[toc]

课上测试

ch06

作业题目：商用密码接口实现

完成下面任务（29分）

1 在 Ubuntu 或 openEuler 中完成任务（推荐openEuler）

2 参考GM/T0018 2023和实验代码，说明SDF接口调用的一般过程是什么（5分）

3 参考课程代码sdfproject，使用gmssl定义一个私有函数 static int getRandom(char *r, int length), 获取length个字节的随机数（7’）

4 把上述函数集成到src中的sdf.c中的SDF_GenerateRandom中,实现相关代码（10'）

5 在test中的main.c调用SDF_GenerateRandom进行测试，至少测试1个字节，5个字节，20个字节三种情况。（4’）

6 提交git log结果（3分）

作业提交要求 (1')

0. 记录实践过程和 AI 问答过程，尽量不要截图，给出文本内容
1. (选做)推荐所有作业托管到 gitee或 github 上
2. (必做)提交作业 markdown文档，命名为“学号-姓名-作业题目.md”
3. (必做)提交作业 markdown文档转成的 PDF 文件，命名为“学号-姓名-作业题目.pdf”

- github链接

作业内容

参考GM/T0018 2023和实验代码，说明SDF接口调用的一般过程是什么

- 一般过程如下：
    1. **初始化与设备打开**：加载头文件，用`SDF_OpenDevice`打开设备获设备句柄，失败则结束流程。
    2. **会话创建**：通过`SDF_OpenSession`传入设备句柄创建会话获会话句柄，失败要关闭设备。
    3. **按需进行其他操作（比如密钥）**：如导入根密钥与设备序列号、生成备份导入导出密钥对等。
    4. **数据处理操作**：包含哈希计算、加密解密、签名验证等具体的数据处理动作。
    5. **资源释放与关闭**：依次调用`SDF_CloseSession`关闭会话、`SDF_CloseDevice`关闭设备来释放资源。

参考课程代码sdfproject，使用gmssl定义一个私有函数 static int getRandom(char *r, int length), 获取length个字节的随机数（7’）

- 函数实现与测试代码：

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <gmssl/rand.h>  // GMSSL 提供的随机数头文件

// 定义静态函数，用于生成指定长度的随机数
static int getRandom(char *r, int length) {
    if (r == NULL || length <= 0) {
        return -1; // 错误的输入
    }

    // 使用 GMSSL 提供的 rand_bytes 函数生成随机字节
    int result = rand_bytes((unsigned char*)r, length);

    if (result != 1) {
        return -2; // 随机数生成失败
    }

    return 0; // 成功
}

int main() {
    char random_bytes[16]; // 定义一个缓冲区来存储16个字节的随机数

    // 调用 getRandom 获取随机数
    int ret = getRandom(random_bytes, sizeof(random_bytes));

    if (ret == 0) {
        printf("随机数生成成功:\n");
        for (int i = 0; i < sizeof(random_bytes); i++) {
            printf("%02X ", (unsigned char)random_bytes[i]);
        }
        printf("\n");
    } else {
        printf("随机数生成失败，错误代码: %d\n", ret);
    }

    return 0;
}
```

- 编译运行测试代码。根据结果不难得知实现成功。

```
root@Youer:~/shiyan/test/bestidiocs2024/ch06/test# nano random_test.c
root@Youer:~/shiyan/test/bestidiocs2024/ch06/test# gcc -o random_test
random_test.c -lgmssl
root@Youer:~/shiyan/test/bestidiocs2024/ch06/test# ./random_test
随机数生成成功:
```

