

# 컴퓨터 보안

(Term Project)



학과:컴퓨터공학과

학번:201312799

성명:김태형

제출 일자:2019/11/16

담당 교수님:서경룡 교수님

## <목차>

1. Term Project 개요 및 목표-----	3
1.1 Term Project 개요	
1.2 Term Project 목표	
2. Term Project 진행 및 사용-----	4
2.1 1-step Basic	
2.2 2-step Symmetric Key Encryption	
2.3 3-step Diffie-Hellman	
2.4 4-step RSA	
2.5 5-step RSA+Diffie-Hellman	
2.6 6-step RSA+Diffie-Hellman+Hash	
2.7 7-step Openssl	
3. 정리 및 요약-----	25
3.1 전체과정 요약	
3.2 보안상 문제점 및 개선된 내용	
3.3 느낀 점	
4. 참조-----	26
5. 코드-----	26

## 1. Term Project 개요 및 목표

### 1.1 Term Project 개요

암호화와 프로토콜을 이해하기 위하여 Socket Programming을 이용하여 암호화 및 프로토콜이 적용된 Server 및 Client를 작성합니다. Server 및 Client가 포함하는 기능은 가장 기본인 Version부터 최종 목표까지 포함된 Version까지 진행합니다. 각 기능들은 다음과 같습니다.

### 1.2 Term Project 목표

Term Project의 목표는 다음과 같습니다.

1. 사용자 인증이 가능하고 간단한 연산을 수행하는 Server-Client Program을 작성합니다.
  - 사용자 정보를 pw 파일에 보관하고 ID와 PWD로 인증합니다.
  - 서버의 기능은 하나의 정수를 입력받아 제곱 값을 Client로 전송합니다.
  - 동작을 test 하고 결과를 보입니다.
  - Trudy가 사용자 정보를 획득할 수 있는 방법을 제시합니다.
  - Trudy가 어떤 값을 입력했는지 알 수 있는 방법을 제시합니다.
  - 위의 결과를 토대로 Program의 보안상 문제점과 개선방법을 설명합니다.
2. Symmetric Key를 사용하여 사용자 인증과정을 암호화합니다.
3. Diffie-Hellman Key 교환 방법을 사용하여 인증과정과 통신내용을 암호화합니다.
4. RSA 방식을 활용하여 프로그램을 개선합니다.
5. OpenSSL을 활용하여 프로그램을 개선합니다.
6. 전체과정을 요약하고 보안상 문제점과 개선된 내용을 정리합니다.

위 목표들을 아래의 Term Project를 통하여 확인하고 결과를 확인합니다. Program 및 실행 결과들은 Ubuntu Linux를 사용하여 진행하였습니다.

## 2. Term Project 진행 및 사용

### 2.1 1-step Basic

첫 번째 단계에서는 다음과 같은 과정을 실험하고 결과를 확인합니다.

1) 사용자 인증이 가능하고 간단한 연산을 수행하는 Server-Client Program을 작성합니다.

- 사용자 정보를 pw 파일에 보관하고 ID와 PWD로 인증합니다.

먼저 userData.txt 파일을 만들어 유저 정보를 저장하였습니다.



- 서버의 기능은 하나의 정수를 입력받아 제공 값을 Client로 전송합니다.

Server측에서는 파일에서 문자열을 읽어 Client로 부터 받은 입력 값을 문자열과 비교하여 ID와 PWD가 같은 지 비교를 하여 같으면 next라는 메시지를 보내고 다르다면 ID 인증 실패라는 문구를 출력하고 Program을 종료합니다.

```
fgets(buf1,sizeof(buf1),user);
fgets(buf2,sizeof(buf2),user);
```

```
if(strcmp(buf1,ID)==0){
    write(cliSock,next,sizeof(next));
    bzero(PWD,sizeof(PWD));
    printf("PWD :");
    read(cliSock,PWD,sizeof(PWD));
    ~
    ~
    ~
else{
    printf("ID Authecation Failed\n");
    fclose(user);
    break;
}
```

Client는 ID를 입력하고 next라는 메시지를 받으면 PWD 입력 화면으로 넘어갑니다. 파일의 저장된 ID, PWD 값과 일치하지 않는다면. ID(PWD) 인증실패라는 문구를 출력하고 Program을 종료합니다.

```
if(strncmp(next,"next",4)==0){
    printf("input PWD : ");
    bzero(PWD,sizeof(PWD));
    n = 0;
    while((PWD[n++] = getchar()) != '\n');
    write(servSock,PWD,sizeof(PWD));

    ~
    ~
else{
    printf("ID Authcation Failed\n");
    break;
}
```

• 동작을 test 하고 결과를 보입니다.

#### ① 로그인 성공

다음은 로그인 성공 시 Server측 화면입니다.

```
root@xogud019-15Z980-GA50K:/home/xogud019/Desktop/term/term01# ./1s 10000
socket creation success
binding success
listen success
server accept
connected
ID :dkdksnsnru
PWD :tkfkd214
Authcation Success
Message Receive : 3

Send Message
Message Receive : Server Exit
ID :Server Exit
```

로그인을 성공하면 Client로부터 정수를 입력받고 그 제공 값을 다시 전송합니다.

다음은 성공 시 Client측 화면입니다.

```
root@xogud019-15Z980-GA50K:/home/xogud019/Desktop/term/term01# ./1c
server address:127.0.0.1
port number:10000
socket create success
connet server
input ID : dkdksnsnru
input PWD : tkfkd214
Authcation Success

Send Message : 3

Receive Message : 9
Send Message : exit
client Exit
input ID : exit
ID Authcation Failed
```

Server로부터 제공 값을 받은 것을 확인 할 수 있습니다.

② 로그인 실패 시 화면입니다.

먼저 Server측 화면입니다.

```
root@xogud019-15Z980-GA50K:/home/xogud019/Desktop/term/term01# ./1s 10000
socket creation success
binding success
listen success
server accept
connected
ID :tkfkd214
ID Authecation Failed
root@xogud019-15Z980-GA50K:/home/xogud019/Desktop/term/term01#
```

실패 시 Servrt는 ID 인증 실패라는 문구를 출력하고 Program을 종료합니다.

Client측 화면입니다.

```
root@xogud019-15Z980-GA50K:/home/xogud019/Desktop/term/term01# ./1c
server address:127.0.0.1
port number:10000
socket create success
connet server
input ID : tkfkd214
ID Authecation Failed
root@xogud019-15Z980-GA50K:/home/xogud019/Desktop/term/term01#
```

역시 마찬가지로 ID 인증 실패를 화면에 출력하고 Program을 종료합니다.

• Trudy가 사용자 정보를 획득할 수 있는 방법을 제시합니다.

1-step의 결과로 로그인 가능하고 제공 값을 되돌려 받는 Server-Client Server가 완성되었습니다. 하지만 위의 결과는 전혀 암호화가 되어있지 않기에 Trudy가 정보를 획득하는 방법은 많습니다. 먼저 사용자 정보의 경우에는 WireShark를 사용하여 패킷 스니핑을 하여 암호화 되지 않은 사용자 정보를 획득할 수 있고 혹은 Server측의 Computer를 침입했다고 가정할 시 userData.txt는 암호화가 되어있지 않기에 그 파일을 확인할 수 있습니다.

그 예로 userData.txt 파일은 위의 사진과 같이 암호화 없이 저장되어있습니다. 그리고 WireShark를 통하여 패킷 스니핑의 결과입니다.

The image shows a Wireshark packet capture window. The top menu bar includes File, Edit, View, Go, Capture, Analyze, Statistics, Telephony, Wireless, Tools, and Help. The packet list table shows the following entries:

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	127.0.0.1	127.0.0.1	TCP	76	34178 → 10000 [SYN] Seq=0 Win=65495 Len=0 MSS=65495
2	0.000022840	127.0.0.1	127.0.0.1	TCP	76	10000 → 34178 [SYN, ACK] Seq=0 Ack=1 Win=65483 Len=0
3	0.000045660	127.0.0.1	127.0.0.1	TCP	68	34178 → 10000 [ACK] Seq=1 Ack=1 Win=65536 Len=0 TSva
4	1.571241229	127.0.0.1	127.0.0.1	TCP	1092	34178 → 10000 [PSH, ACK] Seq=1 Ack=1 Win=65536 Len=1
5	1.571291507	127.0.0.1	127.0.0.1	TCP	68	10000 → 34178 [ACK] Seq=1 Ack=1025 Win=64512 Len=0
6	1.571422330	127.0.0.1	127.0.0.1	TCP	1092	10000 → 34178 [PSH, ACK] Seq=1 Ack=1025 Win=65536 Len=1
7	1.571446158	127.0.0.1	127.0.0.1	TCP	68	34178 → 10000 [ACK] Seq=1025 Ack=1025 Win=64512 Len=0
8	2.472695919	127.0.0.1	127.0.0.1	TCP	1092	34178 → 10000 [PSH, ACK] Seq=1025 Ack=1025 Win=65536
9	2.472857281	127.0.0.1	127.0.0.1	TCP	1092	10000 → 34178 [PSH, ACK] Seq=1025 Ack=2049 Win=65536

The packet details pane shows the selected packet (No. 4) with the following information:

- Frame 4: 1092 bytes on wire (8736 bits), 1092 bytes captured (8736 bits) on interface 0
- Linux cooked capture
- Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
- Transmission Control Protocol, Src Port: 34178, Dst Port: 10000, Seq: 1, Ack: 1, Len: 1024
- Data (1024 bytes)

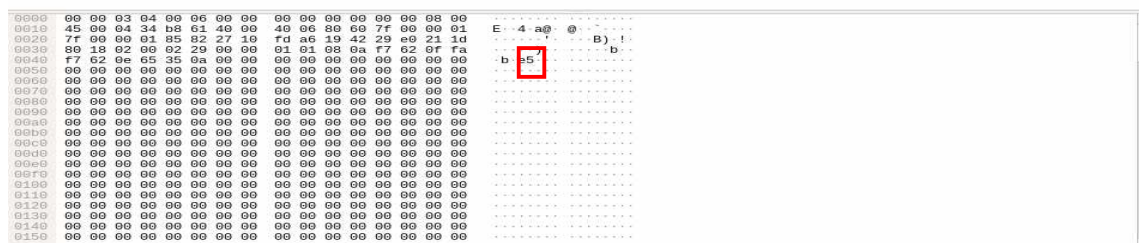
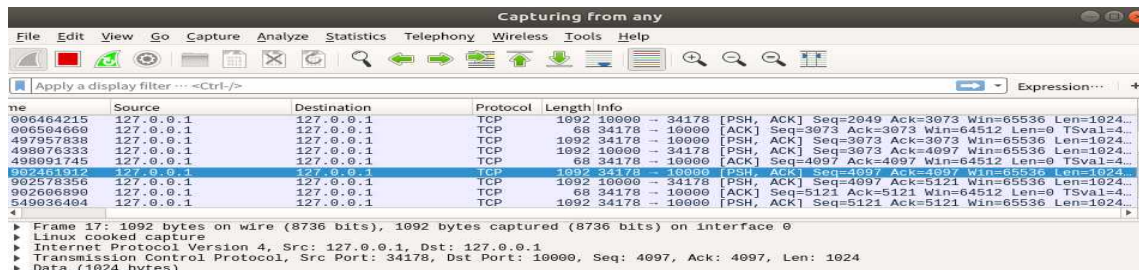
The data field is expanded, showing a hex dump and ASCII representation. The ASCII representation shows the string 'dtkdk:snrnru' highlighted with a red box.



