

# REPORT



과목: 소프트웨어 및 시스템보안

담당교수: 장진수

학과: 컴퓨터융합학부

학번: 202002499

이름: 박종현

제출일: 2024/12/17



**충남대학교**  
Chungnam National University

## [코드 설명]

### TEEencrypt\_ta.h

```
1  #ifndef TA_TEEencrypt_H
2  #define TA_TEEencrypt_H
3
4
5  #define TA_TEEencrypt_UUID \
6      { 0x1d844919, 0x987e, 0x4577, \
7          { 0xa1, 0xd9, 0xc1, 0x73, 0x39, 0x24, 0x2c, 0x7c} }
8
9
10 #define TA_TEEencrypt_CMD_RANDOMKEY_ENC    0
11 #define TA_TEEencrypt_CMD_RANDOMKEY_DEC    1
12 #define TA_TEEencrypt_CMD_RSAKEY_ENC      2
13
14 #endif /*TA_TEEencrypt_H*/
```

caesar방식 암호화, caesar방식 복호화, RSA방식 암호화 총 3개의 함수를 구현하였다.

### Main.c

main.c는 Non-Secure World에서 실행되며 TEE Client API를 통해서 TEE와 통신한다.

```
1  #include <err.h>
2  #include <stdio.h>
3  #include <stdlib.h>
4  #include <string.h>
5  #include <tee_client_api.h>
6  #include <TEEencrypt_ta.h>
7  #define MAX_TEXT_LEN 1024
8  #define RSA_KEY_SIZE 2048
9
10 void read_file(const char *filename, char *buffer) {
11     FILE *file = fopen(filename, "r");
12     if (!file)
13         errx(1, "Failed to open %s", filename);
14     fread(buffer, 1, MAX_TEXT_LEN, file);
15     fclose(file);
16 }
17
18 void write_file(const char *filename, const char *data) {
19     FILE *file = fopen(filename, "w");
20     if (!file)
21         errx(1, "Failed to write to %s", filename);
22     fprintf(file, "%s", data);
23     fclose(file);
24 }
```

암호화, 복호화를 진행하기 위해서는 파일을 읽어야 하므로 파일 읽기/쓰기 기능을 함수화 하였다.

```

27 int main(int argc, char *argv[]) {
28     if (argc != 4) {
29         printf("Usage: TEEencrypt -e [plaintext file] [caeser or rsa] (encryption)\n");
30         printf("    TEEencrypt -d [ciphertext file] [key file] (decryption)\n");
31         return 1;
32     }
33
34     TEEC_Result res;
35     TEEC_Context ctx;
36     TEEC_Session sess;
37     TEEC_Operation op;
38     TEEC_UUID uuid = TA_TEEencrypt_UUID;
39     uint32_t err_origin;
40
41     res = TEEC_InitializeContext(NULL, &ctx);
42     if (res != TEEC_SUCCESS)
43         errx(1, "TEEC_InitializeContext failed with code 0x%x", res);
44
45     res = TEEC_OpenSession(&ctx, &sess, &uuid, TEEC_LOGIN_PUBLIC, NULL, NULL, &err_origin);
46     if (res != TEEC_SUCCESS)
47         errx(1, "TEEC_OpenSession failed with code 0x%x origin 0x%x", res, err_origin);
48
49     memset(&op, 0, sizeof(op));

```

암호화 복호화에 사용되는 인자가 모두 4개이므로 입력 인자를 먼저 확인한다.

- 암호화: TEEencrypt -e plaintext.txt [caeser or rsa]
- 복호화: TEEencrypt -d ciphertext.txt key.txt

그리고 TEE와 연결하기 위한 기본 환경 설정을 한다.

