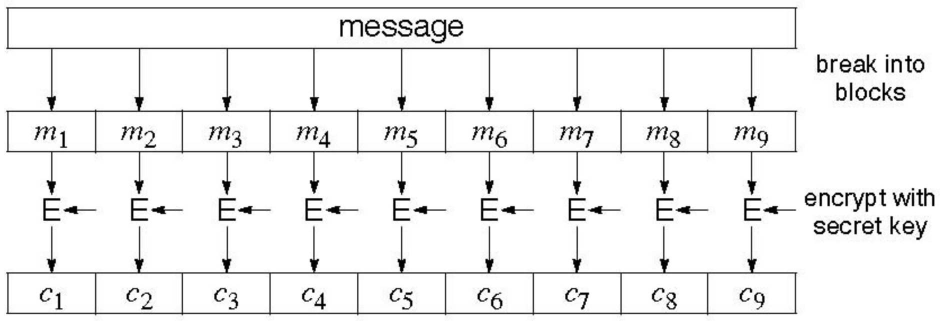
## *Modes of Operation (Encryption modes)*

A block cipher algorithm takes on a fixed-length input, i.e. a block, and output a block of usually the same length. In practice, we want to encrypt files of various lengths that is we need to divide a file into blocks of that given fixed length then let the block cipher works on each block separately.
Operation mode: the manner and structure in which we feed the block cipher with blocks of the plaintext file and then pickup and combine the output blocks to produce the ciphertext file. Note that we can use the same operation mode with different block ciphers to construct different encryption systems.
We now briefly review some popular modes - ECB, CBC, OFB, CFB, CTR – by discussing their main properties (privacy, integrity) and potential attacks against them.

## Electronic codebook (ECB)

The simplest of the encryption modes is the **electronic codebook** (ECB) mode. The message is divided into blocks and each block is encrypted separately. A clear vulnerability is that identical plaintext blocks are encrypted into identical ciphertext blocks, i.e. ECB does not hide data patterns well. Many forms of attack can be drawn from this feature: replay, reorder, and insert or delete selected blocks. Enemy can even manipulated with blocks from different old encrypted messages to create a new legally encrypted one.

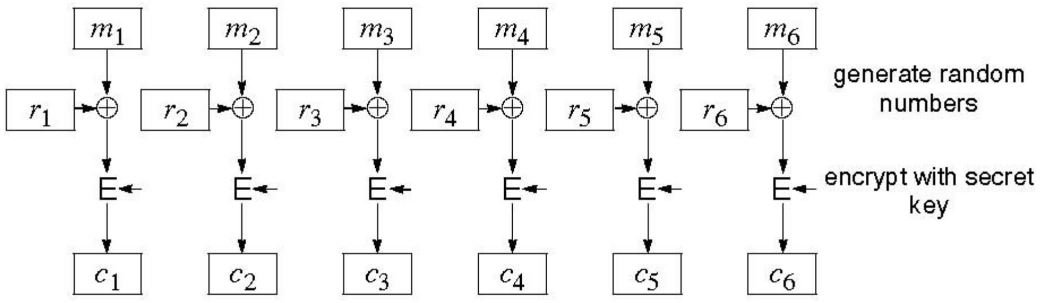
Clearly, ECB should not be used in any communication protocol.


message
break into blocks
$m_1$ $m_2$ $m_3$ $m_4$ $m_5$ $m_6$ $m_7$ $m_8$ $m_9$
E
encrypt with secret key
$c_1$ $c_2$ $c_3$ $c_4$ $c_5$ $c_6$ $c_7$ $c_8$ $c_9$

