

# 네트워크보안 Term Project

## 최종보고서

프로젝트제목 :비밀다이어리

분반 : 101분반

제출일자 : 2019년 12월 23일

작성자 : 201211834 정소비

201511188 배연화

201611983 강륜화

# 목차

<b>I. 프로젝트 개요</b>	<b>3</b>
1. 주제 선정 이유	3
2. 주제 소개	3
3. 개발 환경	3
4. 스케줄	3
5. 팀원간 업무 분장	4
<b>II. 프로젝트 결과물</b>	<b>5</b>
1. 프로그램 동작,구성	5
2. 프로그램 실행 화면	18
<b>III. 결론 및 소감</b>	<b>38</b>
1. 결론 및 소감	38
2. 별도 제출 자료	39

# I. 프로젝트 개요

## 1. 주제선정이유

한 학기 동안 네트워크 보안 수업을 들으면서 여러 가지 다양한 보안 서비스에 대해 학습하였습니다. 클라이언트와 서버 환경을 구축하여 메시지를 주고 받을 수 있으면서, 보안 서비스를 잘 적용할 수 있는 응용 어플리케이션을 구상하다 '비밀 다이어리'라는 아이디어를 떠올렸습니다. 다이어리는 비밀로 작성되며 허가된 사람만 공유하여 열람할 수 있으므로 공유키, 비밀키, handshake 등 네트워크 보안 수업에서 배웠던 보안 서비스들을 적절히 적용할 수 있을 것이라 생각하였습니다.

## 2. 주제 소개

'비밀 다이어리' 프로그램으로써 크게 회원가입, 로그인, 개인다이어리 작성과 읽기, 공유다이어리 작성과 읽기 기능을 가지고 있습니다. 회원가입 시 가입자의 정보가 서버에 저장되고 해당 정보로 로그인을 할 수 있습니다. 개인 다이어리는 자신만 읽고 쓸 수 있는 다이어리이고, 공유 다이어리는 공유할 상대를 지정하고 해당 사용자와 공유하는 다이어리를 읽고 쓸 수 있습니다.

## 3. 개발 환경

- 개발환경 : VISUAL STUDIO 2017
- 운영체제 : WINDOWS 10
- 사용언어 :C++
- 보안 라이브러리 : OpenSSL

## 4. 스케줄

11/11 ~ 11/18 : OpenSSL 개발 환경 갖추기(라이브러리 설치 등), OpenSSL 예제 공부  
11/19 ~ 11/24 : 코드의 전체적인 틀 제작, 클라이언트 - 서버 상호 메시지 교환 환경 구축  
11/25 ~ 11/30 : 작성된 코드에 OpenSSL을 이용한 보안 서비스 적용  
12/1 ~ 12/7 : 작성된 코드 확인, 피드백 과정

12/8 ~ 12/14 : 최종 보고서 작성, 발표자료 PPT 제작

12/15 ~ 12/23 : 최종 검사, 제출

## 5. 팀원간 업무분장

### - 정소비

- 프로젝트 설계
- 서버/클라이언트 모델 구현
- 암호화 모듈 구현
- 기능 구현(회원가입,개인/공유 다이어리 작성)

### - 배연화

- 프로젝트 설계
- 전체 인터페이스, 레이아웃 제작
- PPT, 보고서 작성

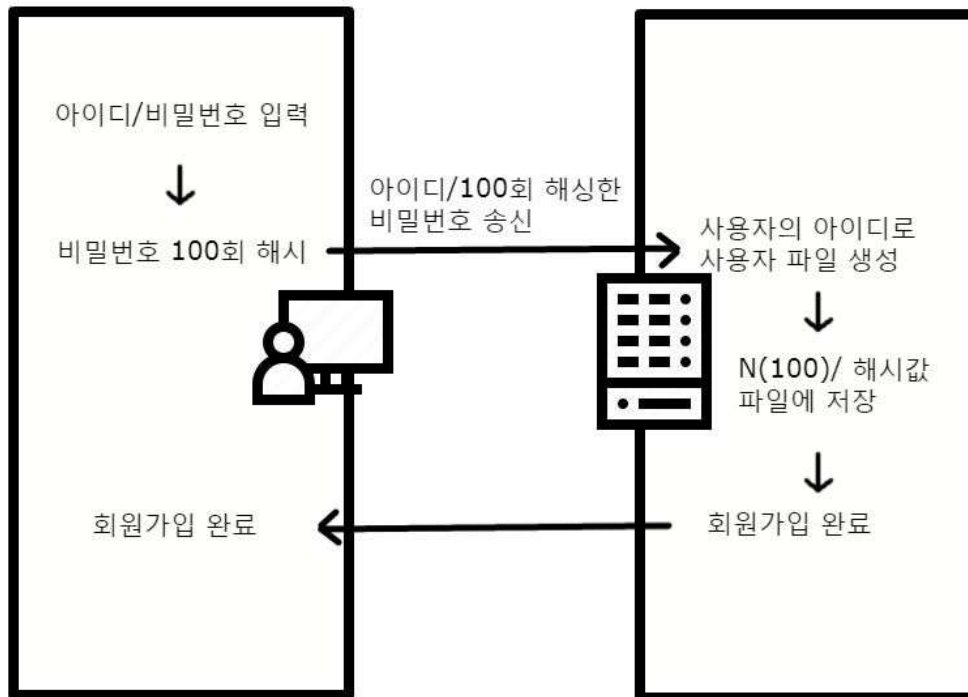
### - 강륜화

- 프로젝트 설계
- 자료수집 및 전체 기능 개요 작성
- 기능 구현(로그인, 개인/공유 다이어리 목록 읽기, 개인/공유 다이어리 읽기)

## II. 프로젝트 결과물

### 1. 프로그램 동작,구성

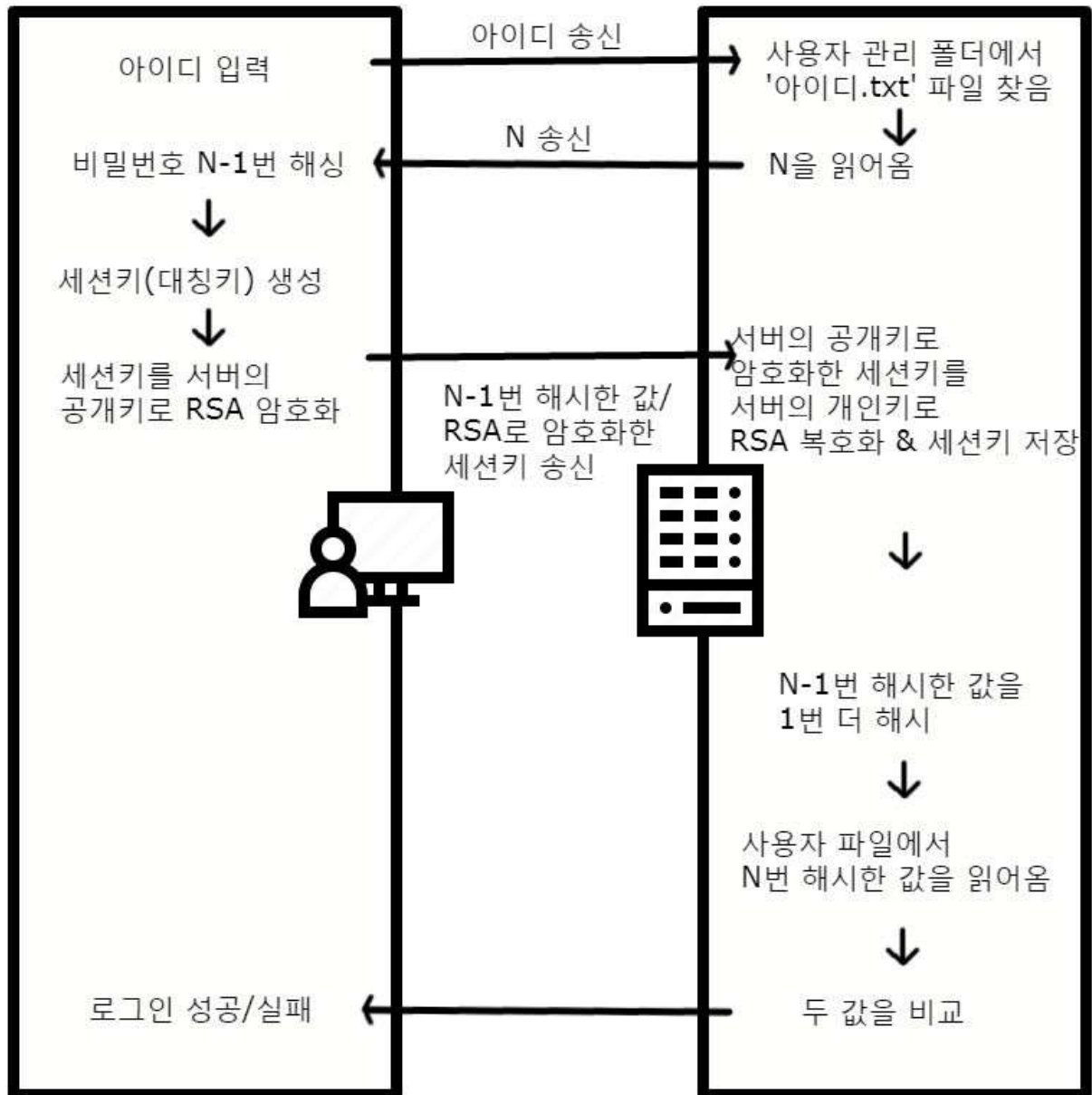
#### (1) 회원가입



```
/*
해시함수
*parameter: 평문메시지, 해시된메시지를넣을변수
*return: 없음
*/
voidhasing_(char* string, char* mdString)

/*
해시함수 n번반복
*parameter: 평문메시지, 해시된메시지, 해시함수반복횟수
*return: 없음
*/
voidhasing(char* string, char* mdString, intn)
```

## (2) 로그인



/\* <로그인>

1. 파일에서사용자의 N번해시한값을받아오고, 사용자가 n-1번해시한값을받아 1번해시하여그값과비교

1) 같으면로그인성공 -> n-1, n-1번해시한값을이전의사용자파일을삭제 하고다시만들어저장

2) 다르면로그인실패 ->저장이이루어지지않음

2. 대칭키 return

\*/

char\* login(AuthServer\* server, char\* user ID)

