UNIVERSITA' DEGLI STUDI DI PARMA
Dipartimento di Ingegneria dell'Informazione

Secret Key (symmetric) Cryptography

Luca Veltri

(mail.to: luca.veltri@unipr.it)

Corso di Sicurezza nelle reti di telecomunicazioni, a.a. 2004/2005

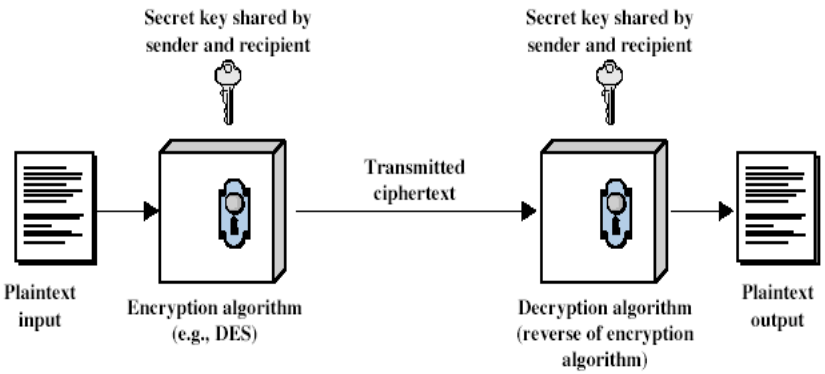
http://www.tlc.unipr.it/veltri

Symmetric Encryption

- Or conventional / private-key / single-key
- Sender and recipient share a common key
- All classical encryption algorithms are private-key
- Was the only type prior to invention of public-key in 1970's

- Secret key cryptographic systems are designed to take a reasonable-length key (e.g. 64 bits) and generating a one-to-one mapping that "looks like completely **random**", to someone doesn't know the key

Symmetric Cipher Model



Symmetric Cipher Model

- two requirements for secure use of symmetric encryption:
  - **a strong encryption algorithm**
  - **a shared secret key known only to sender / receiver**
- plaintext = m
- ciphertext = $c = E_k(m)$
- decrypted plaintext = $D_k(c) = D_k(E_k(m)) = m$

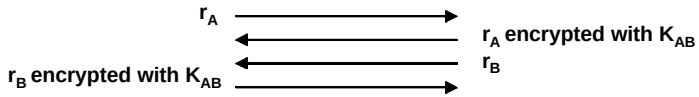
- assume encryption algorithm is known
- implies a secure channel to distribute key

Symmetric Cipher Characteristics

- Richiede una fase iniziale in cui ciascuna coppia di interlocutori si scambia la secret key in maniera sicura
- Il numero delle chiavi per realizzare una comunicazione reciproca tra N utenti (dispositivi) è pari a Nx(N-1)/2 (se le chiavi rimangono sempre le stesse)
- Viene generalmente utilizzato per proteggere, mediante codifica, informazioni (file) in un repository locale o trasmessi
- La robustezza dell'algoritmo è normalmente misurata dalla lunghezza delle chiavi: 40 bit (debole), 128 bit (forte)
- Algoritmi più diffusi: DES, 3DES, RC2, RC4, IDEA, AES

Security uses of secret key cryptography

- Transmitting over an insecure channel
  - **the two parties agree on a shared secret key and use secret key cryptography to send messages**
- Secure storage on insecure media
  - **the user uses a secret key to store and retrieve data on an (insecure) media**
- Authentication
  - **strong authentication through a challenge-response mechanism**



- Integrity check
  - **generating a fixed-length cryptographic checksum associated with a message (Message Integrity Code - MIC)**

Difetti dei sistemi simmetrici

- Metodo per scambio di chiavi assente
- Numero di chiavi troppo grande per gestire dei segreti condivisi tra singoli utenti
  [n(n-1)]/2

Classical Substitution Ciphers

- where letters of plaintext are replaced by other letters or by numbers or symbols
- or if plaintext is viewed as a sequence of bits, then substitution involves replacing plaintext bit patterns with ciphertext bit patterns

Crittografia classica con sostituzione: esempi

Cifrario con shift

Chiave $K \in \{0,1,\ldots,25\}$

$X_i \leftarrow M_i + K \bmod 26$

Cifrario monoalfabetico

Chiave

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
O	C	T	M	B	W	L	A	K	J	D	X	I	N	E	Y	S	U	P	F	Z	R	Q	H	V	G

testo in chiaro: C A S A

testo cifrato: T O P O

Caesar Cipher

- earliest known substitution cipher
- by Julius Caesar
- first attested use in military affairs
- replaces each letter by 3rd letter on
- example:

```
meet me after the toga party
PHHW PH DIWHU WKH WRJD SDUWB
```


Caesar Cipher (cont.)

- can define transformation as:

```
a b c d e f g h i j k l m n o p q r s t u v w x y z
D E F G H I J K L M N O P Q R S T U V W X Y Z A B C
```

- mathematically give each letter a number

```
a b c d e f g h i j k  l  m
0 1 2 3 4 5 6 7 8 9 10 11 12
n  o  p  q  r  s  t  u  v  w  x  y  z
13 14 15 16 17 18 19 20 21 22 23 24 25
```

- then have Caesar cipher as:

  $C = E(p) = (p + k) \bmod (26)$

  $p = D(C) = (C - k) \bmod (26)$

Monoalphabetic Cipher

- rather than just shifting the alphabet
- could shuffle (jumble) the letters arbitrarily
- each plaintext letter maps to a different random ciphertext letter
- hence key is 26 letters long

```
Plain:  abcdefghijklmnopqrstuvwxyz
Cipher: DKVQFIBJWPESCXHTMYAUOLRGZN

Plaintext:  ifwewishtoreplaceletters
Ciphertext: WIRFRWAJUHYFTSDVFSFUUFYA
```

Monoalphabetic Cipher Security

- now have a total of 26! = 4 x 1026 keys
- with so many keys, might think is secure
- but would be WRONG!!
- problem is language characteristics
  - **letter frequencies**
  - **most common words**
  - **two letters frequencies (e.g. "th" in english)**
  - **etc.**