```
1E 58 98 4E A1 35 70 66 2D BF FB 84 9E F1 3C EF
root@Youer:~/shiyan/test/bestidiocs2024/ch06/test# ./random_test
随机数生成成功:
7B 38 C4 D2 7D 85 6D 69 5E AC 6F 43 7E D2 B1 6E
root@Youer:~/shiyan/test/bestidiocs2024/ch06/test# ./random_test
随机数生成成功:
28 D8 E2 AA 26 77 BB BB C6 6D 76 3E 86 93 7B 2C
root@Youer:~/shiyan/test/bestidiocs2024/ch06/test# ./random_test
随机数生成成功:
F5 FF B9 49 C0 E8 7D CA 7F F5 3F 2C 57 4A A1 06
root@Youer:~/shiyan/test/bestidiocs2024/ch06/test# ./random_test
随机数生成成功:
EB A7 46 46 BD E3 A1 17 F2 58 8C AB A9 96 9C DA
```

把上述函数集成到src中的sdf.c中的SDF_GenerateRandom中,实现相关代码（10'）

- 修改后的sdf.c代码如下：

```c
#include "sdf.h"
#include <string.h>
#include <time.h>
#include <stdlib.h>
#include <gmssl/rand.h>

//*********************************
//设备管理
//*********************************

int SDF_OpenDevice(void ** phDeviceHandle){
    return SDR_OK;
}


int SDF_CloseDevice(void *hDeviceHandle){

    return SDR_OK;
}

int SDF_GetDeviceInfo( void * hSessionHandle, DEVICEINFO * pstDeviceInfo) {

    DEVICEINFO di;
    strcpy(di.IssuerName,"RocSDF");
    strcpy(di.DeviceName,"SDFBESTI181x");
    strcpy(di.DeviceSerial,"2021040001");
    di.DeviceVersion = 1;
    //...

    //pstDeviceInfo = &di;
    *pstDeviceInfo = di;

    return SDR_OK;
```

```
}

/**
 * 定义静态函数，用于生成指定长度的随机数
 */
static int getRandom(unsigned char *r, unsigned int length) {
    if (r == NULL || length <= 0) {
        return -1; // 错误的输入
    }

    int result = rand_bytes(r, length);

    if (result != 1) {
        return -2; // 随机数生成失败
    }

    return 0; // 成功
}

/**
 * 功能：获取指定长度的随机数
 */
int SDF_GenerateRandom(void * hSessionHandle, unsigned int uiLength, unsigned char
* pucRandom) {
    // 参数检查
    if (pucRandom == NULL || uiLength <= 0) {
        return -1; // 参数错误
    }

    // 调用 getRandom 函数生成随机数
    int result = getRandom(pucRandom, uiLength);

    // 返回结果
    return result;
}
```

在test中的main.c调用SDF_GenerateRandom进行测试，至少测试1个字节，5个字节，20个字节三种情况。（4’）

- 修改main.c代码如下：

```
#include "sdf.h"
#include <stdio.h>
#include <stdlib.h>

int main(){
    void * pdh = NULL; // 设备句柄
    int ret;

    // 打开设备
    ret = SDF_OpenDevice(&pdh);
```

```
    if(ret != SDR_OK){
        printf("error opening device!\n");
        return -1;
    } else {
        printf("device opened!\n");
    }

    // 获取设备信息
    DEVICEINFO testdi;
    ret = SDF_GetDeviceInfo(pdh, &testdi);
    if(ret != SDR_OK){
        printf("error getting device info!\n");
        SDF_CloseDevice(pdh); // 确保关闭设备
        return -1;
    } else {
        printf("Issuer Name: %s\n", testdi.IssuerName);
        printf("Device Name: %s\n", testdi.DeviceName);
        printf("Device Serial: %s\n", testdi.DeviceSerial);
        printf("Device Version: %d\n", testdi.DeviceVersion);
    }

    // 测试生成从1到20字节的随机数
    unsigned char pRandom[20]; // 增加数组大小以测试最多20字节

    for (unsigned int dataLen = 1; dataLen <= 20; ++dataLen) {
        ret = SDF_GenerateRandom(pdh, dataLen, pRandom);
        if(ret != SDR_OK){
            printf("error generating random bytes for length %u!\n", dataLen);
            break;
        } else {
            printf("Generated %u random byte(s): ", dataLen);
            for(unsigned int i = 0; i < dataLen; ++i)
                printf("%02x ", pRandom[i]); // 打印为两位十六进制数
            printf("\n");
        }
    }