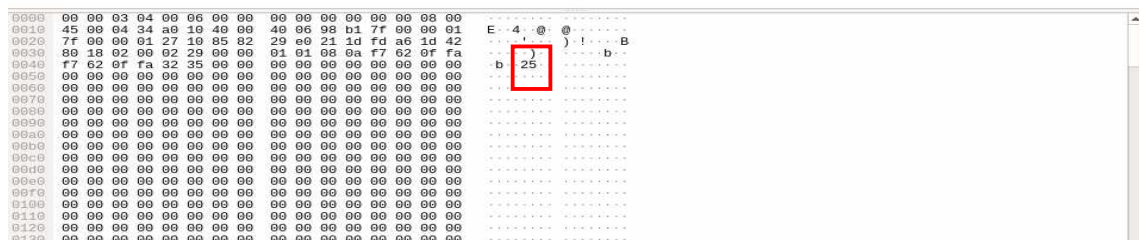
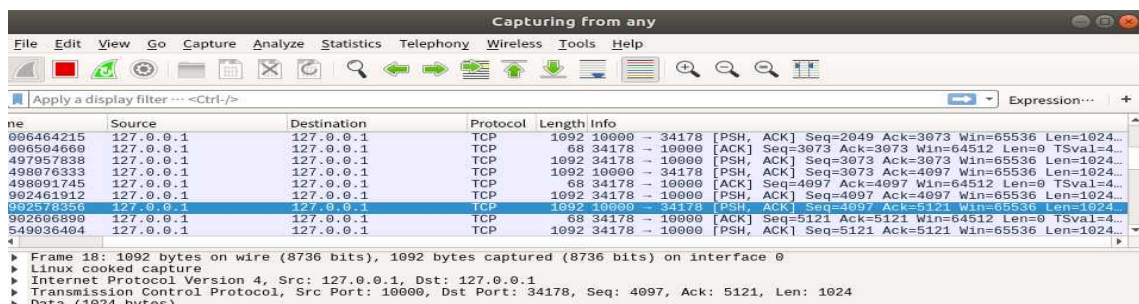
다음과 같이 ID가 노출되어 있는 것을 확인할 수 있습니다.

- Trudy가 어떤 값을 입력했는지 알 수 있는 방법을 제시합니다.

값 입력 역시 패킷 스니핑을 통하여 Trudy가 알아낼 수 있습니다. 혹은 중간자 공격을 통해 중간에서 패킷을 가로채어 확인할 수 있습니다.



다음과 같이 5라는 입력을 한 것을 확인할 수 있습니다. 마찬가지로 Server측에서 보내는 메시지 역시 확인 할 수 있습니다.



- 위의 결과를 토대로 Program의 보안상 문제점과 개선방법을 설명합니다.

위의 결과를 통하여 중간자 공격을 통한 패킷 가로채기와 메시지와 통신내용에 그 어떤 암호화를 하지 않았기 때문에 패킷 스니핑에 취약한 것을 확인 할 수 있습니다.

개선방법은 인증과정과 메시지를 암호화하여 전송하여 패킷 스니핑과 가로채기를 방지 할 수

있습니다. 암호화 방식을 무엇을 선택하느냐에 따라 보안이 강력할 수도 있고 암호화가 되었음에도 불구하고 취약할 수도 있습니다. 남은 과정을 통해 단계별로 보안을 강화합니다.

## 2.2 2-step Symmetric Key Encryption

먼저 Symmetric Key를 이용하여 암호화를 진행하였습니다. 이번 Project에서는 암호화하고 전송하고 복호화하는 과정을 살펴보기 위하여 최대한 간단한 Symmetric 암호를 사용하였습니다. Caesar 암호를 사용하여 Project를 진행하였습니다.

```
for(i = 0; (i < 100 && ID[i] != '\0'); i++)  
    ID[i] = ID[i] + 3;  
write(servSock, ID, sizeof(ID));
```

다음과 같이 for문을 사용하여 각 문자열을 3씩 증가시켜 암호화를 진행합니다. 이렇게 암호화를 한 후에 Server-Client Server를 실행합니다.

먼저 Client측에만 암호화를 설정하였을 시 결과입니다. 먼저 Server측 화면입니다.

```
root@xogud019-15Z980-GA50K:/home/xogud019/Desktop/term/term02# ./2s 10000  
socket creation success  
binding success  
listen success  
server accept  
connected  
ID Authcation Failed
```

Client측 화면입니다.

```
root@xogud019-15Z980-GA50K:/home/xogud019/Desktop/term/term02# ./2c  
server address:127.0.0.1  
port number:10000  
socket create success  
connet server  
input ID : dkdksnsnru  
ID Authcation Failed
```

아까와 같이 ID를 입력하였지만 로그인을 실패한 것을 확인 할 수 있습니다. 그리고 양측 모두 암호화를 하였을 시 결과입니다.

먼저 Server측 화면입니다.

```
root@xogud019-15Z980-GA50K:/home/xogud019/Desktop/term/term02# ./2s 10000  
socket creation success  
binding success  
listen success  
server accept  
connected  
ID :dkdksnsnru  
PWD :tkfkd214  
Authcation Success  
Message Receive : 3
```



Client측 화면입니다.

```
root@xogud019-15Z980-GA50K:/home/xogud019/Desktop/term/term02# ./2c
server address:127.0.0.1
port number:10000
socket create success
connet server
input ID : dkdksnsnru
input PWD : tkfkd214
Authecation Success

Send Message : 3

Receive Message : 9
```

다음과 같이 아까와 같이 인증을 성공하고 성공적으로 정수를 보내고 그 제공 값을 다시 되돌려 받은 것을 확인할 수 있습니다.

그리고 양측의 인증과정을 암호화 하였을 시 패킷을 확인한 결과입니다.

① Client -> Server로 보내는 ID입니다.

The image shows a Wireshark packet capture window. The top menu bar includes File, Edit, View, Go, Capture, Analyze, Statistics, Telephony, Wireless, Tools, and Help. The packet list table shows the following packets:

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	127.0.0.1	127.0.0.1	TCP	76	46592 → 11223 [SYN] Seq=0 Win=65495 Len=0 MSS=65495
2	0.000034464	127.0.0.1	127.0.0.1	TCP	76	11223 → 46592 [SYN, ACK] Seq=0 Ack=1 Win=65483 Len=0
3	0.000061718	127.0.0.1	127.0.0.1	TCP	68	46592 → 11223 [ACK] Seq=1 Ack=1 Win=65536 Len=0 TSva
4	1.530610028	127.0.0.1	127.0.0.1	TCP	168	46592 → 11223 [PSH, ACK] Seq=1 Ack=1 Win=65536 Len=1
5	1.530664702	127.0.0.1	127.0.0.1	TCP	68	11223 → 46592 [ACK] Seq=1 Ack=101 Win=65408 Len=0 TS
6	1.530795571	127.0.0.1	127.0.0.1	TCP	168	11223 → 46592 [PSH, ACK] Seq=1 Ack=101 Win=65536 Len
7	1.530827846	127.0.0.1	127.0.0.1	TCP	68	46592 → 11223 [ACK] Seq=101 Ack=101 Win=65536 Len=0
8	2.703752349	127.0.0.1	127.0.0.1	TCP	168	46592 → 11223 [PSH, ACK] Seq=101 Ack=101 Win=65536 L
9	2.703930771	127.0.0.1	127.0.0.1	TCP	168	11223 → 46592 [PSH, ACK] Seq=101 Ack=201 Win=65536 L

The packet details pane shows the selected packet (Frame 4) with the following information:

- Frame 4: 168 bytes on wire (1344 bits), 168 bytes captured (1344 bits) on interface 0
- Linux cooked capture
- Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
- Transmission Control Protocol, Src Port: 46592, Dst Port: 11223, Seq: 1, Ack: 1, Len: 100
- Data (100 bytes)

The packet bytes pane shows the raw data of the selected packet. The data is displayed in hexadecimal and ASCII. The ASCII column shows the string "dkdksnsnru" in red, which is the ID being sent from the client to the server.

