Cryptography 24th August 2015

Lecture 10: DES and AES; Mode of Operations

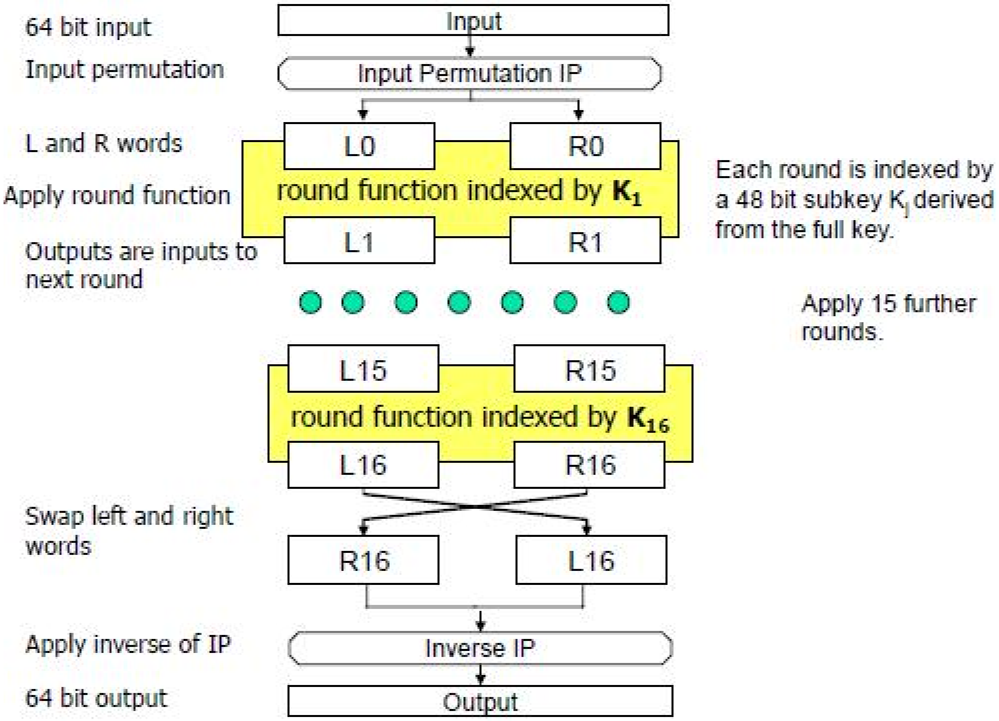
Instructor: Goutam Paul *Scribe: Ayan Das*

1 DATA ENCRYPTION STANDARD (DES)

The Data Encryption Standard was once a predominant symmetric-key algorithm for the encryption of electronic data. It was highly influential in the advancement of modern cryptography in the academic world.Several key points regarding DES are:-
1. The Data Encryption Standard is the most controversial cipher in history.
2. Developed on behalf of the US Govt.
3. Based on previous IBM work.
4. Issued in 1976 as FIPS 46.
5. 56 bit key (64 in fact but there are check bits).
6. Generally has been first choice in commerce (e.g. banking)

2 How DES works

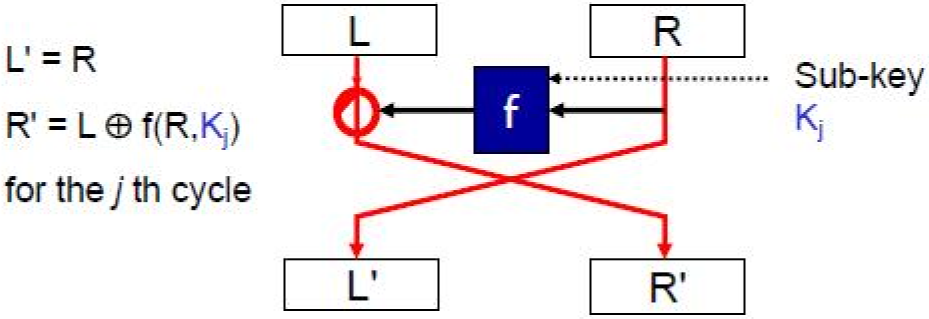
DES is an algorithm that takes a fixed-length string of plaintext bits and transforms it through a series of complicated operations into another ciphertext bitstring of the same length. In the case of DES, the block size is 64 bits. DES also uses a key to customize the transformation, so that decryption can only be performed by those who know the particular key used to encrypt. The key consists of 64 bits; however, only 56 of these are actually used by the algorithm. Eight bits are used solely for checking parity, and are thereafter discarded. Hence the effective key length is 56 bits.


64 bit input
Input
Input permutation
Input Permutation IP
L and R words
L0
R0
Apply round function
round function indexed by $\mathbf{K_1}$
Outputs are inputs to next round
L1
R1
Each round is indexed by a 48 bit subkey K_i derived from the full key.
Apply 15 further rounds.
L15
R15
round function indexed by $\mathbf{K_{16}}$
L16
R16
Swap left and right words
R16
L16
Apply inverse of IP
Inverse IP
64 bit output
Output

2.1 The DES round

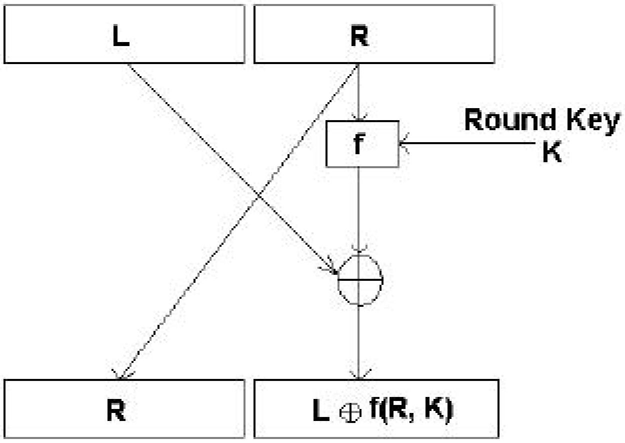
- Initial permutation simply permutes the 64 bits of input according to a specific pattern. It has no security significance.

- A round takes the form given below. The round operates on the left and right 32 bit words. The XOR is a bitwise XOR of 32 bit words.



