2021년 1학기 시스템및네트워크보안 (개인)프로젝트

201802085 김효린

1. 프로젝트 주제

- 신뢰 실행 환경 (TEE, Trusted Execution Environment) 기반 파일 암호화 어플리케이션 구현

- 프로그램 사용 방법 설명 :

(a) 암호화

/home/syssec/devel/optee/build에서 터미널을 열어 make run 실행후
qemu가 실행되면 c를 입력해 초기화를 한다.
그 다음 입력 가능한 터미널 창에서 buildroot login이 뜨면 root를 입력한다.
먼저 echo hello > plaintext.txt를 입력해 plaintext.txt 파일을 생성해준다.
위치는 /root에서 수행한다.
TEEencrypt -e plaintext.txt를 입력해 암호화를 수행한다.
실행이 성공적으로 되면 ciphertext.txt와 encryptedkey.txt가 생성된다.

(b) 복호화

위 과정에 이어서 TEEencrypt -d ciphertext.txt encryptedkey.txt를 입력한다.
복호화할 평문의 내용이 뜨고 다음줄에 복호화한 내용이 뜬다.
ls를 실행해보면 decryptResult.txt 파일이 생성된다.
/root 경로에 저장된다.

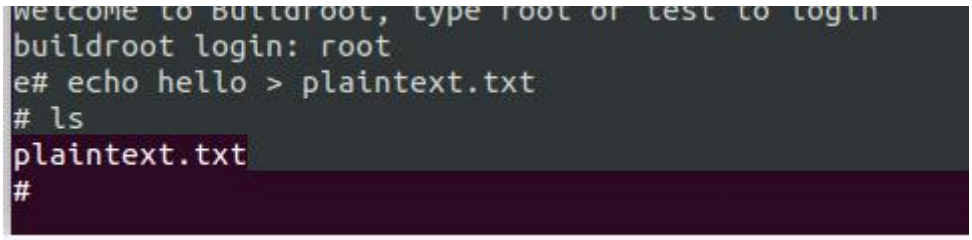
평/암호문 및 암호화키 파일의 경로:/root

2. 주요 코드에 대한 설명 (기능 명세)

가. 기본 기능

(a) 암호화

- 커맨드 : TEEencrypt -e [평문파일]



```
Welcome to Buildroot, type root or test to login
buildroot login: root
# echo hello > plaintext.txt
# cat plaintext.txt
hello
# ls
plaintext.txt
# TEEencrypt -e plaintext.txt
=======================Encryption=======================
Read the Plaintext : hello

Ciphertext : czggj


 # ls
ciphertext.txt    encryptedkey.txt  plaintext.txt
#
```

```
 # ls
ciphertext.txt    encryptedkey.txt  plaintext.txt
# cat ciphertext.txt
czggj
# cat encryptedkey.txt
W#
```

- 구현 기능 :

main.c

```
TEEC_Result res;
TEEC_Context ctx;
TEEC_Session sess;
TEEC_Operation op;
TEEC_UUID uuid = TA_TEEencrypt_UUID;
uint32_t err_origin;
char plaintext[64] = {0,};
char decrypttext[64] = {0,};
char ciphertext[64] = {0,};
char encrypted_key[1]={0};
char decrypted_key[1]={0};
int len=64;
```

필요한 변수를 선언하였다.

plaintext 읽은 평문을 저장할 배열

decrypttext 복호화 시킨 내용을 저장할 배열

ciphertext 암호화 시킨 내용을 저장할 배열

encrypted_key는 암호화된 키를 저장할 배열

decrypted_key는 복호화된 키를 저장할 배열

```
memset(&op, 0, sizeof(op));
op.paramTypes = TEEC_PARAM_TYPES(TEEC_MEMREF_TEMP_OUTPUT, TEEC_NONE,
                                 TEEC_NONE, TEEC_NONE);
op.params[0].tmpref.buffer = plaintext;
op.params[0].tmpref.size = sizeof(plaintext);
```

공유메모리를 설정하였다.

