Bansilal Ramnath Agarwal Charitable Trust's

Vishwakarma Institute of Technology, Pune-37

Department Of Computer Engineering

Lab Manual

SUBJECT NAME : Network Security

SUBJECT CODE : CS42306

Class: - BE Branch:- Computer Engg.

Prepared By: - Prof. S. R. Shinde Year:- 2012

Contributors: - Prof. D. P. Pawar

Examinations: -

Required H/W and S/W: -Linux OS, Windows OS, Nessus, Ethereal

Contents

Experiment Number: 01

TITLE: Implement Caesar Cipher

OBJECTIVES:

To Implement a Caesar Cipher Algorithm to encrypt the given text

ALGORITHMS

The Caesar cipher replaces each alphabet in a text by the alphabet k positions away (in the modulo 26 sense).
For $k = 3$, the substitutions are

D for A, *E* for B, . . . *A* for X, *B* for Y,

etc.

So, "WHAT IS YOUR NAME" becomes

"ZKDW LV BRXU QDPH"

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

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

step 1: Input the text to be encrypted
step 2: Select the Key (randomly) for encryption
step 3: find the index of each letter from the input
step 4: increment the index of each letter by Key
step 5: replace the characters at each place by the characters at new index value.

APPLICATIONS

By replacing these characters we can encrypt the message and send over the network.

FAQS

1. **What is effect of brute force attack on Ceaser Cipher.**
2. **What is the complexity of your algorithm.**

Experiment Number: 02

TITLE: Implement Playfair and Vigenere Cipher.

OBJECTIVES:

To Implement a Playfair and Vigenere Cipher to encrypt the given text

ALGORITHMS

Not even the large number of keys in a monoalphabetic cipher provides security

One approach to improving security was to **encrypt multiple letters**

The **Playfair Cipher** is an example

a 5X5 matrix of letters based on a keyword

fill in letters of keyword (duplicates only once)

fill rest of matrix with other letters

eg. using the keyword MONARCHY

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

plaintext is encrypted two letters at a time

1. **if a pair is a repeated letter, insert filler like 'X' balloon : ba lx lo on**
1. **if both letters fall in the same row, replace each with letter to right (wrapping back to start from end) ar-rm (rightness)**
2. **if both letters fall in the same column, replace each with the letter below it (again wrapping to top from bottom) mu-cm (belowness)**

3. **otherwise each letter is replaced by the letter in the same row and in the column of the other letter of the pair hs-bp and ea-im**

Ex. "smythework"

APPLICATIONS

By replacing the characters we can encrypt the message and send over the network so that sent message is secured.

FAQS

1. **How one can decrypt the Play fair.**
2. **What is the complexity of your algorithm?**
3. **What is advantage of Playfair over Ceaser Cipher?**

Experiment Number: 03

TITLE: Implement Hill Cipher

OBJECTIVES:

To Implement a Hill cipher Algorithm to encrypt the given text

ALGORITHMS

Encryption Algorithm:

Steps:

1. Input the text to be encrypted p
2. Make the group of two characters and find the numeric equivalence of each character
3. Let the block size of Hill Cipher m be 2 and input the keys K
4. We obtain the corresponding cipher text using $\boldsymbol{c} = pK$
5. Find out the character equivalence of the values obtained which is Cipher text.

Decryption Algorithm:

Steps:

1. Take the Cipher text as input generated by Encryption algorithm.
2. Find the matrix inverse of matrix $\begin{pmatrix} a & -b \\ c & d \end{pmatrix}$ by formula =1/ad-bc

   $\begin{pmatrix} d & b \\ -c & a \end{pmatrix}$

a. Switch the elements on the main diagonal
b. Take the opposite of the other two elements, but leave them where they are
c. Find the determinant and divide every element by that. The determinant is the product of the elements on the main diagonal minus the product of the elements off the main diagonal.

3. Then plain text be $p=cK^{-1}$

APPLICATIONS

Hill Cipher is used to encrypt highly sensitive data.

FAQS

1. **What is Hill Cipher**
2. **How to compute gcd using Euclidean algorithm.**
3. **What is the complexity of your algorithm?**

Experiment Number: 04

TITLE: Implement RC4 algorithm

OBJECTIVES:

To Implement a RC4 algorithm to encrypt the given text

ALGORITHMS

A symmetric key encryption algorithm
This is stream cipher
Normally uses 64 bit and 128 bit key sizes.
Consists of 2 parts: Key Scheduling Algorithm (KSA) & Pseudo-Random Generation Algorithm

RC4 Steps:

- Initialize an array of 256 bytes.
- Run the KSA on them
- Run the PRGA on the KSA output to generate keystream.
- XOR the data with the keystream.

Array Initialization

```
unsigned char S[256];
int i;
for(i=0; i< 256; i++)
                S[i] = i;
```

After this the array would like this :
S[] = { 0,1,2,3, ……, 254, 255}

Key Scheduling Algorithm (KSA)

- The initialized array S[256] is now run through the KSA. The KSA uses the secret key to scramble the array.
- C Code for KSA:

```
int  i, j = 0;
for(i=0; i<256; i++)
{
        j = ( j + S[i] + key[ i % key_len] ) % 256;
        swap(S[i], S[j]);
}
```

Pseudo-Random Generation Algorithm

- The KSA scrambled S[256] array is used to generate the PRGA. This is the actual keystream.
- C Code:

```
i = j = m=0;
```

```
        while(end_of_plaintext)
        {
                i = ( I + 1) % 256;
        j = ( j + S[i] ) % 256;
        swap( S[i], S[j] );
        keystream = S[ ( S[i] + S[j] ) % 256 ]
       cipher[m]=plain[m] ^ keystream;
       m++;
}
```

Encryption

- Choose a secret key
- Run the KSA and PRGA using the key to generate a keystream.
- XOR keystream with the data to generated encrypted stream.
- Transmit Encrypted stream.