NS world의 main.c는 TEE\_Context 구조체를 통해서 TEE와 통신하게 된다. TEEC\_InitializeContext는 TEE와 연결을 설정한다. 그리고 TEEC\_OpenSession 함수를 통해서 TEE의 TA와 연결된 세션을 생성한다.

```

D/TA: TA_CreateEntryPoint:18 TA Create Entry Point has been called
I/TA: Session Created Successfully

```

```

51  if (strcmp(argv[1], "-e") == 0) {
52      char plaintext[MAX_TEXT_LEN] = {0};
53      char ciphertext[RSA_KEY_SIZE / 8] = {0};
54      read_file(argv[2], plaintext);
55
56      if (strcmp(argv[3], "caesar") == 0) {
57          op.paramTypes = TEEC_PARAM_TYPES(TEEC_MEMREF_TEMP_INOUT, TEEC_VALUE_OUTPUT, TEEC_NONE, TEEC_NONE);
58          op.params[0].tmpref.buffer = plaintext;
59          op.params[0].tmpref.size = strlen(plaintext) + 1;
60
61          res = TEEC_InvokeCommand(&sess, TA_TEEencrypt_CMD_RANDOMKEY_ENC, &op, &err_origin);
62          if (res != TEEC_SUCCESS)
63              errx(1, "TEEC_InvokeCommand (Caesar encrypt) failed 0x%x origin 0x%x", res, err_origin);
64
65          write_file("ciphertext.txt", plaintext);
66          char encrypted_key[16];
67          snprintf(encrypted_key, sizeof(encrypted_key), "%d", op.params[1].value.a);
68          write_file("encryptedkey.txt", encrypted_key);
69
70          printf("Caesar Encryption complete. Output: ciphertext.txt, encryptedkey.txt\n");
71      }
72      else if (strcmp(argv[3], "rsa") == 0) {
73          op.paramTypes = TEEC_PARAM_TYPES(TEEC_MEMREF_TEMP_INPUT, TEEC_MEMREF_TEMP_OUTPUT, TEEC_NONE, TEEC_NONE);
74          op.params[0].tmpref.buffer = plaintext;
75          op.params[0].tmpref.size = strlen(plaintext) + 1;
76          op.params[1].tmpref.buffer = ciphertext;
77          op.params[1].tmpref.size = sizeof(ciphertext);
78          res = TEEC_InvokeCommand(&sess, TA_TEEencrypt_CMD_RSAKEY_ENC, &op, &err_origin);
79          if (res != TEEC_SUCCESS)
80              errx(1, "TEEC_InvokeCommand (RSA encrypt) failed 0x%x origin 0x%x", res, err_origin);
81
82          FILE *out_file = fopen("ciphertext.txt", "wb");
83          if (!out_file)
84              errx(1, "Failed to write to ciphertext.txt");
85          fwrite(ciphertext, 1, op.params[1].tmpref.size, out_file);
86          fclose(out_file);
87          printf("RSA Encryption complete. Output: ciphertext.txt\n");
88      }
89      else {
90          printf("Wrong Algorithm. Use \"caesar\" or \"rsa\".\n");
91          return 1;
92      }
93  }

```

명령어의 2번째 인자를 통해서 암호화/복호화를 구분한다.

암호화 명령어를 받게 되면 plaintext와 ciphertext를 생성한다.

- plaintext는 평문 데이터를 읽어서 저장한다.
  - ciphertext는 암호화된 텍스트를 저장한다.
- 여기서는 RSA방식으로 암호화된 데이터만 저장한다.

시저 암호화는 단순한 문자 치환이기 때문에 문자열을 직접 변경하여 덮어쓸 수 있어서 입출력에 같은 버퍼[TEEC\_MEMREF\_TEMP\_INOUT]을 사용하여 plaintext에 덮어쓴다.

RSA같은 경우에는 출력되는 암호문을 직접 덮어쓰기 할 수 없기 때문에 별도의 출력 버퍼를 통해서 저장해야 한다. [TEEC\_MEMREF\_TEMP\_INPUT] → [TEEC\_MEMREF\_TEMP\_OUTPUT]

2048비트 길이의 RSA키를 쓰는 암호화 방식에서 암호화된 데이터는 항상 256바이트이다.

