

Cryptography and Network Security

Block Cipher

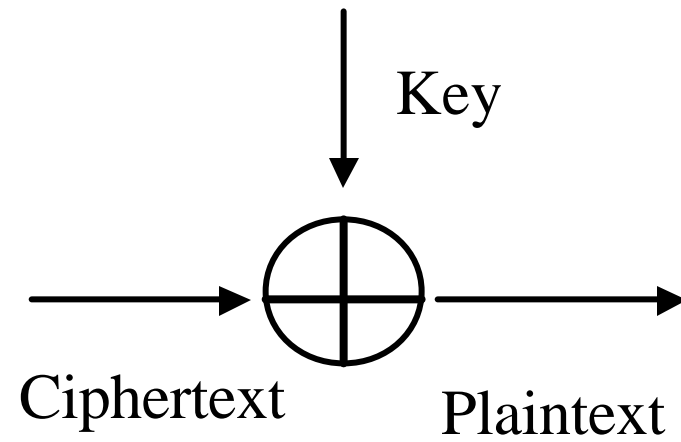
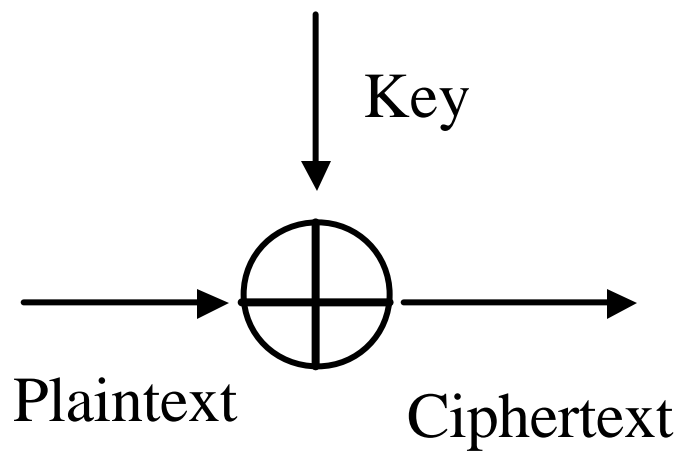
Xiang-Yang Li

Modern Private Key Ciphers

□ Stream ciphers

- The most famous: **Vernam cipher**
- Invented by Vernam, (AT&T, in 1917)
- Process the message bit by bit (as a stream)
- (Also known as the **one-time pad**)
- Simply add bits of message to random key bits

Cont.



Pros and Cons

❑ Drawbacks

- Need as many key bits as message, difficult in practice
- (ie distribute on a mag-tape or CDROM)

❑ Strength

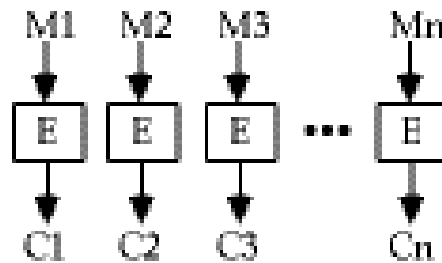
- Is unconditionally secure provided key is truly *random*

Key Generation

- ❑ Why not to generate keystream from a smaller (base) key?
 - Use some pseudo-random function to do this
 - Although this looks very attractive, it proves to be very very difficult in practice to find a good pseudo-random function that is cryptographically strong
- ❑ This is still an area of much research

Block Ciphers

- The message is broken into blocks,
 - Each of which is then encrypted
 - (Like a substitution on very big characters - 64-bits or more)



Substitution and Permutation

- 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

- ❑ A binary word is replaced by some other binary word
- ❑ The whole substitution function forms the key
- ❑ If use n bit words,
 - The key space is $2^n!$
- ❑ 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**

Cont.

