UNIT - II

BLOCK CIPHER PRINCIPLES

Virtually, all symmetric block encryption algorithms in current use are based on a structure referred to as Fiestel block cipher. For that reason, it is important to examine the design principles of the Fiestel cipher. We begin with a **comparison of stream cipher with block cipher.**

- **A stream cipher** is one that encrypts a digital data stream one bit or one byte at a time. E.g, vigenere cipher. **A block cipher** is one in which a block of plaintext is treated as a whole and used to produce a cipher text block of equal length. Typically a block size of 64 or 128 bits is used.

Block cipher principles

- most symmetric block ciphers are based on a **Feistel Cipher Structure** needed since must be able to **decrypt** ciphertext to recover messages efficiently. block ciphers look like an extremely large substitution
- would need table of 264 entries for a 64-bit block
- Instead create from smaller building blocks
- using idea of a product cipher in 1949 Claude Shannon introduced idea of substitution-permutation (S-P) networks called modern substitution-transposition product cipher
these form the basis of modern block ciphers

- S-P networks are based on the two primitive cryptographic operations we have seen before:
  - *substitution* (S-box)
  - *permutation* (P-box)
  - provide *confusion* and *diffusion* of message
  - **diffusion** – dissipates statistical structure of plaintext over bulk of ciphertext
- **confusion** – makes relationship between ciphertext and key as complex as possible

DATA ENCRYPTION STANDARD (DES)

In May 1973, and again in Aug 1974 the NBS (now NIST) called for possible encryption algorithms for use in unclassified government applications response was mostly disappointing, however IBM submitted their Lucifer design following a period of redesign and comment it became the Data Encryption Standard (DES)

it was adopted as a (US) federal standard in Nov 76, published by NBS as a hardware only scheme in Jan 77 and by ANSI for both hardware and software standards in ANSI X3.92-1981 (also X3.106-1983 modes of use) subsequently it has been widely adopted and is now published in many standards around the world cf Australian Standard AS2805.5-1985

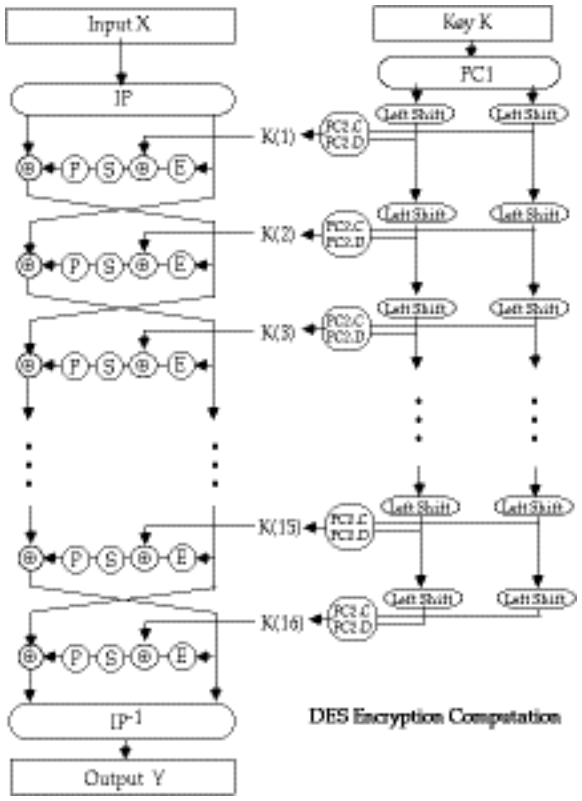
one of the largest users of the DES is the banking industry, particularly with EFT, and EFTPOS

it is for this use that the DES has primarily been standardized, with ANSI having twice reconfirmed its recommended use for 5 year periods - a further extension is not expected however although the standard is public, the design criteria used are classified and have yet to be released there has been considerable controversy over the design, particularly in the choice of a 56-bit key

- recent analysis has shown despite this that the choice was appropriate, and that DES is well designed
- rapid advances in computing speed though have rendered the 56 bit key susceptible to exhaustive key search, as predicted by Diffie & Hellman

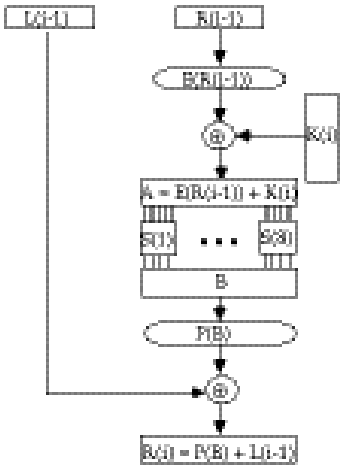
- the DES has also been theoretically broken using a method called Differential Cryptanalysis, however in practice this is unlikely to be a problem (yet)

Overview of the DES Encryption Algorithm



DES Encryption Computation

- the basic process in enciphering a 64-bit data block using the DES consists of:
  - an initial permutation (IP)
  - 16 rounds of a complex key dependent calculation f
  - a final permutation, being the inverse of IP
- in more detail the 16 rounds of f consist of:



- this can be described functionally as

L(i) = R(i-1)