암호화 알고리즘에 따라서 동작이 달라지기 때문에 각 암호화에 맞게 동작을 구현했다.

```

93     } else if (strcmp(argv[1], "-d") == 0) {
94         char ciphertext[MAX_TEXT_LEN] = {0};
95         char plaintext[MAX_TEXT_LEN] = {0};
96         int encrypted_key;
97
98         read_file(argv[2], ciphertext);
99         FILE *key_file = fopen(argv[3], "r");
100         if (!key_file)
101             errx(1, "Failed to open %s", argv[3]);
102         fscanf(key_file, "%d", &encrypted_key);
103         fclose(key_file);
104
105         op.paramTypes = TEEC_PARAM_TYPES(TEEC_MEMREF_TEMP_INOUT, TEEC_VALUE_INPUT, TEEC_NONE, TEEC_NONE);
106         op.params[0].tmpref.buffer = ciphertext;
107         op.params[0].tmpref.size = strlen(ciphertext) + 1;
108         op.params[1].value.a = encrypted_key;
109
110         res = TEEC_InvokeCommand(&sess, TA_TEEencrypt_CMD_RANDOMKEY_DEC, &op, &err_origin);
111         if (res != TEEC_SUCCESS)
112             errx(1, "TEEC_InvokeCommand (decrypt) failed 0x%x origin 0x%x", res, err_origin);
113
114         snprintf(plaintext, MAX_TEXT_LEN, "%s", (char *)op.params[0].tmpref.buffer);
115         write_file("plaintext.txt", plaintext);
116
117         printf("Decryption complete. Output: plaintext.txt\n");
118     } else {
119         printf("Invalid option. Use -e (encryption) or -d (decryption).\n");
120     }
121 }
122
123 TEEC_CloseSession(&sess);
124 TEEC_FinalizeContext(&ctx);
125 return 0;
126 }

```

복호화의 경우, RSA방식은 구현하지 않고 시저 암호화에 대한 복호화 기능만 구현하였다.

TEE에 전송할 인자들을 TEE\_PARAM\_TYPES를 통해서 설정한다.

복호화된 결과는 params[0].tmpref.buffer에 저장되기 때문에 이를 plaintext.txt 파일에 저장한다.

마지막으로 TEE와의 세션을 종료하고 컨텍스트를 정리함으로 TEE 연결과 관련된 자원을 정리한다.

## TEEencrypt\_ta.c

TEE에서 동작하는 TA를 구현한다.

```
1  #include <tee_internal_api.h>
2  #include <tee_internal_api_extensions.h>
3  #include <TEEencrypt_ta.h>
4  #include <string.h>
5
6  #define ROOT_KEY 5
7  #define RSA_KEY_SIZE 2048
8
9  static uint8_t random_key;
10
11 struct rsa_session {
12     TEE_OperationHandle op_handle;
13     TEE_ObjectHandle key_handle;
14 };
15
16
17 TEE_Result TA_CreateEntryPoint(void) {
18     DMSG("TA Create Entry Point has been called");
19     return TEE_SUCCESS;
20 }
21
22 void TA_DestroyEntryPoint(void) {
23     DMSG("TA Destroy Entry Point has been called");
24 }
25
26 TEE_Result TA_OpenSessionEntryPoint(uint32_t __unused param_types,
27                                     TEE_Param __unused params[4],
28                                     void **sess_ctx) {
29     struct rsa_session *sess = TEE_Malloc(sizeof(*sess), 0);
30     if (!sess)
31         return TEE_ERROR_OUT_OF_MEMORY;
32
33     sess->key_handle = TEE_HANDLE_NULL;
34     sess->op_handle = TEE_HANDLE_NULL;
35     *sess_ctx = (void *)sess;
36
37     IMSG("Session Created Successfully\n");
38     return TEE_SUCCESS;
39 }
40
41 void TA_CloseSessionEntryPoint(void *sess_ctx) {
42     struct rsa_session *sess = (struct rsa_session *)sess_ctx;
43
44     if (sess->key_handle != TEE_HANDLE_NULL)
45         TEE_FreeTransientObject(sess->key_handle);
46     if (sess->op_handle != TEE_HANDLE_NULL)
47         TEE_FreeOperation(sess->op_handle);
48
49     TEE_Free(sess);
50     IMSG("Session Closed\n");
51 }
```

클라이언트가 TA에 세션을 열고 닫을 때 동작을 정의한 함수이다.