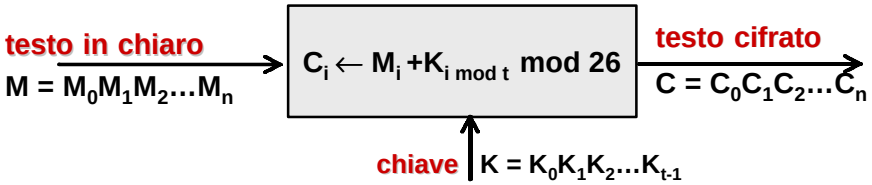
Polyalphabetic Ciphers

- another approach to improving security is to use multiple cipher alphabets
- called **polyalphabetic substitution ciphers**
- makes cryptanalysis harder with more alphabets to guess and flatter frequency distribution
- use a key to select which alphabet is used for each letter of the message
- use each alphabet in turn
- repeat from start after end of key is reached

Vigenère Cipher

- simplest polyalphabetic substitution cipher is the **Vigenère Cipher** [1586]
  - **(Blaise de Vigenère, 1523-1596)**
- effectively multiple caesar ciphers
- key is multiple letters long K = k1 k2 ... kd
- i^{th} letter specifies i^{th} alphabet to use
- use each alphabet in turn
- repeat from start after d letters in message
- decryption simply works in reverse

Vigenère Cipher (cont.)



Esempio

Testo in chiaro: CODICE MOLTO SICURO
Chiave: REBUS

```
      CODIC   EMOLT   OSICU   RO   testo in
chiaro
      REBUS   REBUS   REBUS   RE   chiave
      TSECU   VQPFL   FWJWM   IS   testo
cifrato
```

Cryptanalysis of Vigenère Ciphers

- have multiple ciphertext letters for each plaintext letter
- hence letter frequencies are obscured
- but not totally lost
- start with letter frequencies
  - **see if look monoalphabetic or not**
- if not, then need to determine number of alphabets, since then can attach each

One-Time Pad

- if a truly random key as long as the message is used, the cipher will be secure
- called a One-Time pad
- Advantages:
  - **is unbreakable since ciphertext bears no statistical relationship to the plaintext, and**
  - **for** any plaintext **&** any ciphertext **there exists a key mapping one to other**
- Disadvantages:
  - **can only use the key** once
  - **have problem of safe distribution of key**

Transposition Ciphers

- another tecniquie is that used by classical **transposition** or **permutation** ciphers
- these hide the message by rearranging the letter order (blocks fo bits)
- without altering the actual letters used
- can recognise these since have the same frequency distribution as the original text

Example: Row Transposition Ciphers

- a more complex scheme
- write letters of message out in rows over a specified number of columns
- then reorder the columns according to some key before reading off the rows

```
Key:        4 3 1 2 5 6 7
Plaintext:  a t t a c k p
            o s t p o n e
            d u n t i l t
            w o a m x y z
Ciphertext: TTNAAPTMTSUOAODWCOIXKNLYPETZ
```

Product Ciphers

- Ciphers using substitutions or transpositions are not secure because of language characteristics
- Hence consider using several ciphers in succession to make harder:
  - **two substitutions make a more complex substitution**
  - **two transpositions make more complex transposition**
  - **but a substitution followed by a transposition makes a new much harder cipher**
- This is bridge from classical to modern ciphers

Rotor Machines

- Before modern ciphers, rotor machines were most common product cipher
- Were widely used in WW2
  - **German Enigma, Allied Hagelin, Japanese Purple**
- Implemented a very complex, varying substitution cipher
- Used a series of cylinders, each giving one substitution, which rotated and changed after each letter was encrypted
- with 3 cylinders have $26x26x26=26^3=17576$ alphabets

Steganography

- an alternative to encryption
- hides existence of message
  - **using only a subset of letters/words in a longer message marked in some way**
  - **using invisible ink**
  - **hiding in graphic image or sound file**
- has drawbacks
  - **high overhead to hide relatively few info bits**

Block and Stream Ciphers

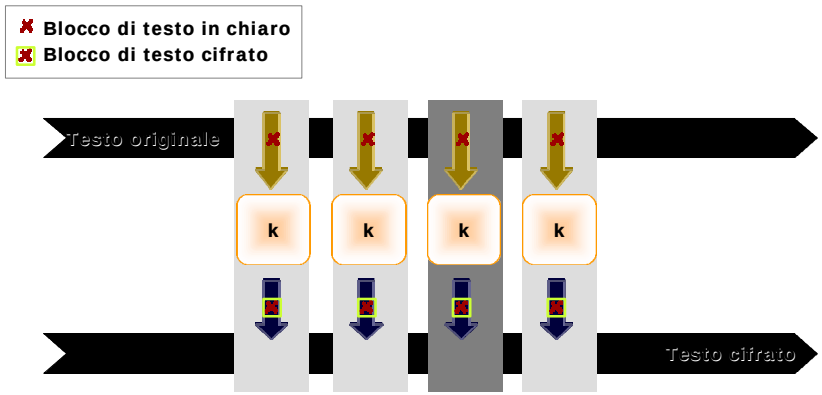
Block and Stream Ciphers

- There are two basic cipher structures
  - **Block ciphers**
    - Block ciphers process messages in into blocks, each of which is then en/decrypted
    - Plaintext is treated as a sequence of n-bit blocks of data
    - Ciphertext is same length as plaintext
    - Like a substitution on very big characters (64-bits or more)
    - Can be made to behave as a stream cipher
  - **Stream ciphers**
    - Stream ciphers process messages (Encryption/Decryption) a bit or byte at a time when en/decrypting
    - Often easier to analyze mathematically
- many current ciphers are block ciphers

Block Ciphers

- A cryptographic algorithm convert a plaintext block into an encrypted block
- How long should the plaintext block be?
  - **having block length too short (say one byte as in monoalphabetic cipher), it could be easier to costruct a decryption table starting from some <plaintext,ciphertext> pairs**
  - **having block length too long, it could be inconvenient due to the increasing of complexity**
- 64bit blocks are often used
  - **it is difficult to obtain all 2^{64} pairs(known plaintext attack)..**