R(i) = L(i-1) (+) P(S(E(R(i-1))(+) K(i)))

and forms one round in an S-P network

- the subkeys used by the 16 rounds are formed by the **key schedule** which consists of:
  - an initial permutation of the key (PC1) which selects 56-bits in two 28-bit halves
  - 16 stages consisting of
  - selecting 24-bits from each half and permuting them by PC2 for use in function f,
  - rotating each half either 1 or 2 places depending on the **key rotation schedule** KS
- this can be described functionally as:

K(i) = PC2(KS(PC1(K),i))

- the **key rotation schedule** KS is specified as:

Round	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
KS	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1
Total Rot	1	2	4	6	8	10	12	14	15	17	19	21	23	25	27	28

- more details on the various DES functions can be found in your textbooks
- following is a walk-through of a DES encryption calculation taken from:

H Katzan, "The Standard Data Encryption Algorithm", Petrocelli Books, New York, 1977

DES Modes of Use

- DES encrypts 64-bit blocks of data, using a 56-bit key
- we need some way of specifying how to use it in practise, given that we usually have an arbitrary amount of information to encrypt
- the way we use a block cipher is called its **Mode of Use** and four have been defined for the DES by ANSI in the standard: ANSI X3.106-1983 Modes of Use)
- modes are either:

Block Modes

Splits messages in blocks (ECB, CBC)

Electronic Codebook Book (ECB)

- Where the message is broken into independent 64-bit blocks which are encrypted

$$C_{(i)} = DES_{(K1)} (P_{(i)})$$

Cipher Block Chaining (CBC)

Again the message is broken into 64-bit blocks, but they are linked together in the encryption operation with an IV C_(i) = DES_(K1) (P_(i)(+)C_(i-1)) C_(-1)=IV

Stream Modes

On bit stream messages (CFB, OFB)

Cipher Feedback (CFB)

- Where the message is treated as a stream of bits, added to the output of the DES, with the result being feedback for the next stage
C_(i) = P_(i)(+) DES_(K1) (C_(i-1)) C_(-1)=IV

Output Feedback (OFB)

- Where the message is treated as a stream of bits, added to the message, but with the feedback being independent of the message
C_(i) = P_(i)(+) O_(i) O_(i) = DES_(K1)(O_(i-1)) O_(-1)=IV

- each mode has its advantages and disadvantages

Limitations of Various Modes

ECB

- repetitions in message can be reflected in ciphertext
  - if aligned with message block
  - particularly with data such graphics
  - or with messages that change very little, which become a code-book analysis problem
- weakness is because enciphered message blocks are independent of each other

Block	Plaintext	Ciphertext
1	T H E Y C A N	60 99 46 42 52 82 22 49
2	H A V E S E	FF BF BC 77 9B BB F2 06
3	V E R A L A C	0D 4D 86 DE B6 CD 92 5D
4	T I V E P E R	99 63 A8 OF 32 D3 E7 E9
5	M A N E N T V	10 49 1F 3B DE 67 21 B7
6	I R T U A L C	BD 2D 6D 61 42 08 C7 B8
7	I R C U I T S	19 F1 01 A4 89 6A AE 4C
8	A N D / O R V	84 DB CC EC 35 18 58 9C
9	I R T U A L C	BD 2D 6D 61 42 08 C7 B8
10	A L L S A T	D4 3C D4 5A 9E 0B A5 ED
11	T H E S A M E	84 52 01 AC 2D FE 98 3A
12	T I M E .	89 F1 89 E9 DB CC CB BB
	Key	01 23 45 67 89 AB CD EF

Fig. 4.1 A weakness in ECB encipherment

CBC

- use result of one encryption to modify input of next
  - hence each ciphertext block is dependent on **all** message blocks before it
  - thus a change in the message affects the ciphertext block after the change as well as the original block

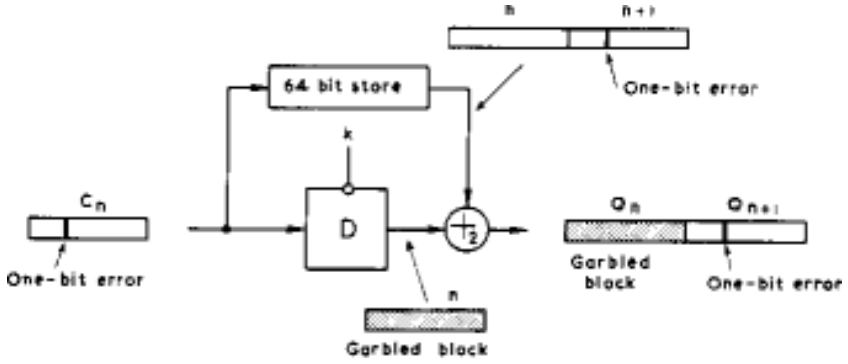


Fig. 4.5 One-bit error in cipher block chaining

to start need an **Initial Value** (IV) which must be known by both sender and receiver

- however 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

- also at the end of the message, have to handle a possible last short block
  - either pad last block (possible with count of pad size), or use some fiddling to double up last two blocks
  - see Davies for examples