시저 암호화 방식은 랜덤 키 값만 전달하면 되지만 RSA방식은 공개키, 개인키 쌍을 생성하고 관리해야하므로 키 객체를 요구한다.

이 코드는 시저 암호화와 RSA암호화 방식을 모두 지원하기 때문에 TEE\_Malloc을 사용하여 세션을 RSA암호화에 대한 세션을 생성한다.

```

53 //Caesar Cipher
54 static TEE_Result generate_random_key(void)
55 {
56     TEE_GenerateRandom(&random_key, sizeof(random_key));
57     random_key = (random_key % 25) + 1;
58     return TEE_SUCCESS;
59 }
60
61 static TEE_Result encrypt_caesar(uint32_t __unused param_types, TEE_Param params[4])
62 {
63     char *text = (char *)params[0].memref.buffer;
64     size_t text_len = params[0].memref.size;
65
66     if (!text || text_len == 0)
67         return TEE_ERROR_BAD_PARAMETERS;
68
69     DMSG("Encrypting text...");
70     DMSG("Plaintext: %s", text);
71     for (size_t i = 0; i < text_len; i++) {
72         if (text[i] >= 'a' && text[i] <= 'z') {
73             text[i] = ((text[i] - 'a' + random_key) % 26) + 'a';
74         } else if (text[i] >= 'A' && text[i] <= 'Z') {
75             text[i] = ((text[i] - 'A' + random_key) % 26) + 'A';
76         }
77     }
78
79     uint8_t encrypted_key = (random_key + ROOT_KEY) % 26;
80     params[1].value.a = encrypted_key;
81
82     DMSG("Ciphertext: %s", text);
83     DMSG("Encrypted Key: %d", encrypted_key);
84     return TEE_SUCCESS;
85 }
86
87 static TEE_Result decrypt_caesar(uint32_t __unused param_types, TEE_Param params[4])
88 {
89     char *text = (char *)params[0].memref.buffer;
90     size_t text_len = params[0].memref.size;
91     uint8_t encrypted_key = params[1].value.a;
92
93     if (!text || text_len == 0)
94         return TEE_ERROR_BAD_PARAMETERS;
95
96     uint8_t decryption_key = (encrypted_key + 26 - ROOT_KEY) % 26;
97
98     DMSG("Decrypting text...");
99     DMSG("Ciphertext: %s", text);
100
101     for (size_t i = 0; i < text_len; i++) {
102         if (text[i] >= 'a' && text[i] <= 'z') {
103             text[i] = ((text[i] - 'a' + 26 - decryption_key) % 26) + 'a';
104         } else if (text[i] >= 'A' && text[i] <= 'Z') {
105             text[i] = ((text[i] - 'A' + 26 - decryption_key) % 26) + 'A';
106         }
107     }
108
109     DMSG("Plaintext: %s", text);
110     return TEE_SUCCESS;
111 }

```

시저 암호화와 복호화 기능을 구현한 함수이다.

generate\_random\_key 함수는 TEE\_GenerateRandom 함수를 통해서 랜덤 값을 생성한다. 이 때 1~25사이가 되어야 하므로 mod 26을 해준다.

encrypt\_caesar 함수는 각 알파벳을 encrypted\_key만큼 움직여준다. 이때 encrypted\_key는 random\_key는 루트키와 함께 계산해서 encrypted\_key로 변환시켜 적용한다.