Decryption

- Use the same secret key as during the encryption phase.
- Generate keystream by running the KSA and PRGA.
- XOR keystream with the encrypted text to generate the plain text.
- Logic is simple :

  (A xor B) xor B = A

  A = Plain Text or Data
  B = KeyStream

APPLICATIONS

Most popular implementation of RC4 is in 802.11 wireless networks

FAQS

4. **What is effect of KSA on the key?**
5. **What is the complexity of your algorithm?**

Experiment Number: 05

TITLE: Implement S-DES

OBJECTIVES:

To implement DES algorithm.

Input : String to encrypt

Output : Encrypted string , Decrypt the encrypted string to get original string

ALGORITHMS

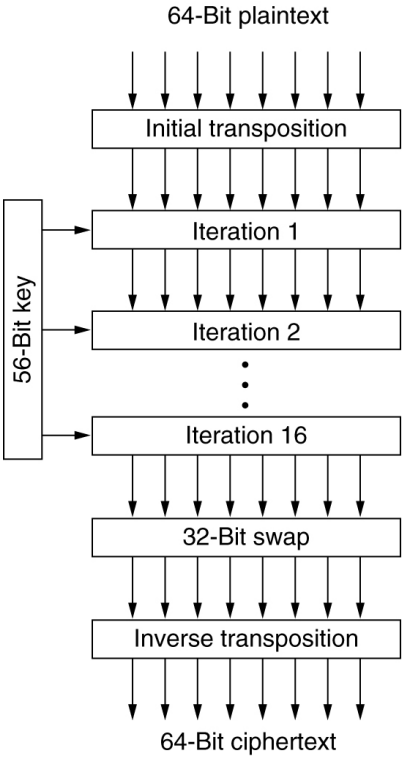
Basic unit is the bit

Encrypts block of 64 bits using a 64 bit key & outputs 64 bits of ciphertext

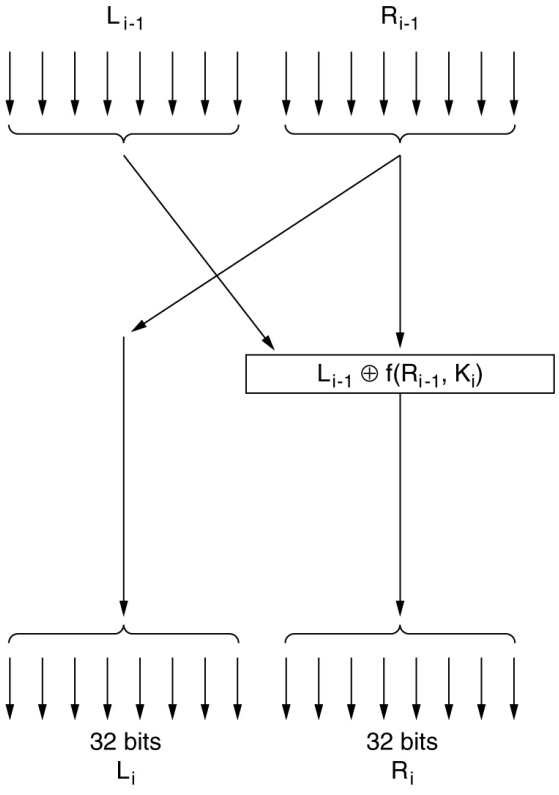
Performs both substitution and transposition (permutation) on the bits

Cipher consists of 16 rounds (iterations) each with a

Round key generated from the user-supplied key



(a)



(b)

ALGORITHM

1 Process the key.

1.1 Get a 64-bit key from the user. (Every 8th bit (the least significant bit of each byte) is considered a parity bit. For a key to have correct parity, each byte should contain an odd number of "1" bits.) This key can be entered directly, or it can be the result of hashing something else. There is no standard hashing algorithm for this purpose.

1.2 Calculate the key schedule.

1.2.1 Perform the following permutation on the 64-bit key. (The parity bits are discarded, reducing the key to 56 bits. Bit 1 (the most significant bit) of the permuted block is bit 57 of the original key, bit 2 is bit 49, and so on with bit 56 being bit 4 of the original key.)

Permuted Choice 1 (PC-1)
57 49 41 33 25 17 09
01 58 50 42 34 26 18
10 02 59 51 43 35 27
19 11 03 60 52 44 36
63 55 47 39 31 23 15
07 62 54 46 38 30 22
14 06 61 53 45 37 29
21 13 05 28 20 12 04

1.2.2 Split the permuted key into two halves. The first 28 bits are called C[0] and the last 28 bits are called D[0].

1.2.3 Calculate the 16 sub keys. Start with i = 1.

1.2.3.1 Perform one or two circular left shifts on both C[i-1] and D[i-1] to get C[i] and D[i], respectively. The number of shifts per iteration are given in the table below.

Iteration # 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16
Left Shifts 01 01 02 02 02 02 02 02 01 02 02 02 02 02 02 01

1.2.3.2 Permute the concatenation C[i]D[i] as indicated below. This will yield K[i], which is 48 bits long.

Permuted Choice 2 (PC-2)
14 17 11 24 01 05
03 28 15 06 21 10
23 19 12 04 26 08
16 07 27 20 13 02
41 52 31 37 47 55
30 40 51 45 33 48
44 49 39 56 34 53
46 42 50 36 29 32

1.2.3.3 Loop back to 1.2.3.1 until K[16] has been calculated.

2 Process a 64-bit data block.

2.1 Get a 64-bit data block. If the block is shorter than 64 bits, it should be padded as appropriate for the application.