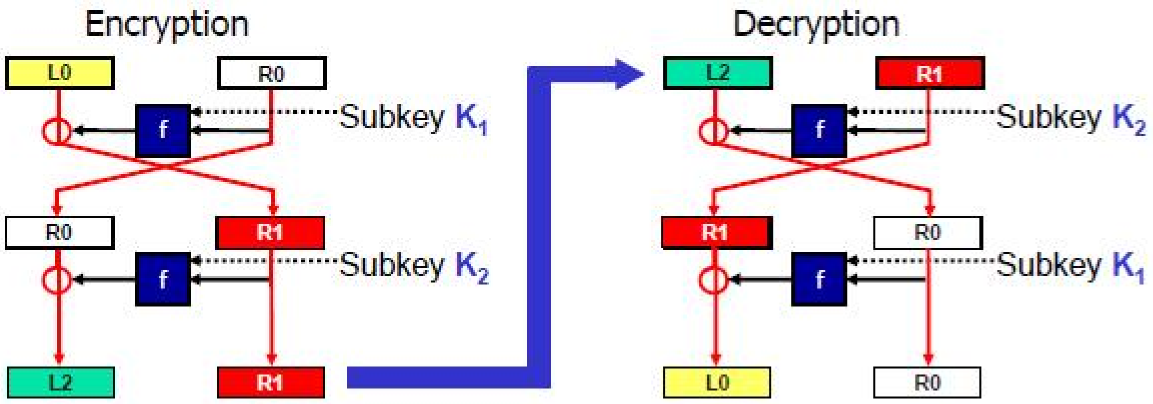
2.2 Fiestel Structure (reversible)

Feistel Cipher is not a specific scheme of block cipher. It is a design model from which many different block ciphers are derived. DES also use Feistel Cipher. A cryptographic system based on Feistel cipher structure uses the same algorithm for both encryption and decryption.



the right half of the block, R, goes through unchanged. But the left half, L, goes through an operation that depends on R and the encryption key. First, we apply an encrypting function f that takes two input the key K and R. The function produces the output f(R,K). Then, we XOR the output of the mathematical function with L.
Once the last round is completed then the two sub blocks, R and L are concatenated in this order to form the ciphertext block.

2.3 DES Decryption



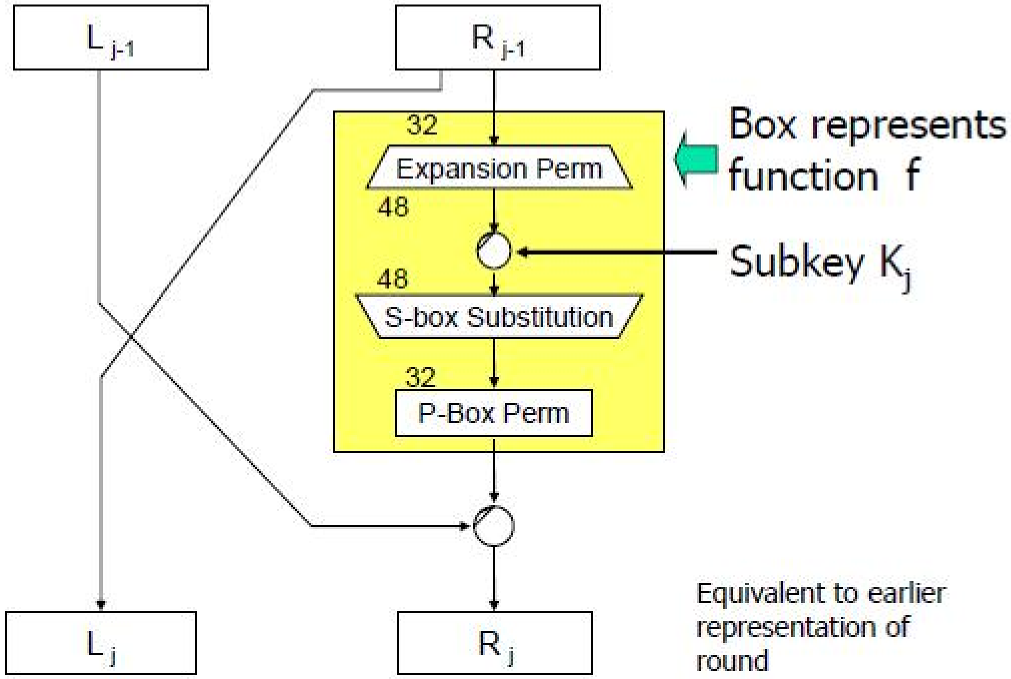
$L_j = R_{j-1}$ $R_j = L_{j-1} \oplus f(R_{j-1}, K_j)$ for the *j* th cycle (j>0)

Can apply same algorithm with subkeys in reverse order. Note final round is followed by swap. Illustrated below for two rounds. (IP omitted for space reasons)

2.4 One Round of DES

This block represent one of 16 round.
standard des operation on 64 bit blocks.