decrypt\_caesar 함수는 encrypted\_key와 루트키를 사용하여 복호화 키를 계산한다. 그리고 텍스트를 복호화 키의 크기만큼 역으로 이동시켜 평문을 복원한다.

```

113 //RSA Cipher
114 static TEE_Result create_rsa_keypair(struct rsa_session *sess) {
115     return TEE_AllocateTransientObject(TEE_TYPE_RSA_KEYPAIR, RSA_KEY_SIZE, &sess->key_handle) ||
116         TEE_GenerateKey(sess->key_handle, RSA_KEY_SIZE, NULL, 0);
117 }
118
119 static TEE_Result encrypt_rsa(struct rsa_session *sess, uint32_t __unused param_types, TEE_Param params[4]) {
120     TEE_Result res;
121     void *plain = params[0].memref.buffer;
122     size_t plain_len = params[0].memref.size;
123     void *cipher = params[1].memref.buffer;
124     uint32_t cipher_len = params[1].memref.size;
125
126     if (!plain || plain_len == 0 || !cipher || cipher_len < (RSA_KEY_SIZE / 8)) {
127         MSG("Invalid buffer sizes: plain_len=%zu, cipher_len=%u", plain_len, cipher_len);
128         return TEE_ERROR_BAD_PARAMETERS;
129     }
130     res = TEE_AllocateOperation(&sess->op_handle, TEE_ALG_RSAES_PKCS1_V1_5, TEE_MODE_ENCRYPT, RSA_KEY_SIZE);
131     if (res != TEE_SUCCESS) {
132         MSG("Failed to allocate RSA operation: 0x%x", res);
133         return res;
134     }
135     res = TEE_SetOperationKey(sess->op_handle, sess->key_handle);
136     if (res != TEE_SUCCESS) {
137         MSG("Failed to set RSA key: 0x%x", res);
138         TEE_FreeOperation(sess->op_handle);
139         return res;
140     }
141     res = TEE_AsymmetricEncrypt(sess->op_handle, NULL, 0, plain, plain_len, cipher, &cipher_len);
142     if (res != TEE_SUCCESS) {
143         MSG("RSA encryption failed: 0x%x", res);
144     } else {
145         params[1].memref.size = cipher_len;
146         MSG("RSA encryption successful, encrypted length: %u", cipher_len);
147     }
148
149     TEE_FreeOperation(sess->op_handle);
150     return res;
151 }

```

이 코드는 RSA암호화 복호화에 관련한 함수들이다.

create\_rsa\_keypair 함수는 RSA키 쌍을 저장할 수 있는 공간을 생성하고, TEE\_GenerateKey를 사용하여 RSA 키 쌍을 생성한다.

encrypt\_rsa 함수는 평문과 암호문의 버퍼의 유효성을 먼저 확인한다. RSA 암호화된 암호문은 RSA 키의 크기인 256바이트와 동일해야 한다. TEE\_AllocateOperation을 통해서 RSA연산 핸들을 생성하고, TEE\_SetOperationKey로 RSA 키를 정한다. TEE\_AsymmetricEncrypt를 통해서 비대칭 암호화를 수행한다.

```

153 TEE_Result TA_InvokeCommandEntryPoint(void *sess_ctx, uint32_t cmd_id, uint32_t param_types, TEE_Param params[4]) {
154     struct rsa_session *sess = (struct rsa_session *)sess_ctx;
155
156     switch (cmd_id) {
157     case TA_TEEencrypt_CMD_RANDOMKEY_ENC:
158         generate_random_key();
159         return encrypt_caesar(param_types, params);
160     case TA_TEEencrypt_CMD_RANDOMKEY_DEC:
161         return decrypt_caesar(param_types, params);
162     case TA_TEEencrypt_CMD_RSAKEY_ENC:
163         if (sess->key_handle == TEE_HANDLE_NULL) {
164             TEE_Result res = create_rsa_keypair(sess);
165             if (res != TEE_SUCCESS) {
166                 MSG("Failed to create RSA keypair: 0x%x", res);
167                 return res;
168             }
169             MSG("RSA keypair created successfully.");
170         }
171         return encrypt_rsa(sess, param_types, params);
172     default:
173         return TEE_ERROR_BAD_PARAMETERS;
174     }
175 }

```

CA에서 전달된 명령 요청에 따라서 TEE에서의 동작을 정의한다.