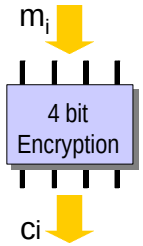
Example of Block Ciphers



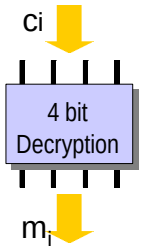
Block Ciphers

- block ciphers look like an extremely large substitution
- If 64bit blocks are used, 2^{64} possible input values are mapped to 2^{64} output values
- The most general way of encrypting could be to specify completely the mapping table
  - **would need table of 2^{64} entries for a 64-bit block $\rightarrow$ 2^{70} bits!**
  - **it is too long for a key.. ;-)**
  - **instead create from smaller building blocks**

Esempio di crittografia a blocchi di 4 bit



Plaintext	Ciphertext
0000	1110
0001	0100
0010	1101
0011	0001
0100	0010
0101	1111
0110	1011
0111	1000
1000	0011
1001	1010
1010	0110
1011	1100
1100	0101
1101	1001
1110	0000
1111	0111



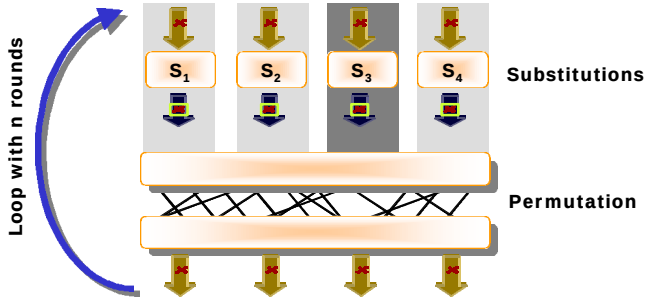
Ciphertext	Plaintext
0000	1110
0001	0011
0010	0100
0011	1000
0100	0001
0101	1100
0110	1010
0111	1111
1000	0111
1001	1101
1010	1001
1011	0110
1100	1011
1101	0010
1110	0000
1111	0101

Block Ciphers

- Two kind of simple transformations are used:
  - **substitutions**
    - specifies for each of the 2^k possible input values the k-bit output
    - this is not practical for 64-bit blocks, but is possible for lower length bloks (e.g. 8bits)
    - to specify a substitution on k-bit blocks, $k \cdot 2^k$ bits are required
  - **permutations**
    - specifies for each of the k input bits the corresponding output position
    - a permutation is a special case of substitution in which each bit of the output gets its value from exactly one bit of the input
    - to specify a permutation on k-bit blocks, $k \cdot \log_2 k$ bits are required

Block Ciphers

- One possible way to build a secret key algorithm is
  - **to break the input into managed-sized chunks (say 8 bits),**
  - **do a substitution on each small chunk,**
  - **and then take the output of all the substitutions and run them through a permuter (big as the input)**
  - **the process is repeated, so that each bit winds up as input to each substitution**
  - **Each time is called *round***



Substitution-Permutation Ciphers

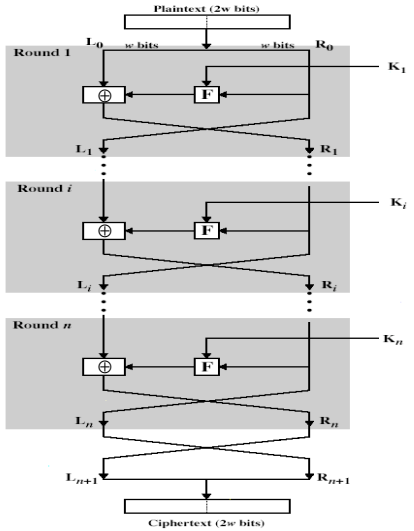
- Cipher needs to completely obscure statistical properties of original message
  - **a one-time pad does this**
- in 1949 Claude Shannon introduced idea of substitution-permutation (S-P) networks
- S-P networks are based on the two primitive cryptographic operations:
  - **substitution**
  - **permutation**
- these form the basis of modern block ciphers
- provide confusion and diffusion of message
  - **confusion**
    - makes relationship between ciphertext and key as complex as possible
  - **diffusion**
    - dissipates statistical structure of plaintext over bulk of ciphertext
    - every single plaintext cipher will influence several ciphertext ciphers
- the operation must be reversible!

Feistel Cipher Structure

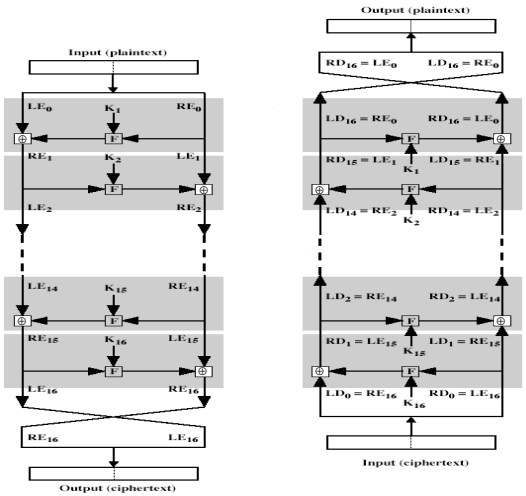
- Horst Feistel devised the **cipher**
  based on concept of invertible product cipher
-
  - **process through multiple rounds which perform a substitution on left data half**
  -
  - **then have permutation swapping halves**

implements Shannon's substitution-permutation network concept

Feistel Cipher Structure



Feistel Cipher Decryption



Design Principles

Block Ciphers are defined in terms of

- **block size**
  - **increasing size improves security, but slows cipher**
- **key size**
  - **increasing size improves security, makes exhaustive key searching harder, but may slow cipher**
- **number of rounds**
  - **increasing number improves security, but slows cipher**
- **subkey generation**
  - **greater complexity can make analysis harder, but slows cipher**
- **round function**
  - **greater complexity can make analysis harder, but slows cipher**

Two other considerations

- **fast software en/decryption & ease of analysis**
  - **are more recent concerns for practical use and testing**

Data Encryption Standard (DES)

Data Encryption Standard (DES)