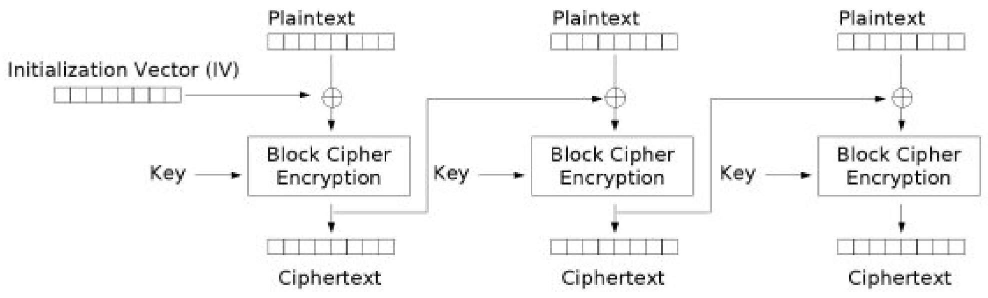
However, because of its nature of encrypting blocks separately, ECB can be used for storage systems, e.g. encrypting data segments in disks or records in relational databases. This allows straightforward updates to any disk segment or database record without interfering with surrounding units (which can not be avoided in CBC or others).

## Cipher-block chaining (CBC)

Let's see what one can do to improve on ECB, i.e. to hide data patterns. A possibly good idea is to add (different) random numbers to the plaintext blocks before encoding. Of course, on decrypting the receiver also need to subtract away these same random numbers.  Such an encryption scheme is depicted by the figure below.

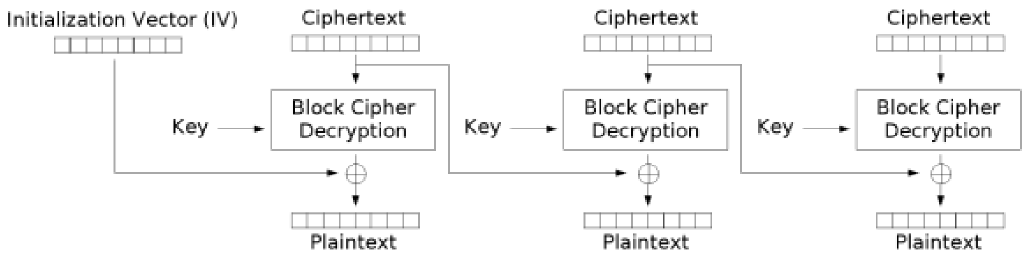

$m_1$ $m_2$ $m_3$ $m_4$ $m_5$ $m_6$
generate random numbers
$r_1$ $r_2$ $r_3$ $r_4$ $r_5$ $r_6$
E
encrypt with secret key
$c_1$ $c_2$ $c_3$ $c_4$ $c_5$ $c_6$

However, we do not want to have the sender generates and sends all the random numbers to the receiver, so a reasonable alternative is using pseudorandom numbers and a smple trick is just use the ciphertext block generated by the previous step.



Cipher Block Chaining (CBC) mode encryption

CBC mode of operation was invented by IBM in 1976. In the **cipher-block chaining** (CBC) mode, each block of plaintext is XORed with the previous ciphertext block before being encrypted. Obviously, each ciphertext block is dependent on all plaintext blocks processed up to that point. Also, initialization vector is used in the first block to make each message unique.



Cipher Block Chaining (CBC) mode decryption

**CBC critics**

There is a clear improvement on ECB:

- Since we use a randomized encryption, repeated text is mapped to different encrypted data.
- CBC provides  provable security: we can prove it assuming that the block cipher has desirable properties and that random IV's are used.
- A ciphertext block depends on all preceding plaintext blocks: thus, reorder affects decryption.

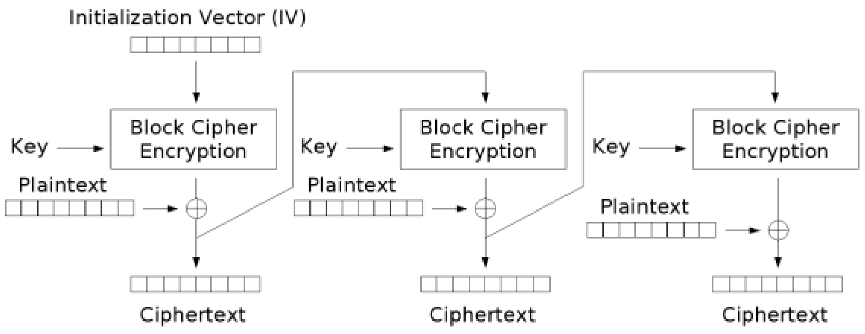
On the other hand, there are minus points:

- Errors in one block propagate to two blocks.
- One bit error in $C_j$ affects all bits in $M_j$ and one bit in $M_{j+1}$.
- CBC is a sequential process wherein we cannot use parallel hardware.
- Observation: if $C_i$=$C_j$ then $E_k$ ($M_i \oplus C_{i-1}$) = $E_k$ ($M_j \oplus C_{j-1}$); thus $M_i \oplus C_{i-1} = M_j \oplus C_{j-1}$**;** thus $M_i \oplus M_j = C_{i-1} \oplus C_{j-1}$.