1) CA에서 평문 텍스트 파일 읽기, TA 호출

main.c

```
//-e , -d function devide need argv[0]=-e or -d argv[1]=filename
if(strcmp(argv[1],"-e")==0){
        printf("========================Encryption=========================\n");
        printf("Read the Plaintext : \n");

        int fd;//read the plaintext.txt
        char str[1024]="/root/";
        strcat(str,argv[2]);
        if((fd = open(str, O_RDONLY)) == -1) {
                fprintf(stderr, "plaintext.txt 파일을 open도중 오류 발생: %s\n",  strerror(errno));
                 return -1;
        }

        read(fd,plaintext,len);
        puts(plaintext);
        close(fd);
        //read plaintext.txt
        printf("\n plaintext.txt %s",plaintext);

        res = TEEC_InvokeCommand(&sess,TA_TEEencrypt_CMD_RANDOMKEY_GET, &op,
                                 &err_origin);//randomkey generate
        if (res != TEEC_SUCCESS)
                errx(1, "TEEC_InvokeCommand failed randomkey code 0x%x origin 0x%x",
                        res, err_origin);

        memcpy(op.params[0].tmpref.buffer, plaintext, sizeof(plaintext));//plaintext send TA

        res = TEEC_InvokeCommand(&sess, TA_TEEencrypt_CMD_ENC_VALUE, &op,
                                 &err_origin);//Ta calling ENC do!!
        if (res != TEEC_SUCCESS)
                errx(1, "TEEC_InvokeCommand failed enc_value code 0x%x origin 0x%x",
                        res, err_origin);
```

main에서 argv로 입력받은 매개인자를 확인할 수 있다. argv[1]에 들어온 값으로 -e 이면 암호화를 시킨다. /root/입력한 파일이름 경로로 평문 파일을 읽는다. plaintext 배열에 저장한다. 랜덤키를 먼저 생성하고 암호화를 실행시켜야 랜덤키를 사용하여 암호화 된다. memcpy를 통해 op.params라는 공유메모리에 plaintext 내용을 복사한다. 평문이 TA에게 전달된다. InvokeCommand로 TA가 호출된다.

2) TA에서 랜덤키 생성

main.c

```
        res = TEEC_InvokeCommand(&sess,TA_TEEencrypt_CMD_RANDOMKEY_GET, &op,
                                 &err_origin);//randomkey generate
        if (res != TEEC_SUCCESS)
                errx(1, "TEEC_InvokeCommand failed randomkey code 0x%x origin 0x%x",
                        res, err_origin);
```

main 부분에서 랜덤키 생성이 호출되는 부분이다.

TEEencrypt_ta.c

```
TEE_Result TA_InvokeCommandEntryPoint(void __maybe_unused *sess_ctx,
                        uint32_t cmd_id,
                        uint32_t param_types, TEE_Param params[4])
{
        (void)&sess_ctx; /* Unused parameter */
        key=1;
        switch (cmd_id) {
        case TA_TEEencrypt_CMD_ENC_VALUE:
                return enc_value(param_types, params);
        case TA_TEEencrypt_CMD_DEC_VALUE:
                return dec_value(param_types, params);
        case TA_TEEencrypt_CMD_RANDOMKEY_GET:
                return randomkey_get(param_types, params);
        case TA_TEEencrypt_CMD_RANDOMKEY_ENC:
                return randomkey_enc(param_types, params);
        case TA_TEEencrypt_CMD_RANDOMKEY_DEC:
                return randomkey_dec(param_types, params);
        default:
                return TEE_ERROR_BAD_PARAMETERS;
        }
}
```

root key는 1 이다.