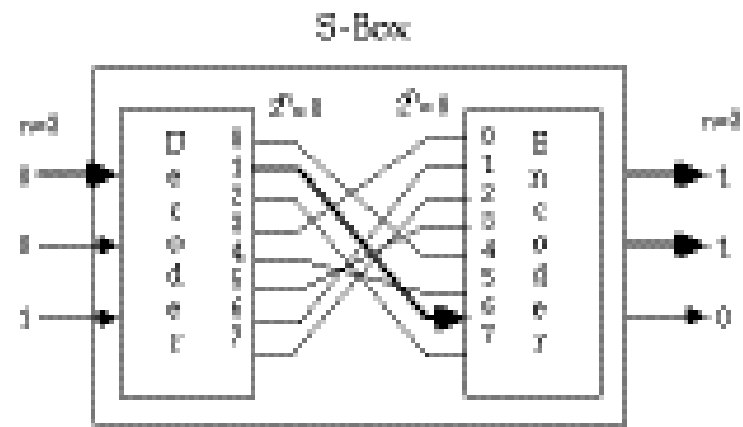


Fig 2.1 Substitution Operation

Permutation

- ❑ A binary word has its bits reordered (permuted)
- ❑ The re-ordering forms the key
- ❑ If use n bit words,
 - The key space is $n!$ (Less secure than substitution)
- ❑ This is equivalent to a wire-crossing in practice
 - (Though is much harder to do in software)
- ❑ Will call these **p-boxes**

Cont.

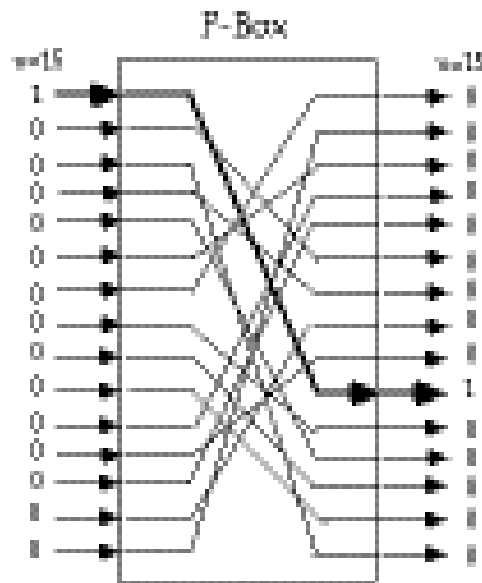


Fig 2.2 - Permutation or Transposition Function

Substitution-permutation Network

- ❑ Shannon combined these two primitives
- ❑ He called these **mixing transformations**
- ❑ A special form of product ciphers where
 - **S-boxes**
 - Provide **confusion** of input bits
 - **P-boxes**
 - Provide **diffusion** across s-box inputs

Confusion and Diffusion

❑ Confusion

- A technique that seeks to make the relationship between the statistics of the ciphertext and the value of the encryption keys as complex as possible. Cipher uses key and plaintext.

❑ Diffusion

- A technique that seeks to obscure the statistical structure of the plaintext by spreading out the influence of each individual plaintext digit over many ciphertext digits.

Desired Effect

❑ Avalanche effect

- A characteristic of an encryption algorithm in which a small change in the plaintext gives rise to a large change in the ciphertext
- Best: changing *one* input bit results in changes of approx *half* the output bits

❑ Completeness effect

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

Practical Substitution-permutation Networks

- ❑ In practice 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

Feistel Cipher

- ❑ Invented by Horst Feistel,
 - working at IBM Thomas J Watson research labs in early 70's,
- ❑ The idea is to partition the input block into two halves, $l(i-1)$ and $r(i-1)$,
 - 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 i th subkey

Cont.

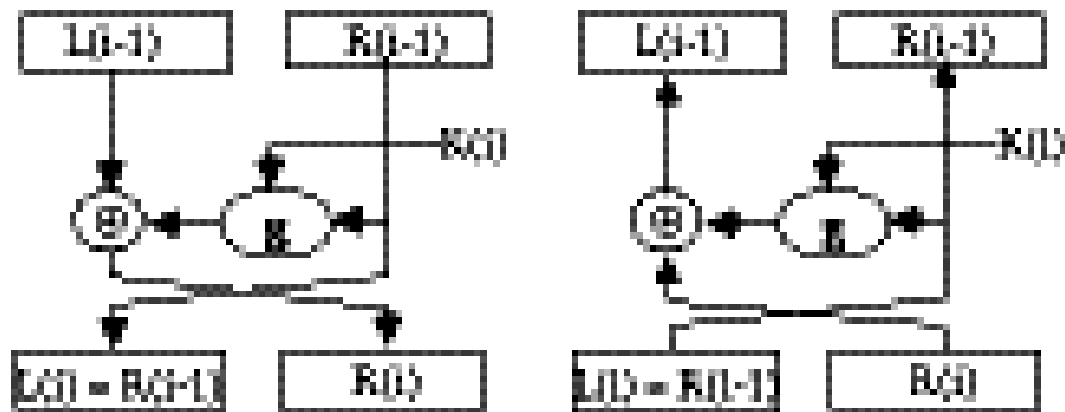


Fig 2.4 - A Round of a Feistel Cipher

Cont.