## Cipher Feedback (CFB)

CFB has a similar look to CBC with this difference: the XOR is being made after the block cipher function rather than before as in CBC. However it makes a conceptual move here because CFB now looks quite close to the one-time pad and this makes it easier to formally show the security of CFB by comparing it with later.

CFB has strengths and weaknesses similar to CBC's, but it can be made into a self-synchronizing stream cipher: we can vary the input block size to any number of bits up to the block size used by the block cipher function.

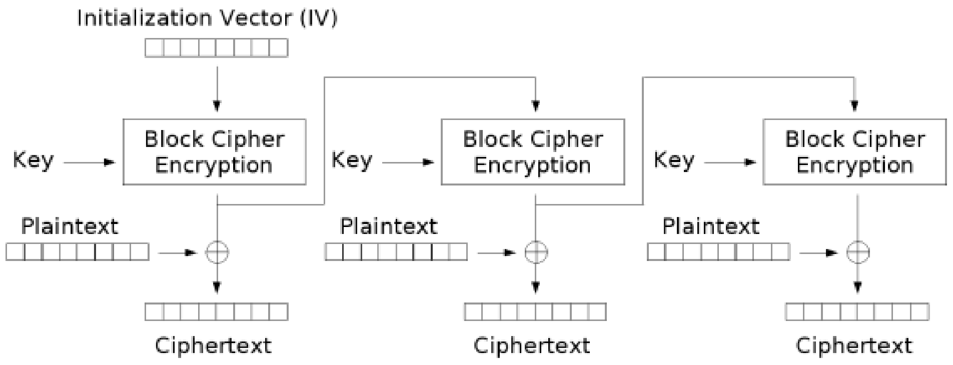


Cipher Feedback (CFB) mode encryption

## Output Feedback (OFB)

OFB makes another move closer to the one-time pad and we could even exploit some offline pre-computation to make it run faster:

- IV is used to generate a stream of blocks (can be pre-computed)
- This stream then is used as an one-time pad and XOR'ed to plaintext

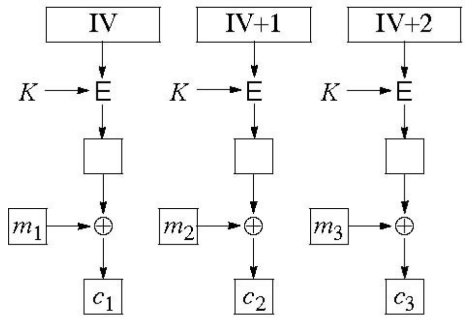


Output Feedback (OFB) mode encryption

Like CFB or CBC, OFB is a sequential encryption, however it allows preprocessing (computing the pad offline) and also allows the on-line XOR computation to be done in parallel. Moreover, error propagation is limited within one single block.

## Counter Mode (CTR)

CTR made another move, in fact a crucial one, towards simplification and efficiency. Simply, IV is incremented and used to generate one-time pad. Note that this incrementing pad should never be repeated or the scheme will be insecure: if the same IV and key is used again, XOR of two encrypted messages equals XOR of plain texts. Thus, if the communication session lasts too long, the key must be changed before the pad numbers making a full circle and repeating themselves.



CTR recorded significant advantages over the previous encryption modes and become widely used now:

- Software and hardware efficiency: different blocks can be encrypted in parallel.
- Preprocessing: the encryption part can be done offline and when the message is known, just do the XOR.
- Random access: decryption of a block can be done in random order, very useful for hard-disk/database encryption (like with ECB).
- Messages of arbitrary length: ciphertext is the same length with the plaintext (i.e., no IV).