TEEencrypt_ta.c

```
static TEE_Result randomkey_get(uint32_t param_types,
        TEE_Param params[4])
{
        uint32_t exp_param_types = TEE_PARAM_TYPES(TEE_PARAM_TYPE_VALUE_INOUT,
                                                   TEE_PARAM_TYPE_NONE,
                                                   TEE_PARAM_TYPE_NONE,
                                                   TEE_PARAM_TYPE_NONE);

        TEE_GenerateRandom(randomkey, sizeof(randomkey));
        //randomnumber 1~25: 25 count
        //A~Z(0~25) matching
        while(randomkey[0]<0||randomkey[0]%26==0){
                TEE_GenerateRandom(randomkey, sizeof(randomkey));
        }
        //regenerated randomkey

                        int temp= randomkey[0]%26;
                        randomkey[0]=temp;
                        DMSG("randomkey is generated : %d\n",randomkey[0]);

        return TEE_SUCCESS;
}
```

randomkey_get 함수에서 void TEE_GenerateRandom(void *randomBuffer, uint32_t randomBufferLen); 를 사용하여 랜덤값을 생성한다. 음수이고 26으로 나눈 나머지가 0이 나온다면 암호화 키로 사용할 수 없기 때문에 랜덤값을 재생성한다. 26으로 나눈 나머지 값이 1~25까지 나오므로 그 값이 randomkey값이 된다.

3) 랜덤키로 평문 암호화, 랜덤키는 TA의 root키로 암호화
(평문 및 TA 랜덤키를 암호화 하는 알고리즘은 모두 시저암호)
(TA의 root키는 미리 TA 내에 정의되어있는 키임)

```
TEE_Result TA_InvokeCommandEntryPoint(void __maybe_unused *sess_ctx,
                        uint32_t cmd_id,
                        uint32_t param_types, TEE_Param params[4])
{
        (void)&sess_ctx; /* Unused parameter */
        key=1;
        switch (cmd_id) {
        case TA_TEEencrypt_CMD_ENC_VALUE:
                return enc_value(param_types, params);
        case TA_TEEencrypt_CMD_DEC_VALUE:
                return dec_value(param_types, params);
        case TA_TEEencrypt_CMD_RANDOMKEY_GET:
                return randomkey_get(param_types, params);
        case TA_TEEencrypt_CMD_RANDOMKEY_ENC:
                return randomkey_enc(param_types, params);
        case TA_TEEencrypt_CMD_RANDOMKEY_DEC:
                return randomkey_dec(param_types, params);
        default:
                return TEE_ERROR_BAD_PARAMETERS;
        }
}
```

root key는 1 이다.

main.c

```
  memcpy(op.params[0].tmpref.buffer, plaintext, sizeof(plaintext));//plaintext send TA

  res = TEEC_InvokeCommand(&sess, TA_TEEencrypt_CMD_ENC_VALUE, &op,
                           &err_origin);//Ta calling ENC do!!
  if (res != TEEC_SUCCESS)
          errx(1, "TEEC_InvokeCommand failed enc_value code 0x%x origin 0x%x",
                  res, err_origin);
```

평문을 암호화하기 위해서 TA가 호출되는 부분이다.

TEEencrypt_ta.c rootkey로 randomkey 암호화

```
static TEE_Result randomkey_enc(uint32_t param_types,
        TEE_Param params[4])
{
        uint32_t exp_param_types = TEE_PARAM_TYPES(TEE_PARAM_TYPE_VALUE_INOUT,
                                                   TEE_PARAM_TYPE_NONE,
                                                   TEE_PARAM_TYPE_NONE,
                                                   TEE_PARAM_TYPE_NONE);

        char * in = (char *)params[0].memref.buffer;//share memory TA send encrypted key to CA

        char encrypted [1]={0};//store encrypted key

        char alpabet[27]={0,};

        for(int i=0; i<26;i++){
                alpabet[i]='A'+i;
        }

        int temp= randomkey[0];
        DMSG("=======Randomkey Encryption===\n");
        encrypted[0]= alpabet[temp];
        //DMSG("temp is %d",temp);
        DMSG ("Encrypt_key :  %s", encrypted);

        for(int i=0; i<1;i++){
                if (encrypted[i] >= 'A' && encrypted[i] <= 'Z') {
                        encrypted[i] -= 'A';
                        encrypted[i] += key;
                        encrypted[i] = encrypted[i] % 26;
                        encrypted[i] += 'A';
                }
        }
        memcpy(in, encrypted, 1);//TA send Encryption key to CA
        //encrypted is encrypted_key

        return TEE_SUCCESS;
}
```

0~25까지 있으면 A~Z로 매칭한다.
rootkey(key) 로 암호화한다. 공유메모리를 통해서 CA로 암호화된 키를 전달한다.