- Most widely used block cipher in world
- Published in 1977 by National Bureau of Standards (now NIST) for use in commercial and unclassified U.S. Government applications
- FIPS PUB 46-3 (Federal Information Processing Standards PUBlication)
  - **U.S. Dept. OF Commerce/NIST (National Institute of Standards and Technology)**
- Has widespread use
- Has been considerable controversy over its security

DES History

- Based on an algorithm known as *Lucifer cipher* (1971)
  - **by an IBM team led by Horst Feistel**
  - **used 64-bit data blocks with 128-bit key**
- then redeveloped as a commercial cipher with input from NSA and others
- in 1973 NBS issued request for proposals for a national cipher standard
- IBM submitted their revised Lucifer which was eventually accepted as the DES
- DES has become widely used, esp in financial applications
- in 1999 NIST published a new version called *triple DES* (*3DES*) or *TDEA*

Data Encryption Standard (DES)

- DES encrypts 64-bit data using 56-bit key
- It consists of
  - **initial permutation of the 64 bits**
  - **16 identical "rounds" of operation where the data is confused and diffused with the key and the previous round**
  - **A final permutation**
- DES can be efficiently implemented in harware
- Relatively slow if implemented in software
  - **this was not a documented goal**
  - **..however people have asserted that it was designed with this in mind**

DES Design Controversy

- although DES standard is public
- was considerable controversy over design
  - **in choice of 56-bit key (vs Lucifer 128-bit)**
  - **and because design criteria were classified**
- subsequent events and public analysis show design was appropriate

Why 56bits keys?

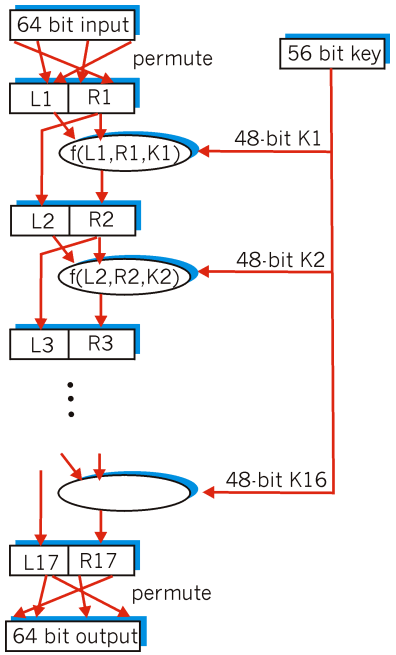
- yes.. 8x7 bits + 8 bits for parity check..
- however, 8 bits for parity check are too small
  - **64 bits of garbage have 1 in 256 chance to look like a valid key**
- people have suggested that key length has been reduced from 64 to 56 to let DES to be broken (only) by the NSA (15 years ago..)

DES

- Uses a 64-bit key that is reduced to 56-bits for parity checking
  - **The 56-bit key is transformed in to 16 48-bit subkeys (one per round)**
- Transforms 64-bit blocks of input M to 64-bit blocks of output C
- Same algorithm for encryption and decryption (sub-keys are used in reverse order for decryption)

DES

- Consists of
  - **An initial permutation (P)**
  - **Key transformation**
  - **16 rounds of:**
    - the rightmost 32 bits of the input are moved to the left 32 bits of the output
    - Then a function f() is run on the left and right halves, and the key
    - The key is shifted for each round
  - **A final permutation (P^{-1})**
- Why permuting?
  - **mhmm..**

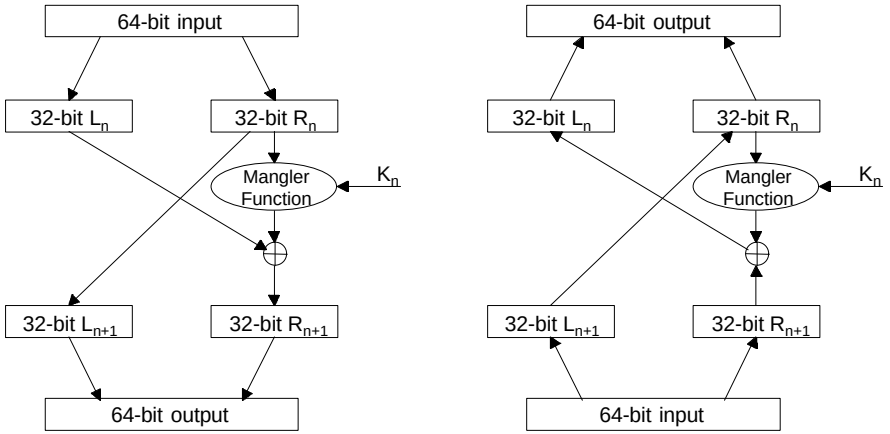


Initial Permutation IP

- first step of the data computation
- IP reorders the input data bits
- even bits to LH half, odd bits to RH half
- quite regular in structure (easy in HW)

- example:
  `IP(675a6967 5e5a6b5a) = (ffb2194d 004df6fb)`

DES Round



Encryption ***Decryption***

DES Round

- uses two 32-bit L & R halves
- as for any Feistel cipher can describe as:

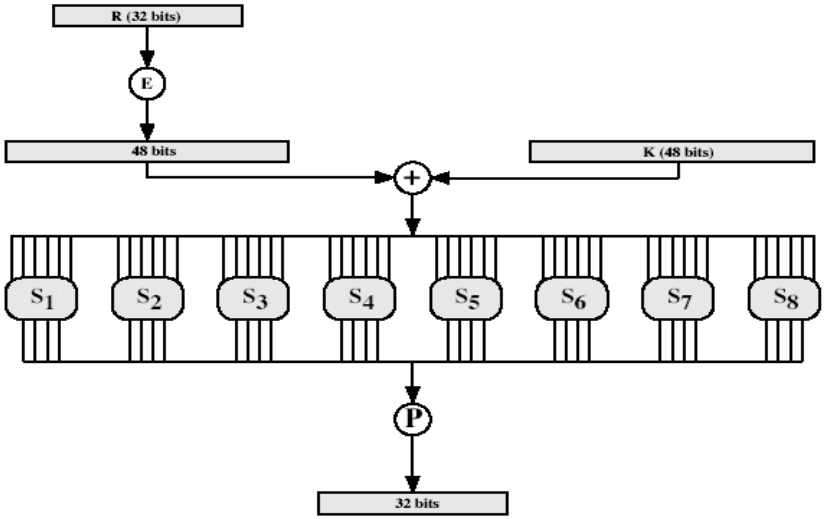
  $L_i = R_{i-1}$

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

- Mangle Function details
  - **Take 48 bits of the shifted key**
  - **Expand the right 32-bits of the data to 48 bits**
  - **XOR the two together, send it through "S-Box"**
  - **The S-BOX is a predefined substitution table**
  - **The S-BOX produces 32 new bits, which is XORed with the left half**

