LAB MANUAL

INFORMATION SECURITY

List of Lab Exercises

Syllabus Programs (JNTUK)

1) Name of the Experiment:

AIM: Working with Sniffers for monitoring network communication (Ethereal).

S/W & H/W Requirements:

S/W: Turbo C

H/W: Pentium IV Processor**,** 256 MB RAM,40 GB HDD.

Algorithm:

Step1: Start
Step2: Declare the following functions
void Handle_ICMP(char *,int)
void Handle_IGMP(char *,int)
void Handle_TCP(char *,int)
void Handle_UDP(char *,int)

Step3: declaring pointer of type ip header
void Handle_IP(char *buf)
Step4: declaring a structure which holds ip address
struct iphdr *ip_hdr
struct in_addr in
FILE *fp
Step5:Read ctl,len
Step6: ip_hdr=(struct iphdr *)(buf+14)

Step7: fp=fopen("./output/ip.txt","a")

Step8: if(!fp)
perror("fopen")
exit(1)

Step9: output(fp,"\t ----------\n")
Step10:system("date>> ./output/ip.txt")
output(fp,"type of service : %d\n",ip_hdr->tos)
output(fp,"protocol id is %d\n",ip_hdr->protocol)
output(fp,"total len %d\n",ntohs(ip_hdr->tot_len))
output(fp,"fragment offset %d\n",ntohs(ip_hdr->frag_off))

Step11: storing ip address in
Step12: struct in_addr so that we can use inet_ntoa to convert
the same into ascii string
in.s_addr=ip_hdr->saddr

fprintf(fp,"source address is %s\n",inet_ntoa(in))

Step13: in.s_addr=ip_hdr->daddr
Step14: output(fp,"destination address is %s\n",inet_ntoa(in))
output(fp,"\t ----------\n")
Step15: ctl=ip_hdr->protocol

Step16: The following statement is used to calculate the combined length of data link header and ip header
len=(ip_hdr->ihl<<2)+14

Step17: The following switch statement can be used to invoke appropriate handle function based on protocol id found in ip header
switch(ctl)
case ICMP
Handle_ICMP(buf,len);
break
case IGMP:
Handle_IGMP(buf,len);
break
case TCP
Handle_TCP(buf,len);
break
case UDP
Handle_UDP(buf,len);
break
fclose(fp)
Step18: This file is a part of sniffer, a packet capture utility and Network monitor.
void Handle_IP(char *)
void Handle_ARP(char *)
void Handle_RARP(char *)
Step19: declaring a structure which hold link level header information
int main()
struct sockaddr_ll sa
Step20: pointer for ethernet header
struct ethhdr *eth_hdr
int sockfd,sl,retval,ctl
char buf[2000],*hadd
FILE *fp;
Step21: The following system call creates raw socket which captures all protocols
sockfd=socket(PF_PACKET,SOCK_RAW,htons(ETH_P_ALL));
Step22: if(sockfd<0)
perror("socket")

```
fp=fopen("./output/ether.txt","a")
if(!fp)
perror("fopen")
```
Step23: `setvbuf(fp,NULL,_IONBF,0)`

Step24: The following infinite while loop is used to continuously capture packets and afterwards the packet will be passed to appropriate handler

```
while(1)
sl=sizeof(struct sockaddr_ll);
retval=recvfrom(sockfd,buf,2000,0,(struct sockaddr *)&sa,&sl)
if(retval==-1)
perror("recvfrom")
eth_hdr=(struct ethhdr *) buf
output(fp,"\t ----------------------------\n")
system("date>>./output/ether.txt")
hadd=ether_ntoa((struct ether_addr *)eth_hdr->h_source)
fprintf(fp,"source ethernet address is %s\n",hadd)
hadd=ether_ntoa((struct ether_addr *)eth_hdr->h_dest)
fprintf(fp,"destination ethernet address is %s\n",hadd)
fprintf(fp,"packet protocol id is %x\n",ntohs(eth_hdr->h_proto))
fprintf(fp,"\t ----------------------------\n")
ctl=ntohs(eth_hdr->h_proto)
```
Step25:
```
switch(ctl)
case 0x0800:
print("received IP packet\n")
Handle_IP(buf)
break

case 0x0806
print("received ARP packet\n")
Handle_ARP(buf)
break

case 0x8035
print("received RARP packet\n")
Handle_RARP(buf)
break
fclose(fp)
```

Step26: This file is a part of sniffer, a packet capture utility and Network Monitor

```
void Handle_ARP(char *buf)
```
Step27:
```
declared pointer of type arp header
struct ether_arp *arp_hdr
```

```
struct in_addr in
char *hadd
FILE *fp
```

Step28: In the following statement we're adjusting the offset so arp header pointer points to currect location

```
arp_hdr=(struct ether_arp *) (buf+14)
fp=fopen("./output/arp.txt","a")
if(!fp)
        perror("fopen")
print(fp,"\t ---------------------------------\n")
system("date>> ./output/arp.txt")
print(fp,"hardware type : %u\n",ntohs(arp_hdr->ea_hdr.ar_hrd))
print(fp,"protocol type : %u\n",ntohs(arp_hdr->ea_hdr.ar_pro))
printf(fp,"hardware length : %d\n",arp_hdr->ea_hdr.ar_hln)
print(fp,"protocol length : %d\n",arp_hdr->ea_hdr.ar_pln)
print(fp,"operation code  : %u\n",ntohs(arp_hdr->ea_hdr.ar_op))
hadd=ether_ntoa((struct ether_addr *)arp_hdr->arp_sha)
print(fp,"sender hardware address : %s\n",hadd)
bcopy(arp_hdr->arp_spa,&in.s_addr,4)
print(fp,"sender protocol address : %s\n",inet_ntoa(in))
bcopy(arp_hdr->arp_tpa,&in.s_addr,4)
print(fp,"target protocol address : %s\n",inet_ntoa(in))
print(fp,"\t ---------------------------------\n")
fclose(fp)
```

Step29: This file is a part of sniffer, a packet capture utility and Network Monitor

```
void Handle_RARP(char *buf)
```

Step30: declaring a pointer of type arp header

```
struct ether_arp *rarp_hdr
struct in_addr in
char *hadd
FILE *fp
```

Step31: The following statement can be used to adjust offset so that arp header points to currect location

```
rarp_hdr=(struct ether_arp *) (buf+14)
fp=fopen("./output/rarp.txt","a")
if(!fp)
perror("fopen")
print(fp,"\t ---------------------------------\n")
system("date>> ./output/rarp.txt")
print(fp,"hardware type : %u\n",ntohs(rarp_hdr->ea_hdr.ar_hrd))
print(fp,"protocol type : %u\n",ntohs(rarp_hdr->ea_hdr.ar_pro))
```

```
print(fp,"hardware length : %d\n",rarp_hdr->ea_hdr.ar_hln)
print(fp,"protocol length : %d\n",rarp_hdr->ea_hdr.ar_pln)
print(fp,"operation code  : %u\n",ntohs(rarp_hdr->ea_hdr.ar_op))
hadd=ether_ntoa((struct ether_addr *)rarp_hdr->arp_sha)
print(fp,"sender hardware address : %s\n",hadd)
bcopy(rarp_hdr->arp_spa,&in.s_addr,4)
print(fp,"sender protocol address : %s\n",inet_ntoa(in))
hadd=ether_ntoa((struct ether_addr *)rarp_hdr->arp_tha)
print(fp,"target harware address : %s\n",hadd)
print(fp,"\t ----------------------------------\n")
fclose(fp)
```

Step32: This file is a part of a sniffer, a packet capture utility and Network monitor

```
void Handle_ICMP(char *,int)
void Handle_IGMP(char *,int)
void Handle_TCP(char *,int)
void Handle_UDP(char *,int)
void Handle_IP(char *buf)
```

Step33: declaring pointer of type ip header

```
struct iphdr *ip_hdr
```

Step34: declaring a structure which holds ip address

```
struct in_addr in
FILE *fp
int ctl,len
```

Step35: In the following statement we're adjusting the offset so that ip pointer can point to currect location

```
ip_hdr=(struct iphdr * )(buf+14)
fp=fopen("./output/ip.txt","a")
if(!fp)
        perror("fopen")
print(fp,"\t ------------------------------\n")
system("date>> ./output/ip.txt")
print(fp,"type of service : %d\n",ip_hdr->tos)
print(fp,"protocol id is %d\n",ip_hdr->protocol)
print(fp,"total len %d\n",ntohs(ip_hdr->tot_len))
print(fp,"fragment offset %d\n",ntohs(ip_hdr->frag_off))
```

Step36: In the following statement we're storing ip address in struct in_addr so that we can use inet_ntoa to convert the same into ascii string

```
in.s_addr=ip_hdr->saddr
print(fp,"source address is %s\n",inet_ntoa(in))
```

```
in.s_addr=ip_hdr->daddr
print(fp,"destination address is %s\n",inet_ntoa(in))
print(fp,"\t ----------\n")
ctl=ip_hdr->protocol
```

Step37: The following statement is used to calculate the combined length of data link header and ip header

```
len=(ip_hdr->ihl<<2)+14
```

Step38: The following switch statement can be used to invoke appropriate handle function based on protocol id found in ip header

```
switch(ctl)
case ICMP:
Handle_ICMP(buf,len);
break;
case IGMP:
Handle_IGMP(buf,len);
break;
case TCP:
Handle_TCP(buf,len);
break;
case UDP:
Handle_UDP(buf,len);
break;
```

Step39: This file is part of sniffer, a packet capturing utility and Network Monitor

```
void Handle_ICMP(char *buf,int len)
```

Step40: declaring pointer of type icmp header

```
struct icmphdr *icmp_hdr
FILE *fp
fp=fopen("./output/icmp.txt","a")
if(!fp)
perror("icmp.fopen")
```

Step41: The following header is used to adjust offset so that icmp header can point to currect location

```
icmp_hdr=(struct icmphdr *) (buf+len);
print(fp,"\t ----------\n")
system("date>> ./output/icmp.txt")
print(fp,"message type : %d\n",icmp_hdr->type)
print(fp,"sub-code : %d\n",icmp_hdr->code)
print(fp,"\t ----------\n")
```

Step 42: void Handle_IGMP(char *buf,int len)
declaring pointer of type igmp header

```
struct igmp *igmp_hdr
declaring a structure to store ip address
struct in_addr in
FILE *fp
igmp_hdr=(struct igmp *) (buf+len)
fp=fopen("./output/igmp.txt","a")
if(!fp)
perror("igmp.fopen")
print(fp,"\t ----------------------------------\n")
system("date>> ./output/igmp.txt")
print(fp,"igmp type : %d\n",igmp_hdr->igmp_type)
```

Step43: Storing ip address in struct in_addr so that we can execute inet_ntoa() to retreive the same in string format

```
in.s_addr=igmp_hdr->igmp_group.s_addr;
print(fp,"igmp group : %s\n",inet_ntoa(in)
print(fp,"\t ----------------------------------\n")
```

Step44: void Handle_TCP(char *buf,int len)

```
struct tcphdr *tcp_hdr; // declaring a pointer of type tcp header
FILE *fp
fp=fopen("./output/tcp.txt","a")
if(!fp)
perror("tcp.fopen")
```

Step45: The following statement is used to adjust the offset so that tcp header pointer can point to currect location

```
tcp_hdr=(struct tcphdr *) (buf+len)
print(fp,"\t ----------------------------------\n")
system("date>> ./output/tcp.txt")
print(fp,"source port no : %u\n",ntohs(tcp_hdr->source))
print(fp,"destination port no : %u\n",ntohs(tcp_hdr->dest))
print(fp,"seq number : %lu\n",ntohl(tcp_hdr->seq))
print(fp,"ack number : %lu\n",ntohl(tcp_hdr->ack_seq))
print(fp,"window size : %u\n",ntohs(tcp_hdr->window))
print(fp,"\t ----------------------------------\n")
```

Step 46: declaring a pointer of type udp header
void Handle_UDP(char *buf,int len)
struct udphdr *udp_hdr
FILE *fp
fp=fopen("./output/udp.txt","a")
if(!fp)
perror("udp.fopen")