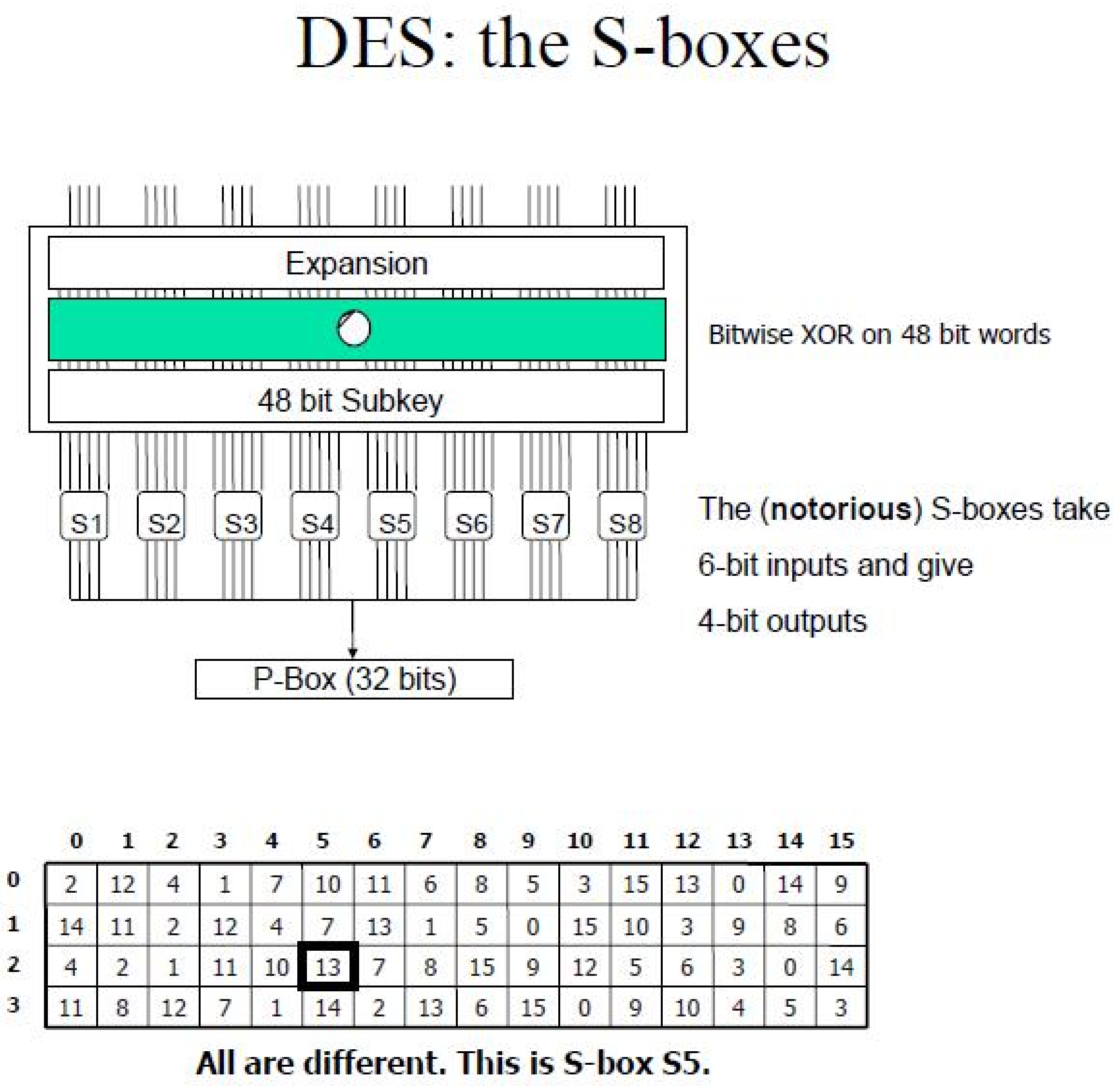
One Round of DES



Box represents function f

Equivalent to earlier representation of round

2.5 DES: the S-boxes



DES: the S-boxes

Expansion
48 bit Subkey
S1
S2
S3
S4
S5
S6
S7
S8
P-Box (32 bits)

Bitwise XOR on 48 bit words

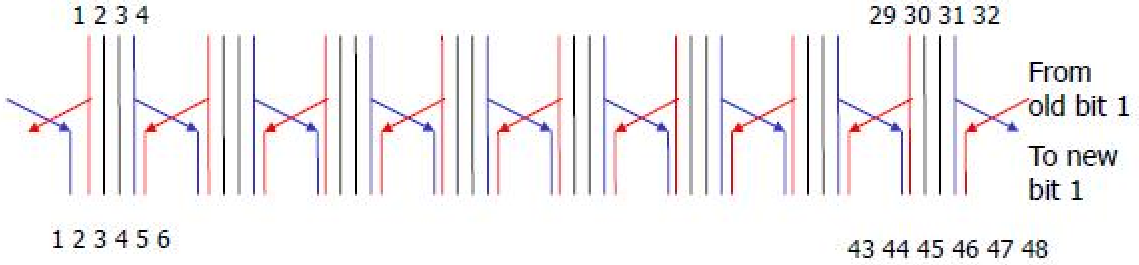
The (**notorious**) S-boxes take 6-bit inputs and give 4-bit outputs

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9
1	14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6
2	4	2	1	11	10	$\boxed{13}$	7	8	15	9	12	5	6	3	0	14
3	11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3

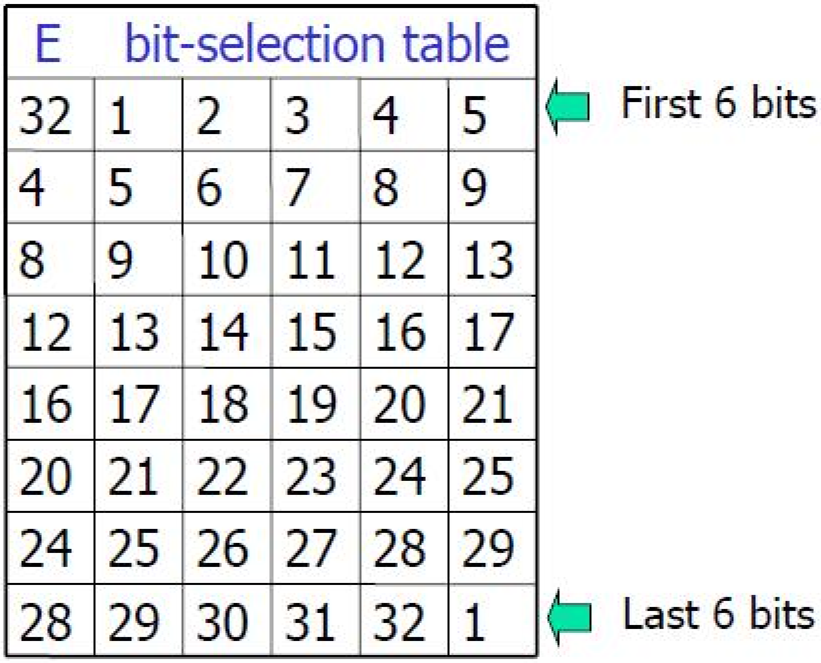
All are different. This is S-box S5.

Suppose we have an input of b1 b2 b3 b4 b5 b6 We interpret b1b6 as the binary for an integer in the range 0..3. This value indexes the row to use..
We interpret b2b3 b4b5 as the binary for an integer in the range 0..15. This value indexes the column to use..
Each entry is represented by a decimal and interpreted as a four bit binary. Thus 101010 corresponds to row 10=2, column 0101=5 and so indexes the value 13 = 1101

2.6 DES: Expansion



Each outer bit in a block of four influences the row used to do substitution in nearest neighbouring block of four (with wrap-around interpretation as shown).



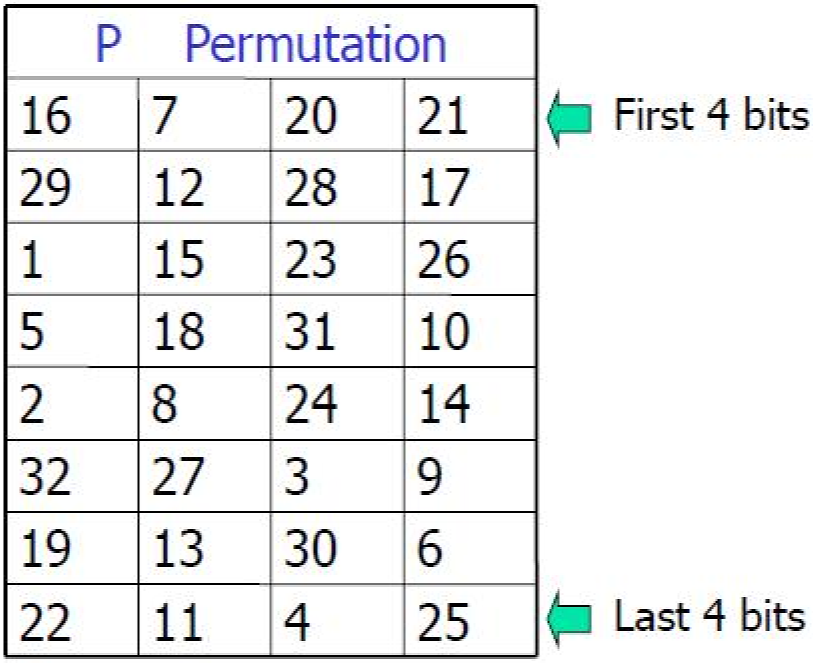
E bit-selection table					
32	1	2	3	4	5
4	5	6	7	8	9
8	9	10	11	12	13
12	13	14	15	16	17
16	17	18	19	20	21
20	21	22	23	24	25
24	25	26	27	28	29
28	29	30	31	32	1

First 6 bits
Last 6 bits

From the diagram it is clear that each of the outer bit is going to the next or privious neighbour .So each 4 bits is converting to 6 bits.