- Incredibly, this process is reversible

DES Round: Mangle Function



DES Round: Mangle Function

- Uses 8 (6x4) S-boxes
  - **6 input bytes yields 4 output bytes**
  - **each S-box is actually 4 little 4 bit boxes**
    - outer bits 1 & 6 select one of the 4 little boxes
    - inner bits 2-5 are substituted
    - result is 8 groups of 4 bits (32 bits)
  - **simply a lookup table of 8 x 4 rows and 16 columns**
    - 4 rows for each S-box
    - outer bits 1 & 6 (**row** bits) select one of the 4 rows
    - inner bits 2-5 (**col** bits) are substituted
- Example:
  - **Given bits 110011 as input and S-box 6 from DES**
    - Take first and last bits "11" to choose row 3
    - Take middle four bits "1001" to choose column 9
    - The value from S-box 6 of DES is 14 ("1110")
    - Substitute "110011" to "1110"
  - **Always count rows and columns from 0 not 1**

DES Round: S-boxes

S_1

14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13

S_2

15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10
3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5
0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15
13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9

S_3

10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8
13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1
13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7
1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12

S_4

7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15
13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9
10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4
3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14

S_5

2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9
14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6
4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14
11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3

S_6

12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11
10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8
9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	6
4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13

S_7

4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1
13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6
1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2
6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12

S_8

13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7
1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2
7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8
2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11

Example:

S(18 09 12 3d 11 17 38 39) = 5fd25e03

DES Decryption

- decrypt must unwind steps of data computation
- with Feistel design, do encryption steps again
- using subkeys in reverse order (SK16 … SK1)
- note that IP undoes final FP step of encryption
- 1st round with SK16 undoes 16th encrypt round
- ….
- 16th round with SK1 undoes 1st encrypt round
- then final FP undoes initial encryption IP
- thus recovering original data value

Avalanche Effect (Effetto valanga)

- key desirable property of encryption alg
  - **where a change of** one **input or key bit results in changing approx** half **output bits**
  - **making attempts to "home-in" by guessing keys impossible**
- DES exhibits strong avalanche

Strength of DES – Key Size

- 56-bit keys have $2^{56} = 7.2 \times 10^{16}$ values
- Originally complaints that the NSA fixed the S-boxes to provide a backdoor. This has never been found
  - **The S-boxes appear to be strong against even differential cryptanalysis (Which means the NSA knew about DC before 1978. It was first described publicly in 1990)**
- Algorithm has never been "broken"
  - **Successfully attacked by brute force**
- Recent advances have shown is possible to break by brute force
  - **in 1997 on Internet in a few months**
  - **in 1998 on dedicated HW ($250K) in a few days (Electronic Frontier Foundation)**
  - **in 1999 above (EFF) combined in 22hrs!**
  - **reasonable for a small business to buy**
- Still must be able to recognize plaintext

3-DES, IDEA, AES

Triple DES: Why?

- The keyspace of DES is too small
- Clear a replacement for DES was needed
  - **theoretical attacks that can break it**
  - **demonstrated exhaustive key search attacks**
- AES is a new cipher alternative
- Prior to this alternative was to use multiple encryption with DES implementations
- Triple-DES is the chosen form

Multiple Encryption DES

- A possible solution is to use the same encryption algorithm more times
- Both Encryption and Decryption algorithms can be see as encryption functions
- How many time should be performed? (2,3,4, 1000..)
- How many keys?
- What combination of E and D can be chosen? (EEE, ED, etc)

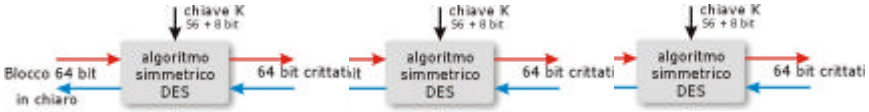
How many time should be performed?

- The more time the block is encrypted the more secure it is
- For computation, no more encryptions than are necessary
- Encrypting twice with the same key

  plaintext ---K---> ---K---> ciphertext
  - **no more secure that single encryption with K: exhaustive search requires trying 2^{56} keys**
- Encrypting twise with two keys

  plaintext ---K_1---> ---K_2---> ciphertext
  - **there is an attack (not very practical) that breaks doubling DES in roughly twice the time of a brute-force breaking single DES**
    - since $X = E_{K1}[P] = D_{K2}[C]$
    - attack by encrypting P with all keys and store
    - then decrypt C with keys and match X value
    - can show takes $0(2^{56})$ steps
- Triple encryption with two keys (EDE)

Triple DES (3-DES)



- **lunghezza blocco = 64 bit**
- **chiave (k, k´,k´´) lunga 56 + 56 + 56 = 168 bit**
- **lunghezza blocco = 64 bit**
- **chiave (k, k´) lunga 56+56 = 112 bit**
  - K_1 to E, K_2 to D, K_1 to E
- **spesso chiamato EDE (acronimo per Encrypt Decrypt Encrypt) or TDEA**
- **adottato negli standard X9.17 e ISO 8732**

Triple-DES with Two-Keys

- Use 2 keys K1 and K2 with E-D-E sequence
  - **$C = E_{K1}[D_{K2}[E_{K1}[P]]]$**
- A key space of 2^{112} possible keys
- Encrypt & decrypt are equivalent in security: there is no advantage to using decryption for the second stage
- however, if K1 = K2 we have backwards compatibility
  - **$E_{k1}(D_{k1}(E_{k1}(P))) == E_{k1}(P)$**
- Standardized in ANSI X9.17 & ISO8732
- No current known practical attacks