2.2 Perform the following permutation on the data block.

Initial Permutation (IP)

58 50 42 34 26 18 10 02
60 52 44 36 28 20 12 04
62 54 46 38 30 22 14 06
64 56 48 40 32 24 16 08
57 49 41 33 25 17 09 01
59 51 43 35 27 19 11 03
61 53 45 37 29 21 13 05
63 55 47 39 31 23 15 07

2.3 Split the block into two halves. The first 32 bits are called L[0], and the last 32 bits are called R[0].

2.4 Apply the 16 sub keys to the data block. Start with i = 1.

2.4.1 Expand the 32-bit R[i-1] into 48 bits according to the bit-selection function below.

Expansion (E)

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

2.4.2 Exclusive-or E(R[i-1]) with K[i].

2.4.3 Break E(R[i-1]) xor K[i] into eight 6-bit blocks. Bits 1-6 are B[1], bits 7-12 are B[2], and so on with bits 43-48 being B[8].

2.4.4 Substitute the values found in the S-boxes for all B[j]. Start with j = 1. All values in the S-boxes should be considered 4 bits wide.

2.4.4.1 Take the 1st and 6th bits of B[j] together as a 2-bit value (call it m) indicating the row in S[j] to look in for the substitution.

2.4.4.2 Take the 2nd through 5th bits of B[j] together as a 4-bit value (call it n) indicating the column in S[j] to find the substitution.

2.4.4.3 Replace B[j] with S[j][m][n].

Substitution Box 1

(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

2.4.4.4 Loop back to 2.4.4.1 until all 8 blocks have been replaced.

2.4.5 Permute the concatenation of B[1] through B[8] as indicated below.

Permutation P
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

2.4.6 Exclusive-or the resulting value with L[i-1]. Thus, all together, your R[i] = L[i-1] xor P(S[1](B[1])...S[8](B[8])), where B[j] is a 6-bit block of E(R[i-1]) xor K[i]. (The function for R[i] is more concisely written as, R[i] = L[i-1] xor f(R[i-1], K[i]).)

2.4.7 L[i] = R[i-1].

2.4.8 Loop back to 2.4.1 until K[16] has been applied.

2.5 Perform the following permutation on the block R[16]L[16]. (Note that block R precedes block L this time.)

Final Permutation (IP**-1)
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

To decrypt, use the same process, but just use the keys K[i] in reverse order. That is, instead of applying K[1] for the first iteration, apply K[16], and then K[15] for the second, on down to K[1].
Short description of algorithm

Key schedule :

C[0]D[0] = PC1(key)
for 1 <= i <= 16
C[i] = LS[i](C[i-1])
D[i] = LS[i](D[i-1])
K[i] = PC2(C[i]D[i])

Encryption :
L[0]R[0] = IP(plain block)
for 1 <= i <= 16
L[i] = R[i-1]
R[i] = L[i-1] xor f(R[i-1], K[i])
cipher block = FP(R[16]L[16])

Decryption :

R[16]L[16] = IP(cipher block)
for 1 <= i <= 16
R[i-1] = L[i]
L[i-1] = R[i] xor f(L[i], K[i])
plain block = FP(L[0]R[0])

APPLICATIONS
DES is most popular and widely used now a day in all application for encryption.

FAQS

1. **How to calculate f function.**
2. **Why the bits are swapped?**
3. **How to calculate expansion permutation?**

Experiment Number: 06

TITLE: Implement S-AES

OBJECTIVES:

To Implement a AES Algorithm to encrypt the given text

ALGORITHMS

AES is a symmetric block cipher.
The plain text and the cipher text are the same size.

AES Block

AES has a fixed block size of 128 bits called a *state*

ABCDEFGHIJKLMNOP

A E I M	41 45 49 4D
B F J N	42 46 4A 4E
C G K O	43 47 4B 4F
D H L P	44 48 4C 50

AES key is either 128 bits, 192 bits or 256 bits

128 bits (4 words):

1122334455667889900AABBCCDDEEFF
11 22 33 44
55 66 77 88
99 00 AA BB
CC DD EE FF

or **192 bits (6 words)**
1122334455667788
9900AABBCCDDEEFF
1122334455667788
11 22 33 44
55 66 77 88
99 00 AA BB
CC DD EE FF
11 22 33 44
55 66 77 88

or 256 bits (8 words)
1122334455667788
9900AABBCCDDEEFF
1122334455667788
9900AABBCCDDEEFF
11 22 33 44
55 66 77 88

99 00 AA BB
CC DD EE FF
11 22 33 44
55 66 77 88
99 00 AA BB
CC DD EE FF

AES Operation:

AES Operates on the binary field GF(2^8).

- This can be represented as a polynomial b(x) with binary coefficients $b\phi \{0,1\}$:

$$b_7x^7 + b_6x^6 + b_5x^5 + b_4x^4 + b_3x^3 + b_2x^2 + b_1x + b_0$$

Multiplication in GF(2^8) consists of multiplying two polynomials modulo an irreducible polynomial of degree 8.