2.7 DES: P-Permutation

As shown in the figure below permutation is happening according to lookup table.
For example 1st bit of the 1st four bit is moved to 16th position.



P Permutation			
16	7	20	21
29	12	28	17
1	15	23	26
5	18	31	10
2	8	24	14
32	27	3	9
19	13	30	6
22	11	4	25

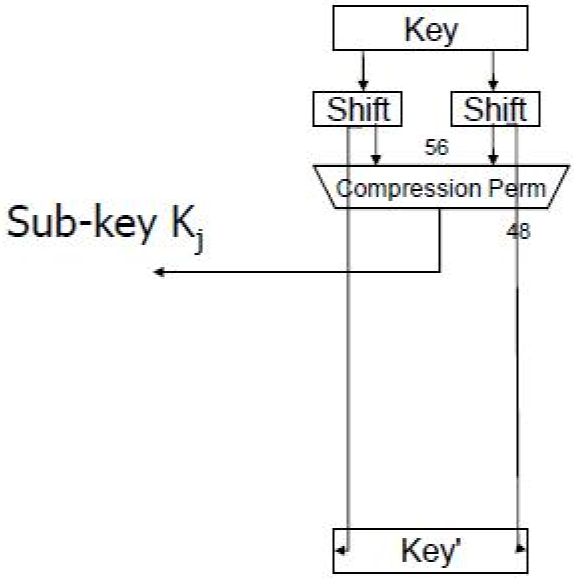
First 4 bits

Last 4 bits

2.8 DES:Keys

- Initial Key = 64 bit (k1k2k64)
- Last bit of every byte is the Parity bit.
- Such that $\sum_{i=1}^{8} k_{8k+i}$= 1 mod 2
- Possible Keys = 256 ’ ’ 7.2 x 1016
- Example Key in Hex : 1334 5779 9BBC DFF1

2.9 Key Schedule



The 56 bit key is split into two 28 bit halves.
Each half is subject to a cyclic shift of 1 or 2 bits in each round as follows (total 16 rounds).
1 1 2 2 2 2 2 2
1 2 2 2 2 2 2 1
After the shifts a sub-key is extracted from the resulting halves (details omitted)..
This is repeated for each round.

2.10 DES Semi-weak Keys

The block cipher DES has a few specific keys termed "weak keys" and "semi-weak keys". These are keys that cause the encryption mode of DES to act identically to the decryption mode of DES (albeit potentially that of a different key).

- Keys which cause the encryption mode of DES to act identically to the decryption mode of DES (potentially that of a different key).
- $E_{k1}(E_{k2}(M))$ = M
- 6 pairs of semi-weak keys

2.11 DES Cryptanalysis

Differential cryptanalysis (Biham and Shamir, 1991)

- 247 (plaintext, ciphertext) pairs.
- IBM (presumably from the NSA) knew about the method of attack 16 or more years before it was discovered and published by leading cryptography academics.

Linear cryptanalysis (Matsui, 1993).

- Reduces no of operations from 2^{56} (brute force) to around 2^{43}

Difference between two :-

- With differential cryptanalysis, the known plaintext/ciphertext pairs must be organized in pairs where both plaintexts differ by a specific difference ("difference" being a XOR, a subtraction... whatever works well algebraically with the algorithm at hand; for DES, this is XOR). The "approximation" is that this input difference will yield another specific difference on the output with a probability which is somewhat higher than what could be obtained with pure randomness, and the exact difference which will most probably appear depends on some of the key bits. Successful differential cryptanalysis normally requires chosen plaintext/ciphertext pairs (the attacker gets to choose the plaintexts or the ciphertexts).
- With linear cryptanalysis, the approximation is a linear formula (i.e. a bunch of XOR on bits) which links together some input bits, some output bits and some key bits, with a probability somewhat higher than what could be obtained with pure random. This "linear formula" works best (i.e. is fulfilled most often) when the hypothesis on the involved key bits is correct, so the analysis yields a few key bits.
- The most salient difference between linear and differential cryptanalysis is the known/chosen plaintext duality. For linear cryptanalysis, known random plaintexts are sufficient, but differential cryptanalysis requires chosen plaintexts, which, depending on the context, may or may not be a significant problem for the attacker.

2.12 DES Brute Force Attack

- Diffie and Hellman proposed a key search machine that would be able to break DES at a cost of doller 20M (1977).

- Detailed hardware design for key search machine by Wiener (1993).

- Cracked in 3 days (1998): Electronic Frontier Foundation, using purpose built hardware:.

- Less than 1 year to build.

- Less than doller 250,000.

- Now 2 hours on an FPGA.

2.13 DES Variants

- Triple DES = TDES.

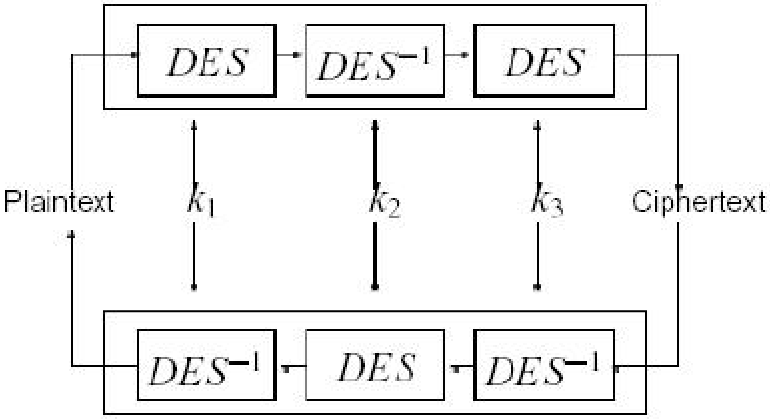
  In cryptography, Triple DES (3DES) is the common name for the Triple Data Encryption Algorithm (TDEA or Triple DEA) symmetric-key block cipher, which applies the Data Encryption Standard (DES) cipher algorithm three times to each data block.

- TDEA : NIST, Recommendation for the Triple Data Encryption Algorithm (TDEA) Block Cipher.

