SGX开发实验

实验要求

在SGX里实现RC4加密算法，设计三个ECALL函数，分别为S盒生成，流密钥生成，解密函数，其中key作为全局变量存放在Enclave中，密文通过ECALL从App传入Enclave中解密，将解密完成的结果通过OCALL输出。

- 密文HEX：1c7b53616e81ce8a45e7af3919bc94aba41258
- key：gosecgosec

RC4原理

RC4加密算法的原理基于伪随机数流和异或运算。

首先，需要一个密钥K，密钥K的长度可以是5到256字节不等，通过密钥编排算法KSA（Key Scheduling Algorithm）对密钥进行处理，生成一个256字节的密钥流S。

接下来，通过生成伪随机数流（Keystream），并将伪随机数流与明文进行异或运算，从而得到密文。伪随机数流的生成过程如下：

- 初始化一个S盒，S盒是一个长度为256的数组，里面存放着0~255的排列。
- 将S盒按照密钥流S进行混淆，生成一个新的S盒。
- 通过循环生成伪随机数流，每次循环生成一个数值，直到生成足够的长度。

在RC4算法中，将伪随机数流与明文进行异或运算，得到的结果就是密文。解密时，将密文与伪随机数流进行异或运算，就可以得到原始明文。

代码实现

Enclave/Enclave.edl

```
enclave {
    trusted {
        public void ecall_rc4_dec([out, size=len] char* buf, size_t len, [in,
size=ctext_len] const char* ciphertext, size_t ctext_len);
    };
};
```

Enclave/Enclave.c（可信代码）

```
#include <stdio.h>
```

```
#include <string.h>
#include <stdint.h>
#include <stdlib.h>

const char* key = "gosecgosec";

// S盒生成
void ecall_sbox_gen(uint8_t S[], uint8_t K[]) {
    int key_len = strlen(key);
    for (int i = 0; i < 256; i++) {
        S[i] = (uint8_t)i;
        K[i] = (uint8_t)key[i % key_len];
    }
    uint8_t j = 0;
    for (int i = 0; i < 256; i++) {
        j = (j + S[i] + K[i]) % 256;
        uint8_t tmp = S[i];
        S[i] = S[j];
        S[j] = tmp;
    }
}

// 流密钥生成
uint8_t ecall_keystream_gen(int& i, int& j, uint8_t S[]) {
    i = (i + 1) % 256;
    j = (j + S[i]) % 256;
    uint8_t tmp = S[i];
    S[i] = S[j];
    S[j] = tmp;
    int t = (S[i] + S[j]) % 256;
    return S[t];
}

// 解密函数
void ecall_rc4_dec(char* buf, size_t len, const char* ciphertext, , size_t ctext_len) {
    uint8_t K[256];
    uint8_t S[256];

    // 初始化S盒
    ecall_sbox_gen(S, K);

    // 密文字符串每两个字符转成一个8位的二进制数
    size_t chex_size = ctext_len / 2; // 密文转化后长度, 也是明文长度
    uint8_t* cipherhex = (uint8_t*)malloc(sizeof(uint8_t)*chex_size);
    char substr[2];
    for (int i = 0; i + 1 < ctext_len; i += 2) {
        substr[0] = ciphertext[i];
        substr[1] = ciphertext[i + 1];
        cipherhex[(i + 1) / 2] = strtol(substr, nullptr, 16);
    }

    // 解密
    char* plaintext = (char*)malloc(sizeof(char) * chex_size);
    for (int n = 0, i = 0, j = 0; n < chex_size; n++) {
        uint8_t keystream = ecall_keystream_gen(i, j, S); // 生成流密钥
        plaintext[n] = (char)(cipherhex[n] ^ keystream); // 分段解密
    }

    // 复制给buffer
    size_t size = len;
    if (strlen(plaintext) < len)
    {
        size = strlen(plaintext) + 1;
    }
    memcpy(buf, plaintext, size - 1);
    buf[size - 1] = '\0';
}
```