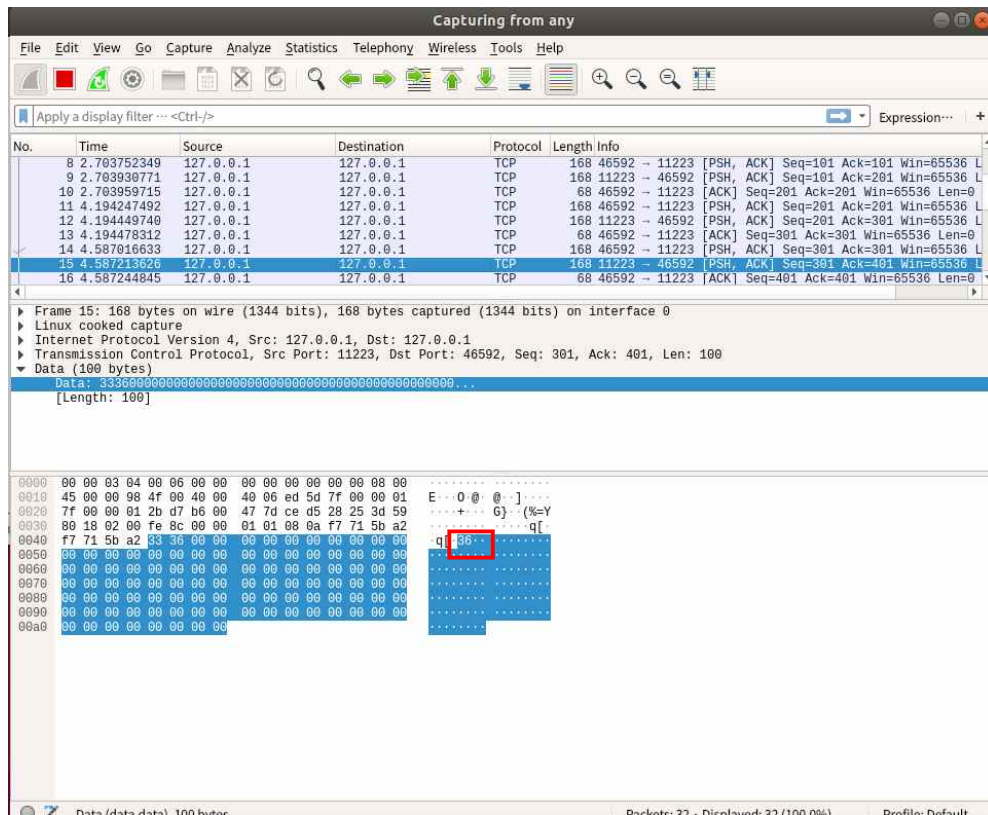
[illegible]

The image shows the Wireshark network protocol analyzer interface. The top menu bar includes File, Edit, View, Go, Capture, Analyze, Statistics, Telephony, Wireless, Tools, and Help. The toolbar contains various icons for file operations, capture, analysis, and display. The main display area is divided into three panes:

- Packet List:** Shows a list of captured packets. The selected packet is a TCP ACK packet with sequence number 12223, acknowledgment number 301, and length 168 bytes.
- Packet Details:** Shows the hierarchical structure of the selected packet. It includes Ethernet II, Internet Protocol Version 4, and Transmission Control Protocol. The TCP section shows the acknowledgment number 301 and the length 168 bytes.
- Packet Bytes:** Shows the raw data of the selected packet in hexadecimal and ASCII. A red box highlights the 'E' flag in the TCP flags field.

The status bar at the bottom indicates that the data is from the 'data.data' file, 100 bytes, and that 32 packets are displayed (100.0%).

④ Server -> Client로 보내는 제곱 값입니다.



ID와 PWD는 위와 같이 대칭 Key를 이용하여 암호화가 성공한 것을 확인할 수 있었습니다. 하지만 메시지는 암호화하지 않았기에 역시 1번의 보내는 메시지는 암호화가 되지 않은 것을 확인할 수 있었습니다. 지금은 서로 고정된 대칭 Key를 이용하여 암호화를 하였지만 rand 함수를 이용하여 랜덤으로 Key를 만들어 사용하여도 그 대칭 Key를 서로 보내는 패킷 역시 암호화를 하지 않았기에 쉽게 Trudy가 알아낼 수 있습니다.

### 2.3 3-step Diffie-Hellman

위의 대칭 Key는 서로 같은 Key를 공유하여 하기에 Key만 알아낸다면 역시나 쉽게 암호문을 풀 수 있습니다. 이를 보안하기 위해 Diffie-Hellman 알고리즘을 이용하여 이 Program을 개선하였습니다. 먼저 rand 함수를 사용하여 개인 Key를 생성하고

```
srand(time(NULL));
int servSecKey =rand()%10+1;
int servPubKey =compute(g,servSecKey,p);
int servKey;
int p = 20;
int g = 13;
```

다음과 같은 compute 함수를 통하여 공개 Key를 생성합니다.

```
int compute(int a, int m, int n){
    int r;
    int y = 1;

    while (m > 0)
    {
        r = m % 2;

        if (r == 1){
            y = (y*a) % n;
        }
        a = a*a % n;

        m = m / 2;
    }

    return y;
}
```

그리고 다음을 통해 서로서로 공개 Key를 보내어 암호화를 복호화할 Key를 계산합니다.

```
bzero(cDHM,sizeof(cDHM));
read(cliSock,cDHM,sizeof(cDHM));
servKey = atoi(cDHM);
servKey = compute(servKey,servSecKey,p);

bzero(sDHM,sizeof(sDHM));
sprintf(sDHM,"%d",servPubKey);
write(cliSock,sDHM,sizeof(sDHM));
printf("%d\n",servKey);
```

그리고 아까와 같은 Symmetric Key 암호화에 이 Key를 사용하여 암호/복호화를 합니다.

```
for(i = 0; (i < 100 && ID[i] != '\0'); i++)
    ID[i] = ID[i] - servKey;
```

이와 같이 Program을 개선하고 실행하여 결과를 확인합니다.

Server측 화면입니다.

```
root@xogud019-15Z980-GA50K:/home/xogud019/Desktop/term/term03# ./3s 12346
socket creation success
binding success
listen success
server accept
connected
17
ID :dkdksnsnru
PWD :tkfkd214
Authcation Success
Message Receive : 3

Send Message
Message Receive : 4

Send Message
Message Receive : 5
```

다음과 같이 Key가 생성된 것을 확인할 수 있습니다.

Client측 화면입니다.

```
root@xogud019-15Z980-GA50K:/home/xogud019/Desktop/term/term03# ./3c
server address:127.0.0.1
port number:12346
socket create success
connet server
17
input ID : dkdksnsnru
input PWD : tkfkd214
Authcation Success

Send Message : 3

Receive Message : 9
Send Message : 4

Receive Message : 16
Send Message : 5

Receive Message : 25
Send Message : exit
```

Client측도 Server측과 같은 Key를 보유하고 있는 것을 확인할 수 있습니다.

다음으로 WireShark로 패킷을 검출한 결과입니다.



The image shows the Wireshark network protocol analyzer interface. The top menu bar includes File, Edit, View, Go, Capture, Analyze, Statistics, Telephony, Wireless, Tools, and Help. The toolbar contains various icons for file operations, capture control, and analysis. The main display area is divided into three panes:

- Packet List:** Shows a list of captured packets. The selected packet is No. 20, which is an HTTP GET request from 127.0.0.1 to 127.0.0.1 on port 80.
- Packet Details:** Shows the hierarchical structure of the selected packet. It includes Ethernet II, Internet Protocol Version 4, Transmission Control Protocol, and Hypertext Transfer Protocol.
- Packet Bytes:** Shows the raw data of the selected packet in hexadecimal and ASCII. A red box highlights the 'U' flag in the TCP header, indicating the Urgent flag is set.

The image shows the Wireshark network protocol analyzer interface. At the top, the title bar reads "Capturing from any". The menu bar includes File, Edit, View, Go, Capture, Analyze, Statistics, Telephony, Wireless, Tools, and Help. Below the menu is a toolbar with various icons for file operations, capture control, and analysis. A filter bar at the top of the packet list shows "Apply a display filter --- <Ctrl-/>".

The packet list pane displays a table of captured packets. The selected packet is number 23, a TCP ACK from 127.0.0.1 to 127.0.0.1, port 201. The details pane shows the structure of this packet: Ethernet II (Type: IPv4), Internet Protocol Version 4 (Src: 127.0.0.1, Dst: 127.0.0.1), and Transmission Control Protocol (Src Port: 47584, Dst Port: 201, Seq: 201, Ack: 301, Len: 100). The data field shows the raw bytes of the packet, with a length of 100 bytes.

The packet bytes pane displays the raw data of the selected packet. The first few bytes are 00 00 03 04 00 06 00 00, which correspond to the Ethernet II header (Type: IPv4). The packet is captured on interface 0.



③ Client -> Server로 보내는 정수입니다.

Wireshark packet capture showing a client sending a packet to a server. The packet is highlighted in blue. The data field shows a sequence of bytes, with a red box highlighting the first few bytes: 00 00 03 04 00 06 00 00 00 00 00 00 08 00.

④ Server -> Client로 보내는 제곱 값입니다.

Wireshark packet capture showing a server sending a packet to a client. The packet is highlighted in blue. The data field shows a sequence of bytes, with a red box highlighting the first few bytes: 00 00 03 04 00 06 00 00 00 00 00 00 08 00.

다음과 같이 성공적으로 암호화가 된것을 확인할 수 있습니다. 매번 연결 시에 다른 Key를

생성하기에 아까보단 조금 더 보안이 잘되었다고 확인할 수 있습니다. 그리고 이 Key 값을 찾는 것은 이산 로그 문제이기에 풀기 어렵기에 안전하다고 생각할 수 있습니다. 하지만 이 Key를 이용하여 간단한 Symmetric Key 암호화에 사용하였기에 역시 전수 조사등으로 메시지나 ID, PWD를 알아낼 수 있습니다.