TEEencrypt_ta.c 평문을 암호화

```
static TEE_Result enc_value(uint32_t param_types,
        TEE_Param params[4])
{
        uint32_t exp_param_types = TEE_PARAM_TYPES(TEE_PARAM_TYPE_VALUE_INOUT,
                                                   TEE_PARAM_TYPE_NONE,
                                                   TEE_PARAM_TYPE_NONE,
                                                   TEE_PARAM_TYPE_NONE);//delete
        DMSG("enc_value come");
        char * in = (char *)params[0].memref.buffer;
        DMSG("enc_value char* in is = %s",in);
        int in_len = strlen (params[0].memref.buffer);
        char encrypted [64]={0,};

        DMSG("=======================Encryption=======================\n");
        DMSG ("Plaintext :  %s", in);
        memcpy(encrypted, in, in_len);

        for(int i=0; i<in_len;i++){
                if(encrypted[i]>='a' && encrypted[i] <='z'){
                        encrypted[i] -= 'a';
                        encrypted[i] += randomkey[0];
                        encrypted[i] = encrypted[i] % 26;
                        encrypted[i] += 'a';
                }
                else if (encrypted[i] >= 'A' && encrypted[i] <= 'Z') {
                        encrypted[i] -= 'A';
                        encrypted[i] += randomkey[0];
                        encrypted[i] = encrypted[i] % 26;
                        encrypted[i] += 'A';
                }
        }
        DMSG ("Ciphertext :  %s", encrypted);
        memcpy(in, encrypted, in_len);

        return TEE_SUCCESS;
}
```

in에 평문의 내용이 들어온다. randomkey로 평문을 암호화 한다.
공유메모리를 통해서 암호화된 내용을 CA로 전달한다.

4) TA에서 CA로 암호문 + 암호화된 TA키 전달

main.c

```
memcpy(op.params[0].tmpref.buffer, plaintext, sizeof(plaintext));//plaintext send TA

res = TEEC_InvokeCommand(&sess, TA_TEEencrypt_CMD_ENC_VALUE, &op,
                         &err_origin);//Ta calling ENC do!!
if (res != TEEC_SUCCESS)
        errx(1, "TEEC_InvokeCommand failed enc_value code 0x%x origin 0x%x",
                res, err_origin);
memcpy(ciphertext, op.params[0].tmpref.buffer, len);//TA send and CA ciphertext receive
printf("Ciphertext : %s\n", ciphertext);

FILE *fw=fopen("/root/ciphertext.txt","w");//save ciphertext.txt
if(0<fw){
        fputs(ciphertext,fw);
        fclose(fw);
}else{
        printf("fail write open\n");
}

memcpy(op.params[0].tmpref.buffer, encrypted_key,1);
res = TEEC_InvokeCommand(&sess,TA_TEEencrypt_CMD_RANDOMKEY_ENC, &op,
                         &err_origin);//ramdonkey encrytpion
memcpy(encrypted_key, op.params[0].tmpref.buffer, 1);
printf("\n ");
//printf("encrypted_key is %s\n",encrypted_key);


FILE *fe=fopen("/root/encryptedkey.txt","w");

if(0<fe){
        fputs(encrypted_key,fe);
        fclose(fe);
}else{
        printf("fail write enc_key open\n");
}
```

ciphertext에는 TA가 전달한 암호화된 내용이 저장된다.