Step47: The following statement is used to adjust the offset so that udp header can point to currect location
udp_hdr=(struct udphdr*) (buf+len)
print(fp,"\t ----------\n")
system("date>>./output/udp.txt")
print(fp,"source port no : %u\n",ntohs(udp_hdr->source))
print(fp,"destination port no : %u\n",ntohs(udp_hdr->dest))
print(fp,"header length : %u\n",ntohs(udp_hdr->len))
print(fp,"\t ----------\n")

Step48: Stop

Test Data:

Valid Data Set: Address has to be given

Invalid Data Set: Invalid address

Limiting Data Set: Not applicable

RESULT:

Fri Jul 16 20:03:51 IST 2004
---
hardware type : 1
protocol type : 2048
hardware length : 6
protocol length : 4
operation code : 1
sender hardware address : 0:2:44:10:f6:41
sender protocol address : 192.168.100.5
target protocol address : 192.168.100.254

Fri Jul 16 20:03:52 IST 2004
---
hardware type : 1
protocol type : 2048

hardware length : 6
protocol length : 4
operation code : 1
sender hardware address : 0:2:44:10:f6:41
sender protocol address : 192.168.100.5
target protocol address : 192.168.100.254

Fri Jul 16 20:03:53 IST 2004

---

hardware type : 1
protocol type : 2048
hardware length : 6
protocol length : 4
operation code : 1
sender hardware address : 0:2:44:10:f6:41
sender protocol address : 192.168.100.5
target protocol address : 192.168.100.254

Fri Jul 16 20:03:55 IST 2004

---

hardware type : 1
protocol type : 2048
hardware length : 6
protocol length : 4
operation code : 1
sender hardware address : 0:2:44:10:f6:41
sender protocol address : 192.168.100.5
target protocol address : 192.168.100.254

Fri Jul 16 20:03:56 IST 2004

---

hardware type : 1
protocol type : 2048
hardware length : 6
protocol length : 4
operation code : 1
sender hardware address : 0:2:44:10:f6:41
sender protocol address : 192.168.100.5
target protocol address : 192.168.100.254

Fri Jul 16 20:03:57 IST 2004

---

hardware type : 1
protocol type : 2048
hardware length : 6

protocol length : 4
operation code : 1
sender hardware address : 0:2:44:10:f6:41
sender protocol address : 192.168.100.5
target protocol address : 192.168.100.254

Fri Jul 16 20:03:59 IST 2004
---
hardware type : 1
protocol type : 2048
hardware length : 6
protocol length : 4
operation code : 1
sender hardware address : 0:2:44:10:f6:41
sender protocol address : 192.168.100.5
target protocol address : 192.168.100.254

Fri Jul 16 20:04:00 IST 2004
---
hardware type : 1
protocol type : 2048
hardware length : 6
protocol length : 4
operation code : 1
sender hardware address : 0:2:44:10:f6:41
sender protocol address : 192.168.100.5
target protocol address : 192.168.100.254

Fri Jul 16 20:04:01 IST 2004
---
hardware type : 1
protocol type : 2048
hardware length : 6
protocol length : 4
operation code : 1
sender hardware address : 0:2:44:10:f6:41
sender protocol address : 192.168.100.5
target protocol address : 192.168.100.254

Fri Jul 16 20:04:03 IST 2004
---
hardware type : 1
protocol type : 2048
hardware length : 6
protocol length : 4
operation code : 1

sender hardware address : 0:2:44:10:f6:41
sender protocol address : 192.168.100.5
target protocol address : 192.168.100.254

Fri Jul 16 20:04:04 IST 2004
---
hardware type : 1
protocol type : 2048
hardware length : 6
protocol length : 4
operation code : 1
sender hardware address : 0:2:44:10:f6:41
sender protocol address : 192.168.100.5
target protocol address : 192.168.100.254

Fri Jul 16 20:04:05 IST 2004
---
hardware type : 1
protocol type : 2048
hardware length : 6
protocol length : 4
operation code : 1
sender hardware address : 0:2:44:10:f6:41
sender protocol address : 192.168.100.5
target protocol address : 192.168.100.254

Fri Jul 16 20:04:07 IST 2004
---
hardware type : 1
protocol type : 2048
hardware length : 6
protocol length : 4
operation code : 1
sender hardware address : 0:2:44:10:f6:41
sender protocol address : 192.168.100.5
target protocol address : 192.168.100.254

Fri Jul 16 20:04:08 IST 2004
---
hardware type : 1
protocol type : 2048
hardware length : 6
protocol length : 4
operation code : 1
sender hardware address : 0:2:44:10:f6:41
sender protocol address : 192.168.100.5

target protocol address : 192.168.100.254

Fri Jul 16 20:04:22 IST 2004
---
hardware type : 1
protocol type : 2048
hardware length : 6
protocol length : 4
operation code : 1
sender hardware address : 0:50:bf:d8:48:a1
sender protocol address : 192.168.100.2
target protocol address : 192.168.100.1

Fri Jul 16 20:04:22 IST 2004
---
hardware type : 1
protocol type : 2048
hardware length : 6
protocol length : 4
operation code : 2
sender hardware address : 0:2:44:10:f6:47
sender protocol address : 192.168.100.1
target protocol address : 192.168.100.2

Viva Voce Questions:

1. What is security attack?
2. What is Security mechanism?
3. What is security services?
4. Name the types of attacks?
5. What is a Sniffer?
6. How is threat different from attack?

(2) Name of the Experiment:

AIM: Understanding of cryptographic algorithms and implementation of the same in c or c++.

S/W & H/W Requirements:

S/W: Turbo C

H/W: Pentium IV Processor**,** 256 MB RAM, 40 GB HDD.

Algorithm:

Step1: Start
Step2: read 64-bit plain text
64-bit key
Step3: Initial permutation
Permutation choice 1
L(i-1)
R(i-1)
C(i-1)
D(i-1)
Step4: Expansion/
Permutation
(E table)
Left shift(s)
Left shift
Permutation/
Contraction
(Permutated choice 2)
Step5: XOR
Substitution choice (S-box)
Permutation
XOR
R (i)
L (i)
Exit
i<=16
C(i)
D(i)

Step6: Stop

Test Data:

Valid Data Set: Numbers should be prime

Invalid Data Set: Integers which are not prime numbers

Limiting Data Set: Not applicable

Result:

Enter the string of 8 chars

SREERAMA

0101001101010010010001010100010101010010010000010100110101000001111111110001001
1010011
0011101
1010000
0000000
0000001
0000000
0010011
r1: 10000000000000000000000000100000000000000010100110

key 0100000111010111100011011011111001100011101010101100000111010111100011011001111100110001100001100
1000temp is 15
1110temp is 11
1111temp is 8
0110temp is 0
0011temp is 7
0011temp is 5
0110temp is 8
0110temp is 11
b1 is
1111
1011
1000
0000
0111
0101
1000
1011
11111011100000000111010110001011after substn
01101000100011101110101110001011xor l[i] with res[i]

left and right for next round
00000000000000000100000000010011
10010111100111011010011101100110r1:
01001010111111001111101111010000111010110000 1101

key11111110001001001110100000010111011011010011 0111
10110100110110000001001111000111100001100011 1010
0110temp is 1
0110temp is 8
0000temp is 13
1001temp is 7
1000temp is 6
1100temp is 1
1100temp is 5
1101temp is 3
b1 is
0001
1000
1101
0111
0110
0001
0101
0011
00011000110101110110000101010011after substn
10000110010111110011100110000010xor l[i] with res[i]
left and right for next round
10010111100111011010011101100110
10000110010111110111100110010001r1:
11000000110000101111111010111111001111001010 0011

key11011010110011100011100001101111000010011101010 0
00011010000011001100011011010000001101010111011 1
0011temp is 1
0000temp is 0
1001temp is 15
0011temp is 3
1010temp is 12
0001temp is 15
1010temp is 5
1011temp is 0
b1 is
0001
0000
1111
0011

1100
1111
0101
0000
00010000111100111100111101010000after substn
10010111011101010010000100001110xor l[i] with res[i]
left and right for next round
10000110010111110111100110010001
00000000111010001000011001101000r1:
00000000000101110101000101000000110000110101000 0

key11001100101110110001111011000001111000001110111 11
11001100101011000100111110000001001000101000111 1
1001temp is 11
0101temp is 11
1000temp is 4
0111temp is 3
0000temp is 4
1001temp is 13
0101temp is 0
0111temp is 4
b1 is
1011
1011
0100
0011
0100
1101
0000
0100
10111011010000110100110100000100after substn
11010000110011010110001000101010xor l[i] with res[i]
left and right for next round
00000000111010001000011001101000
01010110100100100001101110111011r1:
10101010110101001010010000001111011111010111110110

key00100110101111100100111101100111100101101000110 1
10001100011010101110101101101000111010110111101 1
0001temp is 12
0011temp is 14
0101temp is 9
0101temp is 1
1101temp is 0
0111temp is 8
0110temp is 10

1101temp is 5
b1 is
1100
1110
1001
0001
0000
1000
1010
0101
11001110100100010000100010100101after substn
11010100100010001000110100110001xor l[i] with res[i]
left and right for next round
01010110100100100001101110111011
11010100011000000000101101011001r1:
11101010100000110000000000000010101101010111110011

key01101011011111000110001011011010000101011110111 1
10000001111111110110001011011111011111100011100
0000temp is 4
1111temp is 5
1110temp is 2
0001temp is 6
1011temp is 9
1011temp is 7
1110temp is 9
1110temp is 12
b1 is
0100
0101
0010
0110
1001
0111
1001
1100
01000101001001101001011110011100after substn
00101011011000001111000000111101xor l[i] with res[i]
left and right for next round
11010100011000000000101101011001
01111101111100101110101110000110r1:
00111111101111111010010101110101011110000001100

key11101000111011011111100000001110110110111010 1101
11010111010100100101110101111011101001111010000 1
1010temp is 3

1010temp is 7
0100temp is 3
1110temp is 14
1111temp is 9
1101temp is 13
1111temp is 1
0000temp is 2
b1 is
0011
0111
0011
1110
1001
1101
0001
0010
00110111001111101001110100010010after substn
01110111010000100111001001011110xor l[i] with res[i]
left and right for next round
01111101111100101110101110000110
10100011001000100111100100000111r1:
11010000011010010000010000111111001010000000 01111

key11010100111001110001101101010010011111010111110001
00000100100011100001111101101101010101011111111 0
0000temp is 0
0100temp is 6
1100temp is 5
1111temp is 9
1101temp is 9
1010temp is 13
1011temp is 12
1111temp is 8
b1 is
0000
0110
0101
1001
1001
1101
1100
1000
00000110010110011001110111001000after substn
11111101000100010010000001011001xor l[i] with res[i]
left and right for next round
10100011001000100111100100000111

10000000111000111100101111011111r1:
11000000000101110000011111100101011111011111111

key10111110010010111110011011101100111110000101011 1
01111110010111001110000100001001100001101010100 0
1111temp is 8
0010temp is 10
1001temp is 15
0000temp is 3
0001temp is 12
1100temp is 14
1101temp is 10
0100temp is 9
b1 is
1000
1010
1111
0011
1100
1110
1010
1001
10001010111100111100111010101001after substn
11011101111011010000110100001101xor l[i] with res[i]
left and right for next round
10000000111000111100101111011111
01111110110011110111010000001010r1:
00111111111010110010111101011101010000000001010100

key10111010011100110010100110100111110001101111101 0
10000101101001010111011100011101010001101010111 0
0000temp is 15
1101temp is 0
1010temp is 5
1011temp is 11
0011temp is 12
1010temp is 3
1101temp is 10
0111temp is 2
b1 is
1111
0000
0101
1011
1100
0011