CFB

- when data is bit or byte oriented, want to operate on it at that level, so use a stream mode
- the block cipher is use in **encryption** mode at **both** ends, with input being a feed-back copy of the ciphertext
- can vary the number of bits feed back, trading off efficiency for ease of use
- again errors propogate for several blocks after the error

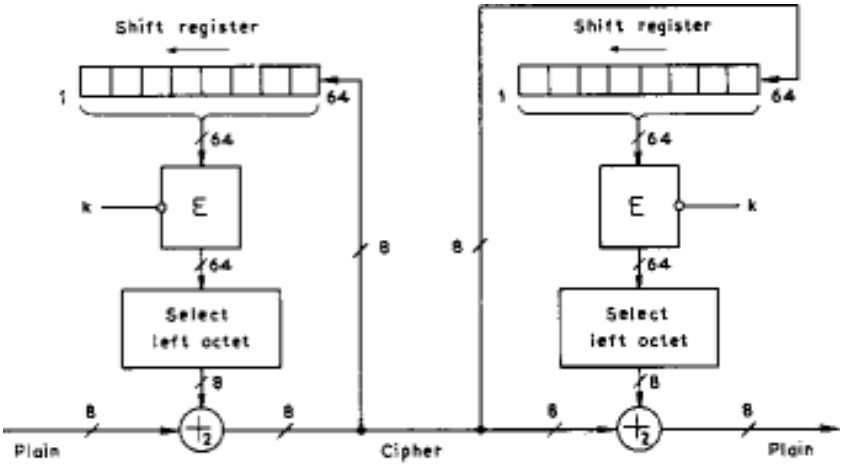


Fig. 4.7 8-bit cipher feedback

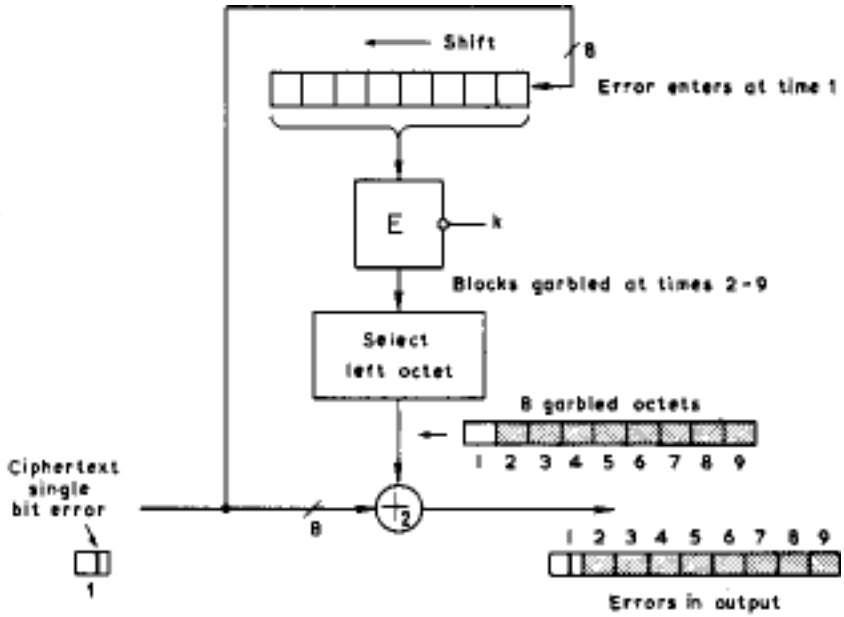


Fig. 4.8 One-bit error in 8-bit cipher feedback

OFB

- also a stream mode, but intended for use where the error feedback is a problem, or where the encryptions want to be done before the message is available
- is superficially similar to CFB, but the feedback is from the output of the block cipher and is independent of the message, a variation of a Vernam cipher
- again an IV is needed
- sender and receiver must remain in sync, and some recovery method is needed to ensure this occurs
- although originally specified with varying m-bit feedback in the standards, subsequent research has shown that only **64-bit OFB** should ever be used (and this is the most efficient use anyway), see

  D Davies, G Parkin, "The Average Cycle Size of the Key Stream in Output Feedback Encipherment" in Advances in Cryptology - Crypto 82, Plenum Press, 1982, pp97-98

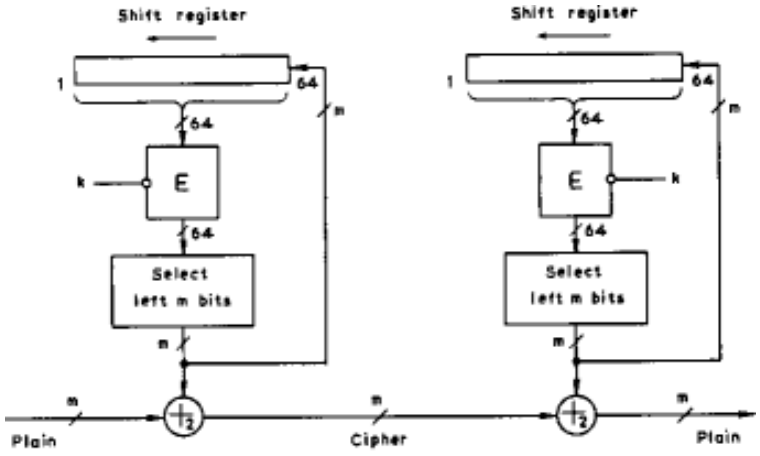


Fig. 4.12 m-bit output feedback

DES Weak Keys

- with many block ciphers there are some keys that should be avoided, because of reduced cipher complexity
- these keys are such that the same sub-key is generated in more than one round, and they include:

Weak Keys

- he same sub-key is generated for every round
- DES has 4 weak keys

Semi-Weak Keys

- only two sub-keys are generated on alternate rounds
- DES has 12 of these (in 6 pairs)

Demi-Semi Weak Keys

- have four sub-keys generated
- none of these cause a problem since they are a tiny fraction of all available keys
- however they MUST be avoided by any key generation program

DES Design Principles

Although the standard for DES is public, the design criteria used are classified and have yet to be released. some information is known, and more has been deduced

L P Brown, "A Proposed Design for an Extended DES", in Computer Security in the Age of Information, W. J. Caelli (ed), North-Holland, pp 9-22, 1989

L P Brown, J R Seberry, "On the Design of Permutation Boxes in DES Type Cryptosystems", in Advances in Cryptology - Eurocrypt '89, Lecture Notes in Computer Science, vol 434, pp 696-705, J.J. Quisquater, J. Vanderwalle (eds), Springer-Verlag, Berlin, 1990.

L P Brown and J R Seberry, "Key Scheduling in DES Type Cryptosystems," in Advances in Cryptology - Auscrypt '90, Lecture Notes in Computer Science, vol 453, pp 221-228, J. Seberry, J. Pieprzyk (eds), Springer-Verlag, Berlin, 1990.

will briefly overview the basic results, for more detailed analyses see the above papers

DES S-Box Design Criteria

Each S-box may be considered as four substitution functions

- these 1-1 functions map inputs 2,3,4,5 onto output bits
- a particular function is selected by bits 1,6
- this provides an **autoclave feature**

DES Design Criteria

- there were 12 criterion used, resulting in about 1000
- possible S-Boxes, of which the implementers chose 8
- these criteria are CLASSIFIED SECRET
- however, some of them have become known
- The following are design criterion:

R1: Each row of an S-box is a permutation of 0 to 15

R2: No S-Box is a linear of affine function of the input

R3: Changing one input bit to an S-box results in changing at least two output bits

R4: S(x) and S(x+001100) must differ in at least 2 bits

- The following are said to be caused by design criteria

R5: S(x) [[pi]] S(x+11*ef* 00) for any choice of *e* and *f*

R6: The S-boxes were chosen to minimize the difference between the number of 1's and 0's in any S-box output when any single input is held constant

R7: The S-boxes chosen require significantly more minterms than a random choice would require

Meyer Tables 3-17, 3-18

DES Permutation Tables

- there are 5 Permutations used in DES:
  - **IP** and **IP^(-1) , P, E, PC1, PC2**
- their design criteria are CLASSIFIED SECRET
- it has been noted that **IP** and **IP^(-1)** and **PC1** serve no cryptological function when DES is used in ECB or CBC modes, since searches may be done in the space generated after they have been applied
- **E, P,** and **PC2** combined with the S-Boxes must supply the required dependence of the output bits on the input bits and key bits (**avalanche** and **completeness** effects)

Ciphertext Dependence on Input and Key

- the role of **P, E,** and **PC2** is distribute the outputs of the S-boxes so that each output bit becomes a function of all the input bits in as few rounds as possible
- Carl Meyer (in Meyer 1978, or Meyer & Matyas 1982) performed this analysis on the current DES design

Ciphertext dependence on Plaintext

- define **G_(i,j)** a 64*64 array which shows the dependence of output bits X(j) on input bits X(i)
- examine **G_(0,j)** to determine how fast complete dependence is achieved
- to build **G_(0,1)** use the following

L(i) = R(i-1)

R(i) = L(i-1) (+) f(K(i), R(i-1))

- DES P reaches complete dependence after 5 rounds
- []

Ciphertext dependence on Key

- Carl Meyer also performed this analysis

- define **F_(i,j)** a 64*56 array which shows the dependence of output bits X(j) on key bits U(i) (after PC1 is used)
- examine **F_(0,j)** to determine how fast complete dependence is achieved
- DES PC2 reaches complete dependence after 5 rounds

Key Scheduling and PC2

- Key Schedule
  - is a critical component in the design
  - must provide different keys for each round otherwise security may be compromized (see Grossman & Tuckerman 1978)
  - current scheme can result in weak keys which give the same, 2 or 4 keys over the 16 rounds
- Key Schedule and PC-2 Design
  - is performed in two 28-bit independent halves
  - C-side provides keys to S-boxes 1 to 4
  - D-side provides keys to S-boxes 5 to 8
  - the rotations are used to present different bits of the key for selection on successive rounds
  - PC-2 selects key-bits and distributes them over the S-box inputs