encrypted_key에는 TA가 전달한 암호화된 키가 저장된다.

```
static TEE_Result enc_value(uint32_t param_types,
        TEE_Param params[4])
{
        uint32_t exp_param_types = TEE_PARAM_TYPES(TEE_PARAM_TYPE_VALUE_INOUT,
                                                   TEE_PARAM_TYPE_NONE,
                                                   TEE_PARAM_TYPE_NONE,
                                                   TEE_PARAM_TYPE_NONE);//delete
        DMSG("enc_value come");
        char * in = (char *)params[0].memref.buffer;
        DMSG("enc_value char* in is = %s",in);
        int in_len = strlen (params[0].memref.buffer);
        char encrypted [64]={0,};

        DMSG("=======================Encryption=======================\n");
        DMSG ("Plaintext :  %s", in);
        memcpy(encrypted, in, in_len);

        for(int i=0; i<in_len;i++){
                if(encrypted[i]>='a' && encrypted[i] <='z'){
                        encrypted[i] -= 'a';
                        encrypted[i] += randomkey[0];
                        encrypted[i] = encrypted[i] % 26;
                        encrypted[i] += 'a';
                }
                else if (encrypted[i] >= 'A' && encrypted[i] <= 'Z') {
                        encrypted[i] -= 'A';
                        encrypted[i] += randomkey[0];
                        encrypted[i] = encrypted[i] % 26;
                        encrypted[i] += 'A';
                }
        }
        DMSG ("Ciphertext :  %s", encrypted);
        memcpy(in, encrypted, in_len);

        return TEE_SUCCESS;
}
```

5) CA는 받은 암호문, 암호화된 키를 파일로 저장한다. 암호화된 키를 별도의 파일에 저장하였다. 읽어 올 평문 텍스트 파일 및 저장되는 암호문 및 암호화키 파일은 /root 디렉토리에 위치한다.

main.c

```
memcpy(op.params[0].tmpref.buffer, plaintext, sizeof(plaintext));//plaintext send TA

res = TEEC_InvokeCommand(&sess, TA_TEEencrypt_CMD_ENC_VALUE, &op,
                         &err_origin);//Ta calling ENC do!!
if (res != TEEC_SUCCESS)
        errx(1, "TEEC_InvokeCommand failed enc_value code 0x%x origin 0x%x",
                res, err_origin);
memcpy(ciphertext, op.params[0].tmpref.buffer, len);//TA send and CA ciphertext receive
printf("Ciphertext : %s\n", ciphertext);

FILE *fw=fopen("/root/ciphertext.txt","w");//change!!!!!!!!
if(0<fw){
        //write(fw,ciphertext,len);
        fputs(ciphertext,fw);
        fclose(fw);
}else{
        printf("fail write open\n");
}

memcpy(op.params[0].tmpref.buffer, encrypted_key,1);
res = TEEC_InvokeCommand(&sess,TA_TEEencrypt_CMD_RANDOMKEY_ENC, &op,
                         &err_origin);//ramdonkey encrytpion
memcpy(encrypted_key, op.params[0].tmpref.buffer, 1);
printf("\n ");
printf("encrypted_key is %s\n",encrypted_key);


FILE *fe=fopen("/root/encryptedkey.txt","w");

if(0<fe){
//write(fe,encrypted_key,len);
        fputs(encrypted_key,fe);
        fclose(fe);
}else{
        printf("fail write enc_key open\n");
}
```

(b) 복호화

- 커맨드 : TEEencrypt -d [암호문파일] [암호화키파일] (별도의 파일에 저장함)

```
W# TEEencrypt -d ciphertext.txt encryptedkey.txt
=======================Decryption==================
read cipher text is czggj

DecryptedResult : hello

# ls
ciphertext.txt     decryptReuslt.txt  encryptedkey.txt   plaintext.txt
# cat decryptReuslt.txt
hello
#
```

