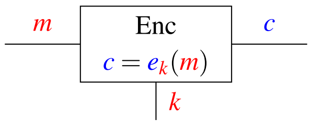
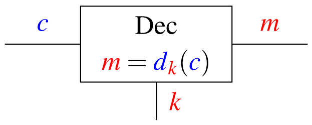
3 Symmetric Cryptography

Symmetric Cryptography

Alice

Bob



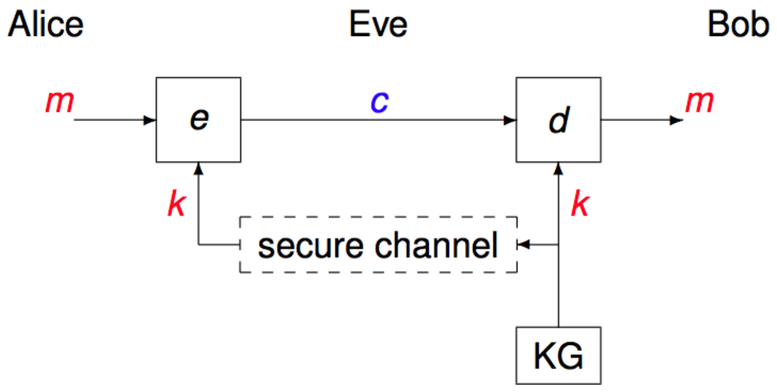


Symmetric cryptography uses the same secret key k for encryption and decryption.

Symmetric Ciphers

- Typically the same (reversible) algorithm is used for encryption and decryption.
- The algorithm(s) used may or may not be kept secret, but in either case, the strength of the algorithm should rest on the size of the keys and the design of the algorithm.
  - In theory, we can recover a symmetric key by performing an exhaustive search.
  - For a secure symmetric cipher, there should not be a more efficient algorithm for finding a key.
- There are numerous symmetric ciphers in use.
  - DES, triple-DES, IDEA, Skipjack, CAST, Blowfish, AES, ...
  - They have different key sizes (e.g., DES - 56 bits, IDEA - 128 bits, AES - 128, 192 or 256 bits).
  - Some are subject to patents in some countries.

Symmetric Key Cryptography



m = plaintext, c = ciphertext, k = key, KG = key generator.

Symmetric Key Cryptography
We write $c = e_k(m)$, where:

- m is the plaintext,
- e is the encryption function,
- k is the secret key,
- c is the ciphertext.

Decryption is given by $m = d_k(c)$.
Both sides need to know the key k, but k needs to be kept secret.

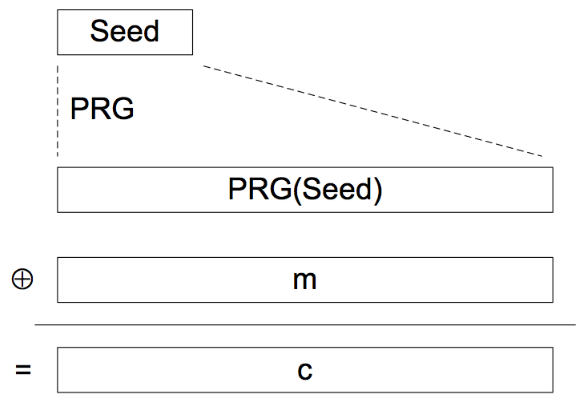
- secret-key, single-key or one-key algorithms.

Symmetric Ciphers
There are two types of symmetric cipher:

- Block ciphers that encrypt one block of data at a time.
  - Typically, blocks consist of 64 bits (8 bytes) or 128 bits (16 bytes) of data.
  - The same plaintext block always encrypts to the same ciphertext block.
  - In certain circumstances, this is a weakness and we need to use some sort of feedback to disguise patterns in the plaintext. This leads to different modes of operation.
- Stream ciphers that encrypt an arbitrary stream of data.
  - Depending on the cipher, the data may consist of a stream of bits or a stream of bytes.
  - Each plaintext bit (or byte) encrypts to a different cipher text bit (or byte) depending on what data occurred earlier in the stream.
  - It is possible to use a block cipher to implement a stream cipher.

3.1 Stream Ciphers

Stream Ciphers
Basic idea: replace the random key in the one-time pad by a pseudo-random sequence, generated by a cryptographic pseudo-random generator (PRG) that is 'seeded' with the key.



PRGs
PRG requirements:

- Randomness
  - Uniformity, scalability, consistency
- Unpredictability
  - Cannot determine what next bit will be despite knowledge of the algorithm and all previous bits.
- Unreproducable
  - Cannot be reliably reproduced.

Characteristics of the seed:

- Must be kept secret
- If known, adversary can determine output
- Must be random or pseudorandom number

Linear Congrentual Generator
Common iterative technique using:

$$X_{n+1} = (aX_n + c) \bmod m$$

Given suitable values of parameters can produce a long random-like sequence
Suitable criteria to have are:

- Function generates a full period
- Generated sequence should appear random
- Efficient implementation with 32-bit arithmetic

Note that an attacker can reconstruct sequence given a small number of values - this can be made harder in practice.

Blum Blum Shub Generator
Find two large primes p, q congruent to 3 (mod 4) where $m = p \times q$
Seed: $X_0 = k^2 \bmod m$ (k relatively prime to m)
Use least significant bit from iterative equation:

$$X_{n+1} = X_n^2 \bmod m$$

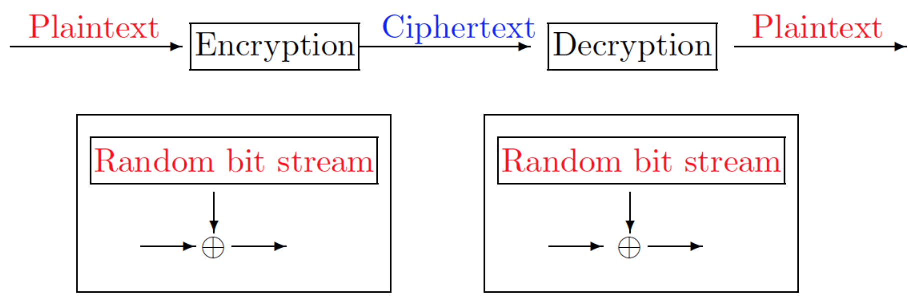
- Unpredictable given any run of bits
  - Passes next-bit test
- Security rests on difficulty of factoring m
- Slow since very large numbers must be used
  - Too slow for cipher use, but good for key generation

Real Random Numbers
In some cryptographic implementations, real random numbers are used.
Such numbers can be generated from random physical events.