AES uses the following irreducible polynomial

$$m(x) = x^8 + x^4 + x^3 + x + 1$$

AES Algorithm:

```
KeyExpansion(byte key[4*Nk], word w[Nb*(Nr+1)],Nk)

Cipher(byte in[4*Nb],byte out[4*Nb],word w[Nb*(Nr+1)])
begin
     byte state[4,Nb]
     state = in
     AddRoundKey(state, w[0, Nb-1])
     for round = 1 step 1 to Nr-1
          SubBytes(state)
          ShiftRows(state)
          MixColumns(state)
          AddRoundKey(state,w[round*Nb,(round+1)*Nb-1])
     end for
     SubBytes(state)
     ShiftRows(state)
     AddRoundKey(state, w[Nr*Nb, (Nr+1)*Nb-1])
     out = state
end
```

Key Expanxion(sample example):
Sample Key:

112233445566778899 00AABBCCDDEEFF

The first 4 (N_k) words are set equal to the key

w[0]	11	22	33	44
w[1]	55	66	77	88
w[2]	99	00	AA	BB
w[3]	CC	DD	EE	FF

For words 4 through 43

i = N_k // N_k = 4
while (i < N_b*(N_r+1)) { // N_b*(N_r+1)= 4*(10+1)= 44 temp = w[i – 1]
If (i%N_k == 0)
rotate word left 1 byte
process each byte through sbox
XOR with RCON[i/N_k-1] // just first byte of w[i]
w[i] = w[i-4] XOR temp
i++}

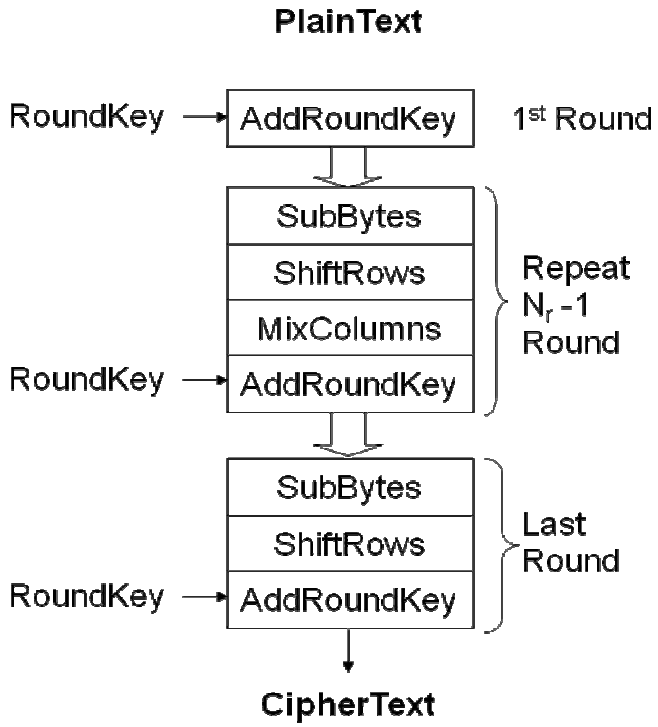
i = N_k // N_k = 4 i=4
while (i < N_b*(N_r+1)) { // N_b*(N_r+1)= 4*(10+1)= 44
temp = w[i - 1] temp = w[3] = CC DD EE FF

If (i%N_k == 0) temp = CC DD EE FF

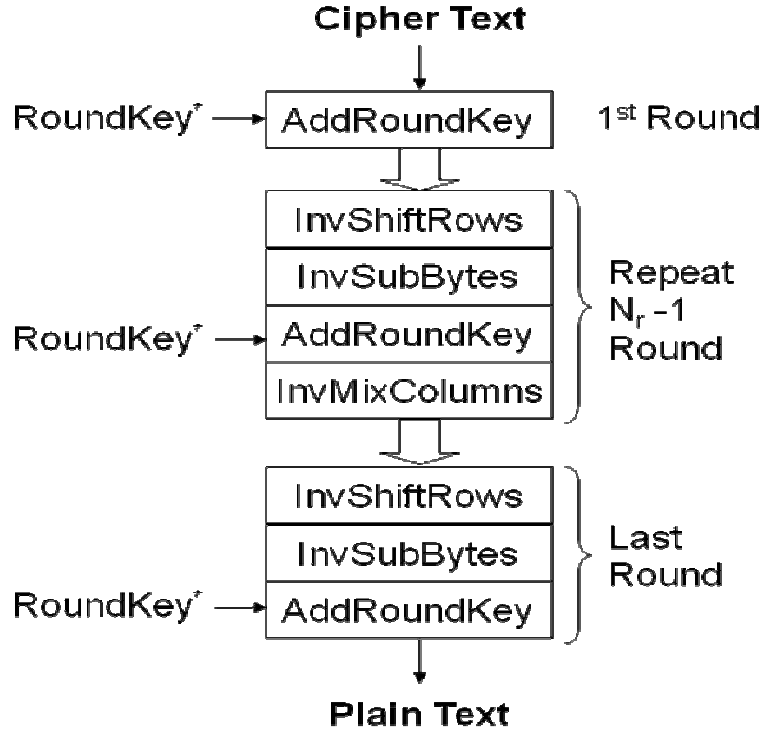
rotate word left 1 byte temp = DD EE FF CC
process each byte through sbox
temp = sbox[DD] sbox[EE] sbox[FF] sbox[CC]
= C1 28 16 4B

XOR with RCON[i/N_k-1] RCON[0] = 01
temp = (C1 01) 28 16 4B
temp = C0 28 16 4B

Encryption:



Decryption:



APPLICATIONS:
Most of the Cyber Security systems are using AES.

FAQS

1. **What are the advantages of AES?**
2. **What are the limitations of AES?**
3. **Which attacks are possible on AES?**
4. **Why padding is used in AES?**

Experiment Number: 07

TITLE: Implement RSA algorithm

OBJECTIVES:

To Implement RSA Algorithm to encrypt the given text

ALGORITHMS

RSA:

- each user generates a public/private key pair by:
- selecting two large primes at random - p, q
- computing their system modulus N=p.q
  - note ø(N)=(p-1)(q-1)
- selecting at random the encryption key e
  - where $1<e<ø(N)$, $gcd(e,ø(N))=1$
- solve following equation to find decryption key d
  - $e.d=1 \bmod ø(N)$ and $0 \le d \le N$
- publish their public encryption key: KU={e,N}
- keep secret private decryption key: KR={d,p,q}
- to encrypt a message M the sender:
  - obtains **public key** of recipient KU={e,N}
  - computes: $C=M^e \bmod N$, where $0 \le M < N$
- to decrypt the ciphertext C the owner:
  - uses their private key KR={d,p,q}
  - computes: $M=C^d \bmod N$
- note that the message M must be smaller than the modulus N (block if needed)