## [실행 결과]

### 평문 생성

```
File Edit View Search Terminal Help
# ls
test.txt
# cat test.txt
hello world
This is test message!
#
```

```
syssec@syssec-VirtualBox: ~/devel/optee/... x Terminal
I/TC: OP-TEE version: 3.9.0-dev (gcc version 8.3.0 (GNU Toolchain for the A-prof
ile Architecture 8.3-2019.03 (arm-rel-8.36))) #2 Sun May 24 07:54:59 UTC 2020 aa
rch64
D/TC:0 0 paged_init_primary:1175 Executing at offset 0 with virtual load address
0xe100000
D/TC:0 0 check_ta_store:635 TA store: "Secure Storage TA"
D/TC:0 0 check_ta_store:635 TA store: "REE"
D/TC:0 0 mobj_mapped_shm_init:447 Shared memory address range: fa00000, 11a00000
I/TC: Initialized
D/TC:0 0 paged_init_primary:1188 Primary CPU switching to normal world boot
D/TC:1 generic_boot_cpu_on_handler:1237 cpu 1: a0 0x0
D/TC:1 select_vector:956 SMCCC_ARCH_WORKAROUND_1 (0x80008000) available
D/TC:1 select_vector:957 SMC Workaround for CVE-2017-5715 used
D/TC:1 init_secondary_helper:1212 Secondary CPU Switching to normal world boot
D/TC:1 tee_entry_exchange_capabilities:102 Dynamic shared memory is enabled
D/TC:1 core_mmu_entry_to_finer_grained:762 xlat tables used 4 / 7
D/TC:? 0 tee_ta_init_pseudo_ta_session:284 Lookup pseudo TA 7011a688-ddde-4053-a
5a9-7b3c4ddf13b8
D/TC:? 0 tee_ta_init_pseudo_ta_session:297 Open device.pta
D/TC:? 0 tee_ta_init_pseudo_ta_session:311 device.pta : 7011a688-ddde-4053-a5a9-
7b3c4ddf13b8
D/TC:? 0 tee_ta_close_session:499 csess 0xe17c9f0 id 1
D/TC:? 0 tee_ta_close_session:518 Destroy session
```

### 시저 암호화

```
File Edit View Search Terminal Help
# TEencrypt
Usage: TEencrypt -e [plaintext file] [caesar or rsa] (encryption)
      TEencrypt -d [ciphertext file] [key file] (decryption)
# TEencrypt -e test.txt caesar
Caesar Encryption complete. Output: ciphertext.txt, encryptedkey.txt
# ls
ciphertext.txt  encryptedkey.txt  test.txt
# cat encryptedkey.txt
8# cat ciphertext.txt
kloor zruog
Wklv lv whvw phvvdjh!
#
```

```
syssec@syssec-VirtualBox: ~/devel/optee/... x Terminal
fa7ae01bbebc
D/TC:? 0 system_open_ta_binary:257 Lookup user TA ELF 1d844919-987e-4577-a1d9-c17
339242c7c (Secure Storage TA)
D/TC:? 0 system_open_ta_binary:260 res=0xffff0008
D/TC:? 0 system_open_ta_binary:257 Lookup user TA ELF 1d844919-987e-4577-a1d9-c17
339242c7c (REE)
D/TC:? 0 system_open_ta_binary:260 res=0x0
D/LD: ldelf:169 ELF (1d844919-987e-4577-a1d9-c17339242c7c) at 0x4007c000
D/TC:? 0 tee_ta_close_session:499 csess 0xe17b360 id 1
D/TC:? 0 tee_ta_close_session:518 Destroy session
D/TA: TA_CreateEntryPoint:18 TA Create Entry Point has been called
I/TA: Session Created Successfully
D/TA: encrypt_caesar:70 Encrypting text...
D/TA: encrypt_caesar:71 Plaintext: hello world
This is test message!
D/TA: encrypt_caesar:83 Ciphertext: kloor zruog
Wklv lv whvw phvvdjh!
D/TA: encrypt_caesar:84 Encrypted Key: 8
D/TC:? 0 tee_ta_close_session:499 csess 0xe17bb60 id 1
D/TC:? 0 tee_ta_close_session:518 Destroy session
I/TA: Session Closed
D/TA: TA_DestroyEntryPoint:23 TA Destroy Entry Point has been called
D/TC:? 0 destroy_context:298 Destroy TA ctx (0xe17bb00)
```