- Measuring radioactive decay.
- Measuring the time to read blocks of data from a disk - this is affected by air turbulence and has a random component.
- Measuring the time between key strokes on a key board.
- Using the least significant bits of a computer's clock.

Using real random numbers is not usually a practical option.

Stream Ciphers
The stream cipher process is as follows:

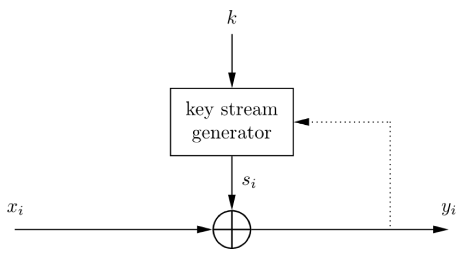


Stream Ciphers
Thus we have $c_i = m_i \oplus k_i$, where:

- m_i are the plaintext bits/bytes.
- k_i are the keystream bits/bytes.
- c_i are the ciphertext bits/bytes.

This means $m_i = c_i \oplus k_i$ and thus decryption is the same as encryption.
Encryption/decryption can be very fast.
No error propagation: one error in ciphertext gives one error in plaintext.
Loss of synchronisation means decryption fails for remaining ciphertext.
No protection against message manipulation.
Same key used twice gives same keystream.

Stream Ciphers



Stream ciphers can be synchronous or asynchronous.
In a synchronous stream cipher, the keystream depends only on the key.
In an asynchronous stream cipher, the keystream depends on the ciphertext (the dotted feedback on the diagram above).

Stream Ciphers
Synchronous stream ciphers:

- Sender and receiver must synchronise their actions for decryption to be successful.
- If digits are added or removed from the message during transmission, synchronisation is lost.
- To restore synchronisation, various offsets can be tried systematically to obtain the correct decryption.
- If a digit is corrupted in transmission only a single digit in the plaintext is affected.
- Errors do not propagate to other parts of the message.
- Useful when the transmission error rate is high.
- Very susceptible to active attacks.

Stream Ciphers
Asynchronous stream ciphers:

- Receiver will automatically synchronise with the keystream generator after receiving n ciphertext digits.
- This makes it easier to recover any digits dropped or added to the message stream.
- Single digit errors are limited in their effect, affecting only up to n plaintext digits.

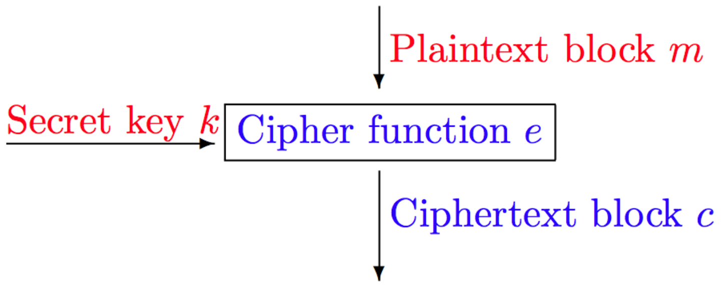
Stream Ciphers
For the stream cipher to be secure, the keystream must:

- Look random, i.e. pass pseudo-random tests.
- Be unpredictable, i.e. have a long period.
- Have large linear complexity (see textbooks).
- Have low correlation between key bits and keystream bits.

Furthermore, the keystream should be efficient to generate.
Most stream ciphers are based on a non-linear combination of LFSRs (Linear Feedback Shift Registers).

3.2 Block Ciphers

Block Ciphers
Plaintext is divided into blocks of fixed length and every block is encrypted one at a time.
A block cipher is a set of ‘code books’ and every key produces a different code book.
The encryption of a plaintext block is the corresponding ciphertext block entry in the code book.



Block Ciphers
We have $c = e_k(m)$ where:

- m is the plaintext block
- k is the secret key
- e is the encryption function
- c is the ciphertext

In addition we have a decryption function d such that:

$$m = d_k(c)$$

Padding
Problem: suppose length of plaintext is not multiple of block length.

- Last block m_t only contains $k < n$ bits.

Padding schemes:

- Append $n-k$ zeroes to last block. Trailing 0-bits in the plaintext cannot be distinguished from this padding, so an extra block has to be added giving the length of the message in bits.
- Append 1 and $n-k-1$ 0-bits. If $k = n$, then create extra block starting with 1 and remaining $n-1$ bits 0.

Block Size
The block size n needs to be reasonably large, $n > 64$ bits, to avoid:

- Text dictionary attacks: plaintext-ciphertext pairs for fixed key.
- Matching ciphertext attacks: uncover patterns in plaintext.

With a very large block size, the cipher becomes inefficient to operate.

- Plaintexts will then need to have a lot of padding added before being encrypted.

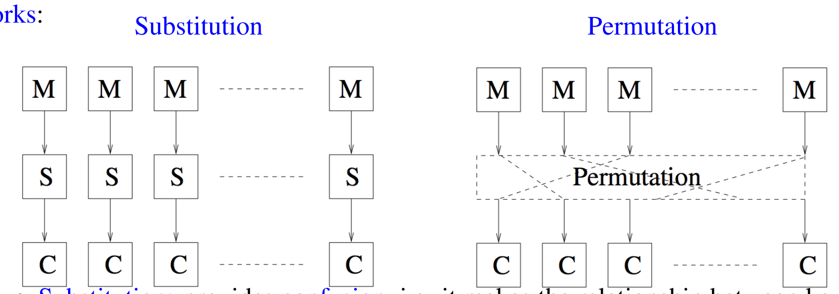
The preferred block size is a multiple of 8 bits as it is easier for implementation since most computer processors handle data in multiples of 8 bits.

Iterated Block Ciphers
An iterated block cipher involves repeated use of a round function.
The idea is to make a strong encryption function out of a weaker round function (easy to implement) by repeatedly using it.
The round function takes an n-bit block to an n-bit block.
Parameters: number of rounds r, blocksize n, keysize s.
Each use of the round function employs a subkey:

$$k_i \text{ for } 1 \leq i \leq r$$

derived from the key k.
For every subkey the round function must be invertible; if not decryption is impossible.

Substitution-Permutation Networks
Ciphers that use substitution and permutation are called substitution-permutation networks:



- Substitution: provides confusion, i.e. it makes the relationship between key and ciphertext complex.
- Permutation: provides diffusion, i.e. each bit (symbol) of the ciphertext depends on many (if possible all) bits (symbols) of the plaintext.