- ❑ This can be described functionally as:
 - $L(i) = R(i-1)$
 - $R(i) = L(i-1) \oplus g(k(i), R(i-1))$
- ❑ This can easily be reversed as seen in the above diagram, working backwards through the rounds
- ❑ In practice link a number of these stages together (typically 16 rounds) to form the full cipher

Data Encryption Standard

- ❑ Adopted in 1977 by the National Bureau of Standards, now the National Institute of Standards and Technology
- ❑ Data are encrypted in 64-bit blocks using a 56-bit key
- ❑ The same algorithm is used for decryption.
- ❑ Subject to much controversy

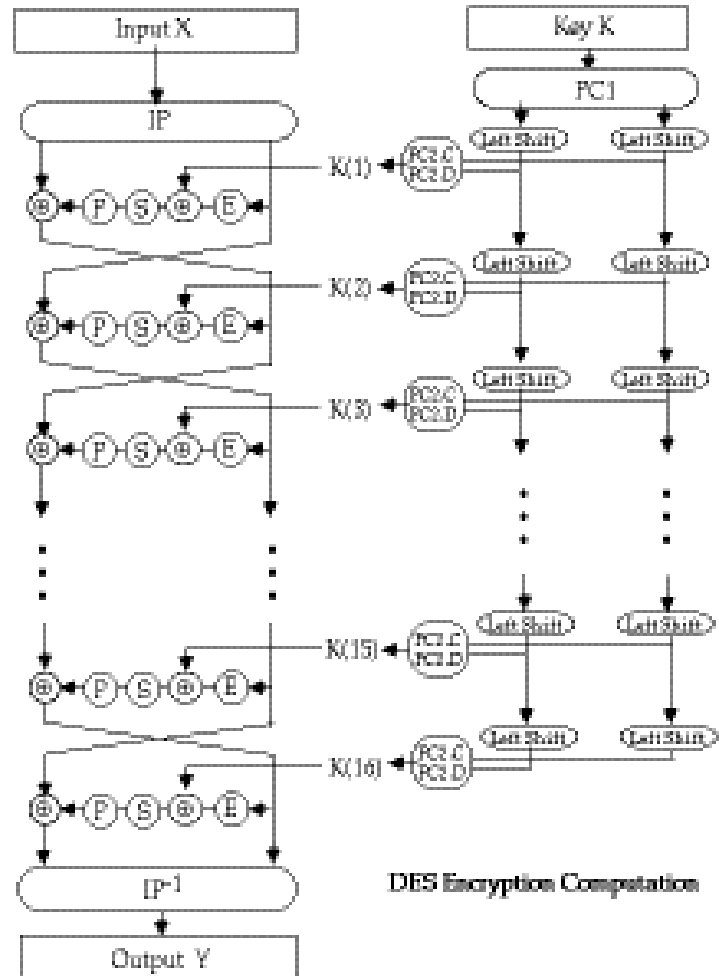
History

- ❑ IBM LUCIFER 60's
 - Uses 128 bits key
- ❑ Proposal for NBS, 1973
- ❑ Adopted by NBS, 1977
 - Uses only 56 bits key
 - Possible brute force attack
 - Design of S-boxes was classified
 - Hidden weak points in in S-Boxes?
 - Wiener (93) claim to be able to build a machine at \$100,00 and break DES in 1.5 days

DES

- ❑ DES encrypts 64-bit blocks of data, using a 56-bit key
- ❑ the basic process consists of:
 - an initial permutation (IP)
 - 16 rounds of a complex key dependent calculation f
 - a final permutation, being the inverse of IP
- ❑ Function f can be described as
 - $L(i) = R(i-1)$
 - $R(i) = L(i-1) \oplus P(S(E(R(i-1)) \oplus K(i)))$