    // 关闭设备
    ret = SDF_CloseDevice(pdh);

    if(ret != SDR_OK){
        printf("error closing device!\n");
        return -1;
    } else {
        printf("device closed!\n");
    }

    return 0;
}
```

- 修改makefile文件

```
# 编译器，这里使用gcc
CC = gcc

# 编译选项，-I 指定头文件路径，-L 指定库文件路径，-l 指定库
CFLAGS = -I./include
LDFLAGS = -lgmssl

# 目标可执行文件名称
TARGET = bin/test

# 获取src和test文件夹下所有的.c源文件
SRC_FILES := $(wildcard src/*.c test/*.c)

# 根据源文件生成对应的.o目标文件，放置在obj文件夹（需要手动创建）
OBJ_FILES := $(patsubst src/%.c, obj/%.o, $(filter src/%.c, $(SRC_FILES))) \
             $(patsubst test/%.c, obj/test/%.o, $(filter test/%.c, $(SRC_FILES)))

# 默认目标，用于生成最终的可执行文件
all: $(TARGET)

# 链接步骤，将所有的.o目标文件链接成最终的可执行文件
$(TARGET): $(OBJ_FILES)
    @mkdir -p $(dir $@) # 确保目标文件夹存在
    $(CC) $(CFLAGS) -o $@ $^ $(LDFLAGS)  # 添加 LDFLAGS 来链接库

# 编译规则，将每个.c源文件编译成对应的.o目标文件
obj/%.o: src/%.c
    @mkdir -p $(dir $@) # 确保目标文件夹存在
    $(CC) $(CFLAGS) -c -o $@ $<

# 编译规则，处理test文件夹下的.c源文件
obj/test/%.o: test/%.c
    @mkdir -p $(dir $@) # 确保目标文件夹存在
    $(CC) $(CFLAGS) -c -o $@ $<

# 清理规则，用于删除生成的目标文件和可执行文件
clean:
    rm -rf $(OBJ_FILES) $(TARGET) obj/*

.PHONY: all clean
```

- 编译结果

```
root@Youer:~/shiyan/test/sdfproject/sdfproject# make
gcc -I./include -c -o obj/sdf.o src/sdf.c
gcc -I./include -o bin/test obj/sdf.o obj/test/main.o -lgmssl    # 添加 LDFLAGS 来
链接库
```

- 运行结果