### 시저 복호화

```
File Edit View Search Terminal Help
# TEencrypt -d ciphertext.txt encryptedkey.txt
Decryption complete. Output: plaintext.txt
# ls
ciphertext.txt  encryptedkey.txt  plaintext.txt  test.txt
# cat plaintext.txt
hello world
This is test message!
#
```

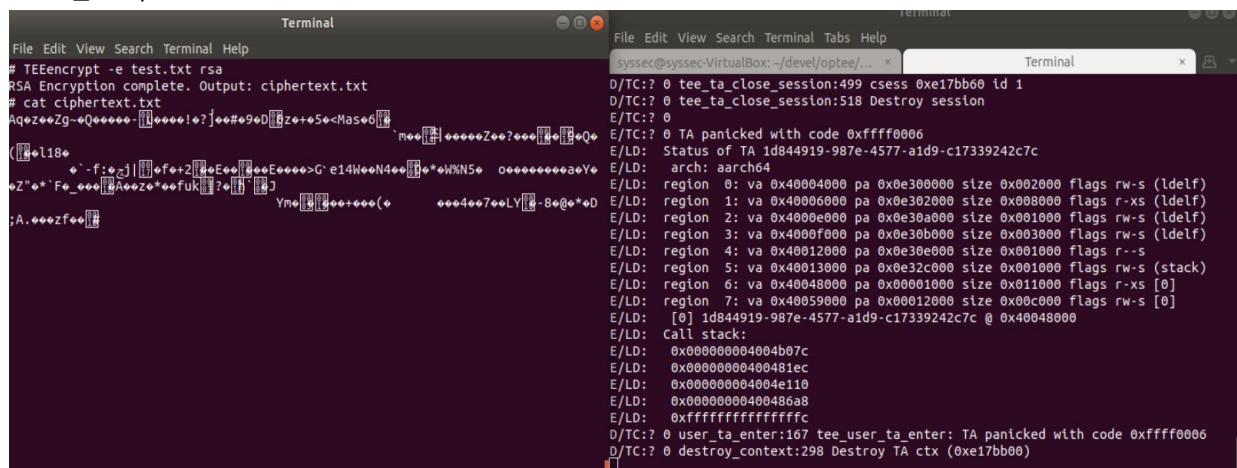
```
syssec@syssec-VirtualBox: ~/devel/optee/... x Terminal
D/TC:? 0 tee_ta_init_session_with_context:573 Re-open TA 3a2f8978-5dc0-11e8-9c2d-
fa7ae01bbebc
D/TC:? 0 system_open_ta_binary:257 Lookup user TA ELF 1d844919-987e-4577-a1d9-c17
339242c7c (Secure Storage TA)
D/TC:? 0 system_open_ta_binary:260 res=0xffff0008
D/TC:? 0 system_open_ta_binary:257 Lookup user TA ELF 1d844919-987e-4577-a1d9-c17
339242c7c (REE)
D/TC:? 0 system_open_ta_binary:260 res=0x0
D/LD: ldelf:169 ELF (1d844919-987e-4577-a1d9-c17339242c7c) at 0x40071000
D/TC:? 0 tee_ta_close_session:499 csess 0xe17b360 id 1
D/TC:? 0 tee_ta_close_session:518 Destroy session
D/TA: TA_CreateEntryPoint:18 TA Create Entry Point has been called
I/TA: Session Created Successfully
D/TA: decrypt_caesar:99 Decrypting text...
D/TA: decrypt_caesar:100 Ciphertext: byffi qilfx
Nbcm cm nymn gymuay!
D/TA: decrypt_caesar:110 Plaintext: hello world
This is test message!
D/TC:? 0 tee_ta_close_session:499 csess 0xe17bb60 id 1
D/TC:? 0 tee_ta_close_session:518 Destroy session
I/TA: Session Closed
D/TA: TA_DestroyEntryPoint:23 TA Destroy Entry Point has been called
D/TC:? 0 destroy_context:298 Destroy TA ctx (0xe17bb00)
```