- There are variations of TDES which use
  Two different keys (2TDES) and
  Three different keys (3TDES)

Note the difference between double DES and 2 key triple DES. Double DES is the operation $DES2$(P):=$DES_{k2}(DES_{k1}(P))$, ie. two DES encryptions with two different keys. 2-key 3DES is the operation 2KEY-$DES3$(P):=$DES_k1(DES_{k2}^{-1}(DES_{k1}(P)))$, ie. 3DES where k1=k3.

2.14 Triple DES



2.15 Digression: Why not 2-DES?

- Suppose we have DES applied twice.
  i.e. C = $E_{k2}(E_{k1}$(M)) k1, k2 of size n = 56 bits.

- Apparently the security is 112 bits.

- Brute force requires:

Space : O(1)
Encryptions : $O(2^{2n})$

- Or is it so?

2.16 Meet in the middle attack

- Known plaintext attack.
  C = $E_{k2}(E_{k1}$(M)) : C and M known.

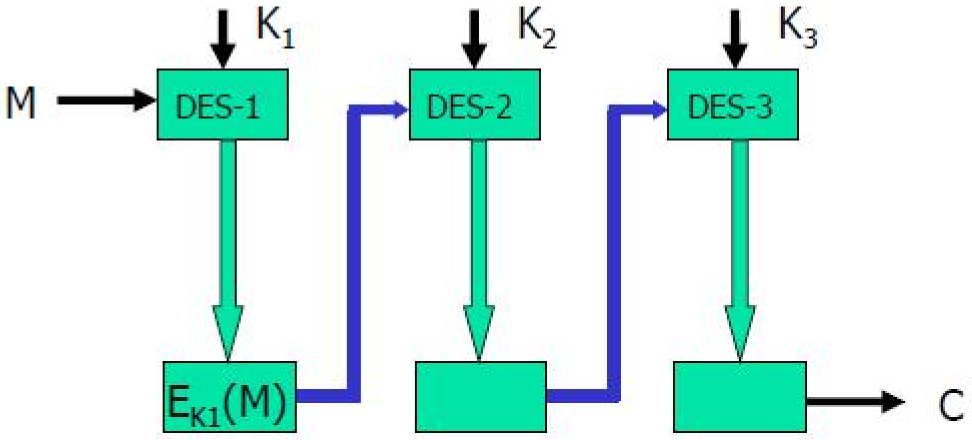
- Store C = E_k(M) for all possible k.

- Find P = D_k(C) for all possible k till it matches with some C.

- Attack requires :
  Space : O(2^n) Encryptions : O(2^{n+1})

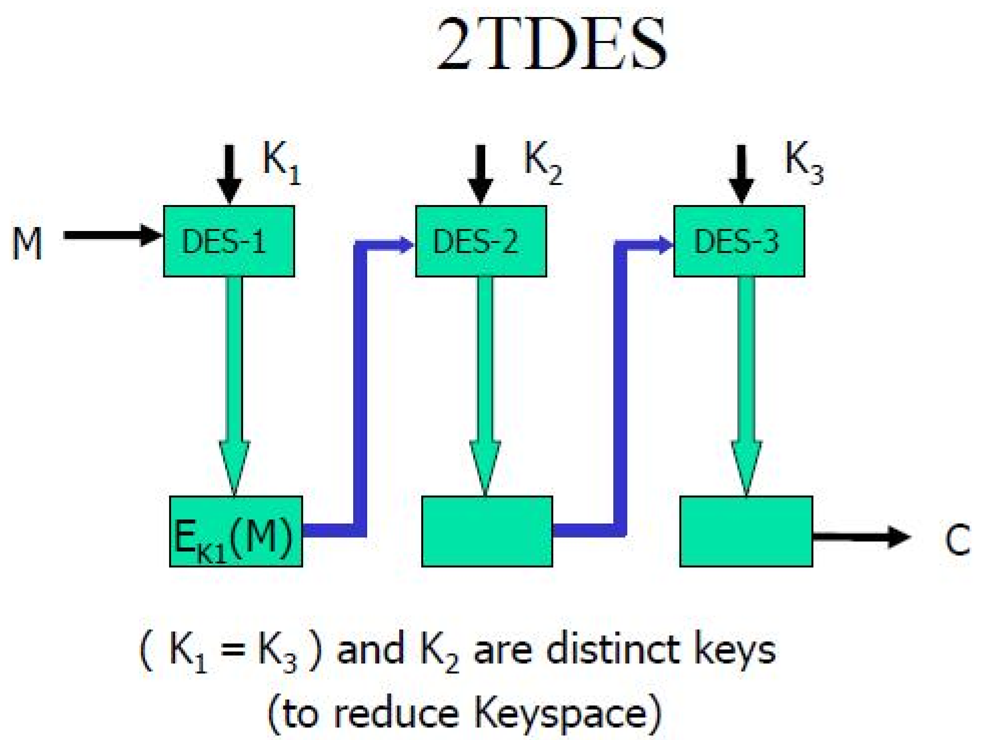
2.17 3TDES

3TDES



K_1, K_2 and K_3 are distinct keys

2.18 2TDES

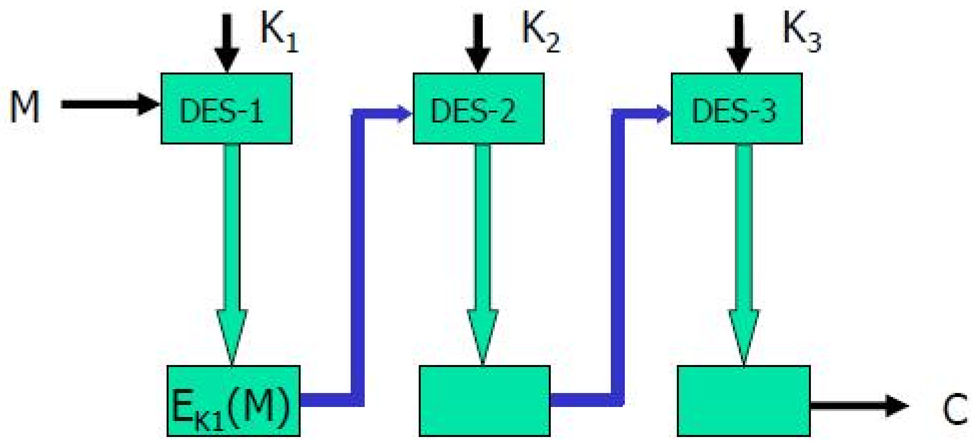


2TDES

($K_1 = K_3$) and K_2 are distinct keys
(to reduce Keyspace)

2.19 DES-EDE

DES-EDE



Encryption-Decryption-Encryption

What if K_1=K_2 ?

- C = $E_{K1}(D_{K2}(E_{K1}(P)))$
- Because encrypt and decrypt equivalent in security
- if K1=K2 then can work with single DES
- Standardized in ANSI X9.17 and ISO 8732

- No current known practical attacks

- 3 Keys version used in PGP, S/MIME

3 ADVANCED ENCRYPTION STANDARD (AES)

AES (acronym of Advanced Encryption Standard) is a symmetric encryption algorithm. The algorithm was developed by two Belgian cryptographer Joan Daemen and Vincent Rijmen. AES was designed to be efficient in both hardware and software.