Triple-DES with Three-Keys

- Triple-DES with Three-Keys
  - **$C = E_{K3}[D_{K2}[E_{K1}[P]]]$**
- Although there are no practical attacks on two-key
- Has been adopted by some Internet applications, eg PGP, S/MIME

Altri cifrari

- IDEA (International Data Encryption Algorithm) [1990]
- SAFER (Secure And Fast Encryption Routine)
  SAFER K-64 [1994], SAFER K-128 [1995]
- RC5 [1995]

cifrario	bit chiave	bit testo
IDEA	128	64
SAFER K-64	64	64
SAFER K-128	128	64
RC5	<256 byte	32,64,128

- Madryga, NewDES, FEAL, REDOC, LOKI, Khufu, Knafre, RC2, MMB, GOST, Blowfish, …
- … AES

IDEA

- International Data Encryption Algorithm
  - **Used in PGP (Pretty Good Privacy)**
- Similar to DES
  - **Works on 64-bit input blocks**
    - Taken as 4 16-bit blocks
  - **Uses a 128-bit key**
    - Uses a total of 52 16-bit subkeys
      (17 rounds : 4 keys+2 keys+4 keys+ .. +4keys=52)
  - **operates in rounds (17 rounds)**
  - **complicated mangler function that does not have to be reversible (it is run in the same direction for both encryption/decryption as for DES)**
  - **Decryption uses same algorithm**
    - Different subkey generation
- Encryption/decryption keys (not as DES) are related in complex manner

Blowfish

- A symmetric block cipher designed by Bruce Schneier in 1993/94
- Characteristics
  - **fast implementation on 32-bit CPUs**
  - **compact in use of memory**
  - **simple structure eases analysis/implemention**
  - **variable security by varying key size (uses a 32 to 448 bit key)**
- Has been implemented in various products

AES

- Designed to replace DES
  - **Organized by NIST**
  - **Chosen from five candidate algorithms**
    - Reviewed by US government (NSA), industry and academia
    - Required a four-year process to pick the algorithm
    - Winning algorithm chosen 2 Oct 2000
    - Rijndael Block Cipher
      - Joan Daemon, Vincent Rijmen (Belgium)
    - Has become a NIST standard

AES

- Key sizes
  - **The AES will specify three key sizes: 128, 192 and 256 bits. In decimal terms, this means that there are approximately:**
    - 3.4 x 10^{38} possible 128-bit keys;
    - 6.2 x 10^{57} possible 192-bit keys; and
    - 1.1 x 10^{77} possible 256-bit keys.
  - **In comparison, DES keys are 56 bits long, which means there are approximately 7.2 x 10^{16} possible DES keys. Thus, there are on the order of 10^{21} times more AES 128-bit keys than DES 56-bit keys**
- Will the AES replace Triple DES and DES?
  - **The AES is being developed to replace DES, but NIST anticipates that Triple DES will remain an approved algorithm (for U.S. Government use)**

AES robustness

- Brute force attack
  - **In the late 1990s, specialized "DES Cracker" machines were built that could recover a DES key after a few hours. In other words, by trying possible key values, the hardware could determine which key was used to encrypt a message**
  - **If you could crack a DES key in one second (i.e., try 2^{56} keys per second), it would take 149 trillion years to crack a 128-bit AES key by brute force at the same speed**
  - **the universe is believed to be less than 20 billion years old**
  - **But, things change…**
- AES robustness
  - **No one can be sure how long the AES - or any other cryptographic algorithm - will remain secure**
  - **However, DES was a U.S. Government standard for approximately twenty years before it was known to be "cracked" by massive parallel computer**
  - **AES has the potential to remain secure well beyond twenty years**

Comparison of symmetric key algorithms

Speed Comparisons of Block Ciphers on a Pentium

Algorithm	Clock cycles per round	# of rounds	# of clock cycles per byte encrypted
Blowfish	9	16	18
RC5	12	16	23
DES	18	16	45
IDEA	50	8	50
Triple-DES	18	48	108

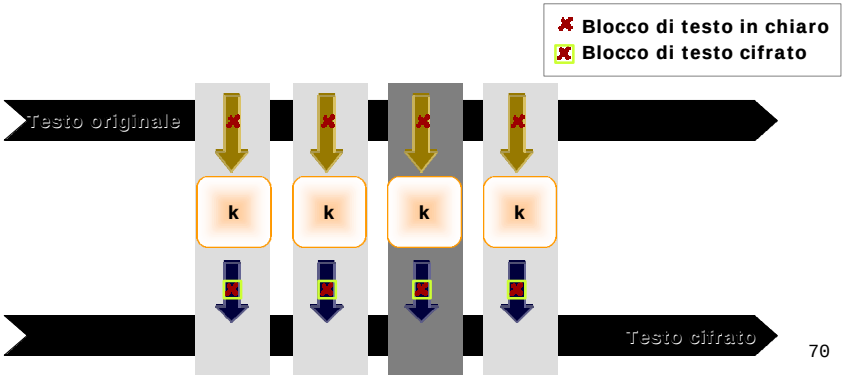
Encrypting Large Messages

Encrypting large messages

- Block ciphers encrypt fixed size blocks
  - **eg. DES encrypts 64-bit blocks, with 56-bit key**
- Need way to use in practise, given usually have arbitrary amount of information to encrypt (longer than 64bits..)
- Four were defined for DES in ANSI standard ANSI X3.106-1983 Modes of Use
- These schemes are applicable for DES, 3DES, IDEA, EAS, etc
- Modes:
  - **Electronic Code Book (ECB)**
  - **Cipher Block Chaining (CBC)**
  - **k-bit Cipher Feedback Mode (CFB)**
  - **k-bit Output Feedback Mode (OFB)**

Electronic Codebook (ECB)

- Consist of doing the obvious thing, and it is usually the worst method.. ;)
- The message is broke into 64-bit blocks, with padding for the last one
- Each block is independently encrypted with the secret key
  - **$C_i = DES_{K1} (P_i)$**
  - **each block is a value which is substituted, like a codebook**