- 구현 기능 :

main.c

```
}else if(strcmp(argv[1],"-d")==0){
        //argv[3]ciphertext.txt argv[4]encryptedkey.txt
        printf("======================Decryption=================\n");

        int fd;//read the ciphertext.txt
        char str[1024]="/root/";
        strcat(str,argv[2]);
        if((fd = open(str, O_RDONLY)) == -1) {
                fprintf(stderr, "chipher.txt 파일을 open도중 오류 발생: %s\n",    strerror(errno));
                 return -1;
        }

        read(fd,ciphertext,len);
        printf("read cipher text is ");
        puts(ciphertext);
        close(fd);

        int fdk;//read the encryptedkeytext.txt
        char strk[1024]="/root/";
        strcat(strk,argv[3]);
        if((fdk = open(strk, O_RDONLY)) == -1) {
                fprintf(stderr, "encryptedkey.txt 파일을 open도중 오류 발생: %s\n",       strerror(errno));
                 return -1;
        }

        read(fdk,decrypted_key,1);//before decrypt it is encryptedkey
        close(fdk);
```

-d를 매개인자로 받으면 복호화가 실행된다. /root/매개인자로 받은 파일명 으로 암호화된 파일을 읽는다. 암호화된 키도 TA로 전달하여 복호화해서 랜덤키를 구해야 암호화된 내용을 복호화 할 수 있기 때문에 암호화키 파일도 읽는다.

1) CA에서 TA로 복호화 요청

```
                //decrypted_key
                memset(&op, 0, sizeof(op));
                op.paramTypes = TEEC_PARAM_TYPES(TEEC_MEMREF_TEMP_OUTPUT, TEEC_NONE,
                                                 TEEC_NONE, TEEC_NONE);

                op.params[0].tmpref.buffer = decrypttext;//clean buffer
                op.params[0].tmpref.size = len;

                memcpy(op.params[0].tmpref.buffer, decrypted_key, 1);
                res = TEEC_InvokeCommand(&sess, TA_TEEencrypt_CMD_RANDOMKEY_DEC, &op,
                                         &err_origin);//ramdomkey decrypt

                op.params[0].tmpref.buffer = decrypttext;//clean buffer
                if (res != TEEC_SUCCESS)
                        errx(1, "TEEC_InvokeCommand failed random key decrypt code 0x%x origin 0x%x",
                                res, err_origin);

                memcpy(op.params[0].tmpref.buffer, ciphertext, len);

                res = TEEC_InvokeCommand(&sess, TA_TEEencrypt_CMD_DEC_VALUE, &op,
                                         &err_origin);//ciphertext decrypt

                if (res != TEEC_SUCCESS)
                        errx(1, "TEEC_InvokeCommand failed decrypt code 0x%x origin 0x%x",
                                res, err_origin);
                memcpy(decrypttext, op.params[0].tmpref.buffer, len);
                printf("DecryptedResult : %s\n", decrypttext);

                FILE *fdw=fopen("/root/decryptReuslt.txt","w");//save decryptResult.txt
                if(0<fdw){
                        fputs(decrypttext,fdw);
                        fclose(fdw);
                }else{
                        printf("fail write open\n");
                }

        }
//////
```

암호화된 키를 복호화 시킨 후에 암호화된 파일을 복호화 시킬 수 있다.

2) TA에서 암호화된 키를 root키로 복호화

TEEencrypt_ta.c

```
static TEE_Result randomkey_dec(uint32_t param_types,
        TEE_Param params[4])
{
        char * in = (char *)params[0].memref.buffer;

        char decrypted [1]={0,};

        DMSG("==========Randomkey Decryption================\n");

        memcpy(decrypted, in, 1);
        //DMSG ("before decrypt :  %s", decrypted);
        for(int i=0; i<1;i++){

                if (decrypted[i] >= 'A' && decrypted[i] <= 'Z') {
                        decrypted[i] -= 'A';
                        decrypted[i] -= key;
                        decrypted[i] += 26;
                        decrypted[i] = decrypted[i] % 26;
                        decrypted[i] += 'A';
                }
        }
        for(int i=0; i<26;i++){
                if(decrypted[0]=='A'+i){
                        randomkey[0]=i;
                }
        }
        //DMSG ("decrypt key :  %s", decrypted);
        //DMSG ("decryptkey :  %d",randomkey[0]);

        return TEE_SUCCESS;
}
```

decrypted에 먼저 암호화된 키를 받고나서 복호화과정을 거친 후 randomkey값은 int여야 하기 때문에 char를 int로 바꿔주는 과정을 거친다. 복호화된 키는 전달하면 안 된다.