Substitution
A substitution box (S-box) substitutes a small block of input bits with another block of output bits.
An S-box can be considered as secure if changing one input bit will change about half of the output bits on average, exhibiting what is known as the avalanche effect.
The following is an example of a S-Box:

	00	01	10	11
0	10	01	11	00
1	00	10	01	11

For input bits $x_0x_1x_2$, x_0 gives the row, and x_1x_2 gives the column.
So, for example, if the input to this S-Box is 010, then the output is 11.

Permutation
A permutation box (P-box) is a permutation of all the bits.
It takes the outputs of all the S-boxes of the current round, permutes the bits, and feeds them into the S-boxes of the next round.
The following is an example of a P-Box:

01	15	02	13	06	17	03	19	09	04	21	11
14	05	12	16	18	07	24	10	23	08	22	20

The table gives the position the input bit is mapped onto in the output.
So, for example, input bit 01 is mapped to output bit 01, input bit 15 is mapped to output bit 02, input bit 14 is mapped to output bit 13, etc.

Data Encryption Standard (DES)
First published in 1977 as a US Federal standard.
Known as Data Encryption Algorithm DEA (ANSI) or DEA-1 (ISO).

- A de-facto international standard for banking security.
- An example of a Feistel Cipher.
- Most analyzed cryptographic algorithm.

Short history of DES:

- Work started in early 1970's by IBM.
- Based on IBM's Lucifer, but amended by NSA.
- Design criteria were kept secret for more than 20 years.
- Supposed to be reviewed every 5 years.

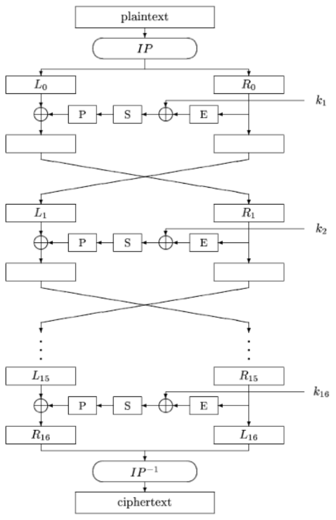
Data Encryption Standard (DES)
Block length is 64 bits, key length 56 bits.
Feistel Cipher:

- Initial permutation of bits.
- Split into left and right half.
- 16 rounds of identical operations, depending on round key.
- Inverse initial permutation.

Round transformation:

- Linear expansion: 32 bits → 48 bits.
- XOR with subkey of 48 bits (key schedule selects 48 bits of key k).
- 8 parallel non-linear S-boxes (6 input bits, 4 output bits).
- Permutation of the 32 bits.

Data Encryption Standard (DES)



Data Encryption Standard (DES)
Each DES round consists of the following six stages:
1. Expansion Permutation:

- Right half (32 bits) is expanded (and permuted) to 48 bits.

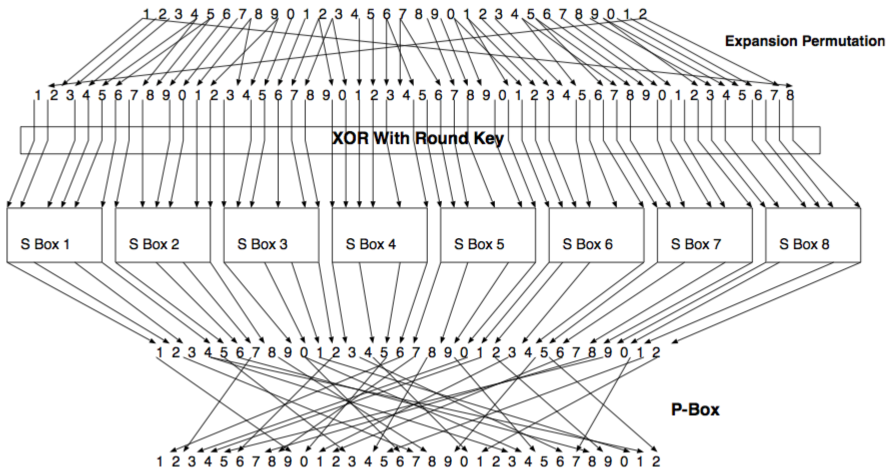
- Diffuses relationship of input bits to output bits.
- Means one bit of input affects two substitutions in output.
- Spreads dependencies.

2. Use of Round Key:
   - 48 bits are XOR-ed with the round key (48 bits).
3. Splitting:
   - Result is split into eight lots of six bit values.

Data Encryption Standard (DES)

4. S-Box:
   - Each six bit value is passed into an S-box to produce a four bit result in a non-linear way. (S = Substitution)
5. P-Box:
   - 32 bits of output are combined and permuted. (P = Permutation)
6. Feistel Part:
   - Output of f is XOR-ed with the left half resulting in new right half.
   - New left half is the old right half.

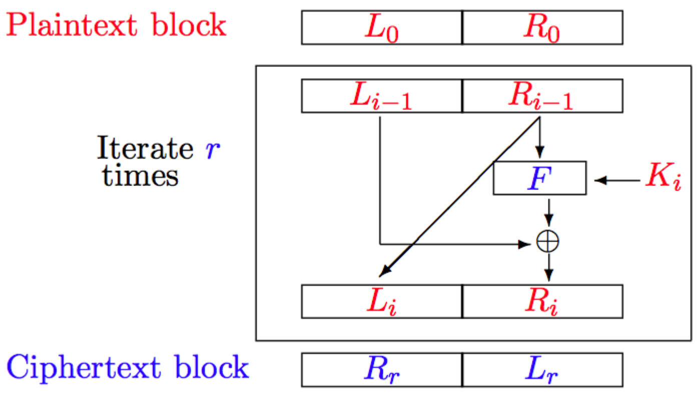
Data Encryption Standard (DES)



Data Encryption Standard (DES)
S-Boxes represent the non-linear component of DES.
There are eight different S-boxes.
The original S-boxes proposed by IBM were modified by NSA.
Each S-box is a table of 4 rows and 16 columns

- The 6 input bits specify which row and column to use.
- Bits 1 and 6 generate the row.
- Bits 2-5 generate the column.

Feistel Cipher



Feistel Cipher
The round function is invertible regardless of the choice of f .
Encryption round is:

- $L_i = R_{i-1}$
- $R_i = L_{i-1} \oplus f(R_{i-1}, k_i)$

Remark: in last step, the left half and right half are swapped:

- Decryption is achieved using the same r-round process.
- But with round keys used in reverse order, i.e. k_r first and k_1 last.