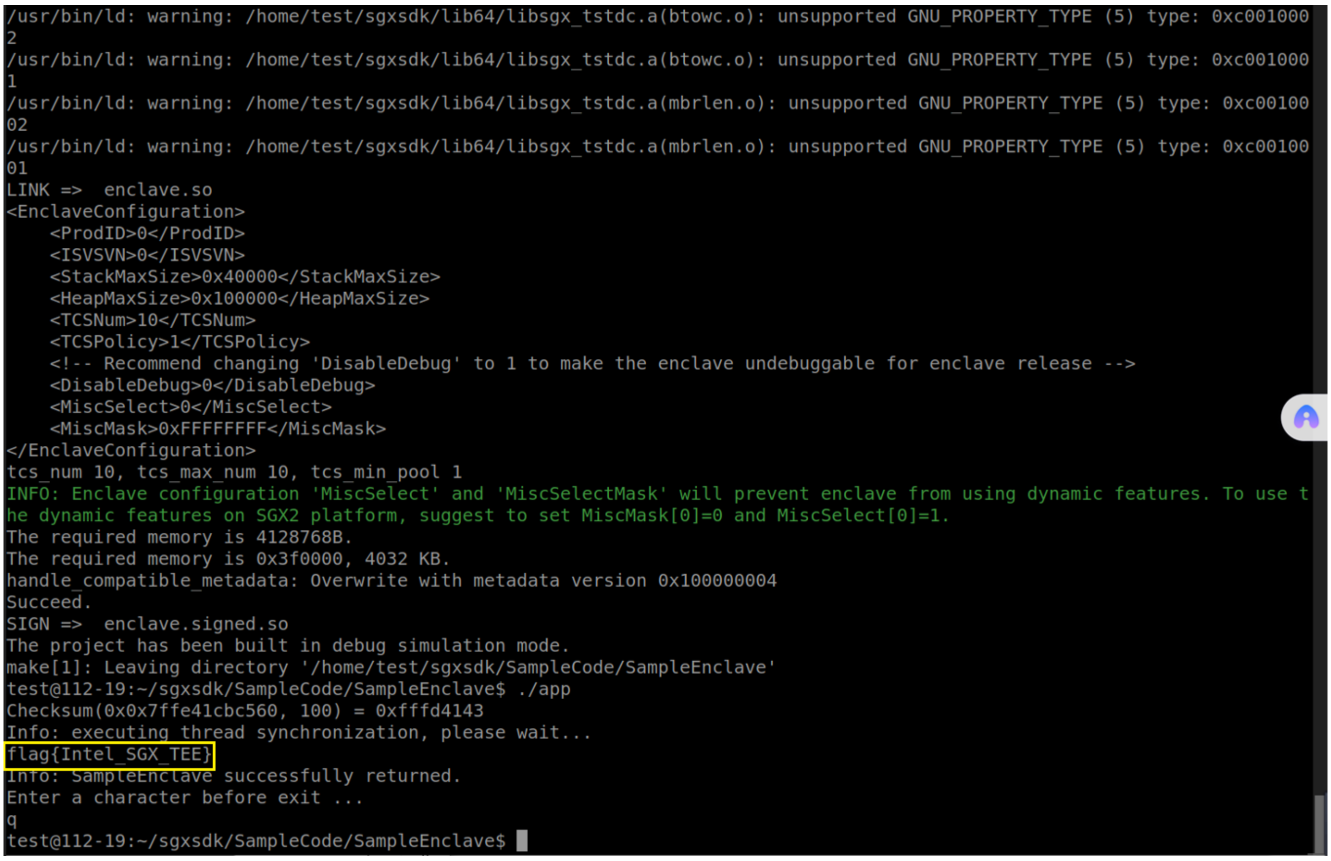
App/App.c（不可信代码）

```
#include <stdio.h>
#include <string.h>
#include <assert.h>
#include <unistd.h>
#include <pwd.h>
#include <string>
#define MAX_PATH FILENAME_MAX
#include "sgx_urts.h"
#include "App.h"
#include "Enclave_u.h"
/* Global EID shared by multiple threads */
sgx_enclave_id_t global_eid = 0;
int initialize_enclave(void)
{
    sgx_status_t ret = SGX_ERROR_UNEXPECTED;
    /* 调用 sgx_create_enclave 创建一个 Enclave 实例 */
    /* Debug Support: set 2nd parameter to 1 */
    ret = sgx_create_enclave(ENCLAVE_FILENAME, SGX_DEBUG_FLAG, NULL, NULL, &global_eid,
NULL);
    if (ret != SGX_SUCCESS) {
        printf("Failed to create enclave, ret code: %d\n", ret);
        return -1;
    }
    return 0;
}
/* 应用程序入口 */
int SGX_CDECL main(int argc, char *argv[])
{
    (void)(argc);
    (void)(argv);


    /* Initialize the enclave */
    if(initialize_enclave() < 0){
        printf("Enter a character before exit ...\n");
        getchar();
        return -1;
    }
    /* Utilize edger8r attributes */
    //edger8r_array_attributes();
    edger8r_pointer_attributes();
    edger8r_type_attributes();
    edger8r_function_attributes();

    /* Utilize trusted libraries */
    ecall_libc_functions();
    ecall_libcxx_functions();
    ecall_thread_functions();


    /* ECALL */
    const size_t max_buf_len = 100;
    char buffer[max_buf_len] = {0};
    char ciphertext[] = "1c7b53616e81ce8a45e7af3919bc94aba41258";
    strcat(ciphertext, "\0");
    const size_t ctext_len = strlen(ciphertext);
    ecall_rc4_dec(global_eid, buffer, max_buf_len, ciphertext, ctext_len);
    printf("%s\n", buffer);

    /* Destroy the enclave */
    sgx_destroy_enclave(global_eid);

    printf("Info: SampleEnclave successfully returned.\n");

    printf("Enter a character before exit ...\n");
    getchar();
    return 0;
}
```