3) 랜덤키로 암호문을 복호화

TEEencrypt_ta.c

```
static TEE_Result dec_value(uint32_t param_types,
        TEE_Param params[4])
{
        char * in = (char *)params[0].memref.buffer;
        int in_len = strlen (params[0].memref.buffer);
        char decrypted [64]={0,};

        DMSG("=======================Decryption=======================\n");
        DMSG ("Ciphertext :  %s", in);
        memcpy(decrypted, in, in_len);

        for(int i=0; i<in_len;i++){
                if(decrypted[i]>='a' && decrypted[i] <='z'){
                        decrypted[i] -= 'a';
                        decrypted[i] -= randomkey[0];
                        decrypted[i] += 26;
                        decrypted[i] = decrypted[i] % 26;
                        decrypted[i] += 'a';
                }
                else if (decrypted[i] >= 'A' && decrypted[i] <= 'Z') {
                        decrypted[i] -= 'A';
                        decrypted[i] -= randomkey[0];
                        decrypted[i] += 26;
                        decrypted[i] = decrypted[i] % 26;
                        decrypted[i] += 'A';
                }
        }
        DMSG ("Plaintext :  %s", decrypted);
        memcpy(in, decrypted, in_len);

        return TEE_SUCCESS;
}
```

in으로 암호화된 내용을 받는다. 복호화 과정을 거쳐서 공유메모리로 복호화된 내용을 CA로 전달한다.

4) TA에서 평문을 CA로 전달

```
static TEE_Result dec_value(uint32_t param_types,
        TEE_Param params[4])
{
        char * in = (char *)params[0].memref.buffer;
        int in_len = strlen (params[0].memref.buffer);
        char decrypted [64]={0,};

        DMSG("=======================Decryption=======================\n");
        DMSG ("Ciphertext :  %s", in);
        memcpy(decrypted, in, in_len);

        for(int i=0; i<in_len;i++){
                if(decrypted[i]>='a' && decrypted[i] <='z'){
                        decrypted[i] -= 'a';
                        decrypted[i] -= randomkey[0];
                        decrypted[i] += 26;
                        decrypted[i] = decrypted[i] % 26;
                        decrypted[i] += 'a';
                }
                else if (decrypted[i] >= 'A' && decrypted[i] <= 'Z') {
                        decrypted[i] -= 'A';
                        decrypted[i] -= randomkey[0];
                        decrypted[i] += 26;
                        decrypted[i] = decrypted[i] % 26;
                        decrypted[i] += 'A';
                }
        }
        DMSG ("Plaintext :  %s", decrypted);
        memcpy(in, decrypted, in_len);

        return TEE_SUCCESS;
}
```