1010
0010
11110000010110111100001110100010after substn
10000101111001111010011001000011xor l[i] with res[i]
left and right for next round
01111110110011110111010000001010
00000101000001000110110110011100r1:
00000000101010000000100000110101101111001111100 0

key10001001000101110111111011001110110011111101000011
10001001101111111011101011010100000100011101110 11
0001temp is 1
1101temp is 9
1110temp is 2
1010temp is 5
0101temp is 13
0001temp is 1
0111temp is 13
1101temp is 5
b1 is
0001
1001
0010
0101
1101
0001
1101
0101
00011001001001011101000111010101after substn
10100011000111000111100000100111xor l[i] with res[i]
left and right for next round
00000101000001000110110110011100
11011101110100110000110000101101r1:
11101111101111101010011010000101100000010101 1011

key11000101010110101101110110011111011000110011101 00
00101010111001000111101100011011010001110010 1111
0101temp is 15
0111temp is 1
1000temp is 2
1101temp is 7
0011temp is 1
1010temp is 4
1110temp is 6
0111temp is 13
b1 is

1111
0001
0010
0111
0001
0100
0110
1101
11110001001001110001010001101101after substn
10101000110100001101111000101110xor l[i] with res[i]
left and right for next round
11011101110100110000110000101101
10101101110101001011001110110010r1:
01010101101111101010100101011010011111010100101

key000101111111101111100000010110011110111111000100
01000010010001010100100100000011100100100110001
1000temp is 3
0010temp is 7
1010temp is 5
0100temp is 6
0000temp is 2
1100temp is 6
0100temp is 4
0000temp is 2
b1 is
0011
0111
0101
0110
0010
0110
0100
0010
00110111010101100010011001000010after substn
01000100011100110101001010011010xor l[i] with res[i]
left and right for next round
10101101110101001011001110110010
10011001101000000101111010110111r1:
11001111001111010000000000101111110101010110101111

key100110100111110111100011101110001110010010011001
01010101010000001110001110010111001100010011010
1010temp is 12
1010temp is 2
0001temp is 7

0001temp is 15
0010temp is 12
1001temp is 14
0010temp is 2
1011temp is 13
b1 is
1100
0010
0111
1111
1100
1110
0010
1101
11000010011111111100111000101101after substn
11011101111001011001110001101100xor l[i] with res[i]
left and right for next round
10011001101000000101111010110111
01110000001100010010111111011110r1:
00111010000000011010001010010101111111011111100

key11111001011001110100110111101011011011000000111
11000011011001101110111101111110100010001111101 1
1000temp is 15
1011temp is 6
1101temp is 11
0111temp is 8
1111temp is 6
0100temp is 2
0001temp is 11
1101temp is 5
b1 is
1111
0110
1011
1000
0110
0010
1011
0101
11110110101110000110001010110101after substn
01000110101001001000111111110111xor l[i] with res[i]
left and right for next round
01110000001100010010111111011110
11011111000001001101000101000000r1:
01101111111010000000100101101010001010100000001

key00101111001111001011111000101011100110111011001 1
01000000110101001011011101000001101100011011001 0
1000temp is 3
0110temp is 8
1001temp is 13
1011temp is 11
1000temp is 8
1101temp is 11
0011temp is 14
1001temp is 6
b1 is
0011
1000
1101
1011
1000
1011
1110
0110
00111000110110111000101111100110after substn
10010101011110110010011101100011xor l[i] with res[i]
left and right for next round
11011111000001001101000101000000
11100101010010100000100010111101
after swapping
11100101010010100000100010111101110111110000010011010001010000000utput is
11001001100100001110000110010101100010010100000111011010110010 01

Viva Voce Questions:

1. What is cryptography
2. What is encryption and decryption
3. What is the key size in blowfish & how many number of rounds are performed.
4. Which are the mathematical operations that are different from other algorithms
5. Explain the convention encryption principles
6. Explain the classification of cryptographic system.

3) Name of the Experiment:

AIM: Using open ssl for web server browser communication.

S/W & H/W Requirements:

S/W: Turbo C

H/W: Pentium IV Processor**,** 256 MB RAM,40 GB HDD.

Algorithm:

Generate Your Private Key

openssl genrsa -des3 -out server.key 1024
[enter a password]
[confirm your password]

Test Data:

Valid Data Set: Not applicable

Invalid Data Set: Not applicable

Limiting Data Set: Not applicable

Result:

```
[root@node2 ssl]# openssl genrsa -des3 -out server.key 1024
Generating RSA private key, 1024 bit long modulus
..++++++
.......++++++
e is 65537 (0x10001)
Enter pass phrase for server.key:
Verifying - Enter pass phrase for server.key:
[root@node2 ssl]#
```

Certificate Signing Request (CSR)

openssl req -key server.key -out server.csr
[enter your private key password]
[enter your two character country code]
[enter your full state or province name]

[enter your city name]
[enter your company name]
[enter your organizational unit or leave it blank]
[enter your common name or fqdn]
[enter your admin email address]
[leave the rest of the attributes blank]

```
[root@node2 ssl]# openssl req -new -key server.key -out server.csr
Enter pass phrase for server.key:
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [GB]:US
State or Province Name (full name) [Berkshire]:California
Locality Name (eg, city) [Newbury]:Los Angeles
Organization Name (eg, company) [My Company Ltd]:XenoCafe
Organizational Unit Name (eg, section) []:Information Technology
Common Name (eg, your name or your server's hostname) []:secure.xenocafe.com
Email Address []:admin@xenocafe.com

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:
[root@node2 ssl]#
```

cat server.csr

```
[root@node2 ssl]# cat server.csr
-----BEGIN CERTIFICATE REQUEST-----
MIIB7jCCAVcCAQAwga0xCzAJBgNVBAYTA1VTMRMwEQYDVQQIEwpDYWxpZm9ybmlh
MRQwEgYDVQQHEwtMb3MgQW5nZWxlczERMA8GA1UEChMIWGVub0NhZmUxHzAdBgNV
BAsTFkluZm9ybWF0aW9uIFR1Y2hub2xvZ3kxHDAaBgNVBAMTE3N1Y3VyZS54ZW5v
Y2FmZS5jb20xITAfBgkqhkiG9w0BCQEWEmFkbWluQHhlbm9jYWZlLmNvbTCBnzAN
BgkqhkiG9w0BAQEFAAOBjQAwgYkCgYEAwWahFT+MdssMM4t5ED1mdDpLCUQhUIGk
Oz25zIOWsXq6H9rTLgDmaPIAK/aGq8JoVt4oAj9pbsS28g8zsnyEQbLNPhKA5jqH
01FoHEb98WL48GJ6O0EQknxhpZJdPOX1rQqhjSzqFllkwF6KbusTHbCA3jy+qxy1
Ys7PhUMD/zkCAwEAAaAAMA0GCSqGSIb3DQEBBAUAA4GBACIlJ3CevtERLr7Fym9f
W3wvF0d4Wi/GFhaf0UQB0VLjscwNaxyDI9IUd/7yhft4X+Ge9KiqLjVZh6cezf0O
VUVuAkM6cgVku9191Jx8vQ1jtuAv20OYDPElzItUdMMbKJD4wHewbGWNPAYiahA3
uJcCMiyBPuWy2kEWnqzDuMq5
-----END CERTIFICATE REQUEST-----
[root@node2 ssl]# _
```

Sign Your Certificate Signing Request

openssl x509 -req -days 365 -in server.csr -signkey server.key -out server.crt
[enter your private key password]

```
[root@node2 ssl]# openssl x509 -req -days 365 -in server.csr -signkey server.key
 -out server.crt
Signature ok
subject=/C=US/ST=California/L=Los Angeles/O=XenoCafe/OU=Information Technology/C
N=secure.xenocafe.com/emailAddress=admin@xenocafe.com
Getting Private key
Enter pass phrase for server.key:
[root@node2 ssl]#
```

Remove the PassPhrase From Your Private Key

cp server.key server.key.secure
openssl rsa -in server.key.secure -out server.key
[enter your private key password]

```
[root@node2 ssl]# cp server.key server.key.secure
[root@node2 ssl]# openssl rsa -in server.key.secure -out server.key
Enter pass phrase for server.key.secure:
writing RSA key
[root@node2 ssl]# _
```

Install Your Certificate and Private Key

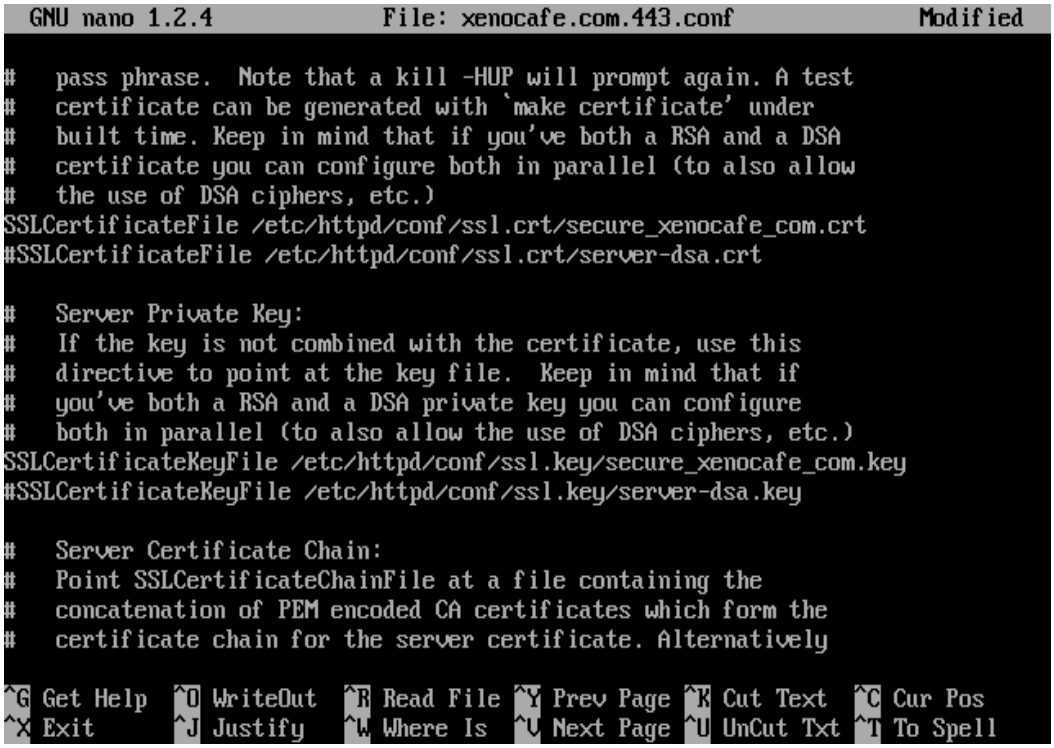
mv server.csr /etc/httpd/conf/ssl.csr/host_domain_tld.csr
mv server.crt /etc/httpd/conf/ssl.crt/host_domain_tld.crt
mv server.key /etc/httpd/conf/ssl.key/host_domain_tld.key
mv server.key.secure /etc/httpd/conf/host_domain_tld.key.secure

```
[root@node2 ssl]# mv server.csr /etc/httpd/conf/ssl.csr/secure_xenocafe_com.csr
[root@node2 ssl]# mv server.crt /etc/httpd/conf/ssl.crt/secure_xenocafe_com.crt
[root@node2 ssl]# mv server.key /etc/httpd/conf/ssl.key/secure_xenocafe_com.key
[root@node2 ssl]# mv server.key.secure /etc/httpd/conf/ssl.key/secure_xenocafe_c
om.key.secure
[root@node2 ssl]#
```

chmod 400 /etc/httpd/conf/ssl.csr/host_domain_tld.csr
chmod 400 /etc/httpd/conf/ssl.crt/host_domain_tld.crt
chmod 400 /etc/httpd/conf/ssl.key/host_domain_tld.key
chmod 400 /etc/httpd/conf/ssl.key/host_domain_tld.key.secure

```
[root@node2 ssl]# chmod 400 /etc/httpd/conf/ssl.csr/secure_xenocafe_com.csr
[root@node2 ssl]# chmod 400 /etc/httpd/conf/ssl.crt/secure_xenocafe_com.crt
[root@node2 ssl]# chmod 400 /etc/httpd/conf/ssl.key/secure_xenocafe_com.key
[root@node2 ssl]# chmod 400 /etc/httpd/conf/ssl.key/secure_xenocafe_com.key.secu
re
[root@node2 ssl]# _
```

Configure Your Apache SSL Virtual Host



service httpd restart

Test Your Apache SSL Virtual Host

Viva-Voce Questions:

1. What is open SSL.
2. Explain the architecture of secure socket layer
3. What are the two services that SSl recprd protocol provides.
4. How many keys are used in open ssl communication
5. How does a cipher text look in openssl after first key use

(4) Name of the Experiment:

AIM: Using GNU PGP.