## [프로그램 사용법]

- vi, echo 등의 명령어를 사용하여 txt파일을 생성한다.
- TEEencrypt -e [txt파일] [암호화방식: (caesar, rsa)] 을 입력하면 txt파일을 암호화 방식에 맞게 암호화한다.
- 암호화 파일과, 키는 cat 명령어를 통해서 확인이 가능하다.
- TEEencrypt -d [암호화파일] [키파일] 을 입력하면 시저 암호화에 대한 복호화된 데이터가 담긴 plaintext.txt 파일을 생성한다.

## [추가 기능 구현 여부] [O]

### RSA 암호화



The image shows two terminal windows. The left window displays the execution of the TEEencrypt program with the command `TEEencrypt -e test.txt rsa`, followed by the output `RSA Encryption complete. Output: ciphertext.txt` and the command `cat ciphertext.txt` which shows a base64-encoded string. The right window shows system logs from a TEE environment, including session management messages and a panic message: `E/TC?: TA panicked with code 0xffff0006`. Below the panic message, memory dump details are shown for various regions (0-7) and the call stack.

```
File Edit View Search Terminal Help
# TEEencrypt -e test.txt rsa
RSA Encryption complete. Output: ciphertext.txt
# cat ciphertext.txt
AqozZg~eQ*****!*****e9eDz+e5eMas6
m*****Z*****Q
<l18
'-f:zj|f+2E*****G>e14W+N4*****%N5 o*****aYe
Z"e*'F*****A+Z*****fuk?*****
;A.***zf*****

syssec@syssec-VirtualBox: ~/devel/optee/...
Terminal
D/TC?: tee_ta_close_session:499 csess 0xe17bb0 id 1
D/TC?: tee_ta_close_session:518 Destroy session
E/TC?:
E/TC?: TA panicked with code 0xffff0006
E/LD: Status of TA 1d844919-987e-4577-a1d9-c17339242c7c
E/LD: arch: aarch64
E/LD: region 0: va 0x40004000 pa 0x0e300000 size 0x002000 flags rw-s (ldelf)
E/LD: region 1: va 0x40006000 pa 0x0e302000 size 0x008000 flags r-xs (ldelf)
E/LD: region 2: va 0x4000e000 pa 0x0e30a000 size 0x001000 flags rw-s (ldelf)
E/LD: region 3: va 0x4000f000 pa 0x0e30b000 size 0x003000 flags rw-s (ldelf)
E/LD: region 4: va 0x40012000 pa 0x0e30e000 size 0x001000 flags r--s
E/LD: region 5: va 0x40013000 pa 0x0e32c000 size 0x001000 flags rw-s (stack)
E/LD: region 6: va 0x40048000 pa 0x00001000 size 0x011000 flags r-xs [0]
E/LD: region 7: va 0x40059000 pa 0x00012000 size 0x00c000 flags rw-s [0]
E/LD: [0] 1d844919-987e-4577-a1d9-c17339242c7c @ 0x40048000
E/LD: Call stack:
E/LD: 0x000000004004b07c
E/LD: 0x00000000400481ec
E/LD: 0x000000004004e110
E/LD: 0x00000000400486a8
E/LD: 0xfffffffffffffffc
D/TC?: user_ta_enter:167 tee_user_ta_enter: TA panicked with code 0xffff0006
D/TC?: destroy_context:298 Destroy TA ctx (0xe17bb00)
```

Github: <https://github.com/Hyeoninii/TEEencrypt>