## 2.4 4-step RSA

RSA알고리즘을 이용하여 Program을 한번 더 개선합니다. RSA를 활용하기 위해 다음과 같은 변수들과 함수를 이용합니다.

```
int num1,N,oN,prime1,prime2,e,d;
```

소수인지 확인을 하는 함수입니다.

```
int checkPrime(int n) {
    int i;
    int m = n / 2;

    for (i = 2; i <= m; i++) {
        if (n % i == 0) {
            return 0; // Not Prime
        }
    }

    return 1; // Prime
}
```

GCD를 찾는 함수입니다.

```
int findGCD(int n1, int n2) {
    int i, gcd;

    for(i = 1; i <= n1 && i <= n2; ++i) {
        if(n1 % i == 0 && n2 % i == 0)
            gcd = i;
    }

    return gcd;
}
```

암/복호화하는 함수입니다.

```
int powMod(int a, int b, int n) {
    long long x = 1, y = a;

    while (b > 0) {
        if (b % 2 == 1)
            x = (x * y) % n;
        y = (y * y) % n; // Squaring the base
        b /= 2;
    }

    return x % n;
}
```

그리고 다음을 통해 e, d 값을 찾습니다.

```
N = prime1*prime2;

oN = (prime1-1)*(prime2-1);

e = 0;
for (e = 5; e <= 100000; e++) {
    if (findGCD(oN, e) == 1)
        break;
}

d = 0;
for (d = e + 1; d <= 100000; d++) {
    if ( ((d * e) % oN) == 1)
        break;
}
```

이 중 e와 N 값은 Client에게 전송합니다.

```
bzero(RSAM,sizeof(RSAM));
sprintf(RSAM,"%d",N);
write(cliSock,RSAM,sizeof(RSAM));

bzero(RSAM,sizeof(RSAM));
sprintf(RSAM,"%d",e);
write(cliSock,RSAM,sizeof(RSAM));
```

그리고 다음을 통해 암호화하여 메시지를 전송합니다.

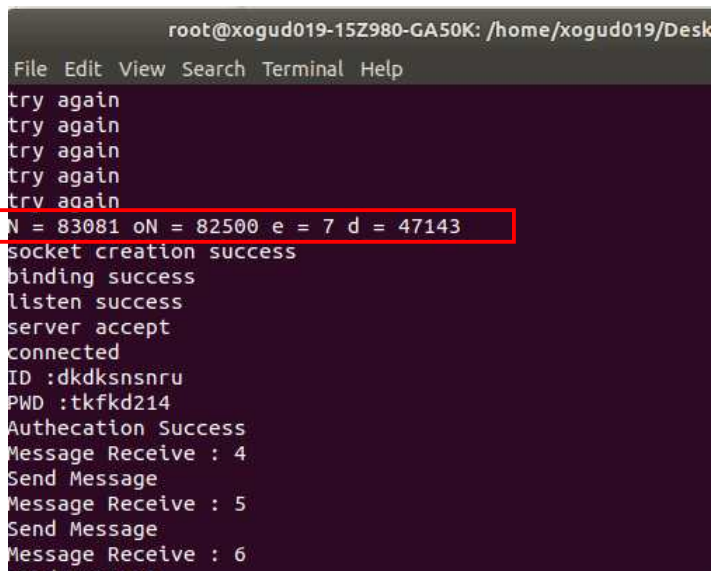
```
num = powMod(num,d,N);
sprintf(sendM,"%d",num);
write(cliSock,sendM,sizeof(sendM))
```

그리고 다음을 통해 복호화하여 메시지를 수신합니다.

```
num1 = powMod(num1,d,N);
sprintf(sendM,"%d",num1)
```

\*Client측은 e를 사용하여 암호/복호화합니다.

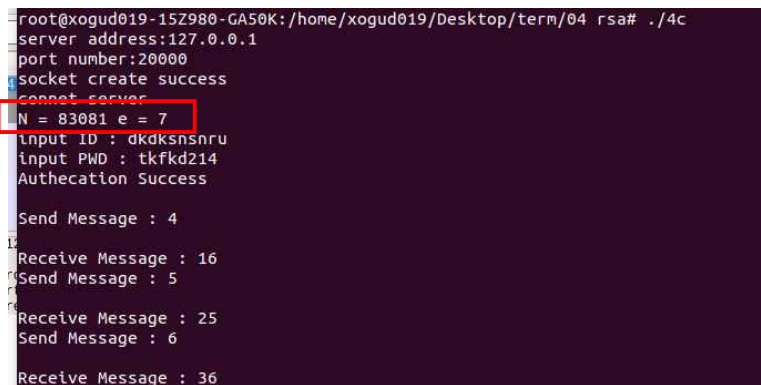
이렇게 Program을 개선하여서 실행 후 결과를 확인합니다.  
Server측 화면입니다.



```
root@xogud019-15Z980-GA50K: /home/xogud019/Desk
File Edit View Search Terminal Help
try again
try again
try again
try again
try again
N = 83081 oN = 82500 e = 7 d = 47143
socket creation success
binding success
listen success
server accept
connected
ID :dkdksnsru
PWD :tkfkd214
Authcation Success
Message Receive : 4
Send Message
Message Receive : 5
Send Message
Message Receive : 6
Send Message
```

다음과 같이 값들이 생성된 것을 확인할 수 있습니다.

Client측 화면입니다.



```
root@xogud019-15Z980-GA50K: /home/xogud019/Desktop/term/04 rsa# ./4c
server address:127.0.0.1
port number:20000
socket create success
connect server
N = 83081 e = 7
input ID : dkdksnsru
input PWD : tkfkd214
Authcation Success

Send Message : 4
Receive Message : 16
Send Message : 5
Receive Message : 25
Send Message : 6
Receive Message : 36
```

다음과 같이 N과 e 값을 Server로부터 받은 것을 확인할 수 있습니다.

**Capturing from any**

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

Apply a display filter ... «Ctrl+» Expression...

No.	Time	Source	Destination	Protocol	Length	Info
9	7.985856118	127.0.0.1	127.0.0.1	TCP	168	34396 → 20900 [PSH, ACK] Seq=1 Ack=201 Win=65536 Len=100 TSval=2448162...
7	7.985892042	127.0.0.1	127.0.0.1	TCP	68	20900 → 34396 [ACK] Seq=201 Ack=181 Win=65536 Len=0 TSval=2448162...
11	7.985985675	127.0.0.1	127.0.0.1	TCP	168	20900 → 34396 [PSH, ACK] Seq=201 Ack=101 Win=65536 Len=100 TSval=...
12	7.986069875	127.0.0.1	127.0.0.1	TCP	68	34396 → 20900 [ACK] Seq=101 Ack=301 Win=65536 Len=0 TSval=...
13	7.983287238	127.0.0.1	127.0.0.1	TCP	168	34396 → 20900 [PSH, ACK] Seq=101 Ack=301 Win=65536 Len=100 TSval=...
14	9.134096547	127.0.0.1	127.0.0.1	TCP	168	20900 → 34396 [PSH, ACK] Seq=301 Ack=201 Win=65536 Len=100 TSval=...
9	9.134125149	127.0.0.1	127.0.0.1	TCP	68	34396 → 20900 [ACK] Seq=201 Ack=401 Win=65536 Len=0 TSval=2448163...
16	11.218332207	127.0.0.1	127.0.0.1	TCP	68	34396 → 20900 [PSH, ACK] Seq=201 Ack=501 Win=65536 Len=0 TSval=...
17	11.218319932	127.0.0.1	127.0.0.1	TCP	168	20900 → 34396 [PSH, ACK] Seq=401 Ack=301 Win=65536 Len=100 TSval=...
18	11.218841344	127.0.0.1	127.0.0.1	TCP	68	34396 → 20900 [ACK] Seq=301 Ack=501 Win=65536 Len=0 TSval=2448165...

▶ Frame 16: 168 bytes on wire (1344 bits), 168 bytes captured (1344 bits) on interface 0  
 ▶ Linux cooked capture  
 ▶ Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1  
 ▶ Transmission Control Protocol, Src Port: 34396, Dst Port: 20900, Seq: 201, Ack: 401, Len: 100  
 ▶ Data (100 bytes)

② Server -> Client로 보내는 제공 값입니다.

Wireshark packet capture interface showing a list of network packets. The interface includes a menu bar (File, Edit, View, Go, Capture, Analyze, Statistics, Telephony, Wireless, Tools, Help), a toolbar with icons for various functions, and a packet list pane. The packet list pane shows 18 packets, with packet 17 selected. The details pane for packet 17 shows the IP header information: 17.11.2188416932 to 127.0.0.1, TCP Seq=401, Ack=301, Win=65536, Len=100.

No.	Time	Source	Destination	Protocol	Length	Info
9	7.985856118	127.0.0.1	127.0.0.1	TCP	168	34396 → 20000 [PSH, ACK] Seq=1 Ack=201 Win=65536 Len=100 TSval=24...
10	7.985892042	127.0.0.1	127.0.0.1	TCP	68	20000 → 34396 [ACK] Seq=201 Ack=101 Win=65536 Len=0 TSval=2448162...
11	7.985985675	127.0.0.1	127.0.0.1	TCP	168	20000 → 34396 [PSH, ACK] Seq=201 Ack=101 Win=65536 Len=100 TSval=...
12	7.986069075	127.0.0.1	127.0.0.1	TCP	68	34396 → 20000 [ACK] Seq=101 Ack=301 Win=65536 Len=0 TSval=2448162...
13	13.9320730	127.0.0.1	127.0.0.1	TCP	168	34396 → 20000 [PSH, ACK] Seq=101 Ack=301 Win=65536 Len=100 TSval=...
14	9.134096547	127.0.0.1	127.0.0.1	TCP	168	20000 → 34396 [PSH, ACK] Seq=301 Ack=201 Win=65536 Len=100 TSval=...
15	9.134125149	127.0.0.1	127.0.0.1	TCP	68	34396 → 20000 [ACK] Seq=201 Ack=401 Win=65536 Len=0 TSval=2448163...
16	11.218633267	127.0.0.1	127.0.0.1	TCP	168	34396 → 20000 [PSH, ACK] Seq=201 Ack=401 Win=65536 Len=100 TSval=...
17	11.218816932	127.0.0.1	127.0.0.1	TCP	168	20000 → 34396 [PSH, ACK] Seq=401 Ack=301 Win=65536 Len=100 TSval=...
18	11.218841344	127.0.0.1	127.0.0.1	TCP	68	34396 → 20000 [ACK] Seq=301 Ack=501 Win=65536 Len=0 TSval=2448165...

▶ Frame 17: 168 bytes on wire (1344 bits), 168 bytes captured (1344 bits) on interface 0

- Linux cooked capture
- Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
- Transmission Control Protocol, Src Port: 20000, Dst Port: 34396, Seq: 401, Ack: 301, Len: 100
- Data (100 bytes)

[illegible]

다음과 같이 암호화가 된 것을 확인할 수 있습니다. 이 RSA 알고리즘 역시 소수의 곱과 GCD

를 이용하여 암호화를 하기에 패킷 스니핑으로 e 값을 알더라도 이를 복호화 하기 위해서 d 값을 알아야 하기에 쉽게 Trudy가 쉽게 내용을 알 수 없습니다.