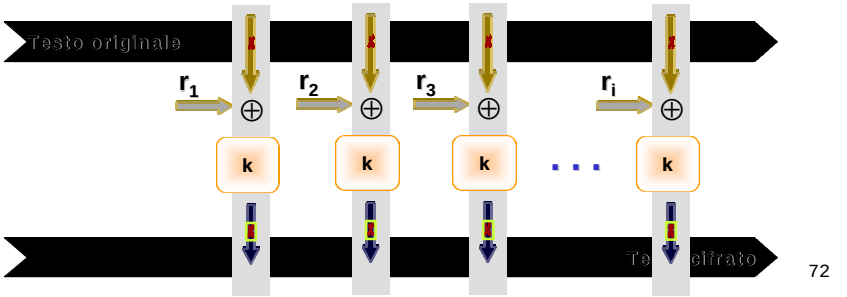


Advantages and Limitations of ECB

- At end of message, handle possible last short block
  - **by padding either with known non-data value (eg nulls)**
  - **or pad last block with count of pad size**
    - eg. [b1 b2 b3 0 0 0 0 5] <- 3 data bytes, then 5 bytes pad+count
- There are a number of problem that arise and that don't show up in the single block case
  - **Repetitions in message may show in ciphertext if aligned with message block**
  - **if a message contains 2 identical 64-bit blocks, the corresponding cipher blocs are identical; it can be a problem**
    - in some cases it can be possible to guess a portion of the message
    - in some cases it can be possible to alter the message
- As result, ECB is rarely used
  - **main use is sending a few blocks of data**

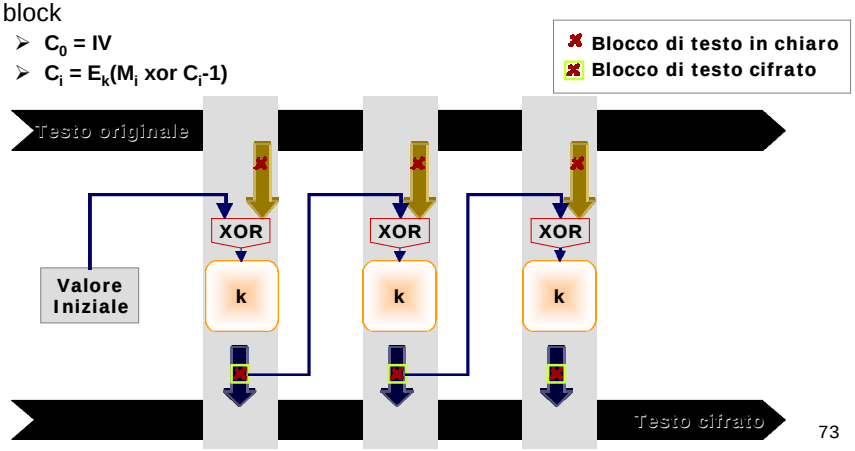
Cipher Block Chaining (CBC)

- CBC avoids some problems in ECB
- If the same block repeats in the plain text, it will not cause repeats of ciphertext
- Adds a feedback mechanism to the cipher
- Plaintext is more difficult to manipulate
- Basic idea:



Cipher Block Chaining (CBC)

- Plaintext patterns are concealed by XORing this block of M with the previous block of C
- Requires an IV (Initialization vector) of random data to encrypt the first block
  - C_0 = IV
  - $C_i = E_k(M_i \text{ xor } C_i\text{-}1)$



Advantages and Limitations of CBC

- Decryption is simple because ⊕ is reversible
- CBC has the same performance of ECB, except for the cost of generating and transmitting the IV
- Each ciphertext block depends on **all** message blocks
  - **thus a change in the message affects all ciphertext blocks after the change as well as the original block**
- It can be used the value of 0 as IV, however it can lead to some problems
  - **e.g. if a message is transmitted weekly, it is possible to guess if changes occurred**
  - **Moreover, IV!=0 prevents attackers for supplying chosen plaintext**
- If IV is sent in the clear, an attacker can change bits of the first block, and change IV to compensate
  - **hence either IV must be a fixed value (as in EFTPOS) or it must be sent encrypted in ECB mode before rest of message**

CBC Threat 1 - Modifing ciphertext blocks

- Using CBC does not eliminate the problem of someone modifying the message in transit
- If the attacker changes the ciphertext block c_n, c_n gets ⊕'d with the decrypted c_{n+1} to yield m_{n+1}
  - **changing bit h of c_n has predictable effect to bit h of m_{n+1}**
  - **the attacker cannot know the new m_n (a new random 64-bit value, as side effect)**
- This thread can be combated by adding a CRC to the plaintext before encrypt

CBC Threat 2 - Rearranging ciphertext blocks

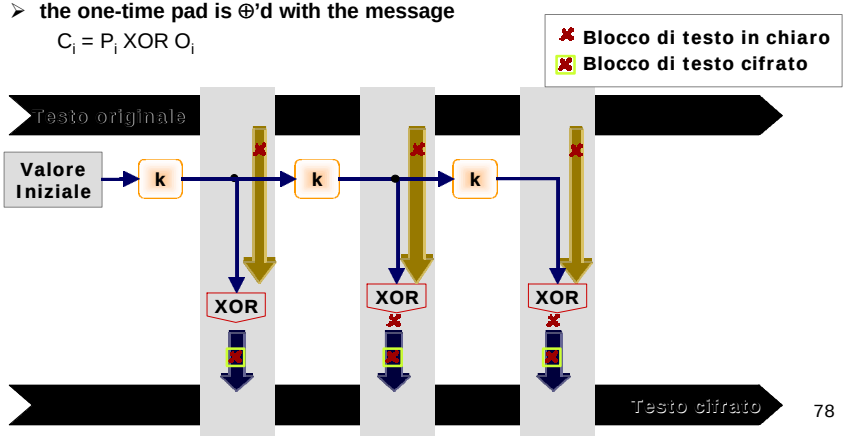
- Knowing the plain text, the corresponding ciphertext and IV, it is possible to rearrange the c_1, c_2, c_3, .. (building blocks), in such a way to obtain a new m_1, m_2, m_3 ..
- A CRC can help but not solve the problem (1 in 2^{32} chance that the CRC will work; if the attack consist only in modifying the message, it is possible to try several combination)