Background:

Prime Number:

- prime numbers only have divisors of 1 and self
  - they cannot be written as a product of other numbers
  - note: 1 is prime, but is generally not of interest
- eg. 2,3,5,7 are prime, 4,6,8,9,10 are not
- to factor a number n is to write it as a product of other numbers: $n=a \times b \times c$
- two numbers a, b are relatively prime if have no common divisors apart from 1
- eg. 8 & 15 are relatively prime since factors of 8 are 1,2,4,8 and of 15 are 1,3,5,15 and 1 is the only common factor

Euler Totient Function:

- when doing arithmetic modulo n
- complete set of residues is: 0..n-1
- reduced set of residues is those numbers (residues) which are relatively prime to n
  - eg for n=10,
  - complete set of residues is {0,1,2,3,4,5,6,7,8,9}
  - reduced set of residues is {1,3,7,9}

- number of elements in reduced set of residues is called the Euler Totient Function ø(n)
- to compute ø(n) need to count number of elements to be excluded
- in general need prime factorization, but
  - for p (p prime) ø(p) = p-1
  - for p.q (p,q prime) ø(p.q) = (p-1)(q-1)
- eg.
  - ø(37) = 36
  - ø(21) = (3–1)×(7–1) = 2×6 = 12

APPLICATIONS:
This is widely used in public key cryptography

FAQS

1. **What is factorization problem?**
2. **What is Euler Totient function?**

Experiment Number: 08

TITLE: Implement Diffie-Hellman key exchange technique.

OBJECTIVES:

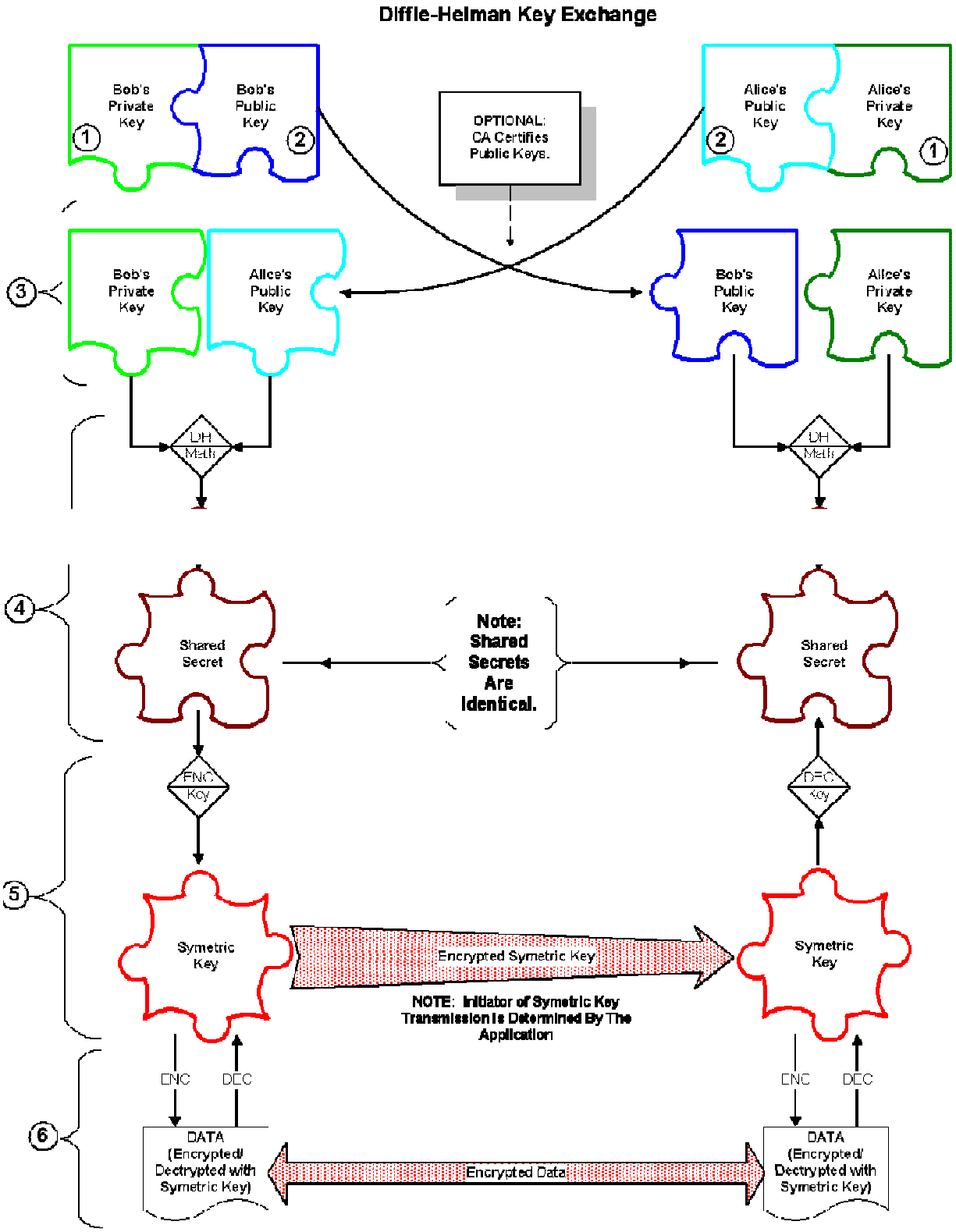
To Implement Diffie-Hellman key exchange Algorithm to encrypt the given text