```
root@Youer:~/shiyan/test/sdfproject/sdfproject# ./bin/test
device opened!
Issuer Name: RocSDF
Device Name: SDFBESTI181x
Device Serial: 2021040001
Device Version: 1
Generated 1 random byte(s): a5
Generated 2 random byte(s): 13 3f
Generated 3 random byte(s): 59 b3 b9
Generated 4 random byte(s): 70 fb b5 22
Generated 5 random byte(s): df 2d 2e 7b c1
Generated 6 random byte(s): 46 9c d2 83 f1 e7
Generated 7 random byte(s): 92 b8 86 01 94 03 33
Generated 8 random byte(s): c6 ae df b5 71 23 92 ef
Generated 9 random byte(s): 7d f6 5c b5 fb e4 ab 95 30
Generated 10 random byte(s): 40 5e 05 83 6a 07 21 95 73 14
Generated 11 random byte(s): b7 b8 e4 ee 81 9a 7c 40 b1 69 45
Generated 12 random byte(s): 7f 95 b4 77 e1 4b 4e ec fa e0 67 ce
Generated 13 random byte(s): 1e a9 48 db 43 67 ef 2e 6a df 12 bf 45
Generated 14 random byte(s): 04 4d 08 6f 4c ff 49 1f a8 f6 31 9d 2e b2
Generated 15 random byte(s): cb 84 0b 7c 0d c7 1a 95 e9 9d b9 ce a8 4d 6e
Generated 16 random byte(s): 82 79 de 6c 97 66 67 24 22 91 53 e8 aa df ee 46
Generated 17 random byte(s): 2f 30 80 51 4f ea da 7f 9d b8 55 1a 64 fc c2 b3 0b
Generated 18 random byte(s): 97 86 94 90 e9 e5 f0 db cf 50 80 4b fa 56 5c 9a 71 31
Generated 19 random byte(s): ec 93 11 b3 61 81 1d 69 6e ca 61 32 7c 2d 34 94 41 94
63
Generated 20 random byte(s): f3 1a 32 25 82 ba 52 7d 6e 76 85 87 da ae ee 58 cc 96
92 31
device closed!
root@Youer:~/shiyan/test/sdfproject/sdfproject# ./bin/test
device opened!
Issuer Name: RocSDF
Device Name: SDFBESTI181x
Device Serial: 2021040001
Device Version: 1
Generated 1 random byte(s): e1
Generated 2 random byte(s): ec a3
Generated 3 random byte(s): e2 7d 14
Generated 4 random byte(s): 1d 8c 15 27
Generated 5 random byte(s): b6 3d 25 ed f4
Generated 6 random byte(s): d2 ba e8 cb 1d 1f
Generated 7 random byte(s): 7b 62 00 07 84 7f 30
Generated 8 random byte(s): fb 9a 59 ad dc 38 3c 42
Generated 9 random byte(s): 31 84 fc 05 78 9d 07 6c 80
Generated 10 random byte(s): 5b 26 d8 06 80 e1 09 28 1b 00
Generated 11 random byte(s): 9a ac 62 b7 3b 85 b4 2e a5 63 27
Generated 12 random byte(s): 40 df 10 90 4d 42 f4 52 78 85 c7 e3
Generated 13 random byte(s): 6e 0c 25 2e 73 c8 b6 2a b0 b0 9e 97 9d
Generated 14 random byte(s): 06 f5 10 bd f5 6c 76 32 52 51 50 c5 18 95
Generated 15 random byte(s): ff 0d 6e 28 7a d6 d4 74 97 ed a0 2b 8c a7 cf
Generated 16 random byte(s): 37 bd 2a d2 a6 4c e9 15 4b 71 c2 a4 2e 12 c4 dd
Generated 17 random byte(s): 88 14 89 f0 d5 22 31 35 28 cd 75 5d f2 ab ff 24 91
Generated 18 random byte(s): 2a 46 d8 b6 e6 53 4f 1c cb 08 d6 b8 f4 13 3a 96 38 d0
```

```
Generated 19 random byte(s): 40 e5 8c e7 79 79 ae b6 04 18 9a 6e 2c 57 91 f2 70 ee
e0
Generated 20 random byte(s): c2 90 cb 9a 5e d0 16 19 eb 88 b6 53 08 f7 77 cd 12 de
5b 8c
device closed!
```

提交git log结果（3分）

- git log记录

```
commit 781aed821774058847da93f86d5454231657d350 (HEAD -> master)
Author: 徐鹿鸣 <xlm20040219@qq.com>
Date:   Tue Dec 17 12:11:15 2024 +0800

    change makefile and test

commit 1f9588ccd5276b217ca1c2b968459446aa6d11b6
Author: 徐鹿鸣 <xlm20040219@qq.com>
Date:   Tue Dec 17 11:57:20 2024 +0800

    change sdf.c and testsdf.c

commit 2e3d1947b1b0db7a1f02d1dfd4078e296822fca8
Author: 徐鹿鸣 <xlm20040219@qq.com>
Date:   Tue Dec 17 11:49:22 2024 +0800

    finish random_test.c
```