3.1 History

- NIST wanted replacement for DES : 1997

- Requirements :-
  Block length : 128 bits
  Key lengths : 128, 192, 256 bits
  Available on Royalty free basis world-wide.

- 21 submissions : June 1998

- 15 met design criteria : AES Candidates
  1st AES Candidates Conference : Aug 1998

- March 1999 :
  2nd AES Candidates Conference

- August 1999 :
  5 Finalists

- Finalists :
  MARS, RC6, Rijndael, Serpent, Twofish

- April 2000
  3rd AES Candidates Conference

- October 2000
  Rijndael Selected !!!

- Feb 2001 :
  Draft FIPS for public review and comment

- Nov 2001 :
  Adoption as a standard

- December 2001 :
  Published as FIPS 197 in Federal Register

- Open and with International Flavor

- Authors from 12 different countries !

- Rijndael : Daemen and Rijnmen
  Belgian researchers

- 2nd Candiadates Conference held in Rome, Italy.

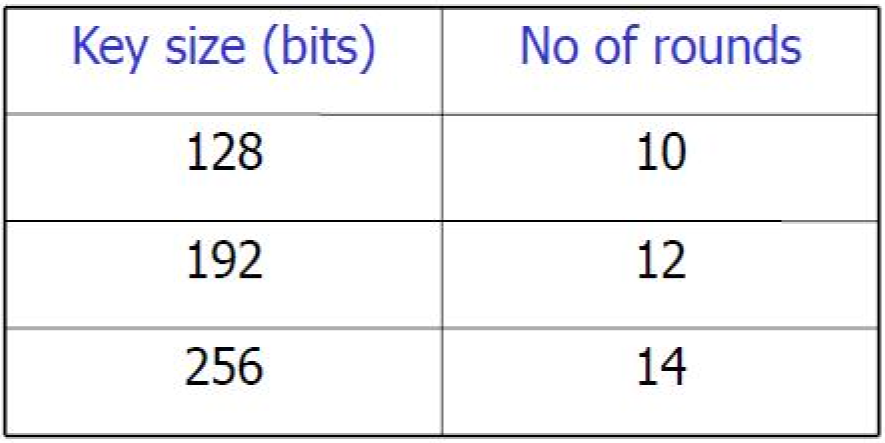
3.2 Evaluation Criteria of Candidates

- Security Absolutely essential Not secure == not to be considered further
- Cost Speed and memory for various implementations (Software, Hardware, Smart-Cards)
- Algorithm and implementation characteristics Flexibility and simplicity of the algorithm

3.3 AES Details

- Key sizes supported: 128, 192, 256 bits.
- Plain text block = 128 bits = 16 bytes.
- Cipher text block = 128 bits = 16 bytes.
- No of rounds Depend on key size

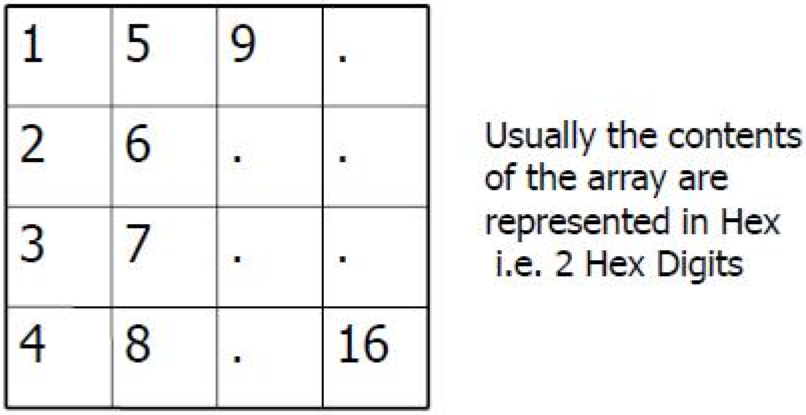
3.4 No of Rounds



Key size (bits)	No of rounds
128	10
192	12
256	14

3.5 State Array

Read 16 bytes plain-text, arrange in 4 x 4 matrix in column major form (State).



1	5	9	.
2	6	.	.
3	7	.	.
4	8	.	16

Usually the contents
of the array are
represented in Hex
i.e. 2 Hex Digits

3.6 AES Description

- Initialize state with plaintext x, perform add-round-key, xor round key with state
- Next r-1 rounds: sub-bytes, shift-rows, mix-columns, add-round-key
- Sub-bytes, shift-rows, add-round-key
- Generates ciphertext y

- Apply a Round transformation on state
- Total no of rounds : Already defined
- Each round has a Round Key, which is generated from the secret key
- A Key schedule is used to generate the Round Keys

3.7 Round Transformation

Each Round has 4 transformations :-
1. ByteSub
2. ShiftRow
3. MixColumn
4. AddRoundKey

3.8 One Round of Rijndael

```
Round(State, RoundKey){
ByteSub(State);
ShiftRow(State);
MixColumn(State);
AddRoundKey(State,RoundKey);
}
```

3.9 ByteSub Transformation

- All 32 bytes of State are replaced by their S-box values.
- S-box constructed in 2 steps :
  x maps to x^{-1} (with respect to the Field generated by an irreducible polynomial).
  Then apply an affine transformation Ax+b.

3.10 Addition in GF(2^8)

- Given 2 polynomials in GF(28)
- Addition is obtained by the coefficients summed modulo 2
- Example :
  Hex(57) = bin(01010111) = poly($x^6 + x^4 + x^2 + x + 1$)
  Hex(83) = bin(10000011) = poly(x^7+x+1)
  57 + 83 = D4
  $(x^6 + x^4 + x^2 + x + 1) + (x^7 + x + 1) = (x^7 + x^6 + x^4 + x^2)$

3.11 Multiplication in GF(28)

- Usual multiplication of polynomials modulo an Irreducible polynomial
- Example : Irreducible polynomial = 11B = $x^8 + x^4 + x^3 + x + 1$
  57 * 83 = C1

3.12 Affine Transformation

$$\begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \\ y_7 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \end{bmatrix} * \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}$$

3.13 ByteSub Transformation

- The steps may look complicated.

- For programming the transformation, you dont need to know so much maths.
- AES S-Box Table is available.
- Byte array of 256 locations.

3.14 ShiftRow Transformation

- Rows of the State are shifted by some bytes in a cyclic order.
- Row i shifted by Ci bytes.
- $C_1 = 0$, $C_2 = 1$, $C_3 = 2$, $C_4 = 3$.

3.15 MixColumn Transformation

- All 4 columns of the State are changed to new columns.
- Each column (cubic polynomial) is multiplied by c(x) and then modulo (x4+1) is taken.
- c(x) = 03 x^3 + 01 x^2+ 01 x + 02.