/\*

RSA 키 파일 불러오기

\*parameter: 키 타입( 0: 개인키, 1: 공개키 ), 파일 이름

\*return: RSA 구조체

\*/

RSA\* load\_RSA(int pem\_type, char\* file\_name)

```

/*
서버공개키로 암호화
*parameter: 평문메시지, 암호화된메시지가저장될문자열변수
*return: 암호화된메시지길이
*/
int publicEncrypt(char* encrypt, char* result)

int public_encrypt(unsignedchar* data, int data_len, RSA* key, unsignedchar* encrypted)

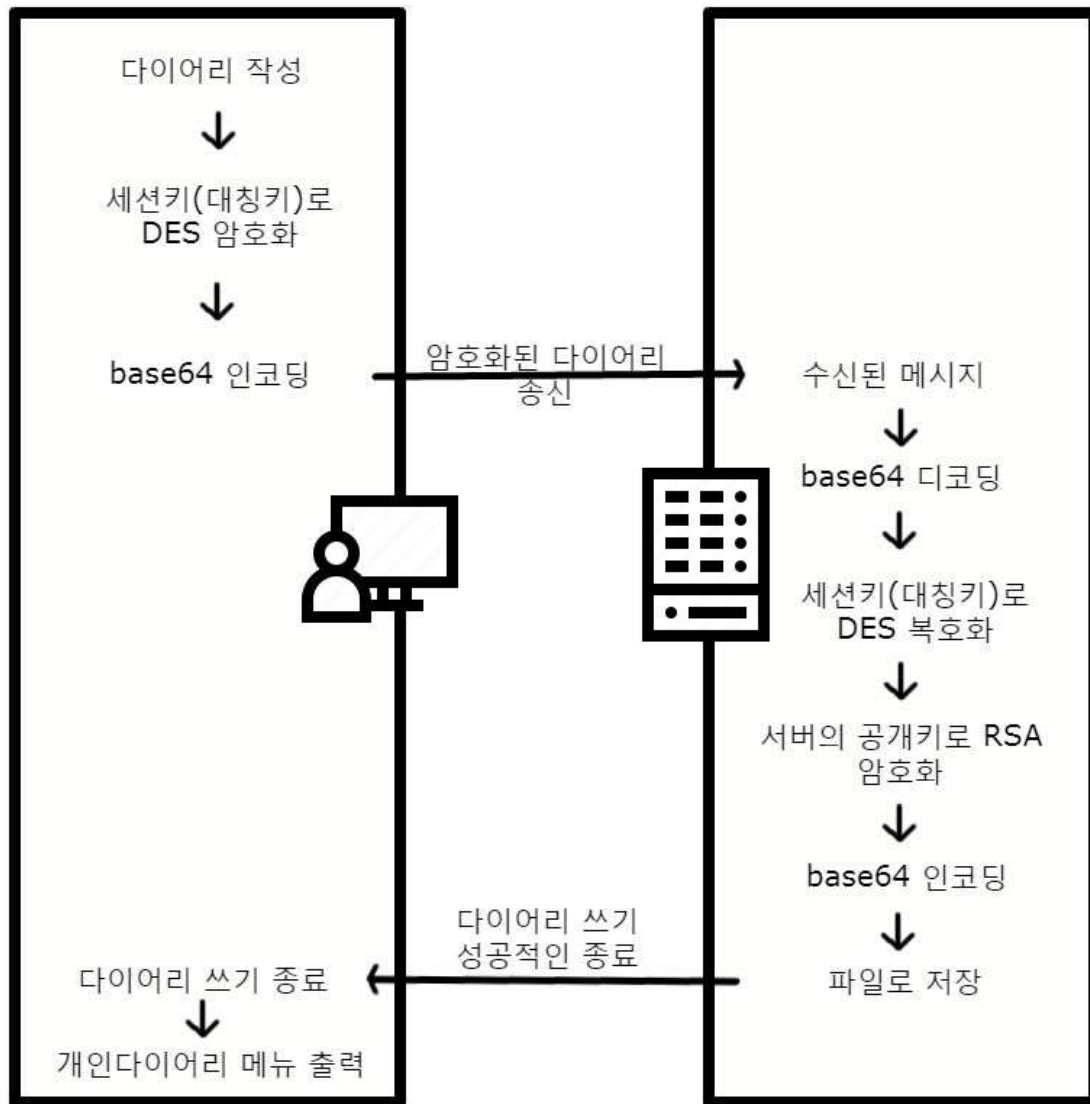

/*
서버비밀키로 복호화
*parameter: 암호화된메시지, 원본메시지의길이, 비밀키, 복호화된메시지가저장될변수
*return: 복호화된메시지의길이
*/
int private_decrypt(unsignedchar* enc_data, int data_len, RSA* key, unsignedchar* decrypted)


/*
해시 함수
*parameter: 평문메시지, 해시된메시지를넣을변수
*return: 없음
*/
void hashing_(char* string, char* mdString)


/*
해시 함수 n번 반복
*parameter: 평문메시지, 해시된메시지, 해시 함수 반복 횟수
*return: 없음
*/
void hashing(char* string, char* mdString, int n)

```

### (3) 개인 다이어리 작성



```

/*
*다이어리작성후, 암호화와 base64 인코딩한후, 서버로전송.
*parameter: 클라이언트객체, 개인/공유다이어리구분값
*return: 없음
*/
void write(AuthClient&client, intn)

/*
아스키문자열을유니코드문자열로변환
*parameter: 아스키문자열
*return: 유니코드문자열
*/
wchar_t* ConverCtoWC(char* str)

/*작성한다이어리내용을서버와의대칭키로암호화한후, Base64인코딩.
*parameter: 클라이언트객체, 암호화할내용, 암호문의길이, 서버와의대칭키
*return: base64 인코딩된문자열
*/
char* msg_encrypt(AuthClient&client, constchar* contents, int&len, char* key)
  
```



```

/*DES 암호화함수
*parameter: 암호화된메시지가저장될변수, 평문메시지, 평문메시지길이, 대칭키값
*return: 암호화된메시지길이
*/
intAuthClient::encrypt_block(unsignedchar* cipherText, unsignedchar* plainText, unsignedintplainTextLen,
unsignedchar* key)

/*
평문메시지를 BASE64 인코딩된메시지로인코딩
*parameter: 평문메시지, 인코딩결과물의길이, 결과물의길이가삽입될변수
*return: BASE64 인코딩된메시지
*/
char* base64(constunsignedchar* input, intlength, int&result_len)

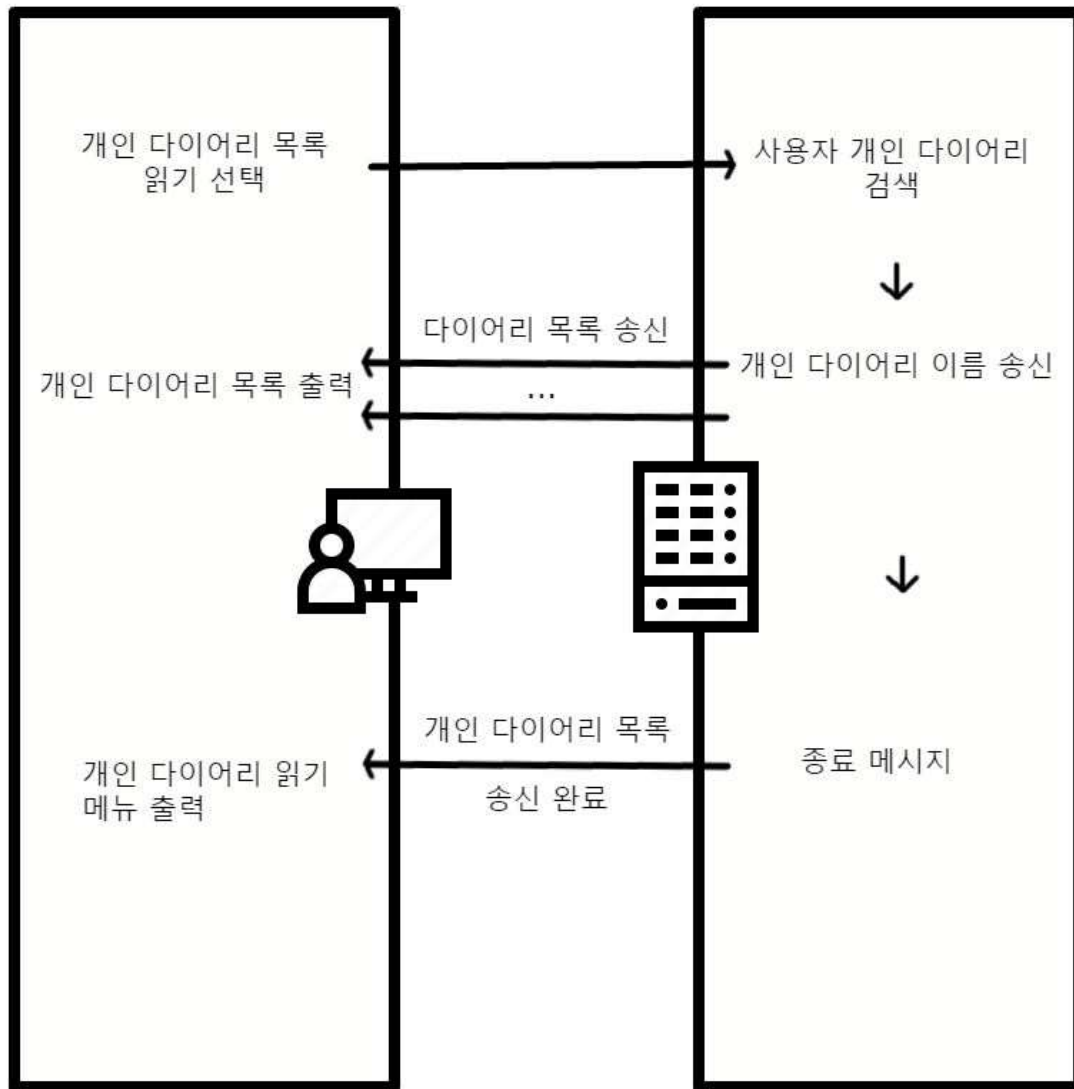
/*
BASE64 인코딩된메시지를본문으로디코딩
*parameter: BASE64 인코딩된문자열, BASE64인코딩된문자열의길이
*return: 디코딩된본문메시지
*/
char* decode64(unsignedchar* input, intlength)

/*클라이언트로부터받은 base64 인코딩된다이러리내용을디코딩한후, 클라이언트와의대칭키로복호화.
*parameter: 서버객체, base64된내용
*return: 복호화된문자열
*/
char* msg_decrypt(AuthServer&server, constchar* contents, intlen, char* key)

/*DES 복호화함수
*parameter: 복호화된메시지가저장될변수, 암호화된메시지본문, 암호화된메시지본문길이, 대칭키값
*return: 복호화된메시지길이
*/
intAuthServer::decrypt_block(unsignedchar* plainText, unsignedchar* cipherText, unsignedintcipherTextLen,
unsignedchar* key)