Possible Techniques for Improving DES

- multiple enciphering with DES
- extending DES to 128-bit data paths and 112-bit keys
- extending the Key Expansion calculation

Triple DES

- DES variant
- standardised in ANSI X9.17 & ISO 8732 and in PEM for key management
- proposed for general EFT standard by ANSI X9
- backwards compatible with many DES schemes
- uses 2 or 3 keys

$$C = DES\_(K1)\ Bbc\{(DES\hat{}(-1)\_(K2)Bbc\{(DES\_(K1)(P)))$$

- no known practical attacks

- brute force search impossible
- meet-in-the-middle attacks need 2^(56) PC pairs per key

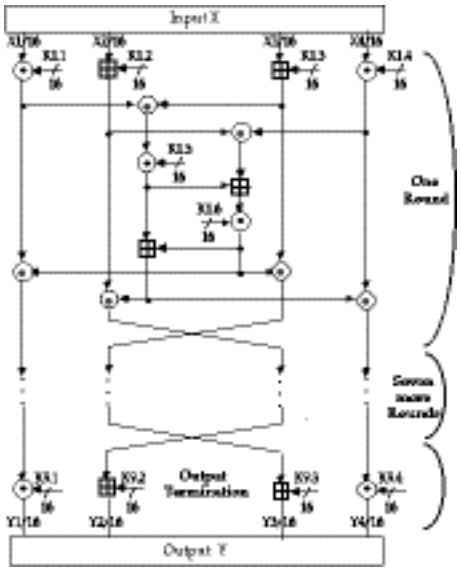
- popular current alternative

IDEA (IPES)

- developed by James Massey & Xuejia Lai at ETH originally in Zurich in 1990, then called IPES :
- Name changed to IDEA in 1992
- encrypts 64-bit blocks using a 128-bit key
- based on mixing operations from different (incompatible) algebraic groups (XOR, Addition mod 2^(16) , Multiplication mod 2^(16) +1)
- all operations are on 16-bit sub-blocks, with no permutations used, hence its very efficient in s/w
- IDEA is patented in Europe & US, however non-commercial use is freely permitted
- used in the public domain PGP secure email system (with agreement from the patent holders)
- currently no attack against IDEA is known (it appears secure against differential cryptanalysis), and its key is too long for exhaustive search

Overview of IDEA

- IDEA encryption works as follows:
  - the 64-bit data block is divided by 4 into: X_(1) , X_(2) , X_(3) , X_(4)
  - in each of eight the sub-blocks are XORd, added, multiplied with one another and with six 16-bit sub-blocks of key material, and the second and third sub-blocks are swapped
  - finally some more key material is combined with the sub-blocks



- IDEA sub-keys
  - the encryption keying material is obtained by splitting the 128-bits of key into eight 16-bit sub-keys, once these are used the key is rotated by 25-bits and broken up again etc
  - the decryption keying material is a little more complex, since inverses of the sub-blocks need to be calculated
- the keys used may be summarised as follows:

Round	Encryption Keys	Decryption Keys
1	K1.1 K1.2 K1.3 K1.4 K1.5 K1.6 K8.6	K9.1-1 -K9.2 -K9.3 K9.4-1 K8.5
2	K2.1 K2.2 K2.3 K2.4 K2.5 K2.6 K7.6	K8.1-1 -K8.3 -K8.2 K8.4-1 K7.5
3	K3.1 K3.2 K3.3 K3.4 K3.5 K3.6 K6.6	K7.1-1 -K7.3 -K7.2 K7.4-1 K6.5
4	K4.1 K4.2 K4.3 K4.4 K4.5 K4.6 K5.6	K6.1-1 -K6.3 -K6.2 K6.4-1 K5.5
5	K5.1 K5.2 K5.3 K5.4 K5.5 K5.6 K4.6	K5.1-1 -K5.3 -K5.2 K5.4-1 K4.5
6	K6.1 K6.2 K6.3 K6.4 K6.5 K6.6 K3.6	K4.1-1 -K4.3 -K4.2 K4.4-1 K3.5
7	K7.1 K7.2 K7.3 K7.4 K7.5 K7.6 K2.6	K3.1-1 -K3.3 -K3.2 K3.4-1 K2.5
8	K8.1 K8.2 K8.3 K8.4 K8.5 K8.6 K1.6	K2.1-1 -K2.3 -K2.2 K2.4-1 K1.5
Output	K9.1 K9.2 K9.3 K9.4	K1.1-1 -K1.2 -K1.3 K1.4-1

where: K1.1^(-1) is the multiplicative inverse mod 2^(16) +1

-K1.2 is the additive inverse mod 2^(16) and the original operations are:

(+) bit-by-bit XOR + additional mod 2^(16) of 16-bit integers

* Multiplication mod 2^(16) +1 (where 0 means 2^(16))