3.16 AddRoundKey Transformation

The State is XORed with the Round Key.

3.17 Inverse of a Round of Rijndael

InvRound(State, RoundKey)
AddRoundKey(State,RoundKey);
InvMixColumn(State);
InvShiftRow(State);
InvByteSub(State);

3.18 InvByteSub Transformation

- Each byte of the State is replaced by its Inverse S-box.
- InvS-box(x) = (i) $x \rightarrow A^{-1}(x + b)$ (ii) $x \Rightarrow x - 1$
- Inverse S-box(x) = y if S-box(y) = x

3.19 InvMixColumn Transformation

- All 4 columns of the State are changed to new columns.
- Each column (cubic polynomial) is multiplied by d(x) and then modulo (x4+1) is taken.
- d(x) = 0B x^3+ 0D x^2+ 09 x + 0E.
- (c(x) . d(x) = 1)

3.20 InvShiftRow

- Undoes the effect of ShiftRow.
- Easy to figure out.
- Rows of the State are shifted by some bytes in a cyclic order.
- Row i shifted by Ci bytes.
- $C_1' = 0, C_2' = 3, C_3' = 2, C_4' = 1$.

3.21 Key Schedule of Rijndael

- 32 byte Cipher Key expanded to 32*15 bytes.
- 14 Rounds + 1 Initial XOR operation requires 15 Round Keys.
- Chunks of 32 bytes from the Expanded Key form the Round Keys.

4 Mode of Operation

4.1 What is Initialization vector

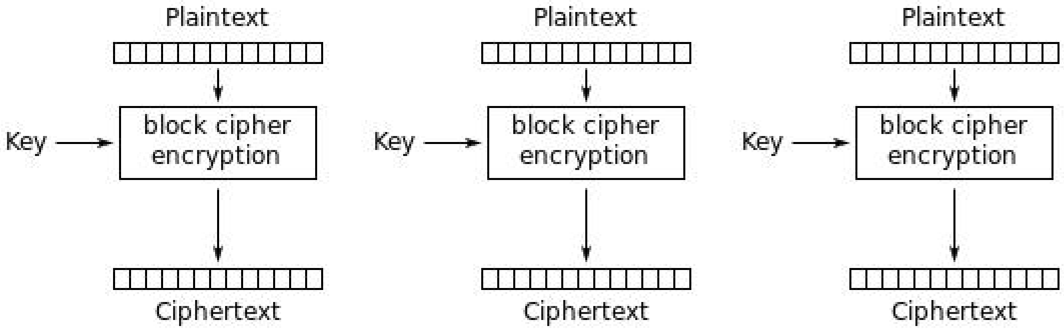
An initialization vector (IV) or starting variable (SV)[5] is a block of bits that is used by several modes to randomize the encryption and hence to produce distinct ciphertexts even if the same plaintext is encrypted multiple times, without the need for a slower re-keying process.

4.2 Padding

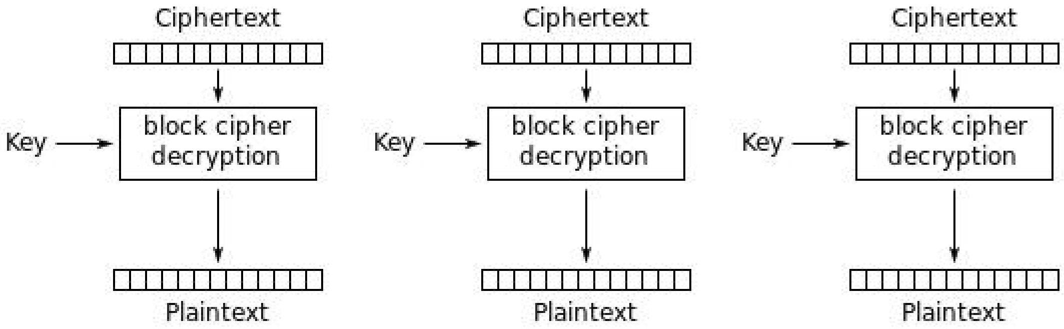
A block cipher works on units of a fixed size (known as a block size), but messages come in a variety of lengths. So some modes (namely ECB and CBC) require that the final block be padded before encryption. Several padding schemes exist. The simplest is to add null bytes to the plaintext to bring its length up to a multiple of the block size.

4.3 Electronic codebook mode

In this mode The message is divided into blocks, and each block is encrypted separately. The disadvantage of this method is that identical plaintext blocks are encrypted into identical ciphertext blocks; thus, it does not hide data patterns well. In some senses, it doesn't provide serious message confidentiality, and it is not recommended for use in cryptographic protocols at all.

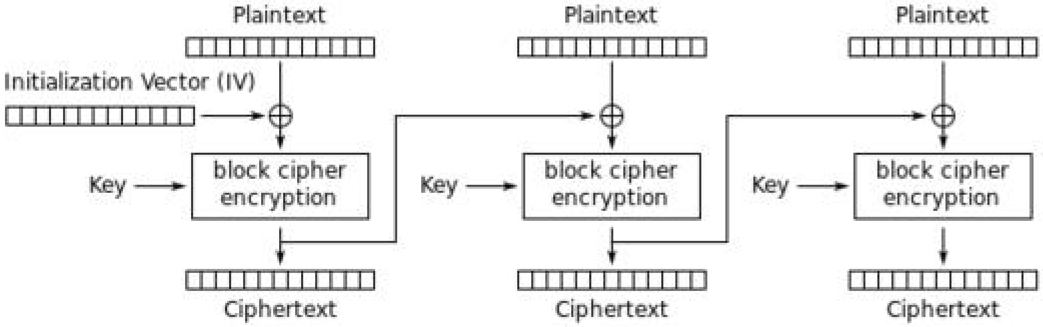


Electronic Codebook (ECB) mode encryption

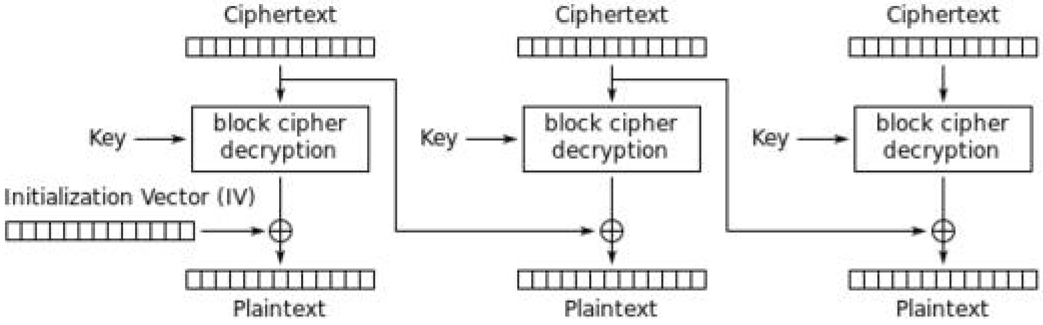


4.4 Cipher Block Chaining

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, The First block is IV.