```

#### (4) 개인 다이어리 목록 불러오기



```

/*
혼자쓰기한파일의제목리스트 client에게보내기
- 혼자쓰기하는폴더이름 : 'private_diary'
- 혼자쓰기하면생기는파일이름 :사용자이름_index.txt
1. 사용자ID가들어간 txt파일의갯수를 count
  1) 0개 ->파일없음 ->오류메시지송신
  2) n개 ->다시파일존재하는지검사하며파일이름보냄 ->파일다보내는것이끝났음을알림
*/
voidreadAloneFiles(AuthServer* server, char* userID)

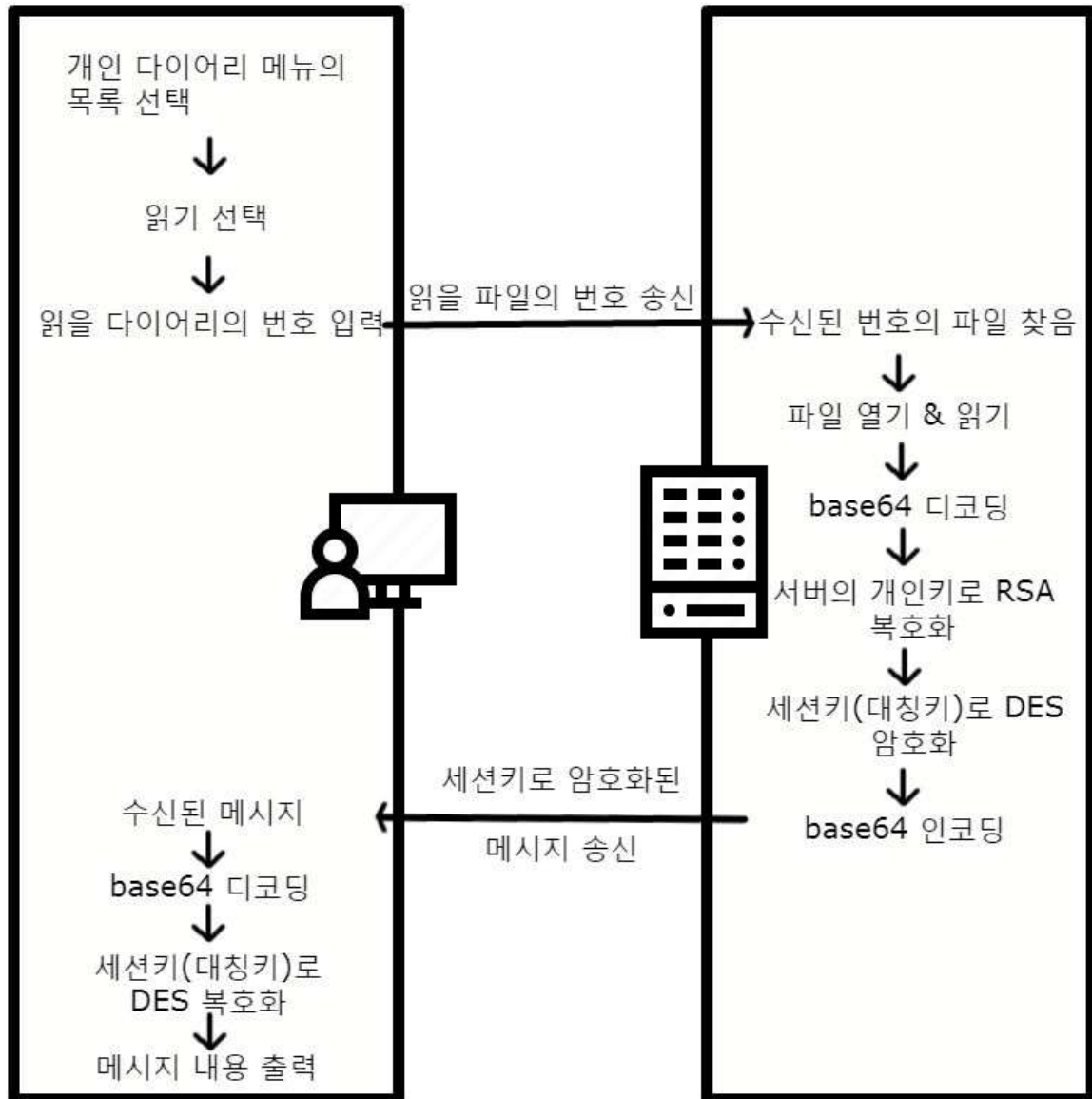
```

```

/* 파일개수 count해서보냄
사용자ID + index값으로파일이름만들어갯수를 count해서보냄
*/
intcountFiles(char* userID, constchar* folderName)

```

## (5) 개인 다이어리 읽기



```

/* 개인다이어리읽기메뉴 1. 읽기 2. 뒤로가기
1. 읽을파일번호를입력받음 ->파일번호를서버에게전송
2. 뒤로가기 ->개인다이어리메뉴로돌아감
*/

```

```

void aloneReadMenu(AuthClient&client)

```

```

// 서버의공개키로암호화된 txt파일을읽어와서디코딩+복호화 = 평문
int getFileEncrypted(char* path, char* plainText)

```

```

/*
BASE64 인코딩된메시지를본문으로디코딩
*parameter: BASE64 인코딩된문자열, BASE64인코딩된문자열의길이
*return: 디코딩된본문메시지
*/
char* decode64(unsignedchar* input, int length)

```

```

/*
서버비밀키로복호화
*parameter: 암호화된메시지, 원본메시지의길이, 비밀키, 복호화된메시지가저장될변수
*return: 복호화된메시지의길이
*/
intprivate_decrypt(unsignedchar* enc_data, intdata_len, RSA* key, unsignedchar* decrypted)

/*다이어리내용을대칭키로암호화한후, Base64인코딩.
*parameter: 서버객체, 암호화할내용, 암호문의길이, 클라이언트와의대칭키
*return: base64 인코딩된문자열
*/
char* msg_encrypt(AuthServer&server, constchar* contents, int&len, char* key)

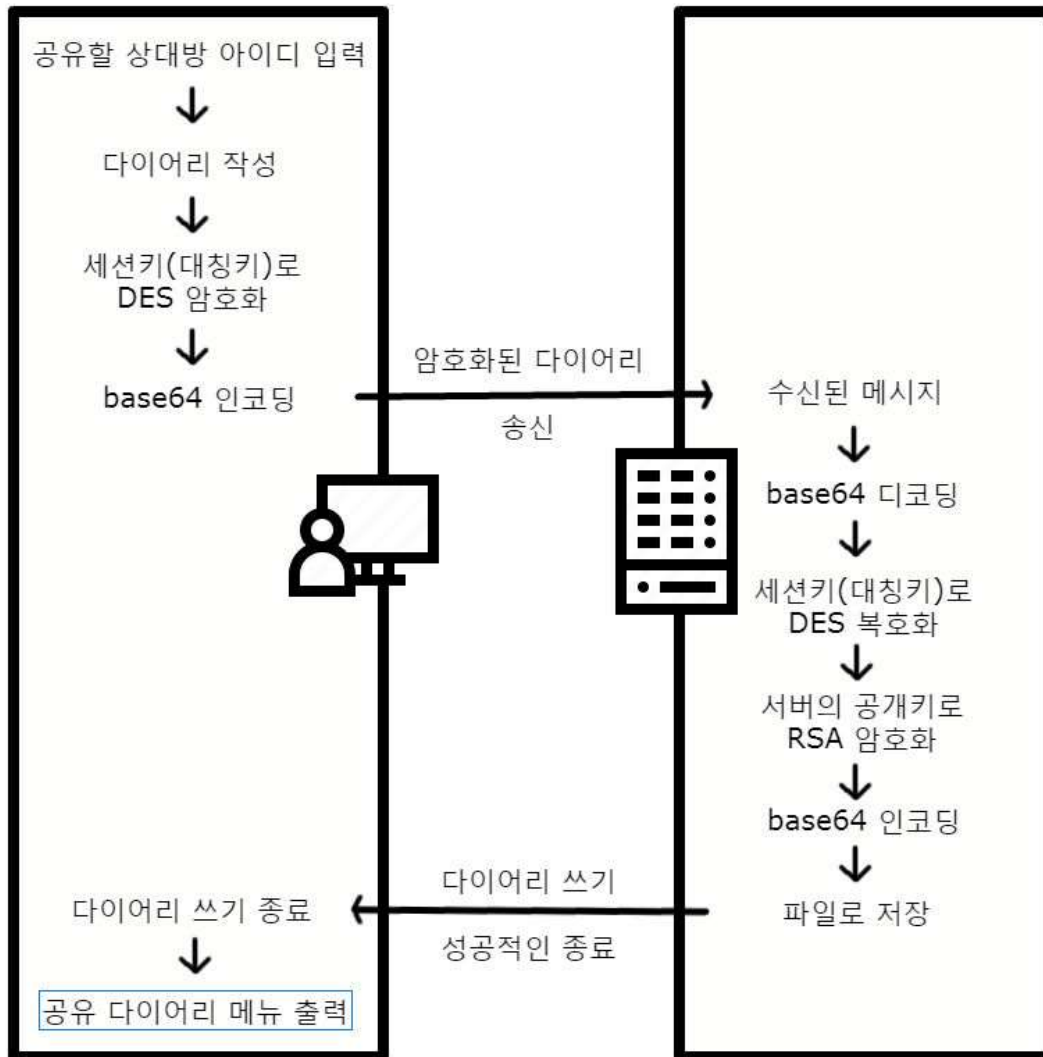
/*DES 암호화함수
*parameter: 암호화된메시지가저장될변수, 평문메시지, 평문메시지길이, 대칭키값
*return: 암호화된메시지길이
*/
intAuthServer::encrypt_block(unsignedchar* cipherText, unsignedchar* plainText, unsignedintplainTextLen,
unsignedchar* key)

/*
평문메시지를 BASE64 인코딩된메시지로인코딩
*parameter: 평문메시지, 인코딩결과물의길이, 결과물의길이가삼입될변수
*return: BASE64 인코딩된메시지
*/
char* base64(constunsignedchar* input, intlength, int&result_len)

/*서버로부터받은 base64 인코딩된다이어리내용을디코딩한후, 서버와의대칭키로복호화.
*parameter: 클라이언트객체, base64된내용
*return: 복호화된문자열
*/
char* msg_decrypt(AuthClient&client, constchar* contents, intlen, char* key)

```

## (6) 공유 다이어리 쓰기



```

/*
*다이어리작성후, 암호화와 base64 인코딩한후, 서버로전송.
*parameter: 클라이언트객체, 개인/공유다이어리구분값
*return: 없음
*/
void write(AuthClient&client, intn)

```

```

/*
아스키문자열을유니코드문자열로변환
*parameter: 아스키문자열
*return: 유니코드문자열
*/
wchar_t* ConverCtoWC(char* str)

```

```

/*작성한다이어리내용을서버와의대칭키로암호화한후, Base64인코딩.
*parameter: 클라이언트객체, 암호화할내용, 암호문의길이, 서버와의대칭키
*return: base64 인코딩된문자열
*/
char* msg_encrypt(AuthClient&client, constchar* contents, int&len, char* key)