IDEA Example Encryption

Key (128-bits) Plain (64-bit) Cipher (64-bit)
7ca110454a1a6e5701a1d6d039776742 690f5b0d9a26939b 1bddb24214237ec7
idea(X=690f 5b0d 9a26 939b)
r=1, X=690f 5b0d 9a26 939b, SK=7ca1 1045 4a1a 6e57 01a1 d6d0
steps=234a 6b52 e440 840f c70a ef5d 3606 2563 0311 3917 205b e751 5245 bd18
r=2, X=205b e751 5245 bd18, SK=3977 6742 8a94 34dc ae03 43ad
steps=460a 4e93 dcd9 3995 9ad3 7706 d13d 4843 4b2d 1c6a 0d27 97f4 52f9 25ff
r=3, X=0d27 97f4 52f9 25ff, SK=a072 eece 84f9 4220 b95c 0687
steps=3320 86c2 d7f2 7410 e4d2 f2d2 57cb 4a9d 04e4 5caf 37c4 d316 da6d 28bf
r=4, X=37c4 d316 da6d 28bf, SK=5b40 e5dd 9d09 f284 4115 2869
steps=8920 b8f3 7776 69e3 fe56 d110 7266 4376 10c0 8326 99e0 67b6 3bd5 eac5
r=5, X=99e0 67b6 3bd5 eac5, SK=0eb6 81cb bb3a 13e5 0882 2a50
steps=9c69 e981 f70f 8efb 6b66 677a b63b 1db5 f5a8 abe3 69c1 02a7 4262 2518
r=6, X=69c1 02a7 4262 2518, SK=d372 b80d 9776 7427 ca11 0454
steps=d39a bab4 d9d8 75d4 0a42 cf60 ba4a 89aa d175 8bbf 02ef 08ad 310b fe6b
r=7, X=02ef 08ad 310b fe6b, SK=a1a6 e570 1a1d 6d03 4f94 2208
steps=3420 ee1d 4b28 1deb 7f08 f3f6 c124 b51a 04bd c5e1 309d 4f95 2bfc d80a
r=8, X=309d 4f95 2bfc d80a, SK=a943 4dca e034 3ada 072e ece8
steps=3df3 9d5f 0c30 0ada 31c3 9785 44a5 dc2a 7253 b6f8 4fa0 7e63 2ba7 bc22
out, X=4fa0 2ba7 7e63 bc22, SK=1152 869b 95c0 6875
= 1bdd b242 1423 7ec7

Differential Cryptanalysis of Block Ciphers

- Differential Cryptanalysis is a recently (in the public research community) developed method which provides a powerful means of analysing block ciphers
- it has been used to analyse most of the currently proposed block ciphers with varying degrees of success
- usually have a break-even point in number of rounds of the cipher used for which differential cryptanalysis is faster than exhaustive key-space search
- if this number is greater than that specified for the cipher, then it is regarded as broken

Overview of Differential Cryptanalysis

- is a statistical attack against Feistel ciphers
- uses structure in cipher not previously used
- design of S-P networks is such that the output from function *f* is influenced by both input and key

$$R(i)=L(i-1) (+) f(K(i)(+)R(i-1))$$

- hence cannot trace values back through cipher without knowing the values of the key

Biham & Shamir's key idea is to compare two separate encryptions (using the same key) and look at the XOR of the S-box inputs and outputs and this is **independent** of the key being used

$$Ra(i)=f(K(i)(+)Ra(i-1))$$
$$Rb(i)=f(K(i)(+)Rb(i-1))$$

hence

$$Y(i)= Ra(i)(+)Rb(i)$$
$$= f(K(i)(+)Ra(i-1)(+)K(i)(+)Rb(i-1))$$
$$= f(Ra(i-1)(+)Rb(i-1)) = f(X(i))$$

- further various input XOR - output XOR pairs occur with different probabilities
- hence knowing information on these pairs gives us additional information on the cipher

XOR Profiles and Characteristics

- start by compiling a table of input vs output XOR values, an **XOR Profile** for each S-box
- a particular input XOR value and output XOR value pair will occur with some probability
- call such a specified pair, a **characteristic**
- can infer information about key value in one round, if find a pair of encryptions matching a characteristic, and hence knowing input and output XOR values
- have several variant forms of differential cryptanalysis, will discuss just the general form used for attacking many rounds (>8) of a cipher
- can describe 1-round characteristic by:

f(x')->y', Pr(p)

(a',b')->(b',a'(+)f(b')) with prob p

i) useful characteristics:

ii) f(0')->0', Pr(1) ie always

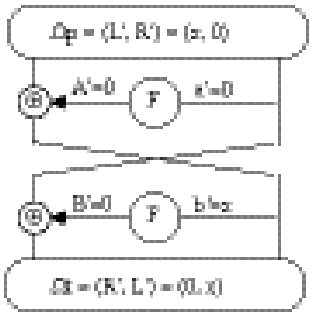
A.(x,0)->(0,x) always

ii) f(x')->0', Pr(p_(0))

B.(0,x)->(x,0) with probability p_(0)

- attack multiple rounds using **n-round characteristics**
- **n-round characteristics** combine one round characteristics whose outputs & inputs match

- probability of **n-round characteristic** is product of the **1-round characteristic** probabilities



2-Round Iterative Characteristic

- some common characteristic.0000c structures are:

  * a 2-round characteristic:

  A.(x,0)->(0,x) always

  B.(0,x)->(x,0) with probability p

  * a 3-round characteristic:

  A.(x,0)->(0,x) always

  B.(0,x)->(x,x) with probability p1

  C.(x,x)->(x,0) with probability p2

- perform attack by repeatedly encrypting plaintext pairs with known input XOR until obtain expected output XOR matching n-round characteristic being used
- if all intermediate rounds also match required XOR (which is unknown) then have a **right pair**, if not then have a **wrong pair**, relative ratio is S/N for attack
- assume know XOR at intermediate rounds (if right pair) then deduce keys values for the rounds - right pairs suggest same key bits, wrong pairs give random values
- for large numbers of rounds, probability is so low that more pairs are required than exist with 64-bit inputs
- optimisations of this attack can be made, trading memory for search time, and number of rounds used
- in their latest paper, Biham and Shamir show how a 13-round iterated characteristic can be used to break the full 16-round DES

Linear Cryptanalysis of Block Ciphers

- Linear Cryptanalysis is another recently developed method for analysing block ciphers
- like differential cryptanalysis it is a statistical method

- again have a break-even point in number of rounds of the cipher used for which linear cryptanalysis is faster than exhaustive key-space search
- if this number is greater than that specified for the cipher, then it is regarded as broken
- In Linear Cryptanalysis want to find a linear approximation which holds with Prob p!=^(1) /_(2)

  P[i1,i2,...,ia](+)C[j1,j2,...,jb]=K[k1,k2,...,kc]
  where ia,jb,kc are bit locations in P,C,K
- can determine one bit of key using maximum likelihood algorithm, using a large number of trial encryptions
- effectiveness of linear cryptanalysis is given by

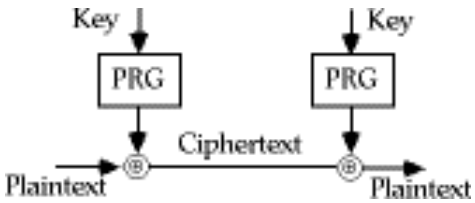
  |p - 1/2|
- DES can be broken by encrypting 2^(47) known plaintexts

  PL[7,18,24](+) PR[12,16](+) CL[15](+) CR[7,18,24,29](+) F16(CR,K16)[15] = K1[19,23](+)K3[22](+) K4[44](+) K5[22](+)K7[22](+) K8[44](+) K9[22](+) K11[22](+) K12[44](+) K13[22](+) K15[22]
- this will recover some of the key bits, the rest must be searched for exhaustively
- LOKI with 12 or more rounds cannot be broken using linear cryptanalysis

Stream Ciphers and the Vernam cipher

- Process the message bit by bit (as a stream)
- The most famous of these is the **Vernam cipher** (also known as the **one-time pad**)
- invented by Vernam, working for AT&T, in 1917
- simply add bits of message to random key bits
- need as many key bits as message, difficult in practise (ie distribute on a mag-tape or CDROM)
- is unconditionally secure provided key is truly random

- suggest generating keystream from a smaller (base) key



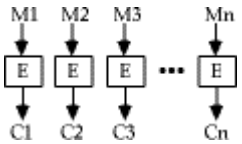
- use some pseudo-random function to do this

Modern Private Key Ciphers (part 1)

- now want to concentrate on modern encryption systems
- these usually consider the message as a sequence of bits
  - (eg as a series of ASCII characters concatenated)
- have two broad families of methods
  - stream ciphers and block ciphers

Block Ciphers

- in a block cipher the message is broken into blocks, each of which is then encrypted (ie like a substitution on very big characters - 64-bits or more)
- most modern ciphers we will study are of this form



Shannons Theory of Secrecy Systems

- Claude Shannon wrote some of the pivotal papers on modern cryptology theory in 1949:
  - C E Shannon, "Communication Theory of Secrecy Systems", Bell System Technical Journal, Vol 28, Oct 1949, pp 656-715
  - C E Shannon, "Prediction and Entropy of printed English", Bell System Technical Journal, Vol 30, Jan 1951, pp 50-64
- in these he developed the concepts of:
  - entropy of a message,
  - redundancy in a language,
  - theories about how much information is needed to break a cipher
  - defined the concepts of computationally secure vs unconditionally secure ciphers
- he showed that the Vernam cipher is the only currently known unconditionally secure cipher, provided the key is truly random
- also showed that if try to encrypt English text by adding to other English text (ie a **Book**cipher), this is not secure since English is 80% redundant, giving ciphertext with 60% redundancy, enough to break

- a similar technique can also be used if the same random key stream is used twice on different messages, the redundancy in the messages is sufficient to break this
- as discussed earlier, **exhaustive key search** is the most fundamental attack, and is directly proportional to the size of the key
- can tabulate these for reasonable assumptions about the number of operations possible (& parallel tests):

Key Size (bits)	Time (1us/test)	Time (1us/106test)
24	8.4 sec	8.4 usec
32	35.8 mins	2.15 msec
40	6.4 days	550 msec
48	4.46 yrs	2.35 mins
56	~2000 yrs	10.0 hrs
64	~500000 yrs	107 days