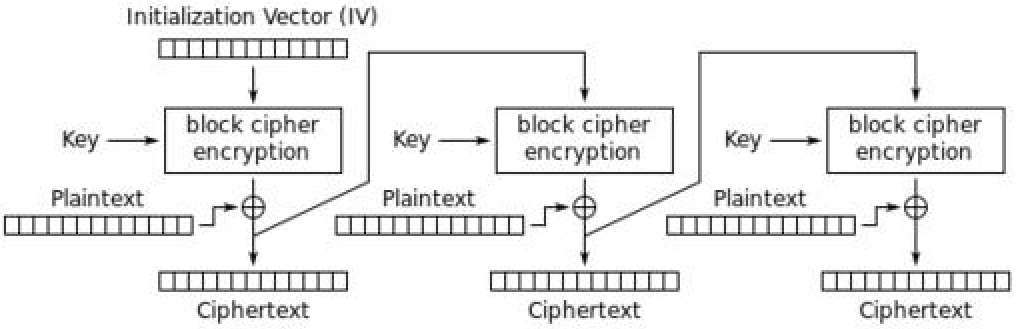
Cipher Block Chaining (CBC) mode encryption



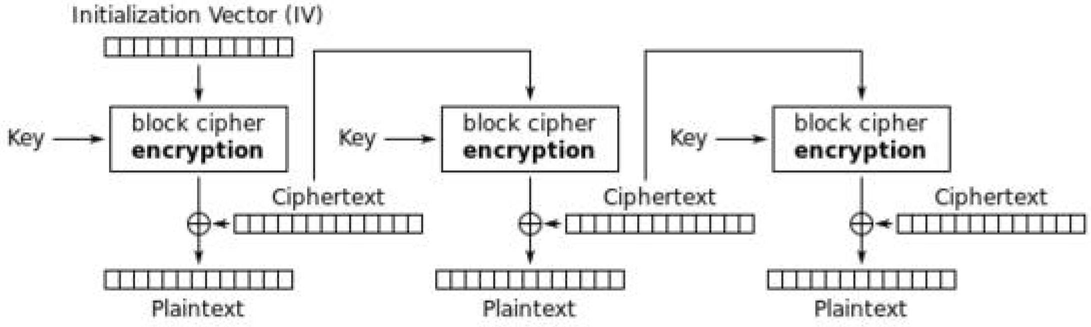
The formula for CBC encryption is :
$C_i = E_k(P_i \oplus C_{i-1})$ where $C_o = IV$
and the formula for decryption is:
$P_i = D_k(C_i) \oplus C_{i-1}$ where $C_o = IV$
CBC has been the most commonly used mode of operation. Its main drawbacks are that encryption is sequential (i.e., it cannot be parallelized), 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 IV affects all following ciphertext blocks

4.5 Cipher Feedback

The Cipher Feedback (CFB) mode, a close relative of CBC, makes a block cipher into a self-synchronizing stream cipher. Operation is very similar; in particular, CFB decryption is almost identical to CBC encryption performed in reverse:



Cipher Feedback (CFB) mode encryption



Cipher Feedback (CFB) mode decryption

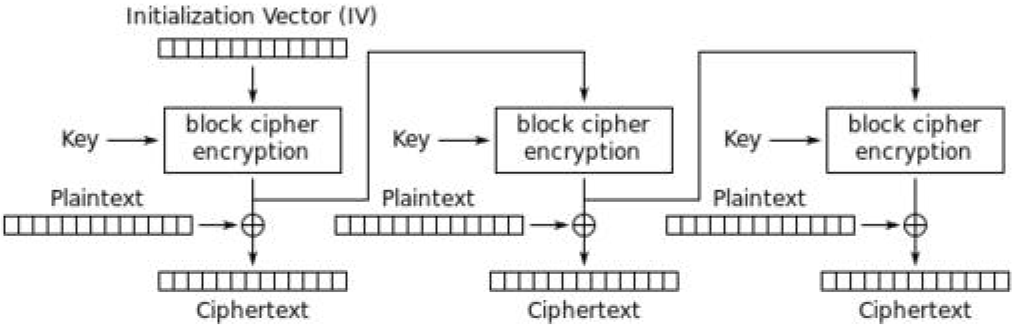
$C_i = E_k(C_{i-1}) \oplus P_i$

$P_i = D_k(C_{i-1}) \oplus C_i$ where $C_o = IV$

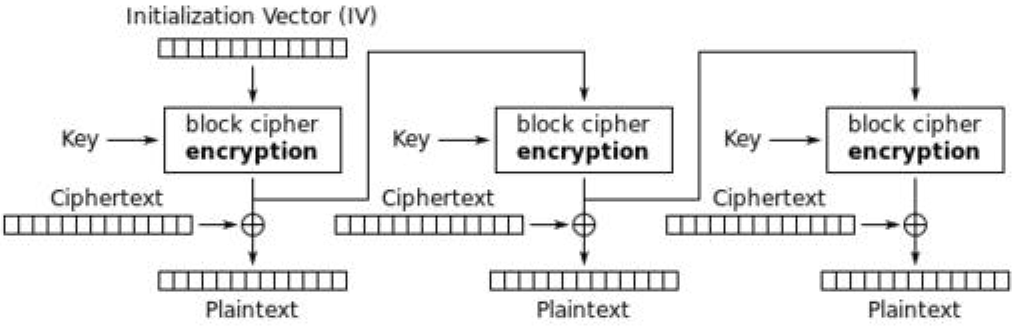
By definition of self-synchronising cipher, if part of the ciphertext is lost (e.g. due to transmission errors), then receiver will lose only some part of the original message (garbled content), and should be able to continue correct decryption after processing some amount of input data. This simplest way of using CFB described above is not any more self-synchronizing than other cipher modes like CBC. Only if a whole blocksize of ciphertext is lost both CBC and CFB will synchronize.but losing only a single byte or bit will permanently throw off decryption. To be able to synchronize after the loss of only a single byte or bit, a single byte or bit must be encrypted at a time. CFB can be used this way when combined with a shift register as the input for the block cipher

4.6 Output Feedback

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. Because of the symmetry of the XOR operation, encryption and decryption are exactly the same:
$C_j = P_j \oplus O_i$
$P_j = C_j \oplus O_i$
$O_j = E_k(I_j)$
$I_j = O_{j-1}$
$I_0 = IV$



Output Feedback (OFB) mode encryption


Initialization Vector (IV)
Key
block cipher **encryption**
Key
block cipher **encryption**
Key
block cipher **encryption**
Ciphertext
Ciphertext
Ciphertext
Plaintext
Plaintext
Plaintext

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.