## 2.5 5-step RSA+Diffie-Hellman

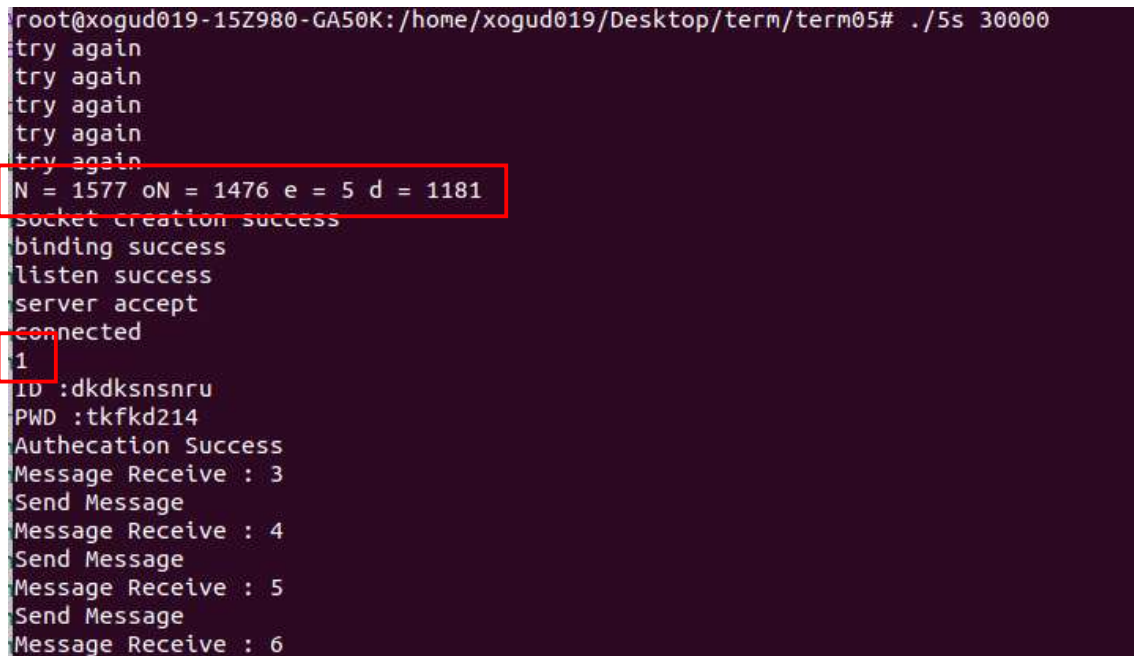
위 두 과정을 이용하여 RSA 알고리즘을 이용하여 메시지를 서명한 후 Diffie-Hellman Key 교환 알고리즘을 이용하여 Key를 교환하여 이것을 사용하여 Symmetric 방식으로 암호화하여 위 과정을 다시 반복하였습니다.

다음과 같은 순서로 암호화 하였습니다.

```
num = powMod(num,d,N);
sprintf(sendM,"%d",num);
for(i = 0; (i < 100 && sendM[i] != '\0'); i++)
sendM[i] = sendM[i] + servKey;
write(cliSock,sendM,sizeof(sendM));
```

다음과 같이 RSA를 사용하여 서명 후 DH Key를 이용하여 암호화한 후 메시지를 전달하였습니다.

Server측 화면입니다.



```
root@xogud019-15Z980-GA50K: /home/xogud019/Desktop/term/term05# ./5s 30000
try again
try again
try again
try again
try again
N = 1577 oN = 1476 e = 5 d = 1181
socket creation success
binding success
listen success
server accept
connected
1
ID :dkdksnsnru
PWD :tkfkd214
Authcation Success
Message Receive : 3
Send Message
Message Receive : 4
Send Message
Message Receive : 5
Send Message
Message Receive : 6
```

다음과 같이 RSA 값들을 생성하고 DH Key 역시 생성된 것을 확인 할 수 있습니다.



```
root@xogud019-15Z980-GA50K: /home/xogud019/Desktop/term/term05# ./5c
server address:127.0.0.1
port number:30000
socket create success
connet server
N = 1577 e = 5
1
input ID : dkdksnsnru
input PWD : tkfkd214
Authcation Success

Send Message : 3

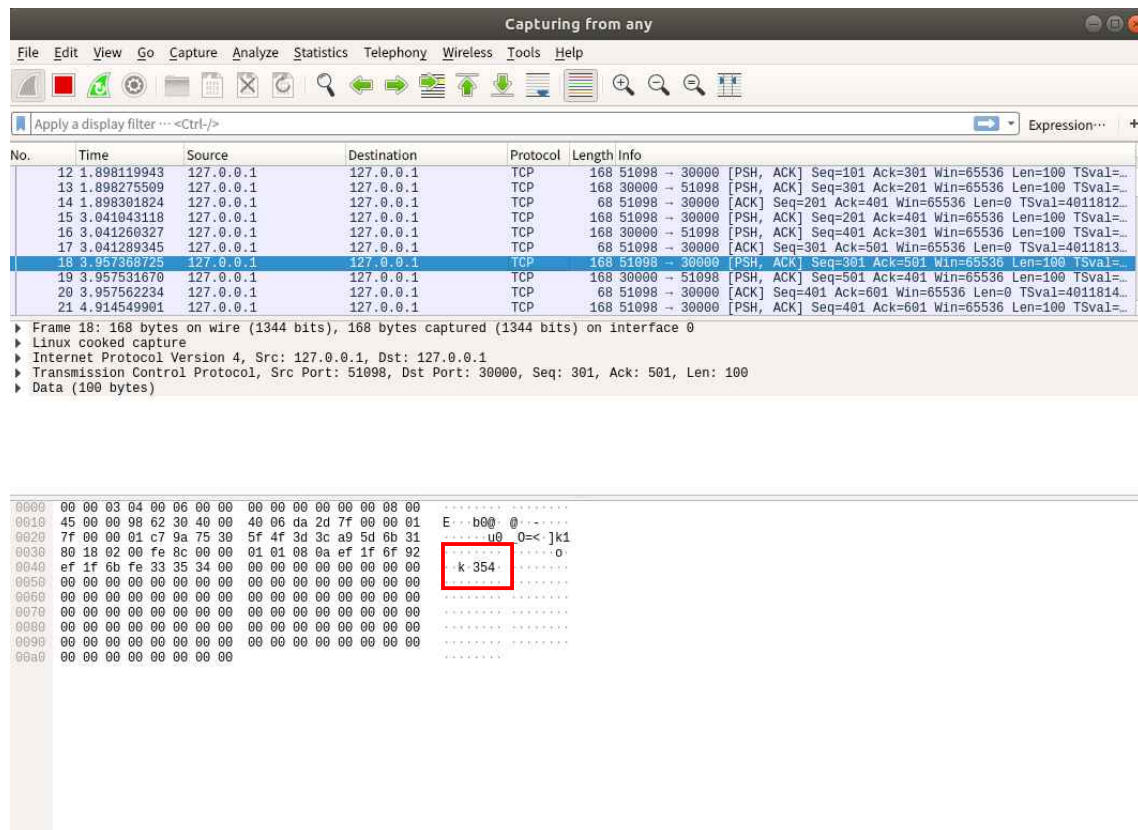
Receive Message : 9
Send Message : 4

Receive Message : 16
Send Message : 5

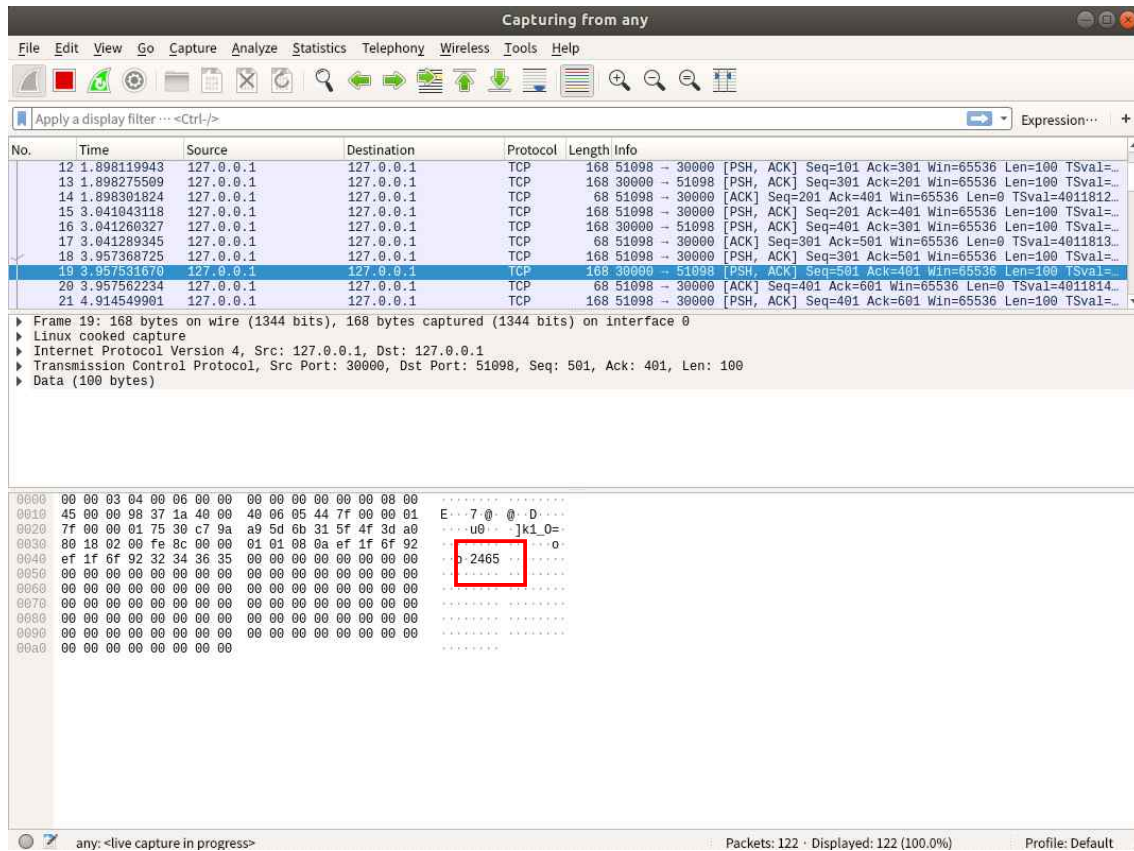
Receive Message : 25
Send Message : 6

Receive Message : 36
```

① Client -> Server로 보내는 정수입니다.