Feistel Cipher
Same algorithm can be used for decryption since we are using the Feistel structure:

$$L_{i-1} \oplus f(R_{i-1}, k_i) \oplus f(R_{i-1}, k_i) = L_{i-1}$$

Remark: for decryption the subkeys are used in the reverse order.
Note that the effect of IP^{-1} is cancelled by IP.
Also note that R_{16}, L_{16} is the input of encryption since halves were swapped and that swapping is necessary.

Security of DES
Exhaustive key search (1 or 2 known plaintext/ciphertext pairs).

- Number of possibilities for DES is $2^{56} = 7.2 \times 10^{16}$.

Software (PC with 3.2 GHz Processor): 2^{48} encryptions per year.

- 2^{23} keys per second, $2^{16.4}$ seconds per day, $2^{8.5}$ days per year.
- 1 PC: 125 years, 125 PC's: 1 year, 1500 PC's: 1 month.

Hardware (ASIC), Cost = $250,000, 'Deep Crack' (EFF, 1998).

- 1 key in 50 hours, less than $500 per key.
- Time halfed by working in conjunction with `distributed.net`
- For $1M: 1 key in 1/2 hour.

Hardware (FPGA), Cost = $10,000, 'COPACABANA', 2006

- 9 days (later reduced to 6 days)
- Reduced to less than one day by successor machine 'RIVYERA'

Security of DES
Export from US: 40-bit keys (SSL, Lotus Notes, S/MIME).

- Obviously much less secure.

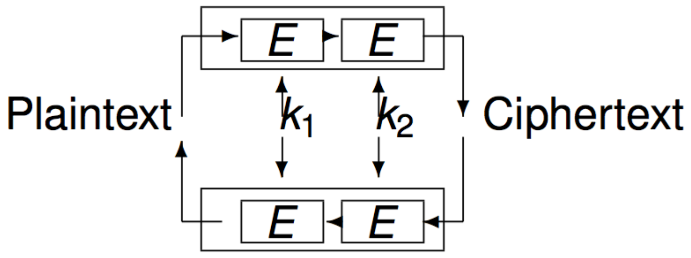
Moore's 'law':

- Computing power doubles every 18 months.
- After 21 years the effective key size is reduced by 14 bits.

Long term: key length and block length of 128 bits.

Multiple Encryption
DES is a ‘standard’: neither key size nor block size nor number of rounds can be changed (easily).
Remember: more rounds bring more security.
Idea: iterating the entire cipher might bring more security.
Double encryption, triple encryption, quadruple encryption, etc.
What makes most sense?

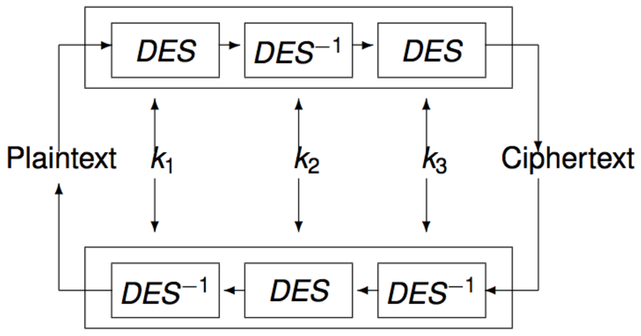
Double Encryption



Time-memory trade-off via a meet-in-the-middle attack is possible:

- Pre-compute for all keys $k_1 : C_1 = E_{k_1}(P)$.
- Given a ciphertext C: invert C and compare with list of C_1. Hit indicates k_2 and gives k_1. Check with one more C.
- Double encryption does not provide double security.

Triple DES



Invented to get around the problem of a short key.
Involves use of 2 or 3 DES keys.

Advanced Encryption Standard (AES)
January 1997: NIST call for algorithms to replace DES.

- Block cipher: 128-bit blocks, 128/192/256-bit keys.
- Strength $\approx 3 \times$ DES, much more efficient.
- Documentation, reference C code, optimised C and JAVA code, test vectors.
- Designers give up all intellectual rights.

Open process: public comments, international submissions.
Website: `http://www.nist.gov/aes/`

Advanced Encryption Standard (AES)
Standard FIPS-197 approved by NIST in November 2001.
Official scope was limited:

- US Federal Administration used AES as Government standard from 26 May 2002.
- Documents that are 'sensitive but not classified'.
- 2003: NSA has approved AES-128 also for secret information, and AES with key sizes larger than 128 for top secret information.
- Significance is huge: AES is the successor of DES.
- Major factors for quick acceptance:
  - No royalties.
  - High quality.
  - Low resource consumption.

Rijndael
Block length, key length: vary between 128 - 256 bits.
Number of rounds is 10/12/14 depending on block and key length.
Uniform and parallel round transformation, composed of:

- Byte substitution.
- Shift rows.
- Mix columns.
- Round key addition.

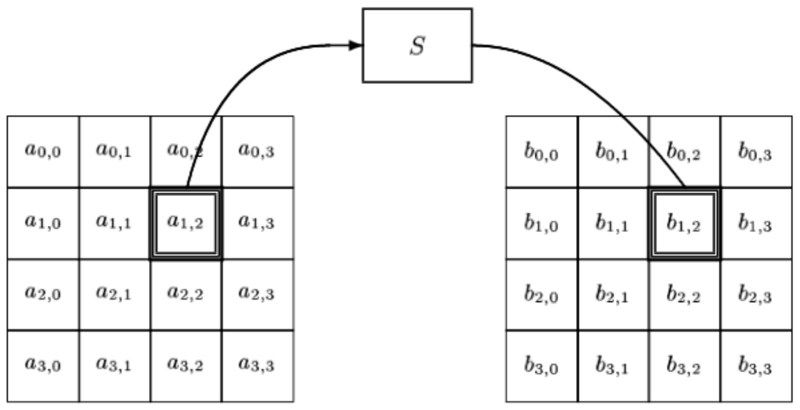
Sequential and light-weight key schedule.
No arithmetic operations.