实验结果

```
$ source /home/test/sgxsdk/environment
$ make SGX_MODE=sim
$ ./app
```

明文：flag{Intel_SGX_TEE}



```
/usr/bin/ld: warning: /home/test/sgxsdk/lib64/libsgx_tstdc.a(btowc.o): unsupported GNU_PROPERTY_TYPE (5) type: 0xc001000
2
/usr/bin/ld: warning: /home/test/sgxsdk/lib64/libsgx_tstdc.a(btowc.o): unsupported GNU_PROPERTY_TYPE (5) type: 0xc001000
1
/usr/bin/ld: warning: /home/test/sgxsdk/lib64/libsgx_tstdc.a(mbrlen.o): unsupported GNU_PROPERTY_TYPE (5) type: 0xc00100
02
/usr/bin/ld: warning: /home/test/sgxsdk/lib64/libsgx_tstdc.a(mbrlen.o): unsupported GNU_PROPERTY_TYPE (5) type: 0xc00100
01
LINK =>  enclave.so
<EnclaveConfiguration>
    <ProdID>0</ProdID>
    <ISVSVN>0</ISVSVN>
    <StackMaxSize>0x40000</StackMaxSize>
    <HeapMaxSize>0x100000</HeapMaxSize>
    <TCSNum>10</TCSNum>
    <TCSPolicy>1</TCSPolicy>
    <!-- Recommend changing 'DisableDebug' to 1 to make the enclave undebuggable for enclave release -->
    <DisableDebug>0</DisableDebug>
    <MiscSelect>0</MiscSelect>
    <MiscMask>0xFFFFFFFF</MiscMask>
</EnclaveConfiguration>
tcs_num 10, tcs_max_num 10, tcs_min_pool 1
INFO: Enclave configuration 'MiscSelect' and 'MiscSelectMask' will prevent enclave from using dynamic features. To use t
he dynamic features on SGX2 platform, suggest to set MiscMask[0]=0 and MiscSelect[0]=1.
The required memory is 4128768B.
The required memory is 0x3f0000, 4032 KB.
handle_compatible_metadata: Overwrite with metadata version 0x100000004
Succeed.
SIGN =>  enclave.signed.so
The project has been built in debug simulation mode.
make[1]: Leaving directory '/home/test/sgxsdk/SampleCode/SampleEnclave'
test@112-19:~/sgxsdk/SampleCode/SampleEnclave$ ./app
Checksum(0x0x7ffe41cbc560, 100) = 0xfffd4143
Info: executing thread synchronization, please wait...
flag{Intel_SGX_TEE}
Info: SampleEnclave successfully returned.
Enter a character before exit ...
q
test@112-19:~/sgxsdk/SampleCode/SampleEnclave$
```