S/W & H/W Requirements:

S/W: Turbo C

H/W: Pentium IV Processor**,** 256 MB RAM,40 GB HDD.

Algorithm:

Step1: Start

Step2: read int listfd,connfd,retval

pid_t childpid

socklen_t clilen

struct sockaddr_in cliaddr, servaddr

listfd = socket(AF_INET, SOCK_STREAM, 0)

Step3: if (listfd < 0)

perror("sock:")

bzero(&servaddr, sizeof(servaddr))

servaddr.sin_family = AF_INET

servaddr.sin_addr.s_addr = htonl(INADDR_ANY)

servaddr.sin_port = htons(8000)

retval = bind(listfd, (struct sockaddr *) &servaddr, sizeof(servaddr))

Step4 : if(retval < 0)

perror("bind:")

listen(listfd, 5)

Step5: while(1)

clilen = sizeof(cliaddr)

connfd = accept(listfd, (struct sockaddr *) &cliaddr,

&clilen)

print(" client connected \n")

print(" client's port no = %d\n",htons(cliaddr.sin_port))

Step6: char *serv_ip = "127.0.0.1"

int sockfd,ret_val

struct sockaddr_in servaddr

sockfd = socket(AF_INET, SOCK_STREAM, 0)

bzero(&servaddr, sizeof(servaddr))

servaddr.sin_family = AF_INET

servaddr.sin_port = htons(8000)

inet_pton(AF_INET, serv_ip, &servaddr.sin_addr)

ret_val = connect(sockfd, (struct sockaddr *) &servaddr, sizeof(servaddr));

if(ret_val <0){

perror("connect:");

printf(" client established connection with server\n")

Step7: Stop

Test Data:

Valid Data Set: System IP Address has to be input

Invalid Data Set: Invalid IP address

Limiting Data Set: Not applicable

Viva Voce Questions:

1. What is PGP?
2. Explain the PGP services.
3. How is confidentiality and authentication in PGP cryptographic function?
4. What is session key
5. Explain how secure hash function helps in PGP

(5) Name of the Experiment

AIM: Performance evaluation of various cryptographic algorithms.

S/W & H/W Requirements:

S/W: Turbo C

H/W: Pentium IV Processor**,** 256 MB RAM,40 GB HDD.

Algorithm:

Step1: Start

Step2: unsigned long F(BLOWFISH_CTX *ctx, unsigned int x)

unsigned short a, b, c, d

unsigned int y

d = x & 0x00FF

x >>= 8

c = x & 0x00FF

x >>= 8

b = x & 0x00FF

x >>= 8

a = x & 0x00FF

y = ctx->S[0][a] + ctx->S[1][b]

```
y = y ^ ctx->S[2][c]

y = y + ctx->S[3][d]

return y
```

Step3: void Blowfish_Encrypt(BLOWFISH_CTX *ctx, unsigned int *xl, unsigned int *xr)

```
unsigned int Xl

unsigned int Xr

unsigned int temp

int   ii,i

Xl = *xl

Xr = *xr

for (i = 0; i < N; ++i)

    Xl = Xl ^ ctx->P[i]

    Xr = F(ctx, Xl) ^ Xr

    temp = Xl

    Xl = Xr

    Xr = temp

temp = Xl

Xl = Xr

Xr = temp

Xr = Xr ^ ctx->P[N]

Xl = Xl ^ ctx->P[N + 1]

*xl = Xl

*xr = Xr
```

Step4: void Blowfish_Decrypt(BLOWFISH_CTX *ctx, unsigned int *xl, unsigned int *xr)

```
unsigned int Xl
unsigned int Xr
unsigned int temp
int ii,i
Xl = *xl
Xr = *xr
for (i = N + 1; i > 1; --i)
    Xl = Xl ^ ctx->P[i]
    Xr = F(ctx, Xl) ^ Xr
    temp = Xl
    Xl = Xr
    Xr = temp
  temp = Xl
Xl = Xr
Xr = temp
Xr = Xr ^ ctx->P[1]
Xl = Xl ^ ctx->P[0]
*xl = Xl
*xr = Xr
```

Step5: void Blowfish_Init(BLOWFISH_CTX *ctx, unsigned char *key, int KeyLen)

```
unsigned int Xl
```

```
int i, j, k
unsigned int data, datal, datar

for (i = 0; i < 4; i++)
    for (j = 0; j < 256; j++)
      ctx->S[i][j] = ORIG_S[i][j]
j = 0
for (i = 0; i < N + 2; ++i)
    data = 0x00000000
    for (k = 0; k < 4; ++k)
       data = (data << 8) | key[j]
       j = j + 1
       if (j >= KeyLen)
         j = 0
  ctx->P[i] = ORIG_P[i] ^ data
datal = 0x00000000
datar = 0x00000000
for (i = 0; i < N + 2; i += 2)
    Blowfish_Encrypt(ctx, &datal, &datar)
    ctx->P[i] = datal
    ctx->P[i + 1] = datar
for (i = 0; i < 4; ++i)
    for (j = 0; j < 256; j += 2)
```

```
            Blowfish_Encrypt(ctx, &datal, &datar)
            ctx->S[i][j] = datal
            ctx->S[i][j + 1] = datar
        int Blowfish_Test(BLOWFISH_CTX *ctx)
          unsigned int L = 1, R = 2
          Blowfish_Init(ctx, (unsigned char*)"TESTKEY", 7)
          Blowfish_Encrypt(ctx, &L, &R)
          if (L != 0xDF333FD2L || R != 0x30A71BB4L) return (-1)
          Blowfish_Decrypt(ctx, &L, &R)
          if (L != 1 || R != 2) return (-1); return (0)
        extern unsigned char    ciphertext_buffer[256]
        extern unsigned char    *ciphertext_string
        int startEncryption(char *plaintext_string,char *key)
          BLOWFISH_CTX ctx
          unsigned int message_left
          unsigned int message_right
          int keylen=strlen(key)
          int block_len,plaintext_len=strlen(plaintext_string)
          int ciphertext_len=0
          ciphertext_string=&ciphertext_buffer[0]
          Blowfish_Init(&ctx, (unsigned char *)key, keylen)
          print("Plaintext message string is: %s\n", plaintext_string)
```

Step 6: encrypt the plaintext message string

print("Encrypted message string is: ")

while (plaintext_len)

message_left = message_right = 0UL

Step7: crack the message string into a 64-bit block (ok, really two 32-bit blocks); pad with zeros if necessary

for (block_len = 0; block_len < 4; block_len++)

message_left = message_left << 8

if (plaintext_len)

message_left += *plaintext_string++

plaintext_len--

else message_left += 0

for (block_len = 0; block_len < 4; block_len++)

message_right = message_right << 8

if (plaintext_len)

message_right += *plaintext_string++

plaintext_len--

else message_right += 0

Step8: encrypt and print the results

Blowfish_Encrypt(&ctx, &message_left, &message_right)

printf("%lx%lx", message_left, message_right)

Step9: save the results for decryption below

*ciphertext_string++ = (unsigned char)(message_left >> 24)

*ciphertext_string++ = (unsigned char)(message_left >> 16)

```
*ciphertext_string++ = (unsigned char)(message_left >> 8)
*ciphertext_string++ = (unsigned char)message_left
*ciphertext_string++ = (unsigned char)(message_right >> 24)
*ciphertext_string++ = (unsigned char)(message_right >> 16)
*ciphertext_string++ = (unsigned char)(message_right >> 8)
*ciphertext_string++ = (unsigned char)message_right
ciphertext_len += 8
print("\n")
ciphertext_string = &ciphertext_buffer[0]
printf("ciphertext_len: %d\n",ciphertext_len)
return ciphertext_len
```

Step10: void startDecryption(unsigned char *ciphertext_string,char ciphertext_len,char *key)

```
BLOWFISH_CTX ctx
unsigned int message_left
unsigned int message_right
int block_len
  int keylen=strlen(key)
Blowfish_Init(&ctx, (unsigned char *)key, keylen)
printf("Decrypted message string is: ")
printf("cipher Len: %d\n",ciphertext_len)
while(ciphertext_len)
  message_left = message_right = 0UL
  for (block_len = 0; block_len < 4; block_len++)
```

```
message_left = message_left << 8
message_left += *ciphertext_string++
if (ciphertext_len)
 ciphertext_len--
for (block_len = 0; block_len < 4; block_len++)
 message_right = message_right << 8
 message_right += *ciphertext_string++
 if (ciphertext_len)
 ciphertext_len--;
Blowfish_Decrypt(&ctx, &message_left, &message_right);
```

Step11: if plaintext message string padded, extra zeros here

```
print("%c%c%c%c%c%c%c%c")
 (int)(message_left >> 24), (int)(message_left >> 16),
 (int)(message_left >> 8), (int)(message_left),
 (int)(message_right >> 24), (int)(message_right >> 16),
 (int)(message_right >> 8), (int)(message_right))
```

Step12: Blowfish algorithm implemented on Server side and on Client side also the same file will be used.

```
typedef struct
  unsigned int  P[16 + 2]
  unsigned int  S[4][256]
BLOWFISH_CTX
static const unsigned long ORIG_P[16 + 2] = 0x243F6A88L,
0x85A308D3L, 0x13198A2EL, 0x03707344L,
```

0xA4093822L, 0x299F31D0L, 0x082EFA98L, 0xEC4E6C89L,
0x452821E6L, 0x38D01377L, 0xBE5466CFL, 0x34E90C6CL,
0xC0AC29B7L, 0xC97C50DDL, 0x3F84D5B5L, 0xB5470917L,
0x9216D5D9L, 0x8979FB1BL

Step13: static const unsigned long ORIG_S[4][256] = {

{ 0xD1310BA6L, 0x98DFB5ACL, 0x2FFD72DBL, 0xD01ADFB7L,
0xB8E1AFEDL, 0x6A267E96L, 0xBA7C9045L, 0xF12C7F99L,
0x24A19947L, 0xB3916CF7L, 0x0801F2E2L, 0x858EFC16L,
0x636920D8L, 0x71574E69L, 0xA458FEA3L, 0xF4933D7EL,
0x0D95748FL, 0x728EB658L, 0x718BCD58L, 0x82154AEEL,
0x7B54A41DL, 0xC25A59B5L, 0x9C30D539L, 0x2AF26013L,
0xC5D1B023L, 0x286085F0L, 0xCA417918L, 0xB8DB38EFL,
0x8E79DCB0L, 0x603A180EL, 0x6C9E0E8BL, 0xB01E8A3EL,
0xD71577C1L, 0xBD314B27L, 0x78AF2FDAL, 0x55605C60L,
0xE65525F3L, 0xAA55AB94L, 0x57489862L, 0x63E81440L,
0x55CA396AL, 0x2AAB10B6L, 0xB4CC5C34L, 0x1141E8CEL,
0xA15486AFL, 0x7C72E993L, 0xB3EE1411L, 0x636FBC2AL,
0x2BA9C55DL, 0x741831F6L, 0xCE5C3E16L, 0x9B87931EL,
0xAFD6BA33L, 0x6C24CF5CL, 0x7A325381L, 0x28958677L,
0x3B8F4898L, 0x6B4BB9AFL, 0xC4BFE81BL, 0x66282193L,
0x61D809CCL, 0xFB21A991L, 0x487CAC60L, 0x5DEC8032L,
0xEF845D5DL, 0xE98575B1L, 0xDC262302L, 0xEB651B88L,
0x23893E81L, 0xD396ACC5L, 0x0F6D6FF3L, 0x83F44239L,