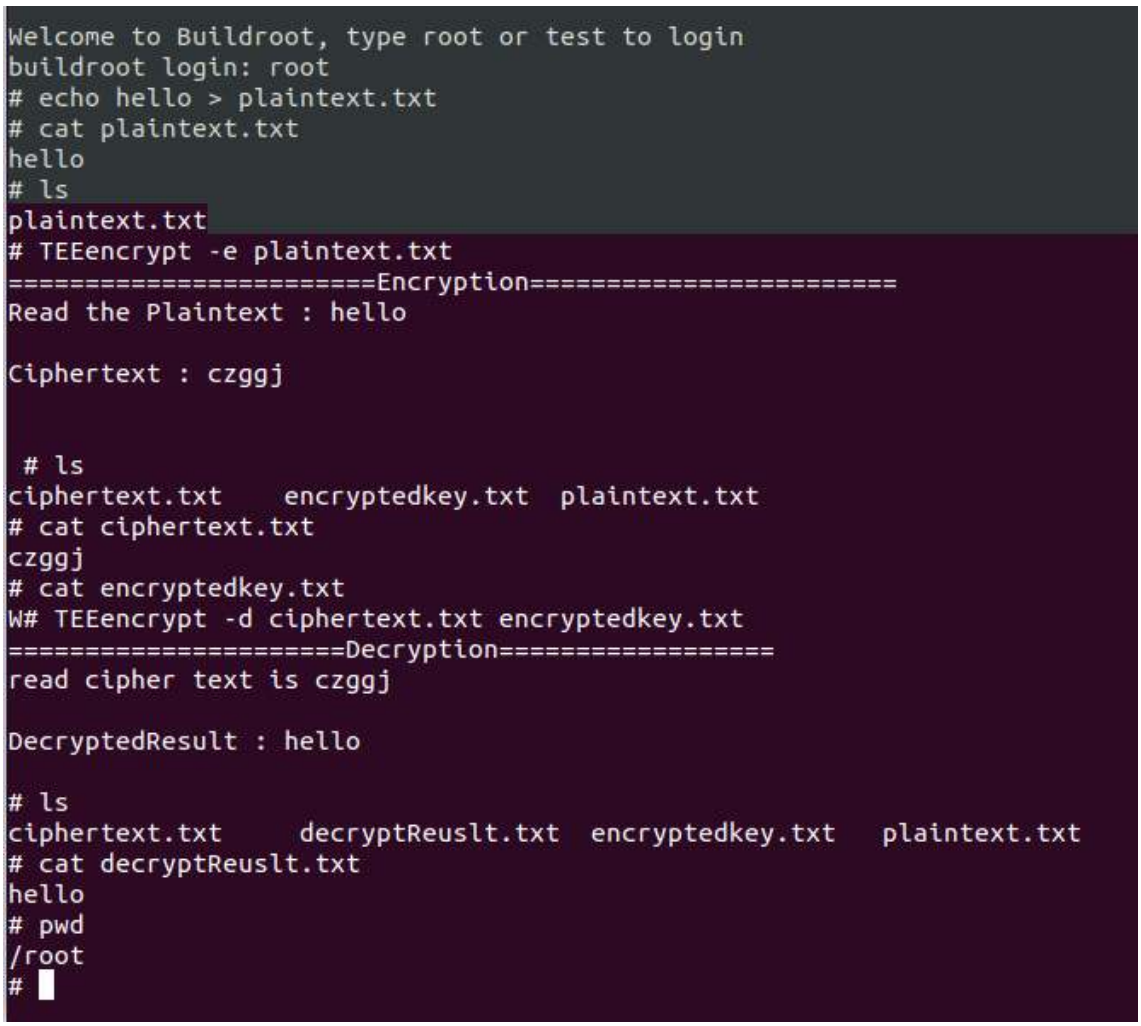
5) CA에서 평문 파일에 저장

```
memcpy(op.params[0].tmpref.buffer, ciphertext, len);
printf("read cipher text is %s",ciphertext);
res = TEEC_InvokeCommand(&sess, TA_TEEencrypt_CMD_DEC_VALUE, &op,
                         &err_origin);//ciphertext decrypt

if (res != TEEC_SUCCESS)
        errx(1, "TEEC_InvokeCommand failed decrypt code 0x%x origin 0x%x",
                 res, err_origin);
memcpy(decrypttext, op.params[0].tmpref.buffer, len);
printf("decrypttext : %s\n", decrypttext);

FILE *fdw=fopen("/root/decryptReuslt.txt","w");//save decryptResult.txt
if(0<fdw){
        fputs(decrypttext,fdw);
        fclose(fdw);
}else{
        printf("fail write open\n");
}
```

decrypttext 배열에 복호화된 내용이 전달되고 decyrptResult 파일을 열어서 저장한다.

- 추가 기능 구현 여부 X

3. 실행 결과에 대한 스크린 샷 :



4.소스코드 (github 링크)
https://github.com/hyorinkim/SystemNetwork.git

- 제출기한: 2020. 5. 10일 월요일 (약 4주간)