DES

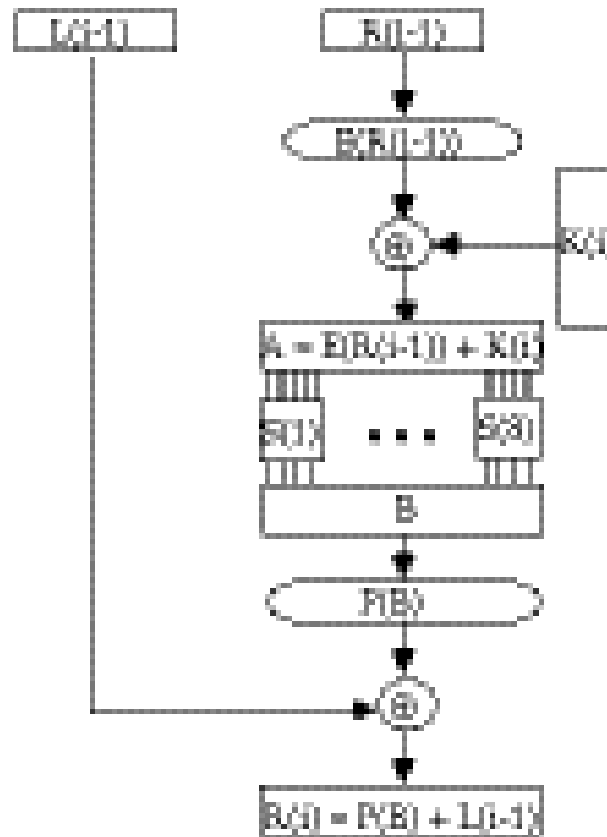


Initial and Final Permutations

□ Inverse Permutations

40	8	48	16	56	24	64	32
39	7	47	15	55	23	63	31
38	6	46	14	54	22	62	30
37	5	45	13	53	21	61	29
36	4	44	12	52	20	60	28
35	3	43	11	51	19	59	27
34	2	42	10	50	18	58	26
33	1	41	9	49	17	57	25

Function f



Expansion Table

- Expands the 32 bit data to 48 bits
 - $\text{Result}(i) = \text{input}(\text{array}(i))$

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

S-Boxes

- S-Box is a fixed 4 by 16 array
- Given 6-bits $B=b_1b_2b_3b_4b_5b_6$,
 - Row $r=b_1b_6$
 - Column $c=b_2b_3b_4b_5$
 - $S(B)=S(r,c)$ written in binary of length 4

Example

□ S-Box 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

Permutation Table

- The permutation after each round

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

Subkey Generation

- ❑ Given a 64 bits key (with parity-check bit)
 - Discard the parity-check bits
 - Permute the remaining bits using fixed table P1
 - Let C_0D_0 be the result (total 56 bits)
- ❑ Let $C_i = Shift_i(C_{i-1})$; $D_i = Shift_i(D_{i-1})$ and K_i be another permutation P2 of C_iD_i (total 56 bits)
 - Where cyclic shift one position left if $i=1,2,9,16$
 - Else cyclic shift two positions left

Permutation Tables

57	49	41	33	25	17	9
1	58	50	42	34	26	18
10	2	59	51	43	35	27
19	11	3	60	52	44	36
63	55	47	39	31	23	15
7	62	54	47	38	30	22
14	6	61	53	45	37	29
21	13	5	28	20	12	4

Permutation table P1

14	17	11	24	1	5
3	28	15	6	21	10
23	19	12	4	26	8
16	7	27	20	13	2
41	52	31	37	47	55
30	40	51	45	33	48
44	49	39	56	34	53
46	42	50	36	29	32

Permutation table P2

DES in Practice

- ❑ DEC (Digital Equipment Corp. 1992) built a chip with 50k transistors
 - Encrypt at the rate of 1G/second
 - Clock rate 250 Mhz
 - Cost about \$300
- ❑ Applications
 - ATM transactions (encrypting PIN and so on)

Model

- ❑ Mode of use
 - The way we use a block cipher
 - Four have been defined for the DES by ANSI in the standard: ANSI X3.106-1983 modes of use)
- ❑ Block modes
 - Splits messages in blocks (ECB, CBC)
- ❑ Stream modes
 - On bit stream messages (CFB, OFB)

Block Modes

❑ Electronic Codebook Book (ECB)

- where the message is broken into independent 64-bit blocks which are encrypted
- $C_i = \text{DES}_{K1}(P_i)$

❑ 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
- $C_i = \text{DES}_{K1}(P_i \oplus C_{i-1})$
- $C_{-1} = \text{IV}$ (initial value)

Stream Model

□ Cipher FeedBack (CFB)

- where the message is treated as a stream of bits, added to the output of the DES, with the result being feed back for the next stage
- $C_i = P_i \oplus \text{DES}_{K1}(C_{i-1})$
- $C_{-1} = \text{IV}$ (initial value)

Cont.

□ 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
- $C_i = P_i \oplus O_i$
- $O_i = \text{DES}_{K1}(O_{i-1})$
- $O_{-1} = \text{IV}$ (initial value)

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:

Cont.

❑ **Weak keys**

- The 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

Cont.