0x2E0B4482L, 0xA4842004L, 0x69C8F04AL, 0x9E1F9B5EL,
0x21C66842L, 0xF6E96C9AL, 0x670C9C61L, 0xABD388F0L,
0x6A51A0D2L, 0xD8542F68L, 0x960FA728L, 0xAB5133A3L,
0x6EEF0B6CL, 0x137A3BE4L, 0xBA3BF050L, 0x7EFB2A98L,
0xA1F1651DL, 0x39AF0176L, 0x66CA593EL, 0x82430E88L,
0x8CEE8619L, 0x456F9FB4L, 0x7D84A5C3L, 0x3B8B5EBEL,
0xE06F75D8L, 0x85C12073L, 0x401A449FL, 0x56C16AA6L,
0x4ED3AA62L, 0x363F7706L, 0x1BFEDF72L, 0x429B023DL,
0x37D0D724L, 0xD00A1248L, 0xDB0FEAD3L, 0x49F1C09BL,
0x075372C9L, 0x80991B7BL, 0x25D479D8L, 0xF6E8DEF7L,
0xE3FE501AL, 0xB6794C3BL, 0x976CE0BDL, 0x04C006BAL,
0xC1A94FB6L, 0x409F60C4L, 0x5E5C9EC2L, 0x196A2463L,
0x68FB6FAFL, 0x3E6C53B5L, 0x1339B2EBL, 0x3B52EC6FL,
0x6DFC511FL, 0x9B30952CL, 0xCC814544L, 0xAF5EBD09L,
0xBEE3D004L, 0xDE334AFDL, 0x660F2807L, 0x192E4BB3L,
0xC0CBA857L, 0x45C8740FL, 0xD20B5F39L, 0xB9D3FBDBL,
0x5579C0BDL, 0x1A60320AL, 0xD6A100C6L, 0x402C7279L,
0x679F25FEL, 0xFB1FA3CCL, 0x8EA5E9F8L, 0xDB3222F8L,
0x3C7516DFL, 0xFD616B15L, 0x2F501EC8L, 0xAD0552ABL,
0x323DB5FAL, 0xFD238760L, 0x53317B48L, 0x3E00DF82L,
0x9E5C57BBL, 0xCA6F8CA0L, 0x1A87562EL, 0xDF1769DBL,
0xD542A8F6L, 0x287EFFC3L, 0xAC6732C6L, 0x8C4F5573L,
0x695B27B0L, 0xBBCA58C8L, 0xE1FFA35DL, 0xB8F011A0L,

0x10FA3D98L, 0xFD2183B8L, 0x4AFCB56CL, 0x2DD1D35BL,
0x9A53E479L, 0xB6F84565L, 0xD28E49BCL, 0x4BFB9790L,
0xE1DDF2DAL, 0xA4CB7E33L, 0x62FB1341L, 0xCEE4C6E8L,
0xEF20CADAL, 0x36774C01L, 0xD07E9EFEL, 0x2BF11FB4L,
0x95DBDA4DL, 0xAE909198L, 0xEAAD8E71L, 0x6B93D5A0L,
0xD08ED1D0L, 0xAFC725E0L, 0x8E3C5B2FL, 0x8E7594B7L,
0x8FF6E2FBL, 0xF2122B64L, 0x8888B812L, 0x900DF01CL,
0x4FAD5EA0L, 0x688FC31CL, 0xD1CFF191L, 0xB3A8C1ADL,
0x2F2F2218L, 0xBE0E1777L, 0xEA752DFEL, 0x8B021FA1L,
0xE5A0CC0FL, 0xB56F74E8L, 0x18ACF3D6L, 0xCE89E299L,
0xB4A84FE0L, 0xFD13E0B7L, 0x7CC43B81L, 0xD2ADA8D9L,
0x165FA266L, 0x80957705L, 0x93CC7314L, 0x211A1477L,
0xE6AD2065L, 0x77B5FA86L, 0xC75442F5L, 0xFB9D35CFL,
0xEBCDAF0CL, 0x7B3E89A0L, 0xD6411BD3L, 0xAE1E7E49L,
0x00250E2DL, 0x2071B35EL, 0x226800BBL, 0x57B8E0AFL,
0x2464369BL, 0xF009B91EL, 0x5563911DL, 0x59DFA6AAL,
0x78C14389L, 0xD95A537FL, 0x207D5BA2L, 0x02E5B9C5L,
0x83260376L, 0x6295CFA9L, 0x11C81968L, 0x4E734A41L,
0xB3472DCAL, 0x7B14A94AL, 0x1B510052L, 0x9A532915L,
0xD60F573FL, 0xBC9BC6E4L, 0x2B60A476L, 0x81E67400L,
0x08BA6FB5L, 0x571BE91FL, 0xF296EC6BL, 0x2A0DD915L,
0xB6636521L, 0xE7B9F9B6L, 0xFF34052EL, 0xC5855664L,
0x53B02D5DL, 0xA99F8FA1L, 0x08BA4799L, 0x6E85076AL };

{ 0x4B7A70E9L, 0xB5B32944L, 0xDB75092EL, 0xC4192623L,
0xAD6EA6B0L, 0x49A7DF7DL, 0x9CEE60B8L, 0x8FEDB266L,
0xECAA8C71L, 0x699A17FFL, 0x5664526CL, 0xC2B19EE1L,
0x193602A5L, 0x75094C29L, 0xA0591340L, 0xE4183A3EL,
0x3F54989AL, 0x5B429D65L, 0x6B8FE4D6L, 0x99F73FD6L,
0xA1D29C07L, 0xEFE830F5L, 0x4D2D38E6L, 0xF0255DC1L,
0x4CDD2086L, 0x8470EB26L, 0x6382E9C6L, 0x021ECC5EL,
0x09686B3FL, 0x3EBAEFC9L, 0x3C971814L, 0x6B6A70A1L,
0x687F3584L, 0x52A0E286L, 0xB79C5305L, 0xAA500737L,
0x3E07841CL, 0x7FDEAE5CL, 0x8E7D44ECL, 0x5716F2B8L,
0xB03ADA37L, 0xF0500C0DL, 0xF01C1F04L, 0x0200B3FFL,
0xAE0CF51AL, 0x3CB574B2L, 0x25837A58L, 0xDC0921BDL,
0xD19113F9L, 0x7CA92FF6L, 0x94324773L, 0x22F54701L,
0x3AE5E581L, 0x37C2DADCL, 0xC8B57634L, 0x9AF3DDA7L,
0xA9446146L, 0x0FD0030EL, 0xECC8C73EL, 0xA4751E41L,
0xE238CD99L, 0x3BEA0E2FL, 0x3280BBA1L, 0x183EB331L,
0x4E548B38L, 0x4F6DB908L, 0x6F420D03L, 0xF60A04BFL,
0x2CB81290L, 0x24977C79L, 0x5679B072L, 0xBCAF89AFL,
0xDE9A771FL, 0xD9930810L, 0xB38BAE12L, 0xDCCF3F2EL,
0x5512721FL, 0x2E6B7124L, 0x501ADDE6L, 0x9F84CD87L,
0x7A584718L, 0x7408DA17L, 0xBC9F9ABCL, 0xE94B7D8CL,
0xEC7AEC3AL, 0xDB851DFAL, 0x63094366L, 0xC464C3D2L,
0xEF1C1847L, 0x3215D908L, 0xDD433B37L, 0x24C2BA16L,

0x12A14D43L, 0x2A65C451L, 0x50940002L, 0x133AE4DDL,
0x71DFF89EL, 0x10314E55L, 0x81AC77D6L, 0x5F11199BL,
0x043556F1L, 0xD7A3C76BL, 0x3C11183BL, 0x5924A509L,
0xF28FE6EDL, 0x97F1FBFAL, 0x9EBABF2CL, 0x1E153C6EL,
0x86E34570L, 0xEAE96FB1L, 0x860E5E0AL, 0x5A3E2AB3L,
0x771FE71CL, 0x4E3D06FAL, 0x2965DCB9L, 0x99E71D0FL,
0x803E89D6L, 0x5266C825L, 0x2E4CC978L, 0x9C10B36AL,
0xC6150EBAL, 0x94E2EA78L, 0xA5FC3C53L, 0x1E0A2DF4L,
0xF2F74EA7L, 0x361D2B3DL, 0x1939260FL, 0x19C27960L,
0x5223A708L, 0xF71312B6L, 0xEBADFE6EL, 0xEAC31F66L,
0xE3BC4595L, 0xA67BC883L, 0xB17F37D1L, 0x018CFF28L,
0xC332DDEFL, 0xBE6C5AA5L, 0x65582185L, 0x68AB9802L,
0xEECEA50FL, 0xDB2F953BL, 0x2AEF7DADL, 0x5B6E2F84L,
0x1521B628L, 0x29076170L, 0xECDD4775L, 0x619F1510L,
0x13CCA830L, 0xEB61BD96L, 0x0334FE1EL, 0xAA0363CFL,
0xB5735C90L, 0x4C70A239L, 0xD59E9E0BL, 0xCBAADE14L,
0xEECC86BCL, 0x60622CA7L, 0x9CAB5CABL, 0xB2F3846EL,
0x648B1EAFL, 0x19BDF0CAL, 0xA02369B9L, 0x655ABB50L,
0x40685A32L, 0x3C2AB4B3L, 0x319EE9D5L, 0xC021B8F7L,
0x9B540B19L, 0x875FA099L, 0x95F7997EL, 0x623D7DA8L,
0xF837889AL, 0x97E32D77L, 0x11ED935FL, 0x16681281L,
0x0E358829L, 0xC7E61FD6L, 0x96DEDFA1L, 0x7858BA99L,
0x57F584A5L, 0x1B227263L, 0x9B83C3FFL, 0x1AC24696L,

0xCDB30AEBL, 0x532E3054L, 0x8FD948E4L, 0x6DBC3128L,
0x58EBF2EFL, 0x34C6FFEAL, 0xFE28ED61L, 0xEE7C3C73L,
0x5D4A14D9L, 0xE864B7E3L, 0x42105D14L, 0x203E13E0L,
0x45EEE2B6L, 0xA3AAABEAL, 0xDB6C4F15L, 0xFACB4FD0L,
0xC742F442L, 0xEF6ABBB5L, 0x654F3B1DL, 0x41CD2105L,
0xD81E799EL, 0x86854DC7L, 0xE44B476AL, 0x3D816250L,
0xCF62A1F2L, 0x5B8D2646L, 0xFC8883A0L, 0xC1C7B6A3L,
0x7F1524C3L, 0x69CB7492L, 0x47848A0BL, 0x5692B285L,
0x095BBF00L, 0xAD19489DL, 0x1462B174L, 0x23820E00L,
0x58428D2AL, 0x0C55F5EAL, 0x1DADF43EL, 0x233F7061L,
0x3372F092L, 0x8D937E41L, 0xD65FECF1L, 0x6C223BDBL,
0x7CDE3759L, 0xCBEE7460L, 0x4085F2A7L, 0xCE77326EL,
0xA6078084L, 0x19F8509EL, 0xE8EFD855L, 0x61D99735L,
0xA969A7AAL, 0xC50C06C2L, 0x5A04ABFCL, 0x800BCADCL,
0x9E447A2EL, 0xC3453484L, 0xFDD56705L, 0x0E1E9EC9L,
0xDB73DBD3L, 0x105588CDL, 0x675FDA79L, 0xE3674340L,
0xC5C43465L, 0x713E38D8L, 0x3D28F89EL, 0xF16DFF20L,
0x153E21E7L, 0x8FB03D4AL, 0xE6E39F2BL, 0xDB83ADF7L },
{ 0xE93D5A68L, 0x948140F7L, 0xF64C261CL, 0x94692934L,
0x411520F7L, 0x7602D4F7L, 0xBCF46B2EL, 0xD4A20068L,
0xD4082471L, 0x3320F46AL, 0x43B7D4B7L, 0x500061AFL,
0x1E39F62EL, 0x97244546L, 0x14214F74L, 0xBF8B8840L,
0x4D95FC1DL, 0x96B591AFL, 0x70F4DDD3L, 0x66A02F45L,