Rijndael
Plaintext block normally is 128 bits or 16 bytes $m_0, \ldots, m_{15}$.
Key normally is 128/192/256 bits or 16/24/32 bytes $k_0, \ldots, k_{31}$.
Both are represented as rectangular array of bytes.

m_0	m_4	m_8	m_{12}
m_1	m_5	m_9	m_{13}
m_2	m_6	m_{10}	m_{14}
m_3	m_7	m_{11}	m_{15}

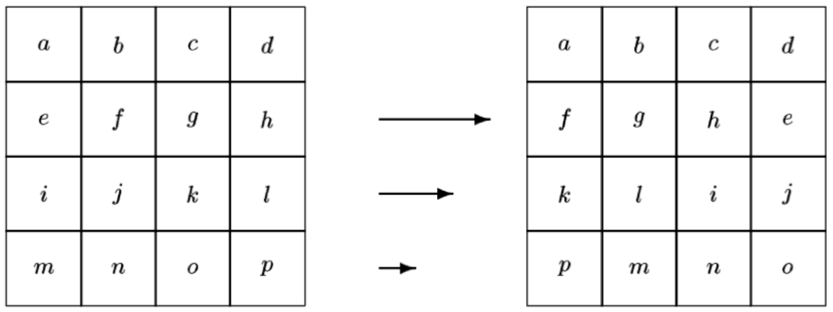
k_0	k_4	k_8	k_{12}	k_{16}	k_{20}
k_1	k_5	k_9	k_{13}	k_{17}	k_{21}
k_2	k_6	k_{10}	k_{14}	k_{18}	k_{22}
k_3	k_7	k_{11}	k_{15}	k_{19}	k_{23}

Rijndael: Byte Substitution
State matrix is transformed byte by byte.
S-box is invertible, else decryption would not work.
Only one S-box for the whole cipher (simplicity).


S
$a_{0,0}$ $a_{0,1}$ $a_{0,2}$ $a_{0,3}$
$a_{1,0}$ $a_{1,1}$ $a_{1,2}$ $a_{1,3}$
$a_{2,0}$ $a_{2,1}$ $a_{2,2}$ $a_{2,3}$
$a_{3,0}$ $a_{3,1}$ $a_{3,2}$ $a_{3,3}$
$b_{0,0}$ $b_{0,1}$ $b_{0,2}$ $b_{0,3}$
$b_{1,0}$ $b_{1,1}$ $b_{1,2}$ $b_{1,3}$
$b_{2,0}$ $b_{2,1}$ $b_{2,2}$ $b_{2,3}$
$b_{3,0}$ $b_{3,1}$ $b_{3,2}$ $b_{3,3}$

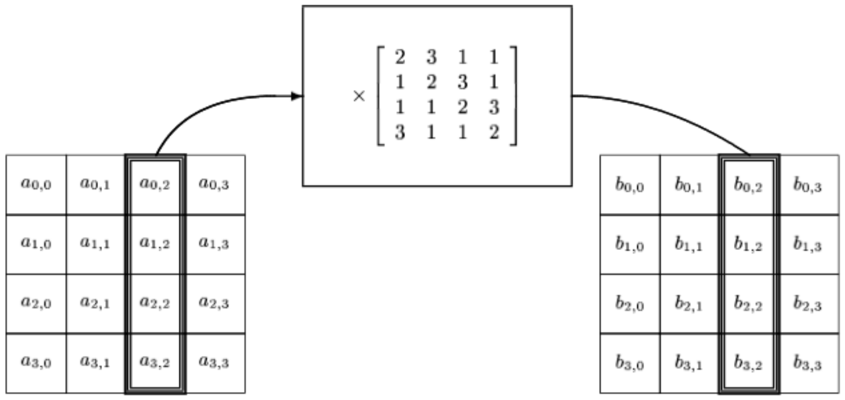
Rijndael: Shift Rows
Rows shifted over different offsets (depending on block length).

Purpose: diffusion over the columns.

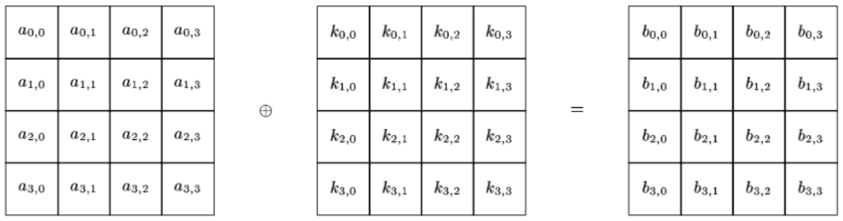


Rijndael: Mix Columns

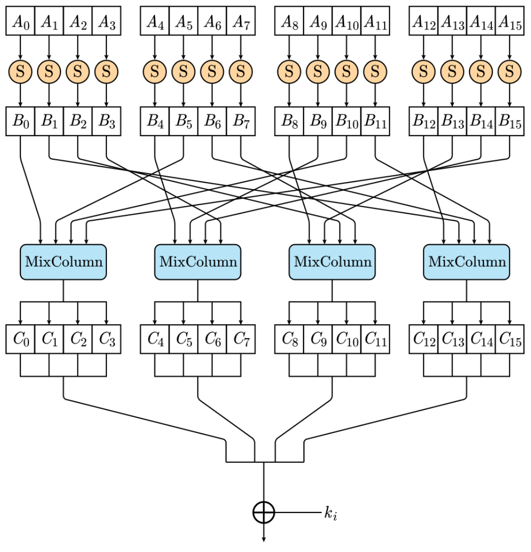
- Bytes in columns are combined linearly.
- Good diffusion properties over rows.


$\times \begin{bmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{bmatrix}$
$a_{0,0}$ $a_{0,1}$ $a_{0,2}$ $a_{0,3}$
$a_{1,0}$ $a_{1,1}$ $a_{1,2}$ $a_{1,3}$
$a_{2,0}$ $a_{2,1}$ $a_{2,2}$ $a_{2,3}$
$a_{3,0}$ $a_{3,1}$ $a_{3,2}$ $a_{3,3}$
$b_{0,0}$ $b_{0,1}$ $b_{0,2}$ $b_{0,3}$
$b_{1,0}$ $b_{1,1}$ $b_{1,2}$ $b_{1,3}$
$b_{2,0}$ $b_{2,1}$ $b_{2,2}$ $b_{2,3}$
$b_{3,0}$ $b_{3,1}$ $b_{3,2}$ $b_{3,3}$

Rijndael: Round Key Addition
Round key is simply XOR-ed with state matrix.