- ❑ None of these causes a problem since they are a tiny fraction of all available keys
- ❑ However they **MUST** be avoided by any key generation program

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

Double DES?

- ❑ Using two encryption stages and two keys
 - $C = E_{k_2}(E_{k_1}(P))$
 - $P = D_{k_1}(D_{k_2}(C))$
- ❑ It is proved that there is no key k_3 such that
 - $C = E_{k_2}(E_{k_1}(P)) = E_{k_3}(P)$
- ❑ But Meet-in-the-middle attack

Meet-in-the-Middle Attack

- ❑ Assume $C = E_{k_2}(E_{k_1}(P))$
- ❑ Given the plaintext P and ciphertext C
- ❑ Encrypt P using all possible keys k_1
- ❑ Decrypt C using all possible keys k_2
 - Check the result with the encrypted plaintext lists
 - If found match, they test the found keys again for another plaintext and ciphertext pair
 - If it turns correct, then find the keys
 - Otherwise keep decrypting C

Triple DES

- ❑ DES variant
- ❑ Standardized 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

Cont.

- ❑ No known practical attacks
- ❑ Brute force search impossible (very hard)
- ❑ Meet-in-the-middle attacks need 2^{56}
Plaintext-Ciphertext pairs per key
- ❑ Popular current alternative

IDEA:

- ❑ Developed by James Massey & Xuejia Lai at ETH originally in Zurich in 1990, then called IPES:
 - X Lai, J L Massey, "A Proposal for a New Block Encryption Standard"
 - in Advances in Cryptology - Eurocrypt '90, Lecture Notes in Computer Science, vol 473, pp 389-404,
 - X Lai, J L Massey, S Murphy, "Markov Ciphers and Differential Cryptanalysis"
 - in Advances in Cryptology - Eurocrypt '91, Lecture Notes in Computer Science, vol 547, pp 17-38,
 - name changed to IDEA in 1992

Basic Features

- ❑ Encrypts 64-bit blocks using a 128-bit key
- ❑ Based on mixing operations from different (incompatible) algebraic groups
 - XOR, $+ \text{ mod } 2^{16}$, $X \text{ mod } 2^{16} + 1$
 - On 16-bit sub-blocks, with no permutations used
- ❑ IDEA is patented in Europe & US, however non-commercial use is freely permitted
 - used in the public domain PGP (with agreement)
 - currently no attack against IDEA is known
 - Seem secure against differential cryptanalysis, brute force