0xBFBC09ECL, 0x03BD9785L, 0x7FAC6DD0L, 0x31CB8504L,
0x96EB27B3L, 0x55FD3941L, 0xDA2547E6L, 0xABCA0A9AL,
0x28507825L, 0x530429F4L, 0x0A2C86DAL, 0xE9B66DFBL,
0x68DC1462L, 0xD7486900L, 0x680EC0A4L, 0x27A18DEEL,
0x4F3FFEA2L, 0xE887AD8CL, 0xB58CE006L, 0x7AF4D6B6L,
0xAACE1E7CL, 0xD3375FECL, 0xCE78A399L, 0x406B2A42L,
0x20FE9E35L, 0xD9F385B9L, 0xEE39D7ABL, 0x3B124E8BL,
0x1DC9FAF7L, 0x4B6D1856L, 0x26A36631L, 0xEAE397B2L,
0x3A6EFA74L, 0xDD5B4332L, 0x6841E7F7L, 0xCA7820FBL,
0xFB0AF54EL, 0xD8FEB397L, 0x454056ACL, 0xBA489527L,
0x55533A3AL, 0x20838D87L, 0xFE6BA9B7L, 0xD096954BL,
0x55A867BCL, 0xA1159A58L, 0xCCA92963L, 0x99E1DB33L,
0xA62A4A56L, 0x3F3125F9L, 0x5EF47E1CL, 0x9029317CL,
0xFDF8E802L, 0x04272F70L, 0x80BB155CL, 0x05282CE3L,
0x95C11548L, 0xE4C66D22L, 0x48C1133FL, 0xC70F86DCL,
0x07F9C9EEL, 0x41041F0FL, 0x404779A4L, 0x5D886E17L,
0x325F51EBL, 0xD59BC0D1L, 0xF2BCC18FL, 0x41113564L,
0x257B7834L, 0x602A9C60L, 0xDFF8E8A3L, 0x1F636C1BL,
0x0E12B4C2L, 0x02E1329EL, 0xAF664FD1L, 0xCAD18115L,
0x6B2395E0L, 0x333E92E1L, 0x3B240B62L, 0xEEBEB922L,
0x85B2A20EL, 0xE6BA0D99L, 0xDE720C8CL, 0x2DA2F728L,
0xD0127845L, 0x95B794FDL, 0x647D0862L, 0xE7CCF5F0L,
0x5449A36FL, 0x877D48FAL, 0xC39DFD27L, 0xF33E8D1EL,

0x0A476341L, 0x992EFF74L, 0x3A6F6EABL, 0xF4F8FD37L,
0xA812DC60L, 0xA1EBDDF8L, 0x991BE14CL, 0xDB6E6B0DL,
0xC67B5510L, 0x6D672C37L, 0x2765D43BL, 0xDCD0E804L,
0xF1290DC7L, 0xCC00FFA3L, 0xB5390F92L, 0x690FED0BL,
0x667B9FFBL, 0xCEDB7D9CL, 0xA091CF0BL, 0xD9155EA3L,
0xBB132F88L, 0x515BAD24L, 0x7B9479BFL, 0x763BD6EBL,
0x37392EB3L, 0xCC115979L, 0x8026E297L, 0xF42E312DL,
0x6842ADA7L, 0xC66A2B3BL, 0x12754CCCL, 0x782EF11CL,
0x6A124237L, 0xB79251E7L, 0x06A1BBE6L, 0x4BFB6350L,
0x1A6B1018L, 0x11CAEDFAL, 0x3D25BDD8L, 0xE2E1C3C9L,
0x44421659L, 0x0A121386L, 0xD90CEC6EL, 0xD5ABEA2AL,
0x64AF674EL, 0xDA86A85FL, 0xBEBFE988L, 0x64E4C3FEL,
0x9DBC8057L, 0xF0F7C086L, 0x60787BF8L, 0x6003604DL,
0xD1FD8346L, 0xF6381FB0L, 0x7745AE04L, 0xD736FCCCL,
0x83426B33L, 0xF01EAB71L, 0xB0804187L, 0x3C005E5FL,
0x77A057BEL, 0xBDE8AE24L, 0x55464299L, 0xBF582E61L,
0x4E58F48FL, 0xF2DDFDA2L, 0xF474EF38L, 0x8789BDC2L,
0x5366F9C3L, 0xC8B38E74L, 0xB475F255L, 0x46FCD9B9L,
0x7AEB2661L, 0x8B1DDF84L, 0x846A0E79L, 0x915F95E2L,
0x466E598EL, 0x20B45770L, 0x8CD55591L, 0xC902DE4CL,
0xB90BACE1L, 0xBB8205D0L, 0x11A86248L, 0x7574A99EL,
0xB77F19B6L, 0xE0A9DC09L, 0x662D09A1L, 0xC4324633L,
0xE85A1F02L, 0x09F0BE8CL, 0x4A99A025L, 0x1D6EFE10L,

0x1AB93D1DL, 0x0BA5A4DFL, 0xA186F20FL, 0x2868F169L,
0xDCB7DA83L, 0x573906FEL, 0xA1E2CE9BL, 0x4FCD7F52L,
0x50115E01L, 0xA70683FAL, 0xA002B5C4L, 0x0DE6D027L,
0x9AF88C27L, 0x773F8641L, 0xC3604C06L, 0x61A806B5L,
0xF0177A28L, 0xC0F586E0L, 0x006058AAL, 0x30DC7D62L,
0x11E69ED7L, 0x2338EA63L, 0x53C2DD94L, 0xC2C21634L,
0xBBCBEE56L, 0x90BCB6DEL, 0xEBFC7DA1L, 0xCE591D76L,
0x6F05E409L, 0x4B7C0188L, 0x39720A3DL, 0x7C927C24L,
0x86E3725FL, 0x724D9DB9L, 0x1AC15BB4L, 0xD39EB8FCL,
0xED545578L, 0x08FCA5B5L, 0xD83D7CD3L, 0x4DAD0FC4L,
0x1E50EF5EL, 0xB161E6F8L, 0xA28514D9L, 0x6C51133CL,
0x6FD5C7E7L, 0x56E14EC4L, 0x362ABFCEL, 0xDDC6C837L,
0xD79A3234L, 0x92638212L, 0x670EFA8EL, 0x406000E0L },
{ 0x3A39CE37L, 0xD3FAF5CFL, 0xABC27737L, 0x5AC52D1BL,
0x5CB0679EL, 0x4FA33742L, 0xD3822740L, 0x99BC9BBEL,
0xD5118E9DL, 0xBF0F7315L, 0xD62D1C7EL, 0xC700C47BL,
0xB78C1B6BL, 0x21A19045L, 0xB26EB1BEL, 0x6A366EB4L,
0x5748AB2FL, 0xBC946E79L, 0xC6A376D2L, 0x6549C2C8L,
0x530FF8EEL, 0x468DDE7DL, 0xD5730A1DL, 0x4CD04DC6L,
0x2939BBDBL, 0xA9BA4650L, 0xAC9526E8L, 0xBE5EE304L,
0xA1FAD5F0L, 0x6A2D519AL, 0x63EF8CE2L, 0x9A86EE22L,
0xC089C2B8L, 0x43242EF6L, 0xA51E03AAL, 0x9CF2D0A4L,
0x83C061BAL, 0x9BE96A4DL, 0x8FE51550L, 0xBA645BD6L,

0x2826A2F9L, 0xA73A3AE1L, 0x4BA99586L, 0xEF5562E9L,
0xC72FEFD3L, 0xF752F7DAL, 0x3F046F69L, 0x77FA0A59L,
0x80E4A915L, 0x87B08601L, 0x9B09E6ADL, 0x3B3EE593L,
0xE990FD5AL, 0x9E34D797L, 0x2CF0B7D9L, 0x022B8B51L,
0x96D5AC3AL, 0x017DA67DL, 0xD1CF3ED6L, 0x7C7D2D28L,
0x1F9F25CFL, 0xADF2B89BL, 0x5AD6B472L, 0x5A88F54CL,
0xE029AC71L, 0xE019A5E6L, 0x47B0ACFDL, 0xED93FA9BL,
0xE8D3C48DL, 0x283B57CCL, 0xF8D56629L, 0x79132E28L,
0x785F0191L, 0xED756055L, 0xF7960E44L, 0xE3D35E8CL,
0x15056DD4L, 0x88F46DBAL, 0x03A16125L, 0x0564F0BDL,
0xC3EB9E15L, 0x3C9057A2L, 0x97271AECL, 0xA93A072AL,
0x1B3F6D9BL, 0x1E6321F5L, 0xF59C66FBL, 0x26DCF319L,
0x7533D928L, 0xB155FDF5L, 0x03563482L, 0x8ABA3CBBL,
0x28517711L, 0xC20AD9F8L, 0xABCC5167L, 0xCCAD925FL,
0x4DE81751L, 0x3830DC8EL, 0x379D5862L, 0x9320F991L,
0xEA7A90C2L, 0xFB3E7BCEL, 0x5121CE64L, 0x774FBE32L,
0xA8B6E37EL, 0xC3293D46L, 0x48DE5369L, 0x6413E680L,
0xA2AE0810L, 0xDD6DB224L, 0x69852DFDL, 0x09072166L,
0xB39A460AL, 0x6445C0DDL, 0x586CDECFL, 0x1C20C8AEL,
0x5BBEF7DDL, 0x1B588D40L, 0xCCD2017FL, 0x6BB4E3BBL,
0xDDA26A7EL, 0x3A59FF45L, 0x3E350A44L, 0xBCB4CDD5L,
0x72EACEA8L, 0xFA6484BBL, 0x8D6612AEL, 0xBF3C6F47L,
0xD29BE463L, 0x542F5D9EL, 0xAEC2771BL, 0xF64E6370L,

0x740E0D8DL, 0xE75B1357L, 0xF8721671L, 0xAF537D5DL,
0x4040CB08L, 0x4EB4E2CCL, 0x34D2466AL, 0x0115AF84L,
0xE1B00428L, 0x95983A1DL, 0x06B89FB4L, 0xCE6EA048L,
0x6F3F3B82L, 0x3520AB82L, 0x011A1D4BL, 0x277227F8L,
0x611560B1L, 0xE7933FDCL, 0xBB3A792BL, 0x344525BDL,
0xA08839E1L, 0x51CE794BL, 0x2F32C9B7L, 0xA01FBAC9L,
0xE01CC87EL, 0xBCC7D1F6L, 0xCF0111C3L, 0xA1E8AAC7L,
0x1A908749L, 0xD44FBD9AL, 0xD0DADECBL, 0xD50ADA38L,
0x0339C32AL, 0xC6913667L, 0x8DF9317CL, 0xE0B12B4FL,
0xF79E59B7L, 0x43F5BB3AL, 0xF2D519FFL, 0x27D9459CL,
0xBF97222CL, 0x15E6FC2AL, 0x0F91FC71L, 0x9B941525L,
0xFAE59361L, 0xCEB69CEBL, 0xC2A86459L, 0x12BAA8D1L,
0xB6C1075EL, 0xE3056A0CL, 0x10D25065L, 0xCB03A442L,
0xE0EC6E0EL, 0x1698DB3BL, 0x4C98A0BEL, 0x3278E964L,
0x9F1F9532L, 0xE0D392DFL, 0xD3A0342BL, 0x8971F21EL,
0x1B0A7441L, 0x4BA3348CL, 0xC5BE7120L, 0xC37632D8L,
0xDF359F8DL, 0x9B992F2EL, 0xE60B6F47L, 0x0FE3F11DL,
0xE54CDA54L, 0x1EDAD891L, 0xCE6279CFL, 0xCD3E7E6FL,
0x1618B166L, 0xFD2C1D05L, 0x848FD2C5L, 0xF6FB2299L,
0xF523F357L, 0xA6327623L, 0x93A83531L, 0x56CCCD02L,
0xACF08162L, 0x5A75EBB5L, 0x6E163697L, 0x88D273CCL,
0xDE966292L, 0x81B949D0L, 0x4C50901BL, 0x71C65614L,
0xE6C6C7BDL, 0x327A140AL, 0x45E1D006L, 0xC3F27B9AL,