$a_{0,0}$ $a_{0,1}$ $a_{0,2}$ $a_{0,3}$
$a_{1,0}$ $a_{1,1}$ $a_{1,2}$ $a_{1,3}$
$a_{2,0}$ $a_{2,1}$ $a_{2,2}$ $a_{2,3}$
$a_{3,0}$ $a_{3,1}$ $a_{3,2}$ $a_{3,3}$
$\oplus$
$k_{0,0}$ $k_{0,1}$ $k_{0,2}$ $k_{0,3}$
$k_{1,0}$ $k_{1,1}$ $k_{1,2}$ $k_{1,3}$
$k_{2,0}$ $k_{2,1}$ $k_{2,2}$ $k_{2,3}$
$k_{3,0}$ $k_{3,1}$ $k_{3,2}$ $k_{3,3}$
=
$b_{0,0}$ $b_{0,1}$ $b_{0,2}$ $b_{0,3}$
$b_{1,0}$ $b_{1,1}$ $b_{1,2}$ $b_{1,3}$
$b_{2,0}$ $b_{2,1}$ $b_{2,2}$ $b_{2,3}$
$b_{3,0}$ $b_{3,1}$ $b_{3,2}$ $b_{3,3}$

Rijndael: Round Function

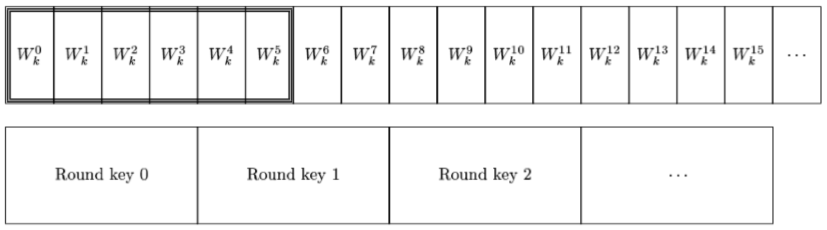

A_0 A_1 A_2 A_3 A_4 A_5 A_6 A_7 A_8 A_9 A_{10} A_{11} A_{12} A_{13} A_{14} A_{15}
S
B_0 B_1 B_2 B_3 B_4 B_5 B_6 B_7 B_8 B_9 B_{10} B_{11} B_{12} B_{13} B_{14} B_{15}
MixColumn
C_0 C_1 C_2 C_3 C_4 C_5 C_6 C_7 C_8 C_9 C_{10} C_{11} C_{12} C_{13} C_{14} C_{15}
k_i

Rijndael: Pseudo-code
Rijndael with 10 rounds is described by the following code:

```
AddRoundKey(S,K[0]);
for (i = 1; i <= 9; i++)
   {
      SubBytes(S);
      ShiftRows(S);
      MixColumns(S);
      AddRoundKey(S,K[i]);
   }
SubBytes(S);
ShiftRows(S);
AddRoundKey(S,K[10]);
```

Rijndael: Key Schedule

Example: key of 192 bits or 6 words of 32 bits.



$$W_k^{6n} = W_k^{6n-6} \oplus f(W_k^{6n-1})$$
$$W_k^{i} = W_k^{i-6} \oplus W_k^{i-1}$$

Cipher key expansion can be done just-in-time: no extra storage required.

Comparing AES and DES

DES:

- S-P Network, iterated cipher, Feistel structure
- 64-bit block size, 56-bit key size
- 8 different S-boxes
- non-invertible round
- design optimised for hardware implementations
- closed (secret) design process

AES:

- S-P Network, iterated cipher
- 128-bit block size, 128-bit (192, 256) key size
- one S-box
- invertible round
- design optimised for byte-orientated implementations
- open design and evaluation process

Non-Linearity in Block Ciphers

Non-linearity is often achieved in block ciphers within the S-Boxes.
This can be achieved by implementing the S-Boxes as non-linear functions.
In DES, the S-Boxes were actually designed by hand. The NSA found that the original S-Boxes designed by IBM did still have a linear relationship between the inputs and outputs, so they slightly modified them to strengthen them against differential cryptanalysis.
In AES, the S-Box is implemented as a non-linear mathematical function (using modular arithmetic), thus ensuring its resistance to cryptanalysis.

Block vs Stream Ciphers
Which is best?
Block ciphers:

- More versatile: can be used as stream cipher.
- Standardisation: DES and AES + modes of operation.
- Very well studied and accepted.

Stream ciphers:

- Easier to do the maths.
- Either makes them easier to break or easier to study.
- Supposedly faster than block ciphers (less flexible).

3.3 Modes of Operation

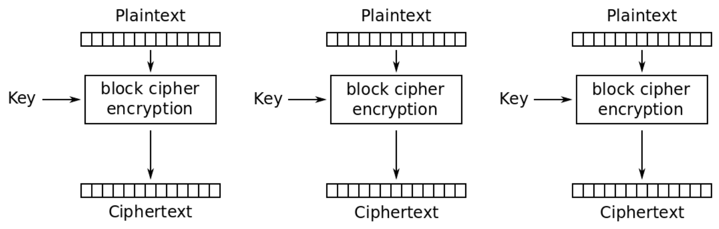
Modes of Operation
If message is longer than blocksize, block cipher can be used in a variety of ways to encrypt the plaintext.
Soon after DES was made a US Federal standard, another US standard appeared giving four recommended ways of using DES for data encryption.
These modes of operation have since been standardised internationally and can be used with any block cipher.

ECB Mode
ECB = Electronic Code Book
Simplest approach to using a block cipher.
Plaintext m is divided into t blocks of n bits $m_1, m_2, \ldots, m_t$ (the last block is padded if necessary).
Ciphertext blocks $c_1, \ldots, c_t$ are defined as follows:

$$c_i = e_k(m_i)$$

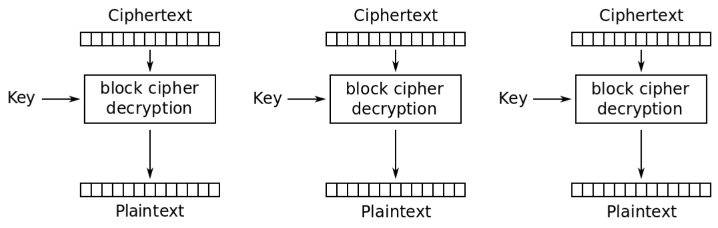
Note that if $m_i = m_j$ then we have $c_i = c_j$; thus patterns in plaintext reappear in ciphertext.

ECB Encipherment



Electronic Codebook (ECB) mode encryption

ECB Decipherment



Electronic Codebook (ECB) mode decryption

ECB Mode

Properties:

- Blocks are independent.
- Does not hide patterns or repetitions.
- Error propagation: expansion within one block.
- Reordering of blocks is possible without too much distortion.
- Stereotyped beginning/ends of messages are common.
- Susceptible to block replay.

Block Replay

Block replay:

- Extracting ciphertext corresponding to a known piece of plaintext.
- Amending other transactions to contain this known block of text.

Countermeasures against block replay:

- Extra checksums over a number of plaintext blocks.
- Chaining the cipher; this adds context to a block.