Operations

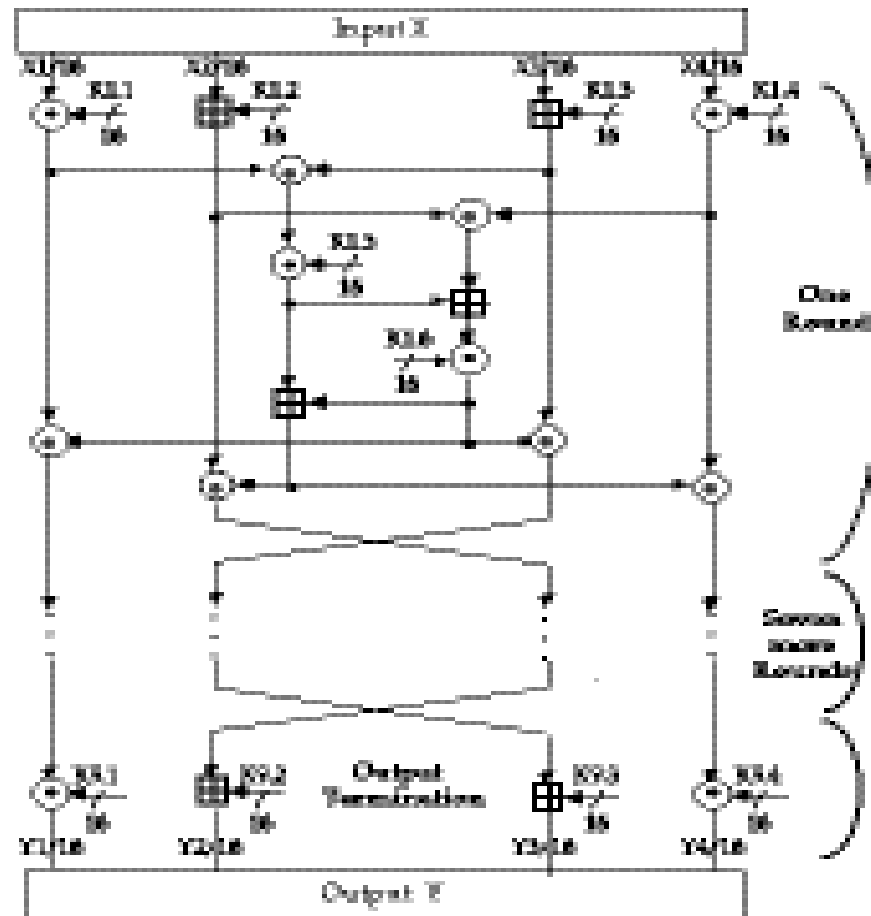
□ Operations

- XOR, Addition mod 2^{16} , multiplication mod $2^{16} + 1$
 - Why these special mod for addition, multiplication
- They do not satisfy the distributive law
- They do not satisfy the associative law

MA: multiplication/addition

- ❑ Multiplication/addition
 - Basic block to provide diffusion
 - Input of MA
 - Two sub-blocks derived from 4 input sub-blocks, 4 sub-keys
 - Two other sub-keys
 - Output
 - Two sub-blocks
 - Needs four operations
 - Four operations are the minimum to provide full diffusion

Overview



Cont.

❑ IDEA encryption works as follows:

- Use 8-rounds
- The 64-bit data is divided into: X_1 , X_2 , X_3 , X_4
- Each round
 - The sub-blocks are added (2,3), multiplied (1,4) with sub-keys
 - The results are XORed [1,3] and [2,4] to 2 sub-blocks
 - The XOR results set as input of MA structure,
 - It outputs two subblocks
 - Results are then XORed with 2,4 and 1,3 subblocks respectively
 - The second and third sub-blocks are swapped
- Finally new sub-keys are combined with the sub-blocks

Sub-Keys

- ❑ Total need $52=6*8+4$ sub-keys
 - First are directly from key in order
 - Left shift of 25 bits, and then next 8 sub-keys
 - Each sub-key is a sub-block of the original key
- ❑ Decryption
 - Much more complicated
 - It needs the inverse of the encryption key
 - For addition, multiplication

Decryption

- ❑ The process of decryption is essentially the same as encryption
 - But with different selection of sub-keys
 - Basic Operations
 - $K1.1^{-1}$ is the multiplicative inverse mod $2^{16} + 1$
 - $-K1.2$ is the additive inverse mod 2^{16}
 - 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})

Decryption Sub-Keys

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

Important Feature

- The size of the sub-block
 - Need $2^{16}+1$ be prime number
 - To compute the inverse for each possible subkey
 - So sub-block size 8 is also possible
 - $2^8+1=257$ is prime number

CAST-128

- ❑ By Carlisle Adams, Stafford Tavares
 - Defined in RFC 2144
 - Use key size varying from 40 to 128 bits
 - Structure of Feistel network
 - 16 rounds on 64-bits data block
 - Four primitive operations
 - Addition, substraction (mod 2^{32})
 - Bitwise exclusive-OR
 - Left-circular rotation

Skipjack and Clipper

□ Skipjack

- used in **Clipper** escrowed encryption scheme(US govt)
- Skipjack is a block cipher, 64-bit data
- hardware only implementation
- 80-bit key (escrowed in 2 halves)
- 32 round
- all design details and descriptions are classified
- has been very considerable debate over its use
- attack by Matt Blaze (ATT) on the LEAF component of the Clipper protocol for secure phone communications

Blowfish Scheme

- ❑ Developed by Bruce Schneier
 - Fast, compact, simple and variably secure
 - Two basic operations: addition, XOR
 - Key ranges from 32 bits to 448 bits
 - Similar to Feistel scheme
 - The sub-key and s-boxes are complicated
 - So not suitable when key changes often
 - Function g is very simple, unlike DES

RC5

- ❑ Developed by R. Rivest
 - Suitable for hardware or software
 - Fast, simple, low memory, data-dependent rotations
 - Adaptable to processors of different word length
 - A family of algorithms determined by word length, number of rounds, size of secret key
 - Decryption and encryption are not the same
 - With little variations
 - Primitive operations
 - Addition, XOR, left circular rotation

Characteristics

- ❑ Key features of advanced sym block cipher
 - Variable key length
 - Mixed operators
 - Data dependent rotation
 - Key dependent rotation
 - Key dependent S-boxes
 - Lengthy key schedule algorithm
 - Variable function F
 - Variable of number of rounds
 - Operation on both halved data each round