```

```

/*DES 암호화함수
*parameter: 암호화된메시지가저장될변수, 평문메시지, 평문메시지길이, 대칭키값
*return: 암호화된메시지길이
*/
intAuthClient::encrypt_block(unsignedchar* cipherText, unsignedchar* plainText, unsignedintplainTextLen,
unsignedchar* key)

/*
평문메시지를 BASE64 인코딩된메시지로인코딩
*parameter: 평문메시지, 인코딩결과물의길이, 결과물의길이가삽입될변수
*return: BASE64 인코딩된메시지
*/
char* base64(constunsignedchar* input, intlength, int&result_len)

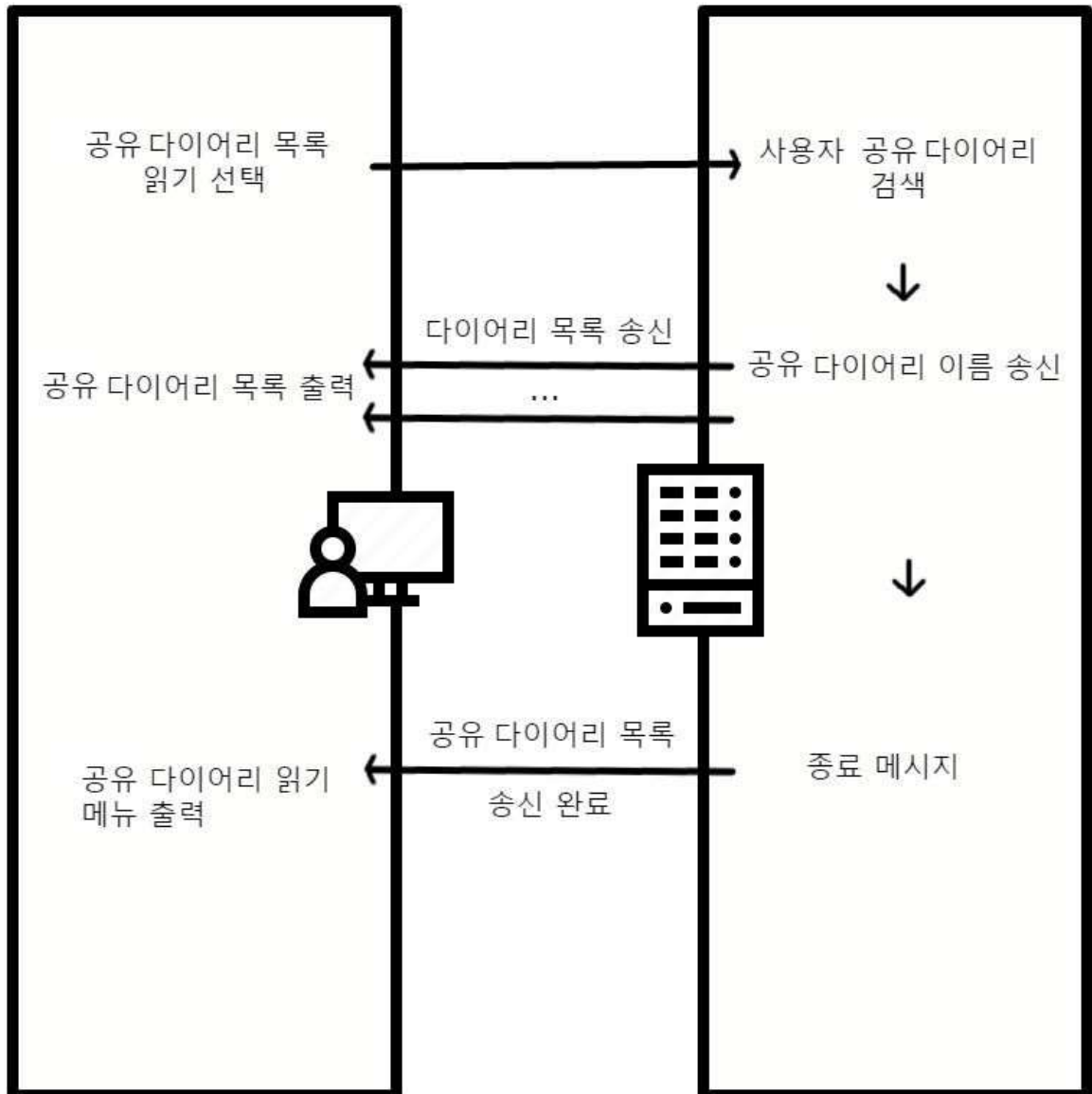
/*
BASE64 인코딩된메시지를본문으로디코딩
*parameter: BASE64 인코딩된문자열, BASE64인코딩된문자열의길이
*return: 디코딩된본문메시지
*/
char* decode64(unsignedchar* input, intlength)

/*클라이언트로부터받은 base64 인코딩된다이러리내용을디코딩한후, 클라이언트와의대칭키로복호화.
*parameter: 서버객체, base64된내용
*return: 복호화된문자열
*/
char* msg_decrypt(AuthServer&server, constchar* contents, intlen, char* key)

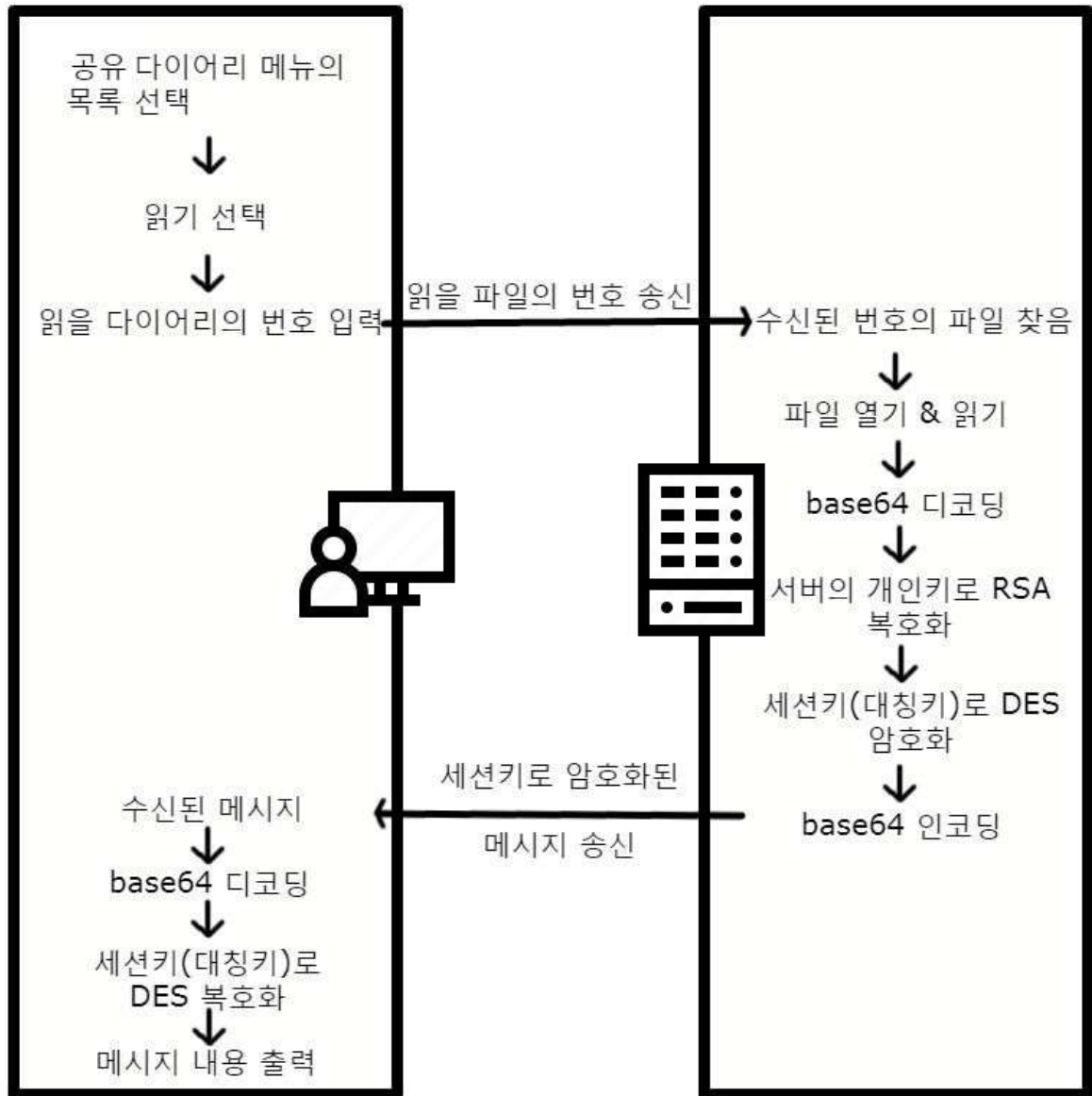
/*DES 복호화함수
*parameter: 복호화된메시지가저장될변수, 암호화된메시지본문, 암호화된메시지본문길이, 대칭키값
*return: 복호화된메시지길이
*/
intAuthServer::decrypt_block(unsignedchar* plainText, unsignedchar* cipherText, unsignedintcipherTextLen,
unsignedchar* key)

```

(7) 공유 다이어리 목록 불러오기



## (8) 공유 다이어리 읽기



```

/* 공유다이어리읽기메뉴 1. 읽기 2. 뒤로가기
1. 읽을파일번호를입력받음 ->파일번호를서버에게전송
2. 뒤로가기 ->공유다이어리메뉴로돌아감
*/

```

```

void sharedReadMenu(AuthClient&client)

```

```

// 서버의공개키로암호화된 txt파일을읽어와서디코딩+복호화 = 평문
int getFileEncrypted(char* path, char* plainText)

```

```

/*
BASE64 인코딩된메시지를본문으로디코딩
*parameter: BASE64 인코딩된문자열, BASE64인코딩된문자열의길이
*return: 디코딩된본문메시지
*/
char* decode64(unsignedchar* input, int length)

```



```

/*
서버비밀키로복호화
*parameter: 암호화된메시지, 원본메시지의길이, 비밀키, 복호화된메시지가저장될변수
*return: 복호화된메시지의길이
*/
intprivate_decrypt(unsignedchar* enc_data, intdata_len, RSA* key, unsignedchar* decrypted)

/*다이어리내용을대칭키로암호화한후, Base64인코딩.
*parameter: 서버객체, 암호화할내용, 암호문의길이, 클라이언트와의대칭키
*return: base64 인코딩된문자열
*/
char* msg_encrypt(AuthServer&server, constchar* contents, int&len, char* key)

/*DES 암호화함수
*parameter: 암호화된메시지가저장될변수, 평문메시지, 평문메시지길이, 대칭키값
*return: 암호화된메시지길이
*/
intAuthServer::encrypt_block(unsignedchar* cipherText, unsignedchar* plainText, unsignedintplainTextLen,
unsignedchar* key)

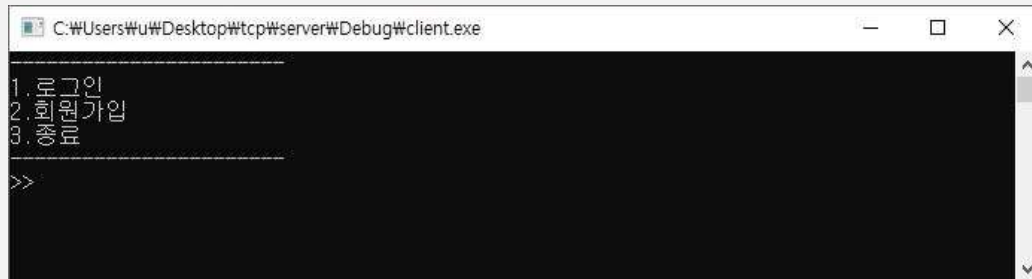
/*
평문메시지를 BASE64 인코딩된메시지로인코딩
*parameter: 평문메시지, 인코딩결과물의길이, 결과물의길이가삼입될변수
*return: BASE64 인코딩된메시지
*/
char* base64(constunsignedchar* input, intlength, int&result_len)

/*서버로부터받은 base64 인코딩된다이어리내용을디코딩한후, 서버와의대칭키로복호화.
*parameter: 클라이언트객체, base64된내용
*return: 복호화된문자열
*/
char* msg_decrypt(AuthClient&client, constchar* contents, intlen, char* key)