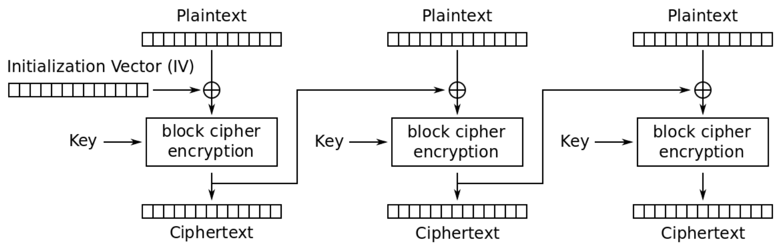
CBC Mode
CBC = Cipher Block Chaining
Plaintext m is divided into t blocks of n bits $m_1, m_2, \ldots, m_t$ (the last block is padded if necessary).
Encryption:

- $c_1 = e_k(m_1 \oplus IV)$.
- $c_i = e_k(m_i \oplus c_{i-1})$ for $i > 1$.

Decryption:

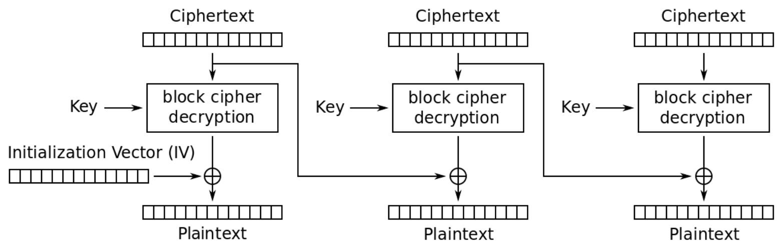
- $m_1 = d_k(c_1) \oplus IV$.
- $m_i = d_k(c_i) \oplus c_{i-1}$ for $i > 1$.

CBC Encipherment



Cipher Block Chaining (CBC) mode encryption

CBC Decipherment



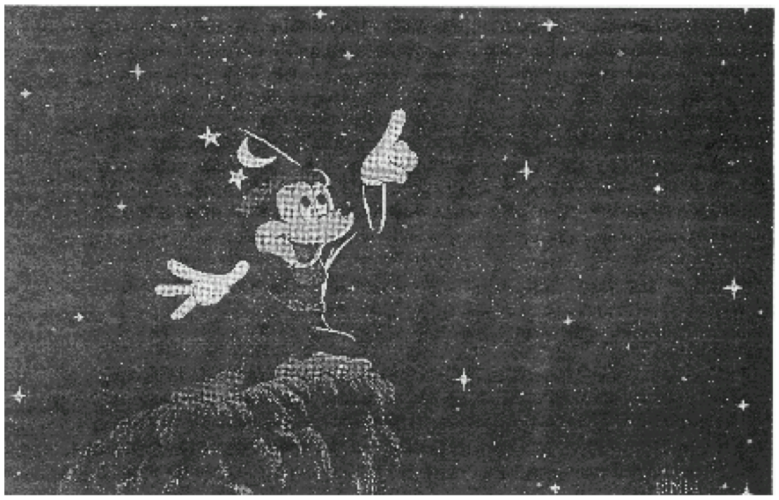
Cipher Block Chaining (CBC) mode decryption

CBC Mode

Properties:

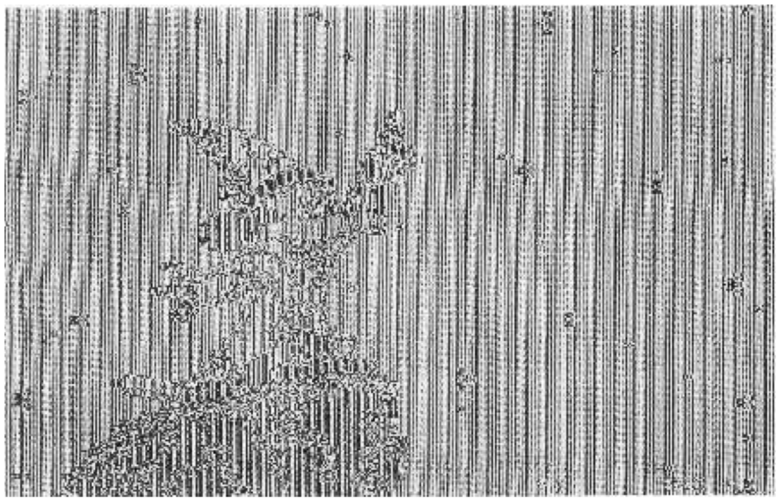
- Ciphertext depends on all previous plaintext blocks (internal memory).
- Different *IV* hides patterns and repetitions.
- Error propagation: expansion in one block, copied in next block.
- Decryption of one block requires only ciphertext of previous block, so CBC is self-synchronizing.
- Rearranging order of blocks affects decryption (not if previous ciphertext block is correct).
- Default mode to use.

Example

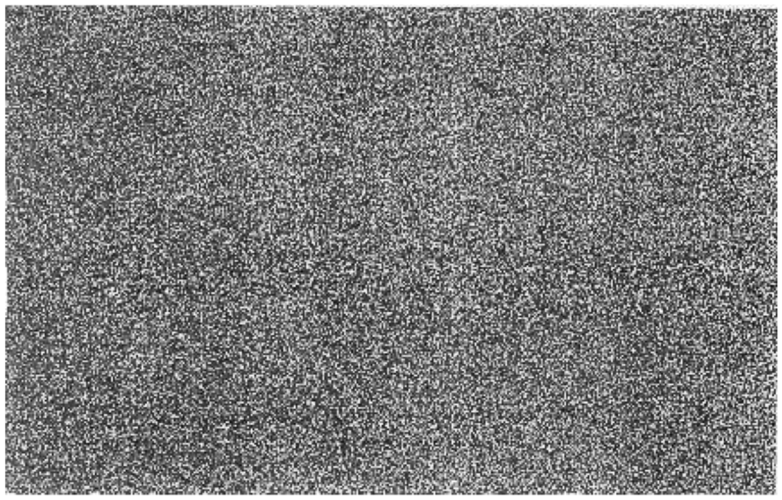


Plaintext : original picture

Example


Ciphertext: ECB Encryption

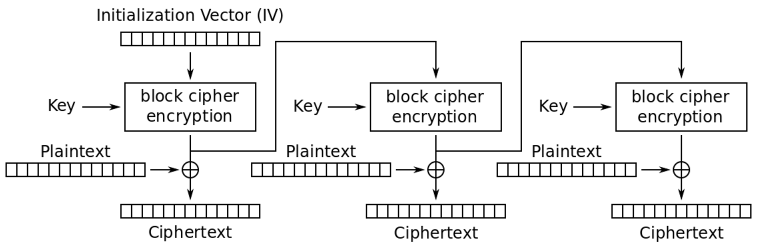
Example


Ciphertext: CBC Encryption

OFB Mode
OFB = Output FeedBack
This mode enables a block cipher to be used as a stream cipher.