Output Feedback (OFB)

- Acts like a pseudorandom number generator
- The message is encrypted by ⊕ing it with the pseudorandom stream generated by the OFB
  - **message is treated as a stream of bits**
- How it works:
  - **A pseudorandom number O_0 is generated (named IV as in CBC)**
  - **O_0 is encrypted (using secret key K) obtaining O_1**
  - **from O_1 is obtained O_2 and so on, as many block are needed**
    $O_i = E_K(O_{i-1})$
    $O_0 = IV$
  - **feedback is independent of message and can be computed in advance**
  - **so, a long pseudorandom string is generated (one-time pad)**
  - **the one-time pad is simply ⊕'d with the message**
    $C_i = P_i$ XOR O_i
- Uses: stream encryption over noisy channels

Output Feedback (OFB)

- OFB in short:
  - **a long pseudorandom string is generated (one-time pad)**
    $O_i = E_K(O_{i-1})$
    $O_0 = IV$
  - **the one-time pad is ⊕'d with the message**
    $C_i = P_i$ XOR O_i



Advantages and Limitations of OFB

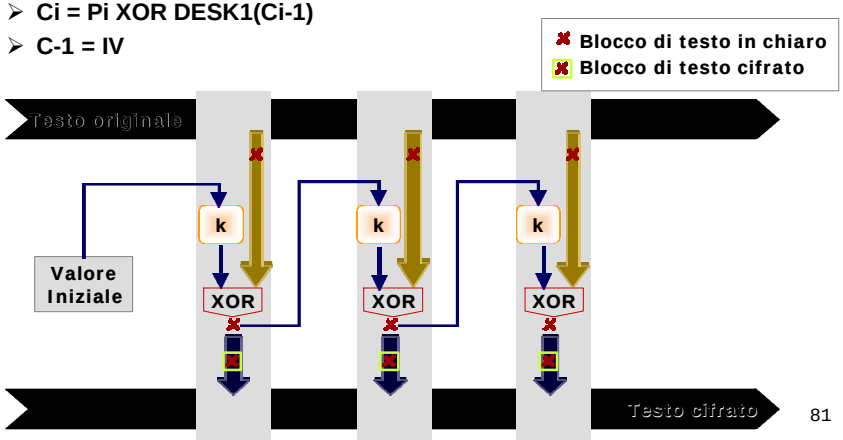
- Advantages of OFB:
  - **one-time pad can be generated in advances**
  - **if some bits of the ciphertext get garbled, only those bits of plaintext get garbled (no error propagation)**
  - **if a message arrives in arbitrary-sized chucks, the associated ciphertext can immediately be transmitted**
- Disadvantages of OFB:
  - **sender and receiver must remain in sync, and some recovery method is needed to ensure this occurs**
  - **if the plaintext and the ciphertext is known by a bad guy, he can modify the plaintext into anything he wants**
  - **hence must never reuse the same sequence (key+IV)**

Cipher FeedBack (CFB)

- Very similar to OFB
- The k-bit shifted in to the encryption module are the k-bit of the ciphertext from the previous block
- added to the output of the block cipher
- result is feed back for next stage (hence name)
  $C_i = P_i$ XOR $DES_{K1}(C_{i-1})$
  $C_{-1} = IV$

Cipher Feedback (CFB)

- Very similar to OFB
- The k-bit shifted in to the encryption module are the k-bit of the ciphertext from the previous block
  - **Ci = Pi XOR DESK1(Ci-1)**
  - **C-1 = IV**

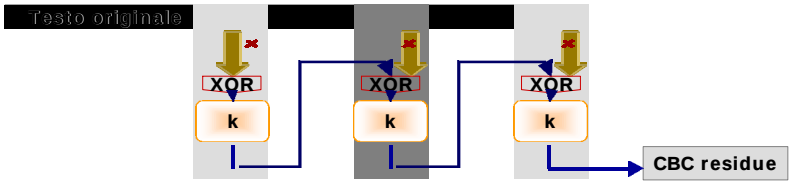


Advantages and Limitations of CFB

- Note: the block cipher is used in **encryption** mode at **both** ends
- The one-time pad cannot be generated in CFB
- Limitation is need to stall while do block encryption after every n-bits
- Errors propogate for several blocks after the error
- it is possible to have k-bit CFB with k different from 64 (e.g. 8)
  - **With OFB or CBC if character are lost in transmission or extra character are added, the rest of trasmission is garbled**
  - **With 8-bit CFB as long as an error is an integral number of bytes, things will be resynchronized**
    - a disadvantage is that DES operation is required each byte

Integrity: Generating MICs

- CBC, CFB and OFB, when properly used, offer good protection against an eavesdropper deciphering a message
- None of these offers good protection against an eavesdropper (who already knows the plaintext or not) modifying it undetected
  - **note that any random string of bits will decrypt into something)**
- In many context message are not secret but integrity must be assured
- A secret key system can be used to generate a cryptographic checksum known as MIC (Message Integrity Code)
- A standard way of protection is to compute the CBC but send only the last block (named CBC residue) along with the plaintext



Ensuring Privacy and Integrity

- To assure privacy it is possible to CBC-encrypt the message
- To assure integrity is appropriate to transmit unencrypted data plus a CBC residue
- To assure both privacy and integrity, is not sufficient to CBC-encrypt the message, nor sending together a CBC-residue
- A possible solution is to send CBC-encrypted message with a CBC-residue, computed with two different keys
  - **double complexity!**

Ensuring Privacy and Integrity

- Other possible solutions:
  - **CBC with a weak cryptography checksum**
    - e.g. Kerberos V4
  - **CBC with cryptography hash**
    - two cryptographic computations such as CBC with two different keys, but more efficient (thanks to hash algorithms)
      - e.g. with MD4, MD5, etc
  - **CBC encryption and CBC residue with related keys**
    - it could be sufficient to switch just one bit
      - e.g. kerberos V5 ⊕s the key with 0xF0F0F0F0F0F0F0F0
      - this has the property of preserving key parity and never transforming non-weak key into a weak-key