```

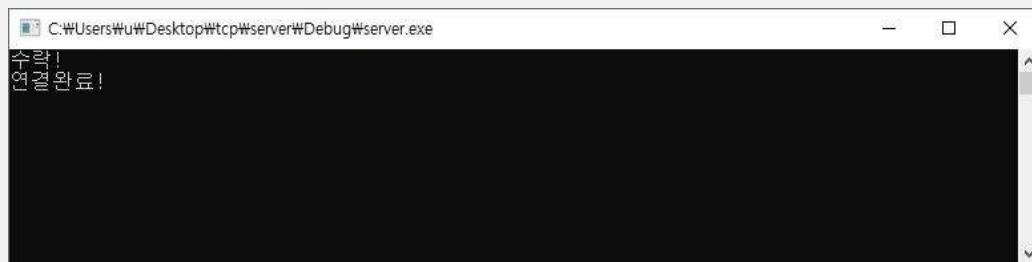
## 2. 프로그램 실행 화면

### 2-1.최초 접속 화면



#### 2-1.1.클라이언트- 최초 접속 화면

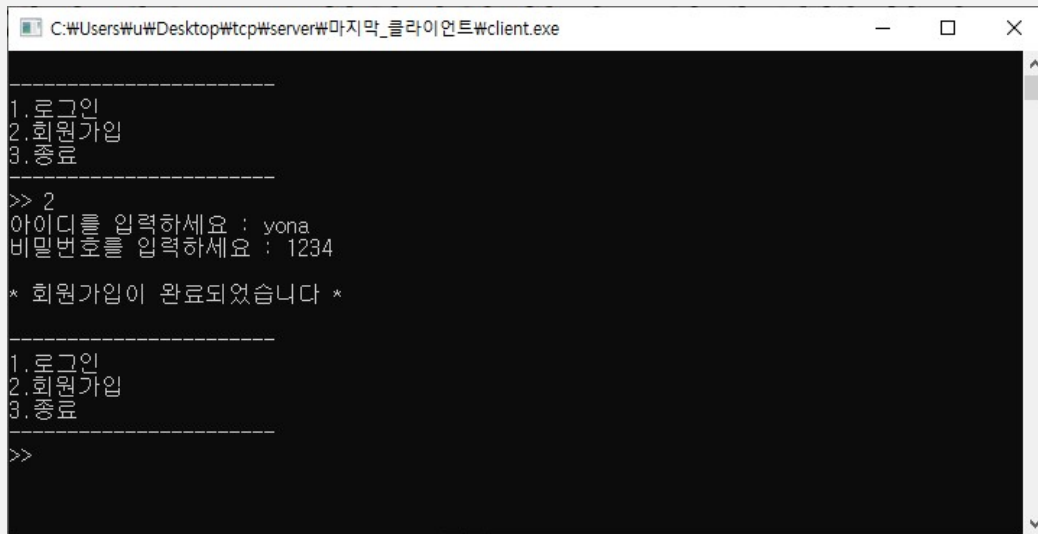
최초 접속 시 노출되는 화면이다.



#### 2-1.2. 서버 - 클라이언트 접속 요청 수락

클라이언트의 접속 요청에 대해 연결을 수락하는 화면이다.

## 2-2. 회원가입 화면



```
-----
1.로그인
2.회원가입
3.종료
-----
>> 2
아이디를 입력하세요 : yona
비밀번호를 입력하세요 : 1234

* 회원가입이 완료되었습니다 *

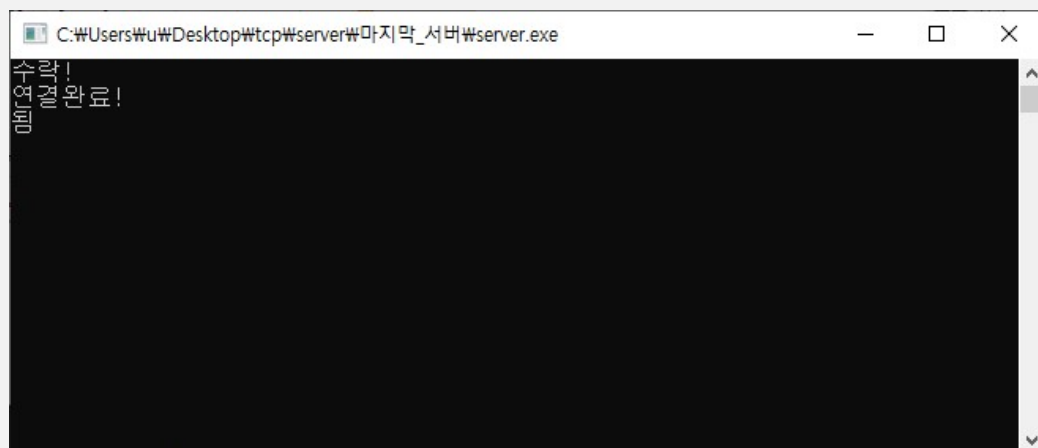
-----
1.로그인
2.회원가입
3.종료
-----
>>
```

### 2-2.1. 클라이언트 – 회원가입 완료 화면

회원가입 시에 아이디와 비밀번호를 입력

아이디와 100번 해시한 값을 서버에게 송신

➔ 서버로부터 회원가입 완료 메시지 받음



```
수락!
회원가입 완료!
```

### 2-2.2 서버 – 회원가입 완료 화면

사용자로부터 아이디, 비밀번호 100번 해시한 값을 받음

사용자 아이디로 파일 생성하여 N값 100과 받은 해시값 저장

회원가입 완료 메시지 송신

## 2-3.로그인 시 아이디 입력 화면



```
C:\Users\WU\Desktop\tcp\server\Debug\client.exe
1.로그인
2.회원가입
3.종료
>> 1
아이디를 입력하세요 :
```

2-3.1. 클라이언트 – 아이디 입력 화면

로그인 시 아이디를 입력하는 화면



```
C:\Users\WU\Desktop\tcp\server\Debug\server.exe
수락!
연결 완료!
받은 id : sobi
유저 찾음
유저 파일의 n값 : 100
```

2-3.2 서버 – 아이디 입력 받은 후의 화면

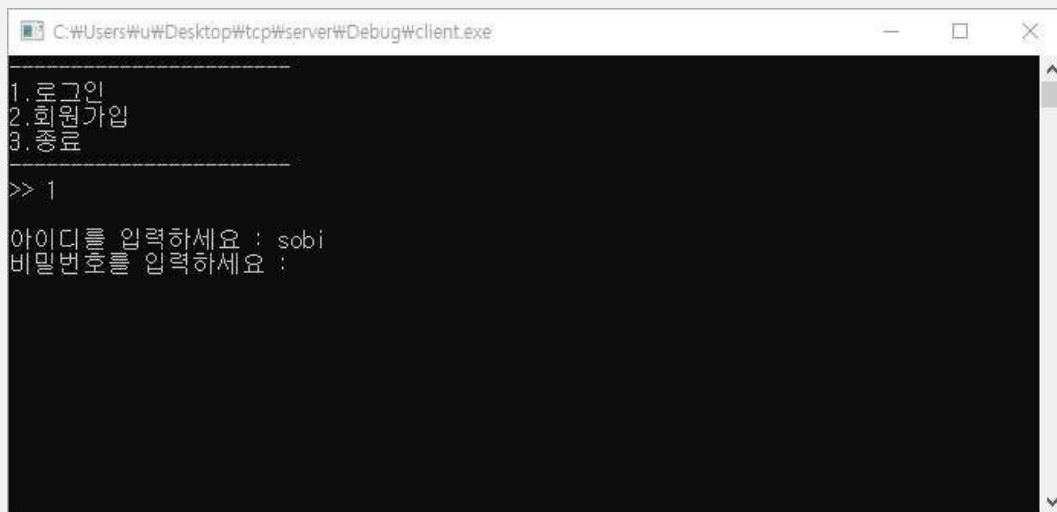
client로부터 로그인 할 ID를 받음

유저 파일은 '사용자이름.txt'로 저장되어 있음 -> 파일 이름으로 검색하여 회원 유무 판단

1) 회원 O -> 사용자의 파일을 읽어 N을 client에게 송신

2) 회원 X -> 사용자가 아님을 client에게 송신

## 2-4.로그인 시 비밀번호 입력 화면



### 2-4.1. 클라이언트 - 이전 회원가입 성공한 사용자일 경우

서버로부터 N을 받아옴

1)세션키 생성 -> 서버의 공개키로 암호화

2) n-1번 해싱

3) 암호화 길이, n-1번 해시한 값, 암호문(세션키) 보냄

```
C:\Users\#u\Desktop\#tcp#server#Debug#server.exe
수락!
연결 완료!
id : sobi
유저 찾을
유저 파일의 n값 : 100

<유저가 보낸 부분>
n-1번 해시값 : 67411d146fa818effe0e9a97312916d81bb8e6fa
클라이언트가 보낸 n-1번 해시값에 한번 더 해시한 값 : a1d6286038e3f15441cc7e553dad877951f3e534

<사용자 파일 변경>
n : 100
해시값 : a1d6286038e3f15441cc7e553dad877951f3e534
↓
n : 99
해시값 : a1d6286038e3f15441cc7e553dad877951f3e534
```

## 2-4.2. 서버 – 비밀번호 입력 화면

1.파일에서 사용자의 N번 해시한 값을 받아오고,

사용자가 n-1번 해시한 값을 받아 1번 해시하여N번 해시한 값과 비교

1) 같으면 로그인 성공

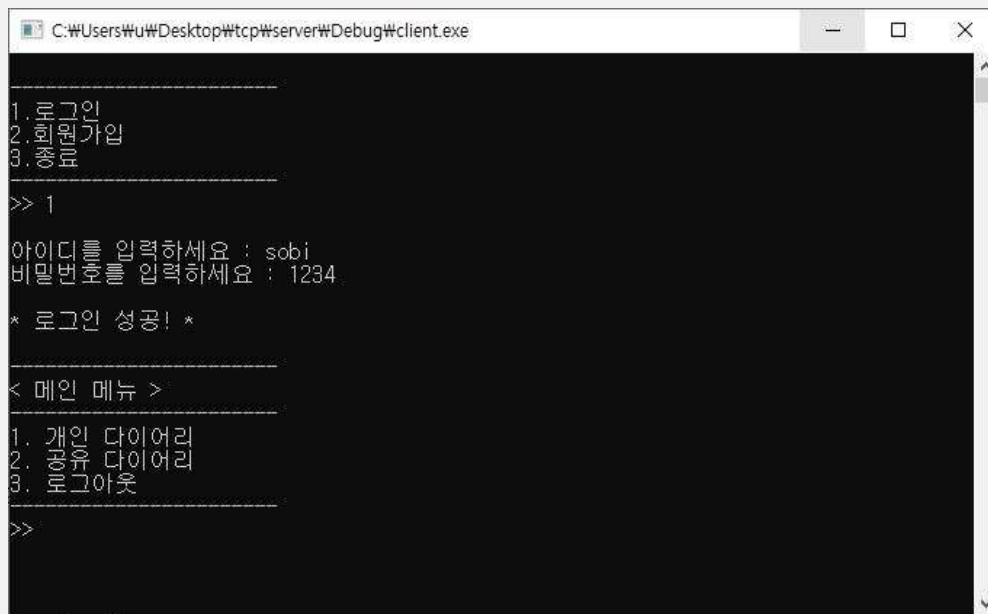
-> n-1, n-1번 해시한 값을 이전의 사용자 파일을 삭제하고 다시 만들어 저장