② Server -> Client로 보내는 제곱 값입니다.



역시 마찬가지로 성공적으로 암호화가 진행된 것을 확인할 수 있습니다. 여기까지 패킷 스니핑을 통한 Key 노출은 줄었지만 Server측의 파일을 직접 탈취하는 경우는 보안이 강화되지는 않았습니다. 이를 해결하기 위해 Hash를 이용하여 Program을 개선합니다.

## 2.6 6-step RSA+Diffie-Hellman+Hash

Hash 알고리즘을 사용하여 저장한 userData.txt는 다음과 같습니다.



사용하기 위해 필요한 변수와 함수는 다음과 같습니다.

먼저 필요한 변수를 선언합니다.

```
long hash_id, hash_pwd;
int legnth = 100;
```

그리고 다음과 같은 함수를 사용하여 Hash를 사용합니다.

```
unsigned int RSHash(const char* str, unsigned int length){
    unsigned int b = 378551;
    unsigned int a = 63689;
    unsigned int hash = 0;
    unsigned int i = 0;

    for (i = 0; i < length; ++str, ++i){
        hash = hash * a + (*str);
        a = a * b;
    }

    return hash;
}
```

해쉬는 가장 기본적인 RS Hash를 사용하였습니다.

다음을 이용하여 Hash를 사용하여 암호화한 ID를 보냅니다.

```
while((ID[n++] = getchar()) != '\n');
hash_id = RSHash(ID, length);
sprintf(ID, "%d", hash_id);
```

위 함수들을 이용하여 Hash를 사용하여 Program을 개선하여 실행 후 확인하였습니다.  
Server측 화면입니다.

```
root@xogud019-15Z980-GA50K: /home/xogud019/Desktop/term/term06# ./6s 50000
try again
try again
try again
N = 2491 oN = 2392 e = 5 d = 957
socket creation success
binding success
listen success
server accept
connected
1
ID :-53037186110
```

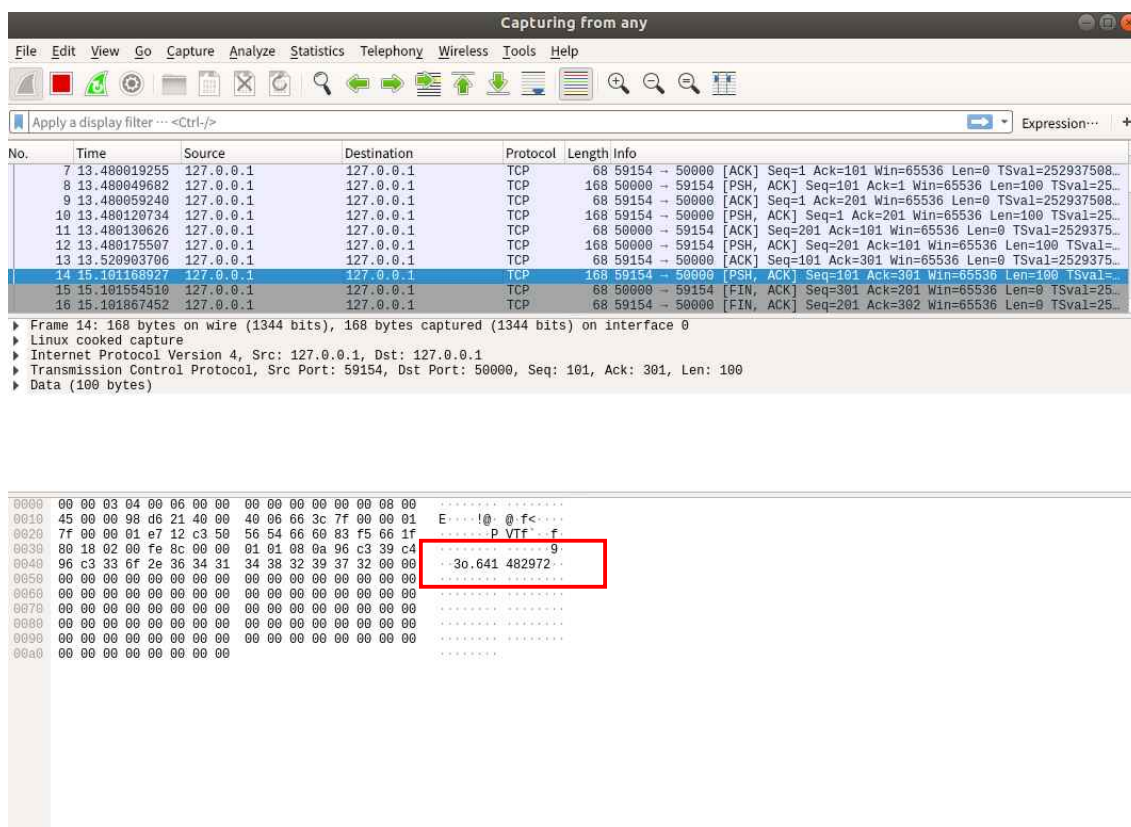
다음과 같이 ID가 Hash 처리된 값으로 넘어온 것을 확인할 수 있습니다.

Client측 화면입니다.

```
root@xogud019-15Z980-GA50K: /home/xogud019/Desktop/term/term06# ./6c
server address:127.0.0.1
port number:50000
socket create success
connet server
N = 2491 e = 5
1
input ID : dkdknsnru
```

Client측에서는 평소와 같이 ID를 입력하는 것을 확인할 수 있습니다.

다음으로 WireShark로 확인한 결과입니다.



다음과 같이 Hash 값이 넘어간 것을 확인할 수 있습니다.

## 2.7 7-step Openssl

마지막으로 OpenSSL을 활용한 Program 개선입니다. OpenSSL로 새로운 Socket Program을 만들었지만 구동을 못하여서 확인을 못하였습니다. 코드는 아래에 추가로 첨부합니다.

### 3. 정리 및 요약

#### 3.1 전체과정 요약

- 처음으로 만든 Server-Client Server는 간단한 ID, PWD 인증을 통하여 연결을 하여 Client가 정수 값 하나를 보내면 Server측에서는 그 정수의 제곱 값을 반환하는 Server를 작성하였습니다.

- ID, PWD 인증과정(이하 로그인 과정)을 Symmetric Key 암호화를 사용하여 (이 TermProject에서는 Caesar 암호를 사용하였습니다.) 로그인 과정을 암호화/복호화하여 통신을 하였습니다.

- Diffie-Hellman Key 알고리즘을 사용하여 개인 Key와 공유 Key를 만들고 공유 Key를 공개함으로써 암호/복호화 Key를 만들어서 이를 이용하여 암호/복호화하여 위 과정을 다시 반복하여 진행하였습니다.

- RSA 알고리즘을 활용하여 Key를 생성하여 이 Key를 통하여 Client와 Server가 주고 받는 메시지를 암호화/복호화를 진행하게 Program을 개선하였습니다.

- RSA 알고리즘과 DH 알고리즘을 같이 사용하여 RSA Key를 이용하여 서명하고 DH 알고리즘을 이용한 Symmetric Key 암호화를 통하여 한번 더 암호화 후 통신을 하게 개선을 하였습니다.

- 마지막으로 Hash 함수를 적용하여 개인정보가 저장된 File을 Hash 처리하고 이것을 받아서 로그인 인증을 하는 방식으로 최종 개선을 하였습니다.

#### 3.2 보안상 문제점 및 개선된 내용

처음으로 만든 Program에서는 패킷 스니핑, 중간자 가로채기, Server측 컴퓨터 접속 후 File 탈취, 전수조사 등이 있었습니다. 처음 Symmetric Key 암호화를 적용하여 패킷 스니핑과 중간자 가로채기로 훔쳐보는 것들을 방지하였습니다. 하지만 그 방식도 Key를 탈취하는 방법이 존재하기에 DH 알고리즘과 RSA 알고리즘을 사용하여 공개 Key가 가로채여도 개인 Key가 없으면 풀 수 없는 방식으로 개선하였습니다. 그리고 역으로 풀기 힘든 방식(예로 DH 알고리즘의 이산 로그 문제)을 사용하여 Program을 개선하였습니다. 그리고 저장된 File 훔쳐도 알아낼 수 없게 Hash를 적용하여 최종적으로 개선하였습니다.

#### 3.3 느낀 점

단순히 그냥 과제로 암호화 프로그램만 사용하였을 때에는 암호화/복호화가 어떤 식으로 되는지 알았습니다. 하지만 어떻게 적용되어 사용되는지는 막연하였는데 이번 TermProject를 통하여서 실제로 만든 Server-Client Program에 적용을 하여 WireShark를 통하여 확인하여 보면서 어떻게 암호화가 진행이 되는지 또 암호화가 진행이 안됐을 시 Packet에 얼마나 노출이 잘되는가를 확인하였습니다. 생각보다 암호화를 하지 않았을 시 얼마나 취약한가에 대해 알았고 또 알려고 한다면 무수히 많은 공격법이 있다는 것을 깨달았습니다. OpenSSL을 확인 못 하여서 아쉬웠지만 추가적으로 더 공부하여 Program을 더 개선하여 최종적으로 확

인해보겠습니다.