0xC9AA53FDL, 0x62A80F00L, 0xBB25BFE2L, 0x35BDD2F6L,

0x71126905L, 0xB2040222L, 0xB6CBCF7CL, 0xCD769C2BL,

0x53113EC0L, 0x1640E3D3L, 0x38ABBD60L, 0x2547ADF0L,

0xBA38209CL, 0xF746CE76L, 0x77AFA1C5L, 0x20756060L,

0x85CBFE4EL, 0x8AE88DD8L, 0x7AAAF9B0L, 0x4CF9AA7EL,

0x1948C25CL, 0x02FB8A8CL, 0x01C36AE4L, 0xD6EBE1F9L,

0x90D4F869L, 0xA65CDEA0L, 0x3F09252DL, 0xC208E69FL,

0xB74E6132L, 0xCE77E25BL, 0x578FDFE3L, 0x3AC372E6L

Step14: unsigned long F(BLOWFISH_CTX *ctx, unsigned int x)

unsigned short a, b, c, d

unsigned int y

d = x & 0x00FF

x >>= 8

c = x & 0x00FF

x >>= 8

b = x & 0x00FF

x >>= 8

a = x & 0x00FF

y = ctx->S[0][a] + ctx->S[1][b]

y = y ^ ctx->S[2][c]

y = y + ctx->S[3][d]

return y

Step15: void Blowfish_Encrypt(BLOWFISH_CTX *ctx, unsigned int *xl, unsigned int *xr)

```
unsigned int Xl
unsigned int Xr
unsigned int temp
int   ii,i
Xl = *xl
Xr = *xr;
for (i = 0; i < N; ++i)
    Xl = Xl ^ ctx->P[i]
    Xr = F(ctx, Xl) ^ Xr
    temp = Xl
    Xl = Xr
    Xr = temp
temp = Xl
Xl = Xr
Xr = temp
Xr = Xr ^ ctx->P[N]
Xl = Xl ^ ctx->P[N + 1]
*xl = Xl
*xr = Xr
```

Step16: void Blowfish_Decrypt(BLOWFISH_CTX *ctx, unsigned int *xl, unsigned int *xr)

```
unsigned int Xl
unsigned int Xr
unsigned int temp
```

```
int   ii,i
Xl = *xl
Xr = *xr
for (i = N + 1; i > 1; --i)
    Xl = Xl ^ ctx->P[i]
    Xr = F(ctx, Xl) ^ Xr
    temp = Xl
    Xl = Xr
    Xr = temp
temp = Xl
Xl = Xr
Xr = temp
Xr = Xr ^ ctx->P[1]
Xl = Xl ^ ctx->P[0]
*xl = Xl
*xr = Xr
```

Step17:

```
void Blowfish_Init(BLOWFISH_CTX *ctx, unsigned char *key, int KeyLen)
unsigned int  Xl
int i, j, k
unsigned int data, datal, datar
for (i = 0; i < 4; i++)
    for (j = 0; j < 256; j++)
      ctx->S[i][j] = ORIG_S[i][j]
```

```
    j = 0
  for (i = 0; i < N + 2; ++i)
      data = 0x00000000
      for (k = 0; k < 4; ++k)
        data = (data << 8) | key[j]
        j = j + 1
        if (j >= KeyLen)
          j = 0
    ctx->P[i] = ORIG_P[i] ^ data;
  datal = 0x00000000
  datar = 0x00000000
  for (i = 0; i < N + 2; i += 2)
```

Step 18: Blowfish_Encrypt(ctx, &datal, &datar)

```
      ctx->P[i] = datal
      ctx->P[i + 1] = datar
  for (i = 0; i < 4; ++i)
      for (j = 0; j < 256; j += 2)
        Blowfish_Encrypt(ctx, &datal, &datar)
        ctx->S[i][j] = datal
        ctx->S[i][j + 1] = datar
```

Step19: int Blowfish_Test(BLOWFISH_CTX *ctx)

```
  unsigned int L = 1, R = 2
```

Blowfish_Init(ctx, (unsigned char*)"TESTKEY", 7)

Blowfish_Encrypt(ctx, &L, &R)

if (L != 0xDF333FD2L || R != 0x30A71BB4L) return (-1)

Blowfish_Decrypt(ctx, &L, &R)

if (L != 1 || R != 2) return (-1); return (0)

extern unsigned char ciphertext_buffer[256]

extern unsigned char *ciphertext_string

extern char plain_buffer[256]

extern char *plain_string

Step20: int startEncryption(char *plaintext_string,char *key)

BLOWFISH_CTX ctx

unsigned int message_left

unsigned int message_right

int keylen=strlen(key)

int block_len,plaintext_len=strlen(plaintext_string)

int ciphertext_len=0

ciphertext_string=&ciphertext_buffer[0]

Blowfish_Init(&ctx, (unsigned char *)key, keylen)

print("Plaintext message string is: %s\n", plaintext_string)

Step21: encrypt the plaintext message string

print("Encrypted message string is: ")

while (plaintext_len)

message_left = message_right = 0UL

Step22: crack the message string into a 64-bit block (ok, really two 32-bit blocks); pad with zeros if necessary

```
for (block_len = 0; block_len < 4; block_len++)
 message_left = message_left << 8
 if (plaintext_len)
    message_left += *plaintext_string++
    plaintext_len--
 else message_left += 0
for (block_len = 0; block_len < 4; block_len++)
 message_right = message_right << 8
 if (plaintext_len)
    message_right += *plaintext_string++
    plaintext_len--
 else message_right += 0
```

Step23: encrypt and print the results

```
Blowfish_Encrypt(&ctx, &message_left, &message_right)
printf("%lx%lx", message_left, message_right)
```

Step24: save the results for decryption below

```
*ciphertext_string++ = (unsigned char)(message_left >> 24)
*ciphertext_string++ = (unsigned char)(message_left >> 16)
*ciphertext_string++ = (unsigned char)(message_left >> 8)
*ciphertext_string++ = (unsigned char)message_left
*ciphertext_string++ = (unsigned char)(message_right >> 24)
*ciphertext_string++ = (unsigned char)(message_right >> 16)
```

```
    *ciphertext_string++ = (unsigned char)(message_right >> 8)
    *ciphertext_string++ = (unsigned char)message_right
    ciphertext_len += 8
    printf("\n")
  ciphertext_string = &ciphertext_buffer[0]
  printf("ciphertext_len: %d\n",ciphertext_len)
  return ciphertext_len
int startDecryption(unsigned char *ciphertext_string,char ciphertext_len,char
*key)
    BLOWFISH_CTX ctx
    unsigned int message_left
    unsigned int message_right
    int block_len
    int keylen=strlen(key)
    int plain_stringLen=0
    plain_string=&plain_buffer[0]
    Blowfish_Init(&ctx, (unsigned char *)key, keylen)
    printf("Decrypted message string is: ")
    print("cipher Len: %d\n",ciphertext_len)
    while(ciphertext_len)
      message_left = message_right = 0UL
      for (block_len = 0; block_len < 4; block_len++)
          message_left = message_left << 8
        message_left += *ciphertext_string++
```

```
if (ciphertext_len)
 ciphertext_len--
for (block_len = 0; block_len < 4; block_len++)
 message_right = message_right << 8
 message_right += *ciphertext_string++
 if (ciphertext_len)
 ciphertext_len--
```

Step25: Blowfish_Decrypt(&ctx, &message_left, &message_right)

```
if plaintext message string padded, extra zeros here
 print("%c%c%c%c%c%c%c%c",
 (int)(message_left >> 24), (int)(message_left >> 16),
 (int)(message_left >> 8), (int)(message_left),
 (int)(message_right >> 24), (int)(message_right >> 16),
 (int)(message_right >> 8), (int)(message_right))
 *plain_string++=(unsigned char)(message_left >>24);
 *plain_string++=(unsigned char)(message_left >>16)
 *plain_string++=(unsigned char)(message_left >>8)
 *plain_string++=(unsigned char)(message_left )
 *plain_string++=(unsigned char)(message_right >>24)
 *plain_string++=(unsigned char)(message_right >>16)
 *plain_string++=(unsigned char)(message_right >>8)
 *plain_string++=(unsigned char)(message_right )
 plain_stringLen+=8
```

plain_string=&plain_buffer[0]

return plain_stringLen

Step 26: Stop

Test Data:

Valid Data Set: 54 key has to be give as input

Invalid Data Set: Not applicable

Limiting Data Set: Not applicable

Viva Voce Questions:

1. Name all the encryption algorithms and the differences between them
2. What is the key size and the plain text size in DES
3. How many rounds are performed in DES and why
4. How is S-box operation performed in DES
5. How secured is DES algorithm

(6) Name of the Experiment:

AIM: Using IP TABLES on Linux and setting the filtering rules.

S/W & H/W Requirements:

S/W: Turbo C

H/W: Pentium IV Processor**,** 256 MB RAM,40 GB HDD.

ALGORITHM

rc.test-iptables.txt
The rc.test-iptables.txt (http://iptables-tutorial.frozentux.net/scripts/rc.test-iptables.txt) script can be used to test all the different chains, but it might need some tweaking depending on your configuration, such as turning on ip_forwarding, and setting up *masquerading* etc. It will work for most everyone who has all the basic set up and all the basic tables loaded into kernel. All it really does is set some LOG targets which will log ping replies and ping requests. This way, you will get information on which chain was traversed and in which order. For example, run this script and then do:
ping -c 1 host.on.the.internet
And tail -n 0 -f /var/log/messages while doing the first command. This should show you all the different chains used, and in which order, unless the log entries are swapped around for some reason.
This script was written for testing purposes only. In other words, it's not a good idea to have rules like this that log everything of one sort since your log partitions might get filled up quickly and it would be an effective Denial of Service attack against you and might lead to real attacks on you that would be unlogged after the initial Denial of Service attack.
rc.flush-iptables.txt
The rc.flush-iptables.txt (http://iptables-tutorial.frozentux.net/scripts/rc.flush-iptables.txt) script should not really be called a script in itself. The rc.flush-iptables.txt (http://iptables-tutorial.frozentux.net/scripts/rc.flush-iptables.txt) script will reset and flush all your tables and chains. The script starts by setting the default policies to ACCEPT on the *INPUT*, *OUTPUT* and *FORWARD* chains of the *filter* table. After this we reset the default policies of the *PREROUTING*, *POSTROUTING* and *OUTPUT* chains of the *nat* table. We do this first so we won't have to bother about closed connections and packets not getting through. This script is intended for actually setting up and troubleshooting your firewall, and hence we only care about opening the whole thing up and resetting it to default values. After this we flush all chains first in the *filter* table and then in the *NAT*

table. This way we know there are no redundant rules lying around anywhere. When all of this is done, we jump down to the next section where we erase all the user specified chains in the *NAT* and *filter* tables. When this step is done, we consider the script done. You may consider adding rules to flush your *mangle* table if you use it.
One final word on this issue. Certain people have mailed me asking me to put this script into the original rc.firewall script using Red Hat Linux syntax where you type something like rc.firewall start and the script starts. However, I will not do that since this is a tutorial and should be used as a place to fetch ideas mainly and it shouldn't be filled up with shell scripts and strange syntax. Adding shell script syntax and other things makes the script harder to read as far as I am concerned and the tutorial was written with readability in mind and will continue being so.

Test Data:

Valid Data Set: Not applicable

Invalid Data Set: Not applicable

Limiting Data Set: Not applicable

Result:

Linux will have IP tables

C:\> apropos IP/apropos IPtab*
C:\> where is iptables

Note: Depending on rehat version as 2.4,2.6
C:\> yom –i install iptables.rpm
C:\> optget –i install iptables
C:\> man squid
C:\> iptables –h
C:\> gip
C:\> kip

Viva Voce Questions:

1. What are IP tables.
2. How is the network monitoring done using IP tables.
3. If an attacker gets IP tables how can he use IP table to attack
4. What does an IP table contain?
5. What are the filtering rules for IP table

(7) Name of the Experiment:

AIM: Configuring S/MIME for email communication

S/W & H/W Requirements:

S/W: Turbo C

H/W: Pentium IV Processor**,** 256 MB RAM,40 GB HDD.