-> 로그인 성공 메시지 송신

2) 다르면 로그인 실패 -> 저장이 이루어지지 않음

-> 로그인 실패 메시지 송신

## 2-5.로그인 성공 후 메인메뉴



```
C:\Users\#u\Desktop#wtcp#server#Debug#client.exe
-----
1. 로그인
2. 회원가입
3. 종료
-----
>> 1

아이디를 입력하세요 : sobi
비밀번호를 입력하세요 : 1234

* 로그인 성공! *

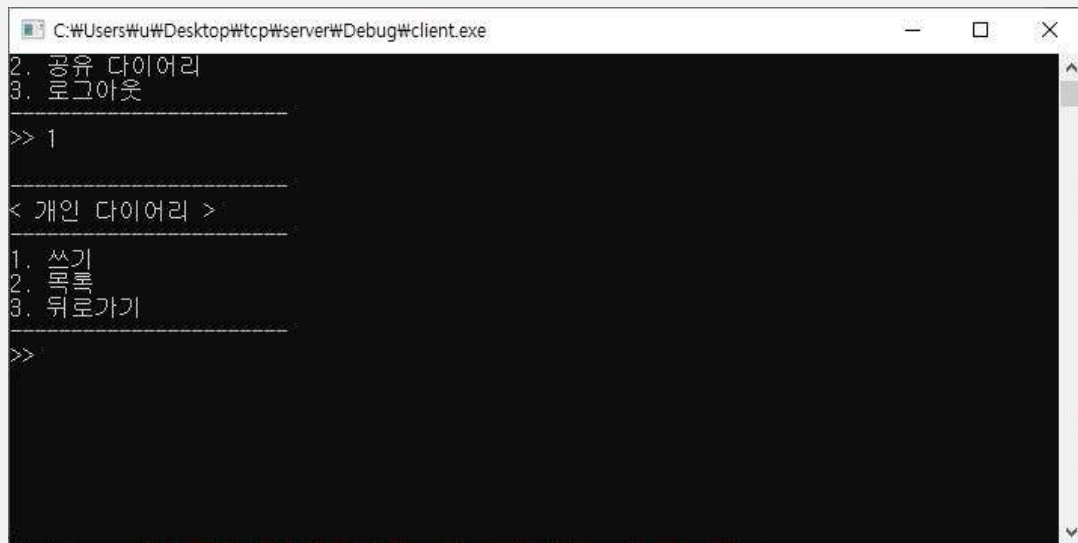
-----
< 메인 메뉴 >
-----
1. 개인 다이어리
2. 공유 다이어리
3. 로그아웃
-----
>>
```

### 2-5.1. 클라이언트 - 메인 메뉴 화면

로그인 성공 시 메인 메뉴 화면이다.

메뉴 번호 입력 시 해당 기능 실행

## 2-6. 개인 다이어리 메뉴

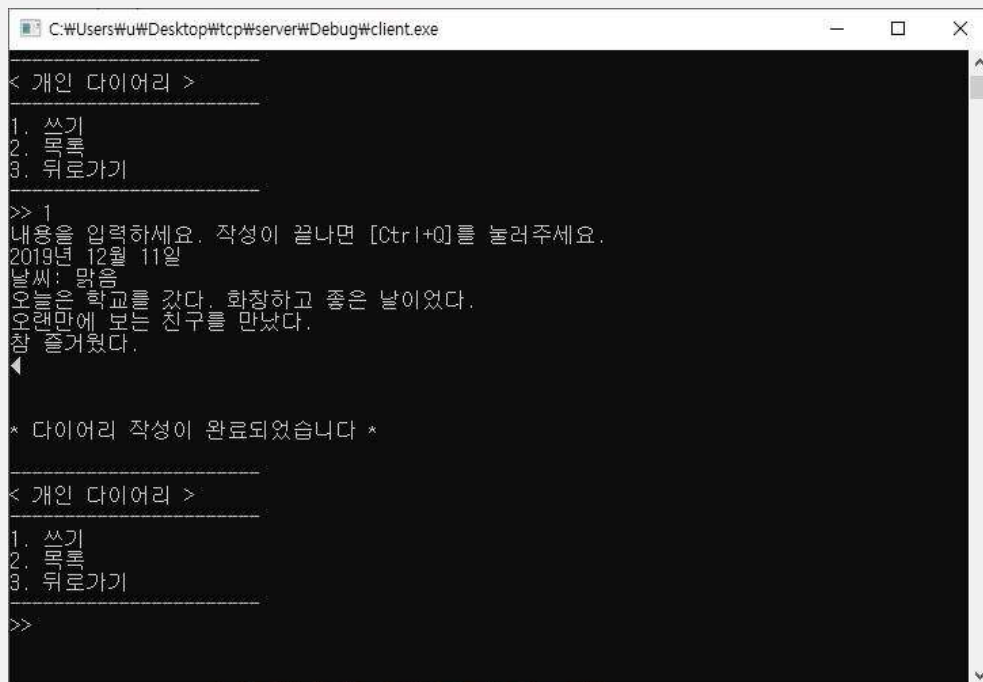


2-6.1. 클라이언트 – 개인 다이어리 메뉴 화면

개인 다이어리 선택 시 노출되는 화면이다.



## 2-7. 개인 다이어리 작성



### 2-7.1. 클라이언트 – 개인 다이어리 작성 화면

개인 다이어리 작성 후 [Ctrl+q]를 누른다.

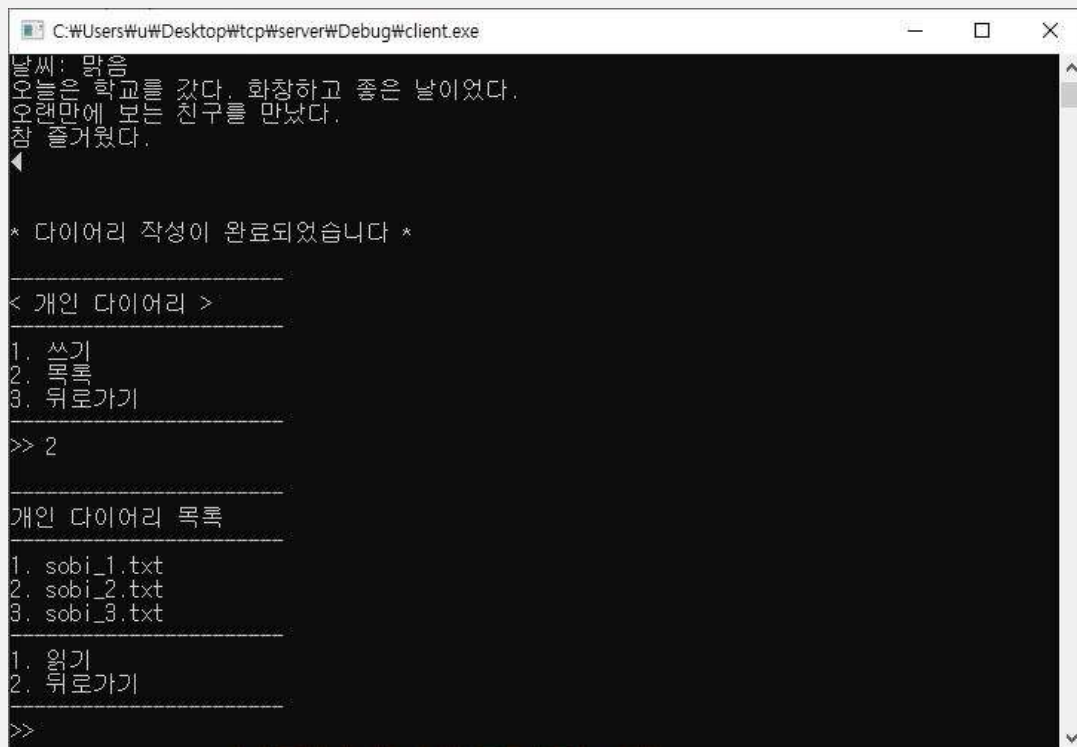
서버와 공유하는 대칭키로DES암호화 되고

BASE64 인코딩 되어

서버로 전송된다.



## 2-8.개인 다이어리 목록



### 2-8.1. 클라이언트 – 개인 다이어리 목록 선택

파일 이름을 받는 중 -> index와 파일 이름 잘라서 출력

목록 출력 끝 -> 혼자 읽기 -> 읽기 메뉴 출력

```
C:\Users#u\Desktop#tcp#server#Debug#server.exe
수락!
연결 완료!
id : sobi
유저 찾을
유저 파일의 n값 : 93

<유저가 보낸 부분>
n-1번 해시값 : 1c60029ccd8429c89a0d36832bcd9ed0e24be1a5
클라이언트가 보낸 n-1번 해시값에 한번 더 해시한 값 : 0cde39e4a0158cdc21f291102789dd746e1caff8

<사용자 파일 변경>
n : 93
해시값 : 0cde39e4a0158cdc21f291102789dd746e1caff8
↓
n : 92
해시값 : 0cde39e4a0158cdc21f291102789dd746e1caff8
찾은 파일의 개수 : 3
```

## 2-8.2. 서버 - 개인 다이어리 목록 송신

개인 다이어리 파일의 제목 리스트 client에게 보내기

- 개인 다이어리 폴더 이름 : 'private\_diary'

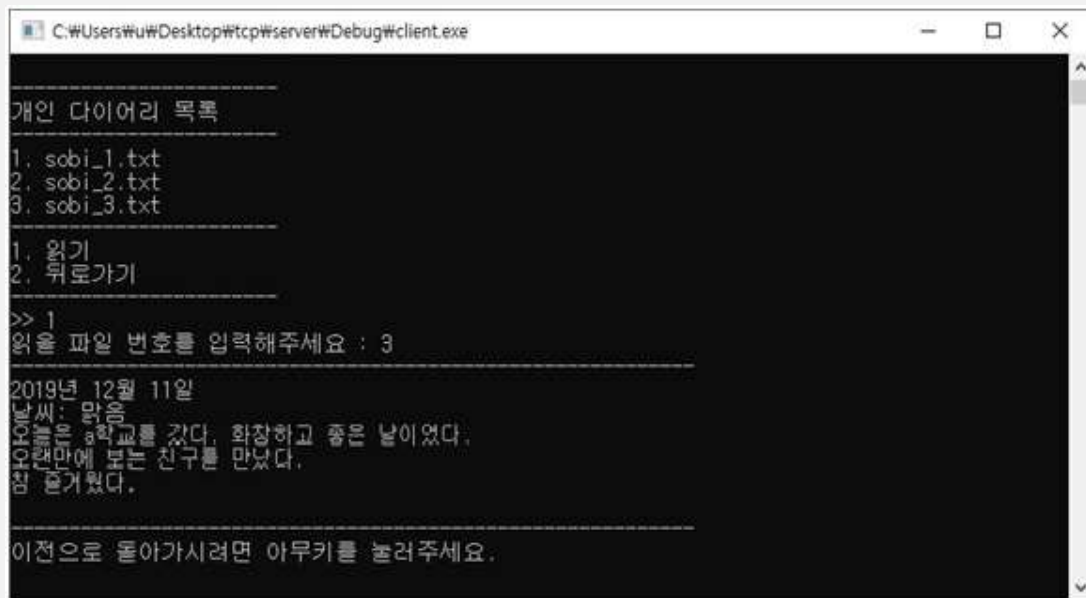
- 개인 다이어리쓰기 하면 생기는 파일 이름 : 사용자이름\_index.txt