#### 4. 참조

[[Beej's Socket Programming](#)]

[스토리로 이해하는 암호화 알고리즘/저자 김수민]

#### 5. 코드

```
Server code입니다.
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <netinet/in.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <sys/un.h>
#include <time.h>

#define MAX 100
#define DEFAULT_PROTOCOL 0

int square(int num);           //find square
int checkPrime(int n);         //rsa start
int findGCD(int n1, int n2);
int powMod(int a, int b, int n); //rsa end
int compute(int a, int m, int n); //dH

int main(int argc, char *argv[]){

    srand(time(NULL));

    int servSock, cliSock, serverlen, clientlen, nRcv;
    int num;
    int n;
    int i;                       //dh
    int p = 20;                  //dh
    int g = 13;                  //dh
    int num1, N, oN, prime1, prime2, e, d; //rsa
    int servSecKey = rand()%10+1; //dh secret key
    int servPubKey = compute(g, servSecKey, p); //dh public key
    int servKey;                 //dh simmetric key
```



```

FILE *user;                                //id,pwd file
char RSAM[MAX];                            //rsa socket
char sDHM[MAX];                            //dh socket
char cDHM[MAX];                            //dh socket
char sendM[MAX];
char next[MAX] = "next";
char ID[MAX],PWD[MAX];
char buf1[MAX],buf2[MAX];                 //id,pwd string save
char *line_p;

struct sockaddr_in serverAddr, clientAddr;

if(argc !=2){
    printf("insert port Number\n");
    exit(1);
}

if((user= fopen("userData.txt","r")) == NULL){
    printf("error");
    exit(2);
}

while(1){
    prime1 = rand()%100+1;
    prime2 = rand()%100+1;
    if (!(checkPrime(prime1) && checkPrime(prime2)))
        printf("try again\n");
    else if (!checkPrime(prime1))
        printf("try again\n");
    else if (!checkPrime(prime2))
        printf("try again\n");
    else
        break;
}

N = prime1*prime2;

oN = (prime1-1)*(prime2-1);

e = 0;
for (e = 5; e <= 100000; e++) {

```

```

        if (findGCD(oN, e) == 1)
            break;
    }

    d = 0;
    for (d = e + 1; d <= 100000; d++) {
        if ( ((d * e) % oN) == 1)
            break;
    }

    printf("N = %d oN = %d e = %d d = %d\n",N,oN,e,d);

    fgets(buf1,sizeof(buf1),user);
//    if((line_p = strchr(buf1,'\n'))!=NULL)*line_p = '\0';
    fgets(buf2,sizeof(buf2),user);
//    if((line_p = strchr(buf2,'\n'))!=NULL)*line_p = '\0';

    servSock = socket(AF_INET,SOCK_STREAM, DEFAULT_PROTOCOL);

    if(servSock == -1){
        printf("socket creation failed\n");
        exit(1);
    }
    else{
        printf("socket creation success\n");
    }

    memset(&serverAddr,0,sizeof(struct sockaddr_in));
    serverAddr.sin_family = AF_INET;
    serverAddr.sin_port = htons(atoi(argv[1]));
    serverAddr.sin_addr.s_addr = htonl(INADDR_ANY);

    if(bind(servSock,(struct sockaddr*)&serverAddr, sizeof(serverAddr))!=0){
        printf("binding failed\n");
        exit(1);
    }
    else{
        printf("binding success\n");
    }

    if(listen(servSock,5) != 0){

```

```

        printf("listen failed\n");
        exit(1);
    }
    else{
        printf("listen success\n");
    }
    clientlen = sizeof(clientAddr);
    cliSock = accept(servSock,(struct sockaddr*)&clientAddr,&clientlen);
    printf("server accept\n");
    printf("connected\n");

    bzero(RSAM,sizeof(RSAM));                //rsa exchange start
    sprintf(RSAM,"%d",N);
    write(cliSock,RSAM,sizeof(RSAM));

    bzero(RSAM,sizeof(RSAM));
    sprintf(RSAM,"%d",e);
    write(cliSock,RSAM,sizeof(RSAM));        //rsa exchange end

    bzero(cDHM,sizeof(cDHM));                //dh start
    read(cliSock,cDHM,sizeof(cDHM));
    servKey = atoi(cDHM);
    servKey = compute(servKey,servSecKey,p);  //key compute

    bzero(sDHM,sizeof(sDHM));
    sprintf(sDHM,"%d",servPubKey);
    write(cliSock,sDHM,sizeof(sDHM));        //dh end
    printf("%d\n",servKey);

    while(1){
        bzero(ID,sizeof(ID));
        printf("ID :");
        read(cliSock,ID,sizeof(ID));
        for(i = 0; (i < 100 && ID[i] != '\0'); i++)
            ID[i] = ID[i] - servKey;
        if(strncmp("exit",ID,4) ==0){
            fclose(user);
            printf("Server Exit\n");
            break;
        }
    }

```

```

printf("%s",ID);
if(strcmp(buf1,ID)==0){
    write(cliSock,next,sizeof(next));
    bzero(PWD,sizeof(PWD));
    printf("PWD :");
    read(cliSock,PWD,sizeof(PWD));
    for(i = 0; (i < 100 && PWD[i] != '\0'); i++)
        PWD[i] = PWD[i] - servKey;
    if(strncmp("exit",PWD,4) ==0){
        fclose(user);
        printf("Server Exit\n");
        break;
    }

    printf("%s",PWD);
    if(strcmp(buf2,PWD)==0){
        printf("Authcation Success\n");
        write(cliSock,next,sizeof(next));
        while(1){
            bzero(sendM,sizeof(sendM));
            printf("Message Receive : ");
            read(cliSock,sendM,sizeof(sendM));
            for(i = 0; (i < 100 && sendM[i] != '\0'); i++)
                sendM[i] = sendM[i] - servKey;
            num1 = atoi(sendM);
            num1 = powMod(num1,d,N);
            sprintf(sendM,"%d",num1);
            if(strncmp("exit",sendM,4) ==0){
                fclose(user);
                printf("Server Exit\n");
                break;
            }

            printf("%s\n",sendM);
            num = atoi(sendM);
            num = square(num);
            num = powMod(num,d,N);
            sprintf(sendM,"%d",num);
            for(i = 0; (i < 100 && sendM[i] != '\0'); i++)
                sendM[i] = sendM[i] + servKey;
            write(cliSock,sendM,sizeof(sendM));

```

```

                                printf("Send Message \n");
                                }
                                }//PWD if
                                else{
                                    fclose(user);
                                    printf("PWd Authecation Failed\n");
                                    break;
                                }
                                }//ID if
                                else{
                                    int ret = strcmp(buf1,ID);
                                    printf("%d\n",ret);
                                    fclose(user);
                                    printf("ID Authecation Failed\n");
                                    break;
                                }
                                }//while
                                }
}

```

```

int square(int num){
    int squares = 0;

    squares = (num*num);
    return squares;
}

```

```

int checkPrime(int n) {
    int i;
    int m = n / 2;

    for (i = 2; i <= m; i++) {
        if (n % i == 0) {
            return 0; // Not Prime
        }
    }

    return 1; // Prime
}

```

```

int findGCD(int n1, int n2) {

```

```

        int i, gcd;

        for(i = 1; i <= n1 && i <= n2; ++i) {
            if(n1 % i == 0 && n2 % i == 0)
                gcd = i;
        }

        return gcd;
    }

int powMod(int a, int b, int n) {
    long long x = 1, y = a;

    while (b > 0) {
        if (b % 2 == 1)
            x = (x * y) % n;
        y = (y * y) % n; // Squaring the base
        b /= 2;
    }

    return x % n;
}

int compute(int a, int m, int n){
    int r;
    int y = 1;

    while (m > 0)
    {
        r = m % 2;

        if (r == 1){
            y = (y*a) % n;
        }
        a = a*a % n;

        m = m / 2;
    }

    return y;
}

```



Client Code입니다.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <netinet/in.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <sys/un.h>
#include <time.h>

#define MAX 100
#define DEFAULT_PROTOCOL 0

unsigned int RSHash(const char* str, unsigned int length); //hash
int powMod(int a, int b, int n); //rsa
int compute(int a, int m, int n); //dh

int main(int argc, char **argv){

    srand(time(NULL));

    int servSock, cliSock, serverlen, clientlen, nRcv;
    int sPort;
    int num;
    long hash_id, hash_pwd; //hash
    int legnth = 100; //hash
    int n=0;
    int p = 20; //dh
    int g = 13; //dh
    int cliSecKey = rand()%10+1; //dh
    int cliPubKey = compute(g, cliSecKey, p); //dh
    int cliKey; //dh
    int i; //dh
    int num1, N, e; //rsa
    char sAddr[15];
    char sendM[MAX];
    char RSAM[MAX]; //rsa socket
    char sDHM[MAX]; //dh socket
```

```

char cDHM[MAX];                                //dh socket
char ID[MAX],PWD[MAX];
char next[MAX];
struct sockaddr_in serverAddr, clientAddr;
struct hostent *host;

printf("server address:");
gets(sAddr);
printf("port number:");
gets(sendM);
sPort = atoi(sendM);

cliSock = socket(AF_INET,SOCK_STREAM, DEFAULT_PROTOCOL);

memset(&serverAddr,0,sizeof(struct sockaddr_in));
serverAddr.sin_family = AF_INET;
serverAddr.sin_port = htons(sPort);
serverAddr.sin_addr.s_addr = inet_addr(sAddr);

if((servSock = socket(AF_INET,SOCK_STREAM,DEFAULT_PROTOCOL)) == -1){
    printf("socket create error \n");
    exit(1);
}
else{
    printf("socket create success\n");
}

if(connect(servSock,(struct sockaddr*)&serverAddr, sizeof(serverAddr))<0){
    printf("connect server failed\n");
    exit(1);
}
else{
    printf("connet server\n");
}

bzero(RSAM,sizeof(RSAM));                    //rsa exchange start
read(servSock,RSAM,sizeof(RSAM));
N =atoi(RSAM);

bzero(next,sizeof(RSAM));

```

```

read(servSock,RSAM,sizeof(RSAM));
e = atoi(RSAM);
printf("N = %d e = %d\n", N,e);          //rsa exchange end

bzero(cDHM,sizeof(cDHM));                //dh start
sprintf(cDHM,"%d",cliPubKey);
write(servSock,cDHM,sizeof(cDHM));

bzero(sDHM,sizeof(sDHM));
read(servSock,sDHM,sizeof(sDHM));
cliKey = atoi(sDHM);
cliKey = compute(cliKey,cliSecKey,p);    //key compute
printf("%d\n",cliKey);                  //dh end

while(1){
    printf("input ID : ");
    bzero(ID,sizeof(ID));
    n = 0;
    while((ID[n++] = getchar()) != '\n');
    hash_id = RSHash(ID,legnth);
    sprintf(ID,"%d",hash_id);
    for(i = 0; (i < 100 && ID[i] != '\0'); i++)
        ID[i] = ID[i] + cliKey;
    write(servSock,ID,sizeof(ID));
    bzero(next,sizeof(next));
    read(servSock,next,sizeof(next));
    if((strcmp(next,"next",4))==0){
        printf("input PWD : ");
        bzero(PWD,sizeof(PWD));
        n = 0;
        while((PWD[n++] = getchar()) != '\n');
        hash_pwd = RSHash(PWD,legnth);
        sprintf(PWD,"%d",hash_pwd);
        for(i = 0; (i < 100 && PWD[i] != '\0'); i++)
            PWD[i] = PWD[i] + cliKey;
        write(servSock,PWD,sizeof(PWD));
        bzero(next,sizeof(next));
        read(servSock,next,sizeof(next));
        if((strcmp(next,"next",4))==0){
            printf("Authcation Success\n");
            while(1){

```