ALGORITHMS

- Diffie-Hellman key agreement protocol
  - Exponential key agreement
  - Allows two users to exchange a secret key
  - Requires no prior secrets
  - Real-time over an untrusted network
  - Security of transmission is critical for many network and Internet applications
  - Requires users to share information in a way that others can't decipher the flow of information
  - Based on the difficulty of computing discrete logarithms of large numbers.
  - No known successful attack strategies*
  - Requires two large numbers, one prime (P), and (G), a primitive root of P
  - P and G are both publicly available numbers
  - P is at least 512 bits
  - Users pick private values a and b

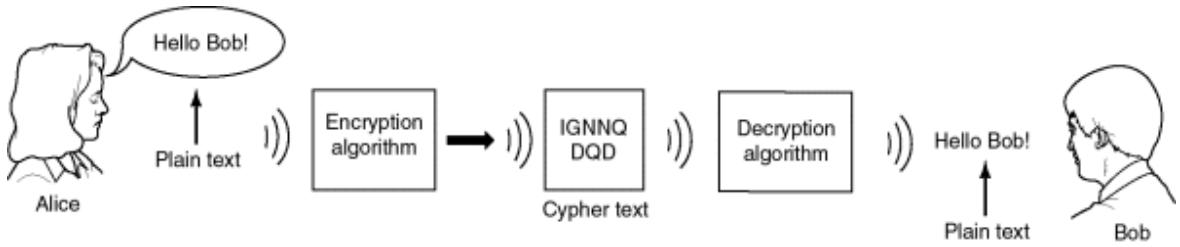
Implementation:

- Compute public values
- $x = g^a \bmod p$
- $y = g^b \bmod p$
- Public values x and y are exchanged
- Compute shared, private key
- $k_a = y^a \bmod p$
- $k_b = x^b \bmod p$
- Algebraically it can be shown that $k_a = k_b$
- Users now have a symmetric secret key to encrypt
-


Diffie-Helman Key Exchange
Bob's Private Key
Bob's Public Key
OPTIONAL: CA Certifies Public Keys.
Alice's Public Key
Alice's Private Key
Bob's Private Key
Alice's Public Key
Bob's Public Key
Alice's Private Key
DH Math
DH Math
Shared Secret
Note: Shared Secrets Are Identical.
Shared Secret
ENC Key
DEC Key
Symetric Key
Encrypted Symetric Key
Symetric Key
NOTE: Initiator of Symetric Key Transmission Is Determined By The Application
ENC
DEC
ENC
DEC
DATA (Encrypted/ Decrypted with Symetric Key)
Encrypted Data
DATA (Encrypted/ Decrypted with Symetric Key)

Example:

- Two Internet users, Alice and Bob wish to have a secure conversation.
  - They decide to use the Diffie-Hellman protocol



- Bob and Alice are unable to talk on the untrusted network.
- Alice and Bob get public numbers
  - P = 23, G = 9
  - Alice and Bob compute public values
  - $X = 9^4 \bmod 23 = 6561 \bmod 23 = 6$
  - $Y = 9^3 \bmod 23 = 729 \bmod 23 = 16$
  - Alice and Bob exchange public numbers
- Alice and Bob compute symmetric keys
  - $k_a = y^a \bmod p = 16^4 \bmod 23 = 9$
  - $k_b = x^b \bmod p = 6^3 \bmod 23 = 9$
- Alice and Bob now can talk securely!

APPLICATIONS

Diffie-Hellman is currently used in many protocols, namely:

- Secure Sockets Layer (SSL)/Transport Layer Security (TLS)
- Secure Shell (SSH)
- Internet Protocol Security (IPSec)
- Public Key Infrastructure (PKI)

FAQS:

1. How DH POP algorithm enhances IPSec Layer?
2. How this algorithm defeats middleperson attack?

Experiment Number: 09

TITLE: Implement ECC

OBJECTIVES:

To Implement ECC Algorithm to encrypt the given text

ALGORITHMS
Background:

- An *elliptic curve* is the set of solutions (x, y) to an equation of the form
  $y^2 = x^3 + ax + b$
  where $4a^3 + 27b^2 \neq 0$, together with a *point at infinity* denoted ***O***

Group Laws

- Closure
- Identity:
  $P + O = O + P = P$
- Inverse:
  $(x, y) + (x, -y) = O$
- Associativity
- Commutativity

Addition Formulae

- Let $P_1 = (x_1, y_1)$ and $P_2 = (x_2, y_2)$ be non-inverses
- Then $P_1 + P_2 = (x_3, y_3)$ where

$x_3 = (\lambda^2 - x_1 - x_2)$
$y_3 = (\lambda (x_1 - x_3) - y_1)$
and λ is the slope of the line:

  - $\lambda = (3x_1^2+a)/2y_1$ if $x_1 = x_2$

$\lambda = (y_2-y_1)/(x_2-x_1)$ otherwise

- ***Scalar multiplication*** **is repeated group addition:**

$cP = P + \cdots + P$ (c times)
where c is an integer

Algorithm: ECC Diffie Hellman key Exchange

1. Let $E_q(a,b)$ be an elliptic curve with parameters a, b, and q, where q is a prime number or an integer of the form 2^m
2. G be the point on elliptic curve whose order is large value n
3. User A Key generation
   a. Select an integer n_A less than n as a private key.
   b. Generate the public key $P_A = n_A * G$; the public key is point in $E_q(a,b)$
4. User B key generation
   a. Select an integer n_B less than n as a private key.
   b. Generate the public key $P_B = n_B * G$; the public key is point in $E_q(a,b)$

5. User A generates the secrete key $K = n_A * P_B$
6. User B generates the secrete key $K = n_B * P_A$
7. Step 5 and 6 has the same output.