- as the ultimate limit, it can be shown from energy consumption considerations that the maximum number of possible elementary operations in 1000 years is about: 3 x 10 ^(48)
- similarly can show that if need say 10 atoms to store a bit of information, then the greatest possible number of bits storable in a volume of say the moon is: 10 ^(45)
- if a cipher requires more operations, or needs more storage than this, it is pretty reasonable to say it is computationally secure
  - eg to test all possible 128-bit keys in Lucifer takes about 3 x 10 ^(48) encryptions, needing 10 ^(19) years

Substitution-Permutation Ciphers

- in his 1949 paper Shannon also introduced the idea of substitution-permutation (S-P) networks, which now form the basis of modern block ciphers
- an S-P network is the modern form of a substitution-transposition product cipher
- S-P networks are based on the two primitive cryptographic operations we have seen before

Substitution Operation

- a binary word is replaced by some other binary word
- the whole substitution function forms the key
- if use n bit words, the key is 2^(n)!bits, grows rapidly

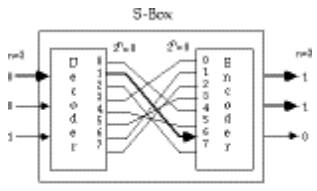


Fig 2.1 Substitution Operation

- can also think of this as a large lookup table, with n address lines (hence 2^(n) addresses), each n bits wide being the output value
- will call them **S-boxes**

Permutation Operation

- a binary word has its bits reordered (permuted)
- the re-ordering forms the key
- if use n bit words, the key is n!bits, which grows more slowly, and hence is less secure than substitution

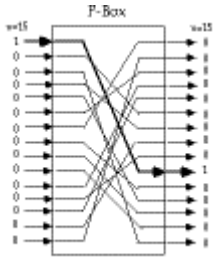


Fig 2.2 - Permutation or Transposition Function

- this is equivalent to a wire-crossing in practise (though is much harder to do in software)
- will call these **P-boxes**

Substitution-Permutation Network

- Shannon combined these two primitives
- he called these **mixing transformations**

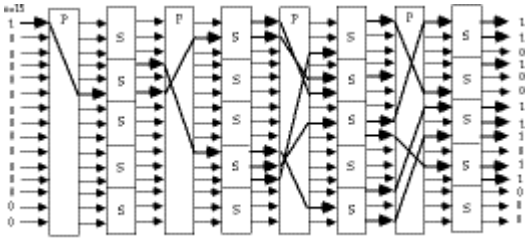


Fig 2.3 - Substitution-Permutation Network, with the Avalanche Characteristic

- Shannons mixing transformations are a special form of product ciphers where

S-Boxes provide **confusion** of input bits
P-Boxes provide **diffusion** across S-box inputs

- in general these provide the following results, as described in:

A F Webster & S E Tavares "On the Design of S-boxes", in Advances in Cryptology - Crypto 85, Lecture Notes in Computer Science, No 218, Springer-Verlag, 1985, pp 523-534

Avalanche effect

- where changing **one** input bit results in changes of approx **half** the output bits

More formally, a function f has a good **avalanche** effect if for each bit i,0<=i<m, if the 2^(m) plaintext vectors are divided into 2^(m-1) pairs X and X_(i) with each pair differing only in bit i; and if the 2^(m-1) exclusive-or sums, termed avalanche vectors

V_(i) = f(X) (+) f(X_(i))

Are compared, then about half of these sums should be found to be 1.

Completeness effect

- where each output bit is a complex function of **all** the input bits

More formally, a function f has a good **completeness** effect if for each bit j,0<=j<m, in the ciphertext output vector, there is at least one pair of plaintext vectors X and X_(i) which differ only in bit i, and for which f(X) and f(X_(i)) differ in bit j

Practical Substitution-Permutation Networks

- in practise we need to be able to decrypt messages, as well as to encrypt them, hence either:
  - have to define inverses for each of our S & P-boxes, but this doubles the code/hardware needed, or
  - define a structure that is easy to reverse, so can use basically the same code or hardware for both encryption and decryption
- Horst Feistel, working at IBM Thomas J Watson Research Labs devised just such a structure in early 70's, which we now call a **feistel cipher**
  - the idea is to partition the input block into two halves, L(i-1)and R(i-1), and use only R(i-1)in each round i (part) of the cipher
  - the function g incorporates one stage of the S-P network, controlled by part of the key K(i)known as the ith subkey

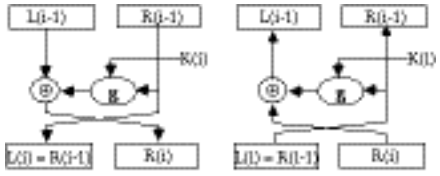


Fig 2.4 - A Round of a Feistel Cipher

- this can be described functionally as:

L(i) = R(i-1)

R(i) = L(i-1) (+) g(K(i), R(i-1))

- this can easily be reversed as seen in the above diagram, working backwards through the rounds
- in practise link a number of these stages together (typically 16 rounds) to form the full cipher