```

        bzero(sendM,sizeof(sendM));
        printf("\nSend Message : ");

        n = 0;
        while((sendM[n++] = getchar()) != '\n');
        num1 = atoi(sendM);
        num1 = powMod(num1,e,N);
        sprintf(sendM,"%d",num1);
        for(i = 0; (i < 100 && sendM[i] != '\0'); i++)
            sendM[i] = sendM[i] + cliKey;
        write(servSock,sendM,sizeof(sendM));
        if((strcmp(sendM,"exit",4))==0){
            printf("client Exit \n");
            break;
        }

        bzero(sendM,sizeof(sendM));
        printf("\nReceive Message : ");
        read(servSock,sendM,sizeof(sendM));
        for(i = 0; (i < 100 && sendM[i] != '\0'); i++)
            sendM[i] = sendM[i] - cliKey;
        num = atoi(sendM);
        num = powMod(num,e,N);
        printf("%d",num);
    }
} //PWD if
else{
    printf("PWD Authecation Failed\n");
    break;
}
} //ID if
else{
    printf("ID Authecation Failed\n");
    break;
}
}

} //while
}

int powMod(int a, int b, int n) {

```

```

        long long x = 1, y = a;

        while (b > 0) {
            if (b % 2 == 1)
                x = (x * y) % n;
            y = (y * y) % n; // Squaring the base
            b /= 2;
        }

        return x % n;
    }
}

```

```

int compute(int a, int m, int n){
    int r;
    int y = 1;

    while (m > 0)
    {
        r = m % 2;

        if (r == 1){
            y = (y*a) % n;
        }
        a = a*a % n;

        m = m / 2;
    }

    return y;
}

```

```

unsigned int RSHash(const char* str, unsigned int length){
    unsigned int b    = 378551;
    unsigned int a    = 63689;
    unsigned long hash = 0;
    unsigned int i    = 0;

    for (i = 0; i < length; ++str, ++i)
    {
        hash = hash * a + (*str);
        a    = a * b;
    }
}

```

```
}  
  
    return hash;  
}
```

OpenSSL Server Code 입니다

```
#include <stdio.h>
#include <unistd.h>
#include <malloc.h>
#include <string.h>
#include <sys/socket.h>
#include <resolv.h>
#include <openssl/ssl.h>
#include <openssl/err.h>

#define FAIL    -1

int OpenListener(int port){
    int sd;
    struct sockaddr_in addr;

    sd = socket(PF_INET, SOCK_STREAM, 0);
    bzero(&addr, sizeof(addr));
    addr.sin_family = AF_INET;
    addr.sin_port = htons(port);
    addr.sin_addr.s_addr = INADDR_ANY;
    if ( bind(sd, &addr, sizeof(addr)) != 0 ){
        perror("can't bind port");
        abort();
    }
    if ( listen(sd, 10) != 0 ){
        perror("Can't configure listening port");
        abort();
    }
    return sd;
}

SSL_CTX* InitServerCTX(void){
    SSL_METHOD *method;
    SSL_CTX *ctx;

    OpenSSL_add_all_algorithms();
    SSL_load_error_strings();
    method = SSLv2_server_method();
    ctx = SSL_CTX_new(method);
}
```



```

    if ( ctx == NULL ){
        ERR_print_errors_fp(stderr);
        abort();
    }
    return ctx;
}

void LoadCertificates(SSL_CTX* ctx, char* CertFile, char* KeyFile){

    if ( SSL_CTX_use_certificate_file(ctx, CertFile, SSL_FILETYPE_PEM) <= 0 ){
        ERR_print_errors_fp(stderr);
        abort();
    }

    if ( SSL_CTX_use_PrivateKey_file(ctx, KeyFile, SSL_FILETYPE_PEM) <= 0 ){
        ERR_print_errors_fp(stderr);
        abort();
    }

    if ( !SSL_CTX_check_private_key(ctx) ){
        fprintf(stderr, "Private key does not match the public certificate\n");
        abort();
    }
}

void ShowCerts(SSL* ssl){
    X509 *cert;
    char *line;

    cert = SSL_get_peer_certificate(ssl);
    if ( cert != NULL )
    {
        printf("Server certificates:\n");
        line = X509_NAME_oneline(X509_get_subject_name(cert), 0, 0);
        printf("Subject: %s\n", line);
        free(line);
        line = X509_NAME_oneline(X509_get_issuer_name(cert), 0, 0);
        printf("Issuer: %s\n", line);
        free(line);
        X509_free(cert);
    }
}

```

```

    else
        printf("No certificates.\n");
}

void Servlet(SSL* ssl) {
    char buf[1024];
    char reply[1024];
    int sd, bytes;
    const char* HTMLEcho("<html><body><pre>%s</pre></body></html>\n\n");

    if ( SSL_accept(ssl) == FAIL )
        ERR_print_errors_fp(stderr);
    else{
        ShowCerts(ssl);
        bytes = SSL_read(ssl, buf, sizeof(buf));
        if ( bytes > 0 ) {
            buf[bytes] = 0;
            printf("Client msg: \"%s\"\n", buf);
            sprintf(reply, HTMLEcho, buf);
            SSL_write(ssl, reply, strlen(reply));
        }
        else
            ERR_print_errors_fp(stderr);
    }
    sd = SSL_get_fd(ssl);
    SSL_free(ssl);
    close(sd);
}

int main(int count, char *strings[]){    SSL_CTX *ctx;
    int server;
    char *portnum;

    if ( count != 2 ) {
        printf("Usage: %s <portnum>\n", strings[0]);
        exit(0);
    }
    portnum = strings[1];
    ctx = InitServerCTX();
    LoadCertificates(ctx, "newreq.pem", "newreq.pem");
    server = OpenListener(atoi(portnum));

```

```

while (1) {
    struct sockaddr_in addr;

    int len = sizeof(addr);
    SSL *ssl;
    int client = accept(server, &addr, &len);
    printf("Connection: %s:%d\n",
        inet_ntoa(addr.sin_addr), ntohs(addr.sin_port));
    ssl = SSL_new(ctx);
    SSL_set_fd(ssl, client);
    Servlet(ssl);
}
close(server);
SSL_CTX_free(ctx);
}

```

OpenSSL Client Code 입니다.

```

#include <stdio.h>
#include <unistd.h>
#include <malloc.h>
#include <string.h>
#include <sys/socket.h>
#include <resolv.h>
#include <netdb.h>
#include <openssl/ssl.h>
#include <openssl/err.h>

#define FAIL    -1

int OpenConnection(const char *hostname, int port){    int sd;
    struct hostent *host;
    struct sockaddr_in addr;

    if ( (host = gethostbyname(hostname)) == NULL ){
        perror(hostname);
        abort();
    }
    sd = socket(PF_INET, SOCK_STREAM, 0);
    bzero(&addr, sizeof(addr));
    addr.sin_family = AF_INET;

```

```

    addr.sin_port = htons(port);
    addr.sin_addr.s_addr = *(long*)(host->h_addr);
    if ( connect(sd, &addr, sizeof(addr)) != 0 ){
        close(sd);
        perror(hostname);
        abort();
    }
    return sd;
}

SSL_CTX* InitCTX(void){
    SSL_METHOD *method;
    SSL_CTX *ctx;

    OpenSSL_add_all_algorithms();
    SSL_load_error_strings();
    method = SSLv2_client_method();
    ctx = SSL_CTX_new(method);
    if ( ctx == NULL ){
        ERR_print_errors_fp(stderr);
        abort();
    }
    return ctx;
}

void ShowCerts(SSL* ssl){
    X509 *cert;
    char *line;

    cert = SSL_get_peer_certificate(ssl);
    if ( cert != NULL ){
        printf("Server certificates:\n");
        line = X509_NAME_oneline(X509_get_subject_name(cert), 0, 0);
        printf("Subject: %s\n", line);
        free(line);
        line = X509_NAME_oneline(X509_get_issuer_name(cert), 0, 0);
        printf("Issuer: %s\n", line);
        free(line);
        X509_free(cert);
    }
    else

```

```

        printf("No certificates.\n");
    }

int main(int count, char *strings[]){
    SSL_CTX *ctx;
    int server;
    SSL *ssl;
    char buf[1024];
    int bytes;
    char *hostname, *portnum;

    if ( count != 3 ){
        printf("usage: %s <hostname> <portnum>\n", strings[0]);
        exit(0);
    }
    hostname=strings[1];
    portnum=strings[2];

    ctx = InitCTX();
    server = OpenConnection(hostname, atoi(portnum));
    ssl = SSL_new(ctx);
    SSL_set_fd(ssl, server);
    if ( SSL_connect(ssl) == FAIL )
        ERR_print_errors_fp(stderr);
    else {
        char *msg = "Hello???";

        printf("Connected with %s encryption\n", SSL_get_cipher(ssl));
        ShowCerts(ssl);
        SSL_write(ssl, msg, strlen(msg));
        bytes = SSL_read(ssl, buf, sizeof(buf));
        buf[bytes] = 0;
        printf("Received: \"%s\"\n", buf);
        SSL_free(ssl);
    }
    close(server);
    SSL_CTX_free(ctx);
}

```