APPLICATIONS

- **Wireless communication devices**
- **Smart cards**
- **Web servers that need to handle many encryption sessions**
- **Any application where security is needed but lacks the power, storage and computational power that is necessary for our current cryptosystems**

FAQS

1. How to calculate 2P, 3P, 4P
2. Explain the calculation of λ in various situations
3. What is the effect of various attacks on ECC?

Experiment Number: 10

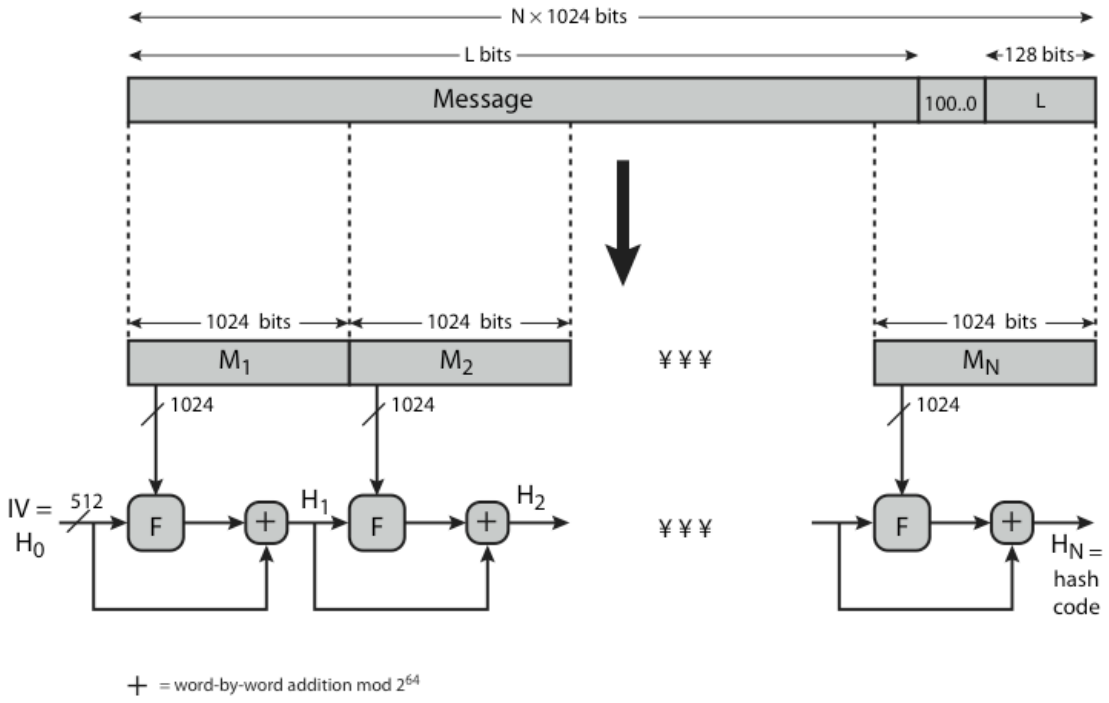
TITLE: Implement Hash Algorithm

OBJECTIVES:

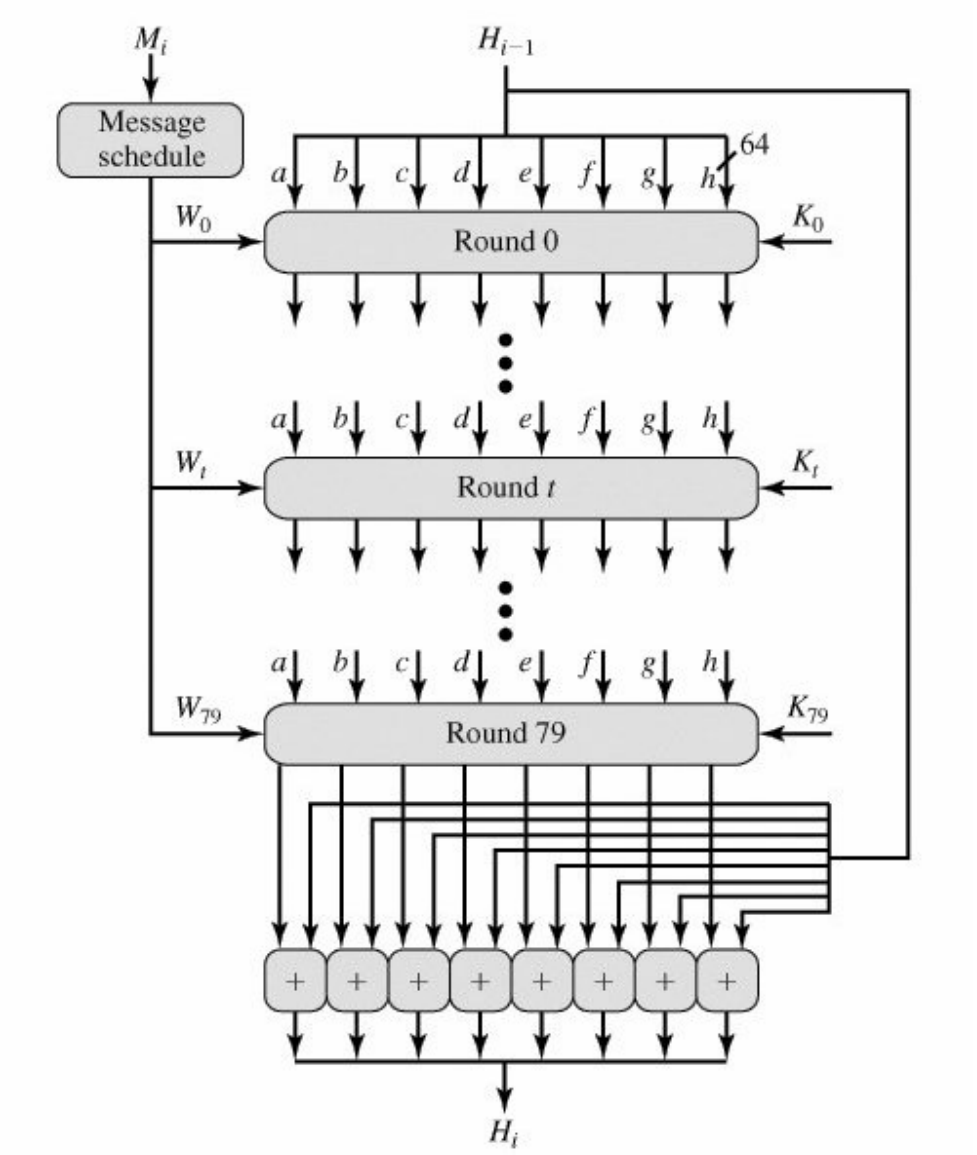
To Implement Hash Algorithm for message integrity