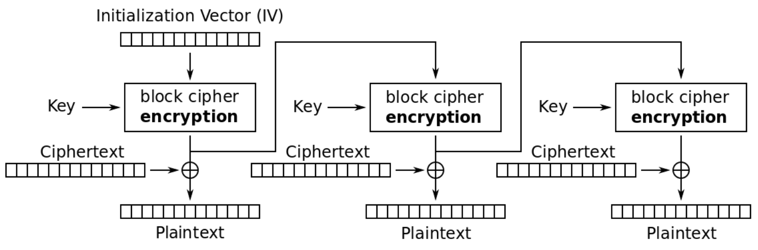
- The block cipher creates the keystream.
- Block length for block cipher is n.
- Can choose to use keystream blocks of $j \leq n$ bits only.
- Divide plaintext into a series of j-bit blocks $m_1, \ldots, m_t$.
- Encryption: $c_i = m_i \oplus e_i$, where e_i is selection of j bits of 'ciphertext' generated by block cipher, starting with IV in shift register.

OFB Encipherment



Output Feedback (OFB) mode encryption

OFB Decipherment



Output Feedback (OFB) mode decryption

OFB Mode
Properties:

- Synchronous stream cipher.
- No linking between subsequent blocks.
- Different IV necessary; otherwise insecure.

- Only uses encryption (no decryption algorithm necessary).
- If $j < n$: more effort per bit.
- Key stream independent of plaintext: can be precomputed.
- No error propagation: errors are only copied.

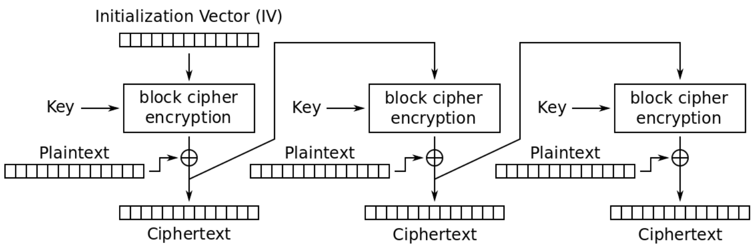
CFB Mode
CFB = Cipher FeedBack
In OFB Mode the keystream is generated by:

- Encrypting the IV.
- Encrypting the output from this encryption.

In CFB Mode the keystream is generated by:

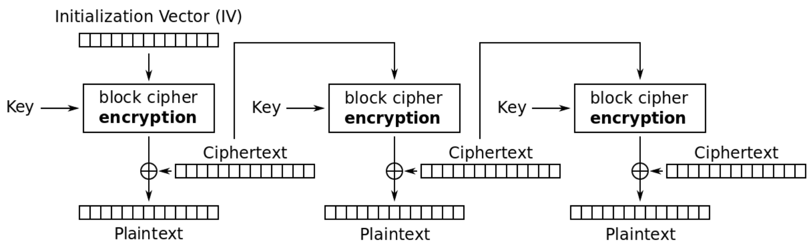
- Encrypting the IV.
- Encrypting n bits of ciphertext.

CFB Encipherment



Cipher Feedback (CFB) mode encryption

CFB Decipherment



Cipher Feedback (CFB) mode decryption

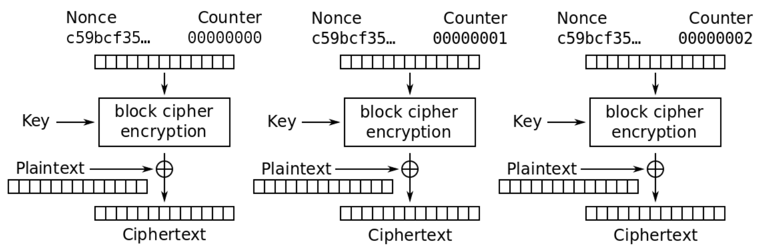
CFB Mode
Properties:

- Self-synchronizing stream cipher.
- Ciphertext depends on all previous plaintext blocks (internal memory).
- Different IV hides patterns and repetitions.
- Only uses encryption (no decryption algorithm necessary).
- If $j < n$: more effort per bit.
- Error propagation: propagates over $\lceil n/j \rceil + 1$ blocks.
- No synchronisation needed between sender and receiver; synchronisation if n bits have been received correctly.
- Often used with $j = 1, 8$ because of synchronisation.

CTR Mode
CTR = Counter

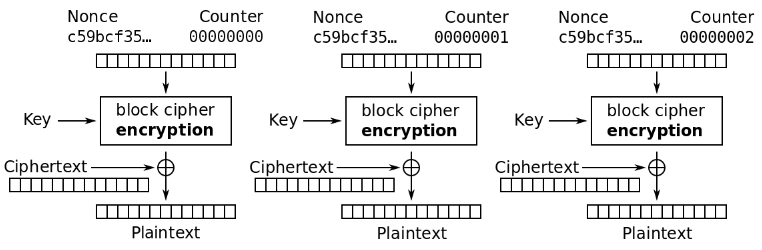
- Proposed more recently.
- Also turns the block cipher into a stream cipher.
- Enables blocks to be processed in parallel.
- Combines many of the advantages of ECB Mode, but with none of the disadvantages.
- Need to select a public IV and a different counter i for each message encrypted under the fixed key k.
- Encryption: $c_i = m_i \oplus e_k(IV + i)$
- Unlike ECB Mode two equal blocks will not encrypt to the same ciphertext value.
- Also unlike ECB Mode each ciphertext block corresponds to a precise position within the ciphertext, as its position information is needed to be able to decrypt it successfully.

CTR Encipherment



Counter (CTR) mode encryption

CTR Decipherment



Counter (CTR) mode decryption

Mode Summary

	ECB	CBC	OFB	CFB	CTR
Patterns	-	+	+	+	+
Repetitions	-	IV	IV	IV	IV
Block length	n	n	j	j	n
Error prop.	1 block	1 block + 1 bit	1 bit	n bits + 1 bit	1 block
Synch.	block	block	exact	j bits	block
Parallel	enc/dec	dec	enc/dec	dec	enc/dec
Application	key enc.	default	no error prop.	synch.	parallel