1. 사용자ID가 들어간 txt파일의 개수를 count

- 1) 0개 -> 파일 없음 -> 오류 메시지 송신

- 2) n개 -> 다시 파일 존재하는지 검사하며 파일 이름 보냄 -> 파일 다 보내는 것이 끝났음을 알림

## 2-9.개인 다이어리 읽기



```
C:\Users\wu\Desktop\wtcp\server\Debug\client.exe

-----
개인 다이어리 목록
-----
1. sobi_1.txt
2. sobi_2.txt
3. sobi_3.txt
-----
1. 읽기
2. 뒤통가기
-----
>> 1
읽을 파일 번호를 입력해주세요 : 3

-----
2019년 12월 11일
날씨: 맑음
오늘은 학교를 갔다, 화창하고 좋은 날이었다.
오랜만에 보는 친구를 만났다.
참 즐거웠다.
-----
이전으로 돌아가시려면 아무키를 눌러주세요.
```

### 2-9.1. 클라이언트 – 개인 다이어리 출력 화면

입력한 번호에 해당하는 파일을 수신하여

BASE64디코딩 하고

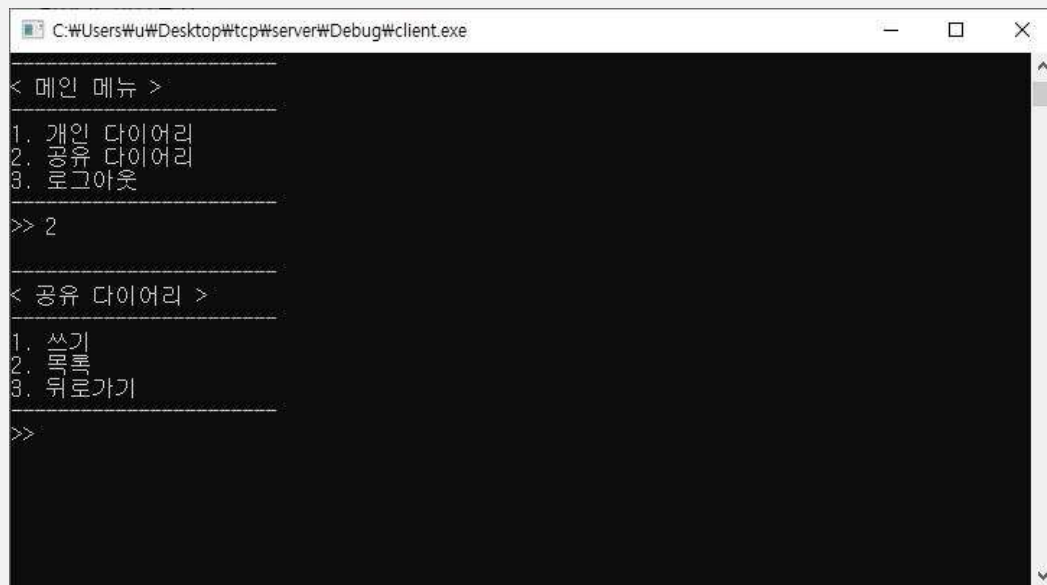
서버와 공유하는 대칭키로DES 복호화 후

내용을 출력한다.

wait ->아무 키를 입력하면 이전 개인 다이어리 메뉴로 돌아감



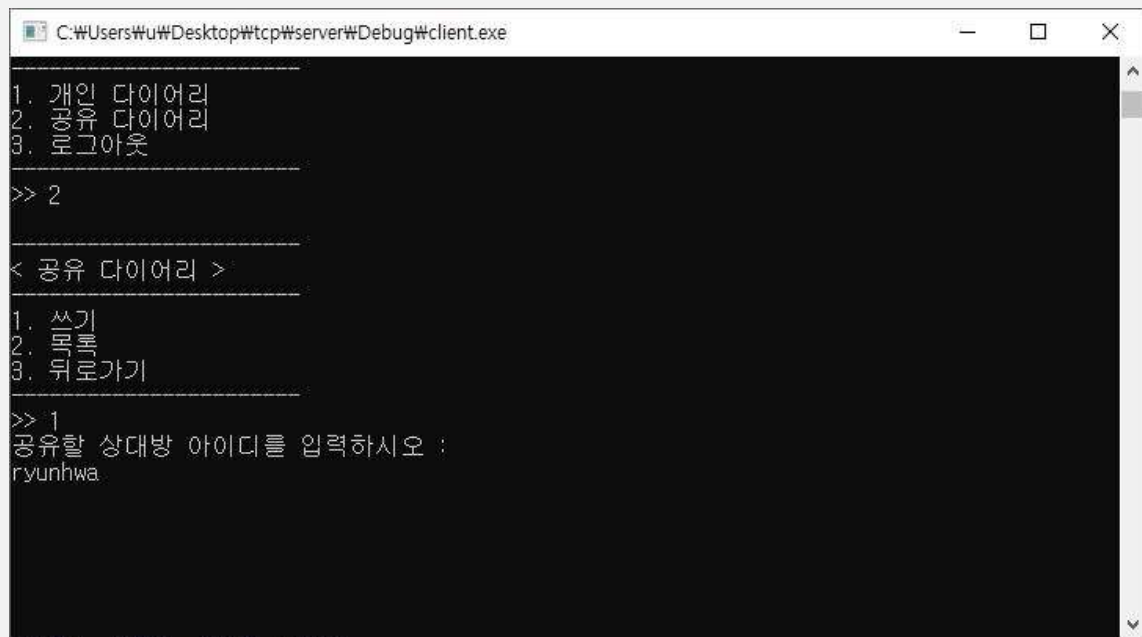
## 2-10. 공유 다이어리 메뉴



2-10.1. 클라이언트 - 공유 다이어리 메뉴 화면

공유 다이어리 선택 시 호출되는 화면이다

## 2-11. 공유 다이어리 작성 (사용자 지정)



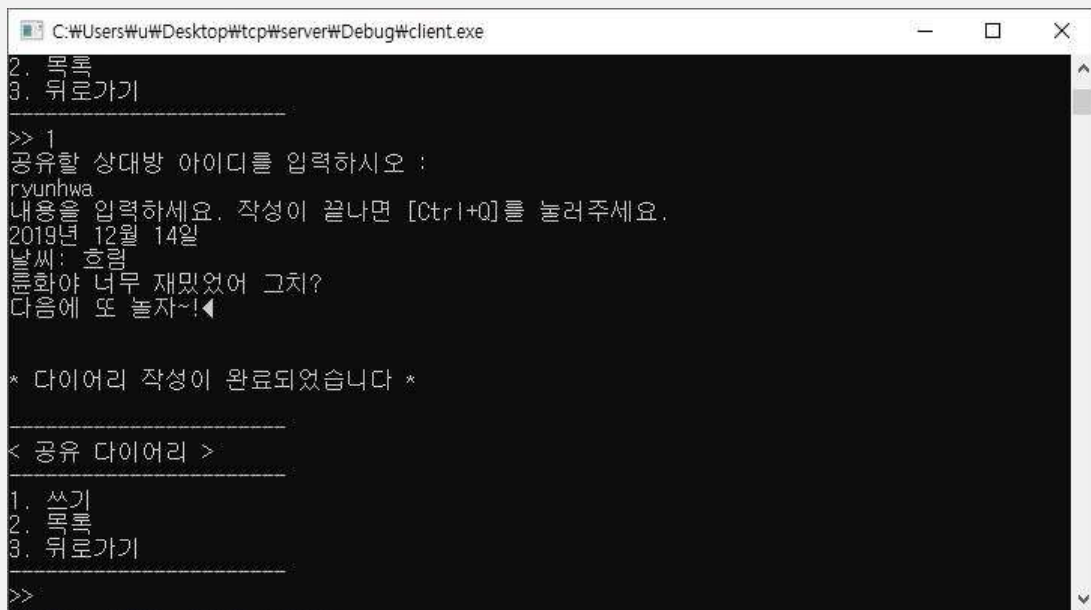
### 2.-11.1. 클라이언트 – 다이어리 공유 상대 지정

공유 다이어리 작성 시

다이어리를 공유할 사용자를 지정하는 화면이다.



## 2-11. 공유 다이어리 작성 (실제 다이어리 내용 작성)



```
C:\Users\wu\Desktop\tcp\server\Debug\client.exe
2. 목록
3. 뒤로가기
-----
>> 1
공유할 상대방 아이디를 입력하시오 :
ryunhwa
내용을 입력하세요. 작성이 끝나면 [Ctrl+Q]를 눌러주세요.
2019년 12월 14일
날씨: 흐림
평화야 너무 재밌었어 그치?
다음에 또 놀자~!

* 다이어리 작성이 완료되었습니다 *

< 공유 다이어리 >
-----
1. 쓰기
2. 목록
3. 뒤로가기
>>
```

### 2-11.2. 클라이언트 - 공유 다이어리 작성 화면

공유 다이어리 작성 후 [Ctrl+q]를 누른다.

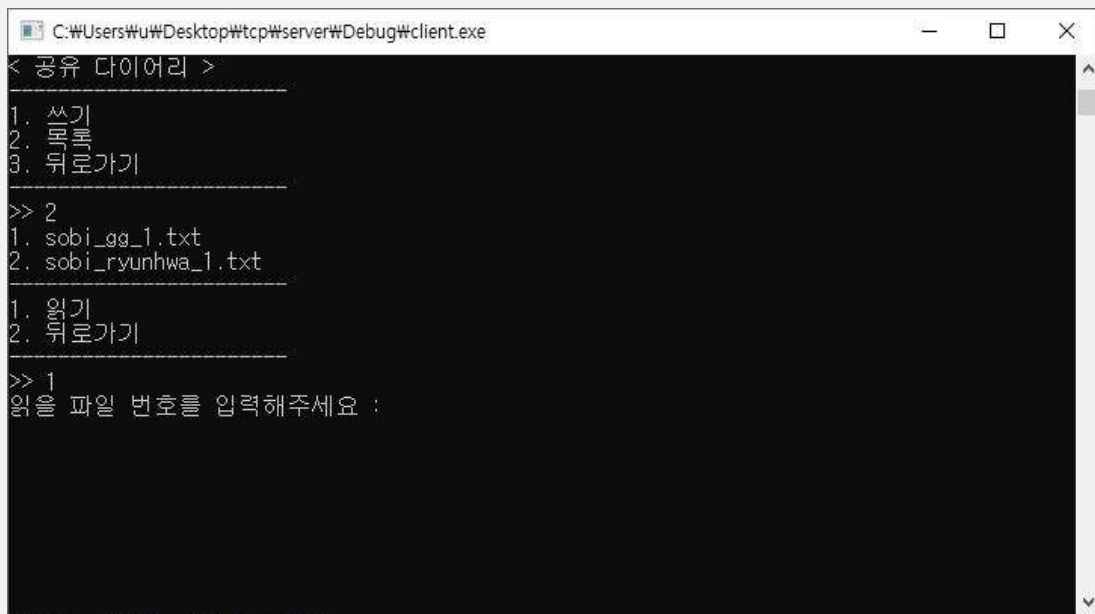
서버와 공유하는 대칭키로DES암호화 되고

BASE64 인코딩 되어

서버로 전송된다.