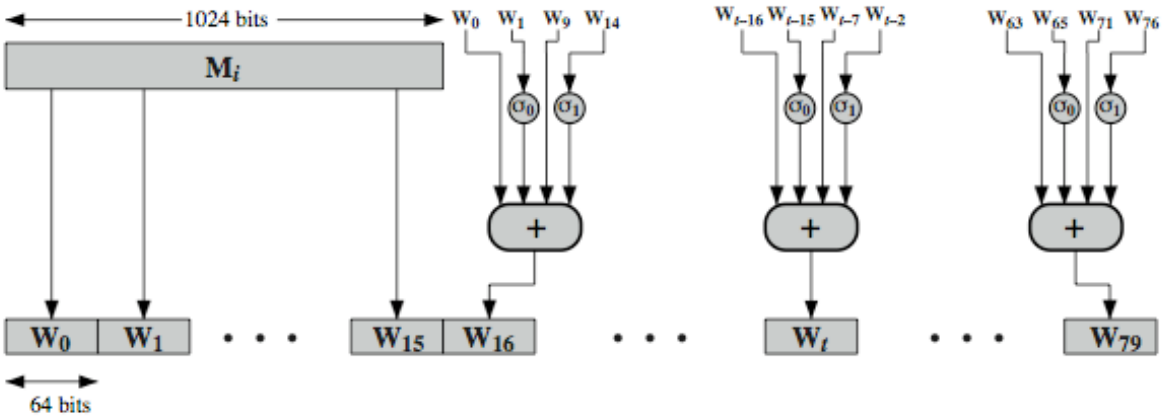
ALGORITHMS

- Hash Functions
  - condense arbitrary size message to fixed size
  - by processing message in blocks
  - through some compression function
  - either custom or block cipher based



- processing message in 1024-bit blocks
- consists of 80 rounds
  - updating a 512-bit buffer
  - using a 64-bit value Wt derived from the current message block
  - and a round constant Kt that represents the first 64 bits of the fractional parts of the cube root of first 80 prime numbers


M_i
H_{i-1}
Message schedule
64
a b c d e f g h
W_0
Round 0
K_0
W_t
Round t
K_t
W_{79}
Round 79
K_{79}
+
H_i


1024 bits
M_i
W_0 W_1 W_9 W_{14}
W_{t-16} W_{t-15} W_{t-7} W_{t-2}
W_{63} W_{65} W_{71} W_{76}
σ_0 σ_1
+
W_0 W_1 . . . W_{15} W_{16} . . . W_t . . . W_{79}
64 bits

Steps:

1. Append padding bits
2. Append length
3. Initialize hash buffer
4. Process message in 1024 bit blocks

APPLICATIONS

1. Security applications that require authentication
2. E-mail
3. Electronic funds transfer
4. Software distribution
5. Data storage

FAQS

1. How to calculate round function in secure hash
2. How the words will be formed from a block of size 1024.

Experiment Number: 11

TITLE: Implementation of Packet Sniffer (Protocol Analyzer)

OBJECTIVES:

Implement Packet sniffer (protocol analyzer) to analyze the network traffic.

ALGORITHMS

1. Putting the Ethernet card in promiscuous mode.
2. Capture all packets
3. Filter these packets according to Source IP, Destination IP , protocol and analyze the protocol.
4. Store them in File.
5. Also check output of ethereal network analyzer

The network card discarding all the packets that do not contain its own MAC address--an operation mode called non promiscuous, which basically means that each network card is reading only the frames directed to it.

Three exceptions to this rule:
1) a frame whose destination MAC address is the special broadcast address (FF:FF:FF:FF:FF:FF) will be picked up by any card

2) a frame whose destination MAC address is a multicast address will be picked up by cards that have multicast reception enabled

3) a card that has been set in **promiscuous mode** will pick up all the packets it sees on cable.

To set a network card to promiscuous mode, all we have to do is issue a particular ioctl() call to an open socket on that card. This call is only allowed for the root user for security purpose.

If ``sock" contains an already open socket then we use following code for setting mode.

```
strncpy(ethreq.ifr_name,"eth0",IFNAMSIZ);
ioctl(sock, SIOCGIFFLAGS, &ethreq);   // reads current value of flag
ethreq.ifr_flags |= IFF_PROMISC;
ioctl(sock, SIOCSIFFLAGS, &ethreq);  // sets new flag value
```

(where ethreq is an ifreq structure, defined in /usr/include/net/if.h).

SIOCGIFFLAGS - Socket I/O Control Get Interface FLAGS
SIOCSIFFLAGS - Socket I/O Control Set Interface FLAGS
IFF_PROMISC - Interface Flag for promiscuous mode

APPLICATIONS

For analyzing network and its traffic.

FAQS

4. **What is promiscuous mode ?**
5. **What are formats of different protocols (TCP,IP)?**
6. **What is advantages of sniffing ?**