Algorithm:

- **MIME-Version:** Must be "1.0" -> RFC 2045, RFC 2046
- **Content-Type:** More types being added by developers (application/word)
- **Content-Transfer-Encoding:** How message has been encoded (radix-64)
- **Content-ID:** Unique identifying character string.
- **Content Description:** Needed when content is not readable text (e.g.,mpeg)

Test Data:

Valid Data Set: Not applicable

Invalid Data Set: Not applicable

Limiting Data Set: Not applicable

Viva Voce Questions:

1. What is S/MIME.
2. Name the five MIME header fields.
3. Explain MIME transfer encoding
4. Explain the format of E-mail and the fields in it.
5. What role does base64 play in S/MIME.

(8) Name of the Experiment:

AIM: Understanding the buffer overflow and format string attacks.

S/W & H/W Requirements:

S/W: Turbo C

H/W: Pentium IV Processor**,** 256 MB RAM,40 GB HDD.

Algorithm:

- Introduction
- Most deadly weapon on the Internet.

- Try to gain partial or complete control over target computer by creating a back door entry.

- Enable the attacker to execute a malicious code on target system.

- Gives root or super access to attacker.

- How it Works…
- Due to casual or careless programming
- Poor Memory Management
- Mismanagement of system variables, pointers and temporary data.
  -by application developers.

THE OVERALL PROCESS:-

1. Identify a vulnerable application.
2. Inject the malicious code.
3. Execute the code.

THE OVERALL PROCESS:-

1. Identify a vulnerable application.
2. Inject the malicious code.
3. Execute the code.
4. Format String Overflows:-
5. Subvert many vulnerable applications
6. These can be executed remotely or locally
7. Exploit a lack of validation in handling and functioning of format strings.
8. In C programs, a large variety of C header files to access std functions
9. printf(), sprintf(), fprintf(), etc
10. Printf("&d",a);

--"&d" represents the decimal data type that is expected to follow
--"a" represents the parameter whose data type is decimal

```
#include<stdio.h>
#include<conio.h>
void main()
{
clrscr();
int input=2;
printf("ABCdef\n\n%n",&input);
printf("value of var:%d",input);
getch();
}
```

- Expected output:
  ABCdef
Value of var: 2

- Real output:
  ABCdef
  Value of var: 8

Test Data:

Valid Data Set: Not applicable

Invalid Data Set: Not applicable

Limiting Data Set: Not applicable

Viva Voce Questions:

1. What is buffer overflow?
2. In a program what is runtime buffer overflow
3. What is Format string attack
4. How format string attack effects the buffer overflow.

(9) Name of the Experiment

AIM: Using NMAP for ports monitoring.

S/W & H/W Requirements:

S/W: Turbo C

H/W: Pentium IV Processor**,** 256 MB RAM,40 GB HDD.

ALGORITHM

Nmap features include:
Ping Sweeping- Identifying computers on a network, for example listing the computers which respond to pings, or which have a particular port open
Port Scanning - Enumerating the open ports on one or more target computers
OS Detection - Remotely determining the operating system and some hardware characteristics of network devices.

Nmap command line prompt:
C:\nmap> nmap -sP 192.168.0.10

Well-Known Ports:
Ports numbered 0 to 1023 are considered well known (also called standard ports) and are assigned to services by the IANA (Internet Assigned Numbers Authority). Here are a few samples:
21 FTP
80 HTTP
110 POP

Table:Scans available through nmap:

Scan Type	Switch	Description
TCP connect() scan	-sT	The most basic form of scanning. This opens a connection to every potentially interesting port on the target machine.
TCP SYN scan	-sS	The "half-open" scan. This scan sends a TCP SYN packet as though it is trying to open the connection. If it receives a SYN-ACK response, it sends an immediate RST to shut down the connection. Because this scan doesn't open the connection, it is less likely to be logged.
Stealth FIN	-sF	This scan attempts to pass through packet filters by sending a TCP FIN packet.
Xmas Tree	-sX	This scan attempts to pass through packet filters by sending a packet with FIN, URG, and PUSH flags set.

Scan Type	Switch	Description
Null	-sN	This scan attempts to pass through packet filters by sending a packet without any flags turned on.
Ping	-sP	This limits the scan to only conducting a ping sweep to look for connected systems. It does not do port scans.
UDP scan	-sU	This sends 0 byte UDP packets to each port on the target machine(s).
ACK scan	-sA	This scan is used to help check packet filters. An ACK packet with random acknowledgment and sequence numbers is sent. If nothing is returned, the port is marked as filtered.
List scan	-sL	Simply lists targets to scan.
FTP bounce scan	-b <ftp relay host>	This scan relives a historical foible of FTP servers. Older FTP servers were able to serve bounced FTP sessions; that is, they connected to another host to deliver data to you. By providing a relay host in the format username:password@server:port, you can use this FTP bounce (mis)feature to scan ports that might otherwise be protected.

Nmap command line prompt:
C:\nmap> nmap -sT 192.168.0.10

Pros:
Fast
Easy to implement
Accurate

Cons:
Easily detected
Traceable

Nmap command line prompt:
C:\nmap> nmap -sS 192.168.0.10

Pros:
Fast
Accurate
Fairly easy to implement
Harder to trace than the TCP connect port-scan method

Cons:
Not totally stealthy
Easily blocked

Nmap command line prompt:
C:\nmap> nmap -sF 192.168.0.7

Pros:
Fairly fast
Easy to implement
Stealthy to a certain extent

Cons:
Inaccurate with certain operating systems

Nmap command line prompt:
C:\nmap> nmap -sX 192.168.0.7

Nmap command line prompt:
C:\nmap> nmap -sN 192.168.0.7

Pros:
Fairly fast
Easy to implement
Stealthy to a certain extent

Cons:
Works only with UNIX and some other operating systems

Nmap command line prompt:
C:\nmap> nmap -sU 192.168.0.10

Nmap command line prompt:
C:\nmap> nmap -sA 68.46.234.161

Nmap command line prompt:
C:\nmap> nmap -sA 68.46.234.161

Option	Explanation
-P0	Tells nmap not to ping hosts before scanning. (This is used to scan hosts that sit behind packet filters that don't allow ICMP traffic.)
-f	Causes nmap to fragment its scanning packets, making it more difficult to block the scan with packet filters.
-v	Puts nmap into verbose mode, causing it to display much more information about what it's doing.
-oN <logfile>	Writes output into a human readable logfile.

Option	Explanation
-oM <logfile>	Writes output into a machine-parsable logfile.
--resume <logfile&;	Resumes an incomplete scan from a logfile.
-iL <logfile>	Causes nmap to read input from a logfile instead of really scanning a host.
-g <portnumber>	Allows you to define the port nmap uses as its source port.
-p <port range>	Allows you to define the range of ports nmap will scan. If no range is given, nmap will scan all the ports listed in its own services file. A range of ports can be given in the following format: -p 20-30,79,6000-6010.

Test Data:

Valid Data Set: IP address

Invalid Data Set: Invalid IP address

Limiting Data Set: Not applicable

Result:

Nmap command line prompt:
C:\nmap> nmap -O www.hackingmobilephones.com

Nmap command line prompt for TCP SYN Scan:
Ex: c:\nmap> nmap -sS www.hackingmobilephones.com
Interesting ports on corp2.net4india.com (202.71.129.91):
Not shown: 1673 filtered ports
PORT STATE SERVICE
7/tcp closed echo
13/tcp closed daytime
21/tcp open ftp
25/tcp open smtp
53/tcp closed domain
80/tcp open http
110/tcp closed pop3

113/tcp closed auth
123/tcp closed ntp
143/tcp closed imap
366/tcp closed odmr
433/tcp closed nnsp
443/tcp closed https
610/tcp closed npmp-local
626/tcp closed unknown
631/tcp closed ipp
1433/tcp closed ms-sql-s
1434/tcp closed ms-sql-m
3306/tcp open mysql
5900/tcp closed vnc
6000/tcp closed X11
6001/tcp closed X11:1
6002/tcp closed X11:2
7938/tcp closed lgtomapper
Nmap run completed at Sat Nov 24 11:21:58 2007 -- 1 IP address (1 host up) scanned in 242.562 seconds

Nmap command line prompt for OS Detection
C:\nmap> nmap -O www.hackingmobilephones.com
Ex: c:\nmap> nmap -oN op.txt -O www.hackingmobilephones.com
Interesting ports on corp2.net4india.com (202.71.129.91):
Not shown: 1673 filtered ports
PORT STATE SERVICE
7/tcp closed echo
13/tcp closed daytime
20/tcp closed ftp-data
21/tcp open ftp
25/tcp open smtp
53/tcp closed domain
80/tcp open http
110/tcp closed pop3
113/tcp closed auth
123/tcp closed ntp
199/tcp closed smux
366/tcp closed odmr
433/tcp closed nnsp
443/tcp closed https
626/tcp closed unknown
631/tcp closed ipp
1433/tcp closed ms-sql-s
1434/tcp closed ms-sql-m
5800/tcp closed vnc-http

5900/tcp closed vnc
6000/tcp closed X11
6001/tcp closed X11:1
6002/tcp closed X11:2
7938/tcp closed lgtomapper
Device type: general purpose
Running: OpenBSD 3.X
OS details: OpenBSD 3.6 x86 with pf "scrub in all"
Uptime: 28.341 days (since Sat Oct 27 02:20:14 2007)
OS detection performed. Please report any incorrect results at http://insecure.org/nmap/submit/ .
Nmap run completed at Sat Nov 24 10:31:00 2007 -- 1 IP address (1 host up) scanned in 264.500 seconds

Viva Voce Questions:

1. What is NMAP.
2. what do you understand by dedicated ports.
3. If a packet does not belong to a particular port number what does a router do
4. How does NMAP help in network monitoring
5. How can a network administrator monitor ports.

10) Name of the Experiment:

AIM: Implementation of proxy based security protocols in c or c++ with feathers like confidentiality, integrity and authentication.

S/W & H/W Requirements:

S/W: Turbo C

H/W: Pentium IV Processor**,** 256 MB RAM,40 GB HDD.

ALGORITHM:

/sbin/iptables –F INPUT
/sbin/iptables –F OUTPUT
/sbin/iptables –t filter-F FORWARD
/sbin/iptables –P INPUT DROP
/sbin/iptables –P OUTPUT ACCEPT
/sbin/iptables –P FORWARD DROP
Iptables - A INPUT –i eth0 state –state RELATED ,ESTABLISHED – J ACCEPT
iptables - A FORWARD –i eth0 state –state RELATED ,ESTABLISHED – J ACCEPT
iptables - A FORWARD –i eth1 – J ACCEPT
iptables - A FORWARD –i eth0 –p tcp –dport 80 –d 200.0.2.1 – J ACCEPT
iptables - A FORWARD –i eth0 –p tcp –dport telnet –d 200.0.2.1 – J ACCEPT
iptables- A INPUT –I eth0 –p tcp –dport http –j ACCEPT
iptables- A INPUT –I eth0 –p tcp –dport ssh –j ACCEPT

Test Data:

Valid Data Set: IP Table

Invalid Data Set: Invalid IP table

Limiting Data Set: Not applicable

Viva Voce Questions:

1. What is confidentiality?
2. What is Authentication?
3. What is Integrity?
4. What is a proxy server and how do u configure it.
5. How an ISP use proxy server
6. In a organization if proxy server is configured, can that work as firewall?

References:

Books:

Network Security Essentials (Applications and Standards) by William Stallings Pearson Education

Hack Proofing your Network by Ryan Russel, dan kaminsky,Rain Forest Puppy,Joe Garnd,David Ahmed,Hal Flynn Ido Dubdrasky,Steve W.MAnzuik and Ryan Permeh,wiley Dreamtech.

Web Sites:

1. http://linuxcommand.org/man_pages/openssl1.html
2. http://www.openssl.org/docs/apps/openssl.html
3. http://www.queen.clara.net/pgp/art3.html

4. http://www.ccs.ornl.gov/~hongo/main/resources/contrib/gpg-howto/gpg- howto.html
5. https://netfiles.uiuc.edu/ehowes/www/gpg/gpg-com-0.htm
6. http://www.ethereal.com/docs/user-guide/