## 2-12. 공유 다이어리 목록 출력



```
C:\Users\Wu\Desktop\tcp\server\Debug\client.exe
< 공유 다이어리 >
1. 쓰기
2. 목록
3. 뒤로가기
>> 2
1. sobi_gg_1.txt
2. sobi_ryunhwa_1.txt
1. 읽기
2. 뒤로가기
>> 1
읽을 파일 번호를 입력해주세요 :
```

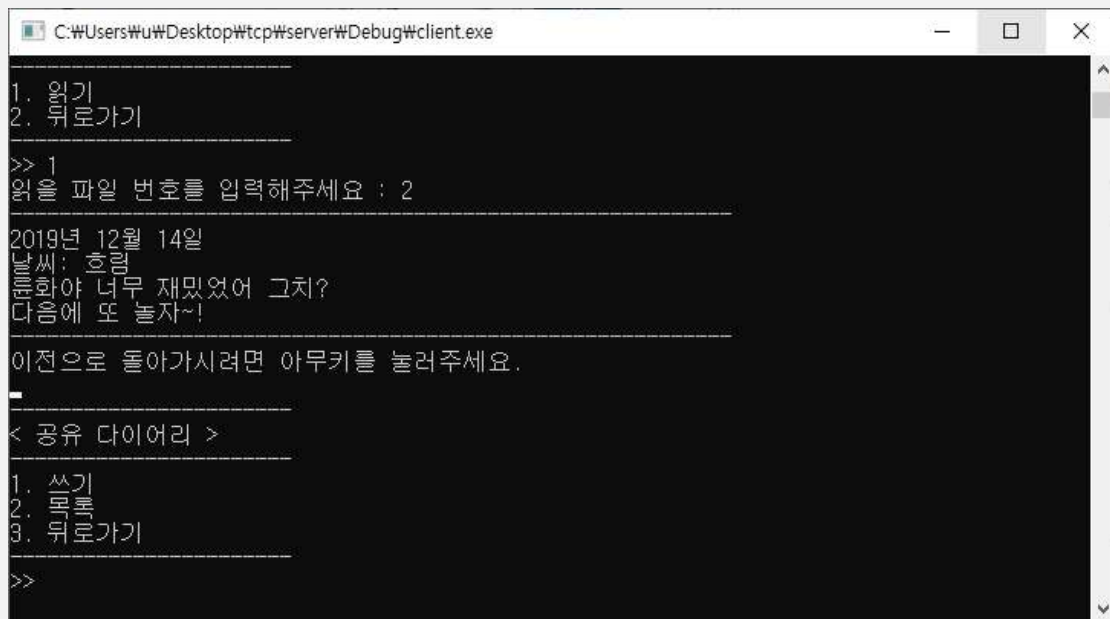
### 2-12.1. 클라이언트 - 공유 다이어리 목록 출력

공유 다이어리의 리스트 출력

출력 끝-> 공유 다이어리 읽기 메뉴로 이동



## 2-13. 공유 다이어리 읽기



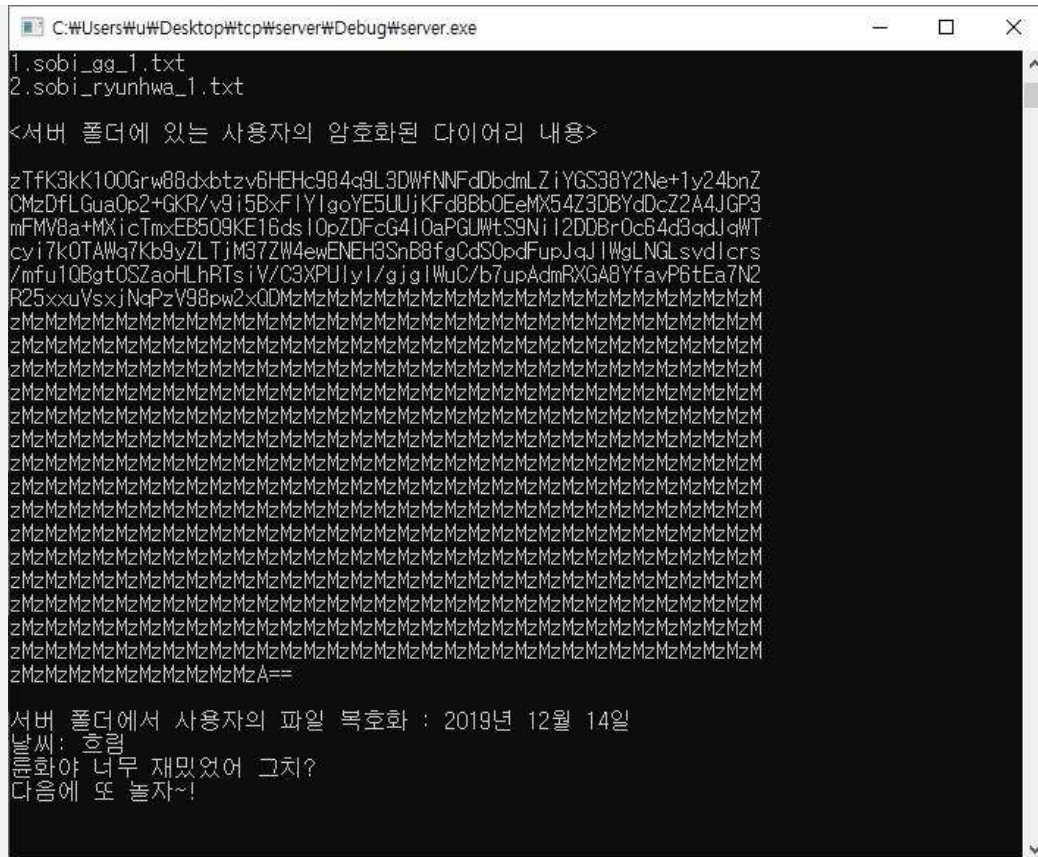
```
C:\Users\#u\Desktop\tcp\server\Debug\client.exe
-----
1. 읽기
2. 뒤로가기
-----
>> 1
읽을 파일 번호를 입력해주세요 : 2
-----
2019년 12월 14일
날씨: 흐림
문화야 너무 재밌었어 그치?
다음에 또 놀자~!
-----
이전으로 돌아가시려면 아무키를 눌러주세요.
_
-----
< 공유 다이어리 >
-----
1. 쓰기
2. 목록
3. 뒤로가기
-----
>>
```

### 2-13.1. 클라이언트 – 공유 다이어리 읽기

공유 다이어리 읽기 메뉴 1. 읽기 2. 뒤로가기

1. 읽을 파일 번호를 입력 받음 -> 파일 번호를 서버에게 전송

2. 뒤로가기 -> 공유 다이어리 메뉴로 돌아감



## 2-13.2. 서버- 공유 다이어리 읽기

사용자가 원하는 파일을 열어서 보냄

- 공유 폴더 이름 : 'shared\_diary'
- 공유 파일 이름 : '사용자1\_사용자2\_index.txt'

1. 사용자는 파일의 index를 보냄
2. 사용자 ID로 파일 검색하면서 count하여 사용자가 보낸 index와 같아지면 파일 처리

1) 파일 있음 : 서버의 공개키로 암호화된 파일을

BASE64 디코딩, 서버의 개인키로 RSA 복호화하여

다시 세션키(대칭키)로 DES암호화하여, BASE64 인코딩 후 송신

2) 파일 없음 : 오류메시지 송신

3) 파일 내용 없음 : 오류메시지 송신

## Ⅲ. 결론 및 소감

### 1. 결론 및 소감

**정소비 :** 프로젝트 설계를 시작하면서, 한 학기 간 수업에서 배웠던 보안 알고리즘을 작품에 최대한 많이 접목시키고자 하는 목적을 가지고 시작하게 되었습니다. 팀원들과의 회의를 통해 여러 가지 암호 알고리즘을 적용시켜 볼 수 있는 주제를 선정하게 되었습니다. 프로젝트 초반에는 개발이 순조롭게 진행되었지만, 서버/클라이언트 모델과 암호화기법을 개발하는 과정에서 생각지 못한 난관에 부딪히기도 하였습니다. 단일 암호화 모듈들을 개발하고 실행하는 부분에서는 문제가 없었지만, 암호화를 하는 과정에서 메시지에 특수기호가 들어가게 되어 메시지 송수신에 문제가 생겼고, 이를 해결하는 과정에서 BASE64 인코딩 기법을 적용시키면 해결할 수 있다는 것 또한 알게 되었습니다.

팀프로젝트를 진행하면서 업무 분장도 큰 과제였습니다. 개발을 어떻게 나눠서 분담해야 할까? 라는 고민도 많이 했습니다. 각 파트별로 나눠 개발을 진행하였지만, 전반적으로 자신이 맡은 파트를 서로 공유하며 다같이 진행을 했고, 결과도 만족스러웠습니다.

프로젝트를 진행하면서 한 학기 동안 배운 내용들을 실제로 작품에 적용시켜 볼 수 있어서 재미있었고, 생각보다 많은 시간이 소요되고, 난관도 많았지만, 네트워크와 보안에 대해 전체적인 구조를 알 수 있었던 계기가 되어 알찬 시간이었습니다.

**강륜화 :** 주제에 맞는 안전한 보호체계를 만들기 위해 어느 부분에 어떤 방식의 보안 방법을 적용해야할지 고민하면서 보안의 전체적인 이해도를 높일 수 있었습니다. 아직은 미숙해서 모든 부분에서 보안을 적용시킬 수는 없었지만 보람있는 팀 프로젝트였습니다. 다음에는 좀 더 OpenSSL 라이브러리를 활용하기위해 OpenSSL 라이브러리에 대해 공부를 해야할 것 같습니다.

**배연화 :** 네트워크 보안에 대해 배우는 것은 이번 수업이 처음이고 내용이 많이 낯설다 보니 팀 프로젝트를 진행한다고 했을 때 팀원들에게 해가 가지 않을까 많이 걱정했었습니다. 하지만 팀원들이 많은 도움을 주었고 덕분에 마지막까지 잘 해낼 수 있었습니다. 팀 프로젝트 주제도 수업 때 배웠던 내용을 잘 적용해볼 수 있어 보람찬 팀프로젝트가 되었습니다.

## 2. 별도 제출 자료

- (\*) 발표자료 - PPT 첨부하였습니다.
- (\*) 소스코드 - 소스코드 첨부하였습니다.