[toc]

课上测试

ch03

作业题目：SM2算法测试

完成下面任务（24分）

1 在 Ubuntu 或 openEuler 中完成任务（推荐openEuler） 2 生成一个文档 sn.txt,内容为全班同学的 8 位学号，把你的学号排到第一个 3 使用 GmSSL 命令产生一对公私钥对。（4 分） 4 使用 GmSSL 编程对sn.txt进行加密解密，提交代码或代码链接，以及编译运行过程（文本或截图）（10 分） 5 使用 GmSSL 编程对sn.txt进行签名验签，提交代码或代码链接，以及编译运行过程（文本或截图）（10 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

生成一个文档 sn.txt,内容为全班同学的 8 位学号，把你的学号排到第一个

- 生成代码

```c
#include <stdio.h>

int main() {
    FILE *fp;
    int student_numbers[28];
    int i, index;

    // 生成学号数组
    for (i = 0; i < 28; i++) {
        student_numbers[i] = 20221401 + i;
    }

    // 找到20221414在数组中的位置并移除它
    for (i = 0; i < 28; i++) {
        if (student_numbers[i] == 20221414) {
            index = i;
            break;
        }
    }
    for (i = index; i < 27; i++) {
        student_numbers[i] = student_numbers[i + 1];
```

```
    }

    // 将20221414插入到数组第一个位置
    student_numbers[0] = 20221414;

    // 打开文件sn.txt用于写入，如果文件不存在则创建，如果存在则覆盖原有内容
    fp = fopen("sn.txt", "w");
    if (fp == NULL) {
        perror("Error opening file");
        return 1;
    }

    // 将学号逐个写入文件，每个学号占一行
    for (i = 0; i < 28; i++) {
        fprintf(fp, "%d\n", student_numbers[i]);
    }

    // 关闭文件
    fclose(fp);

    return 0;
}
```

- 过程

```
root@Youer:~/shiyan/test1210/sm2# nano gen_sn.c
root@Youer:~/shiyan/test1210/sm2# gcc -o gen_sn  gen_sn.c
root@Youer:~/shiyan/test1210/sm2# ls
gen_sn  gen_sn.c  gen_sn.py  sn.txt
root@Youer:~/shiyan/test1210/sm2# ./gen_sn
root@Youer:~/shiyan/test1210/sm2# ls
gen_sn  gen_sn.c  gen_sn.py  sn.txt
root@Youer:~/shiyan/test1210/sm2# cat sn.txt
20221414
20221402
20221403
20221404
20221405
20221406
20221407
20221408
20221409
20221410
20221411
20221412
20221413
20221415
20221416
20221417
20221418
20221419
20221420
```

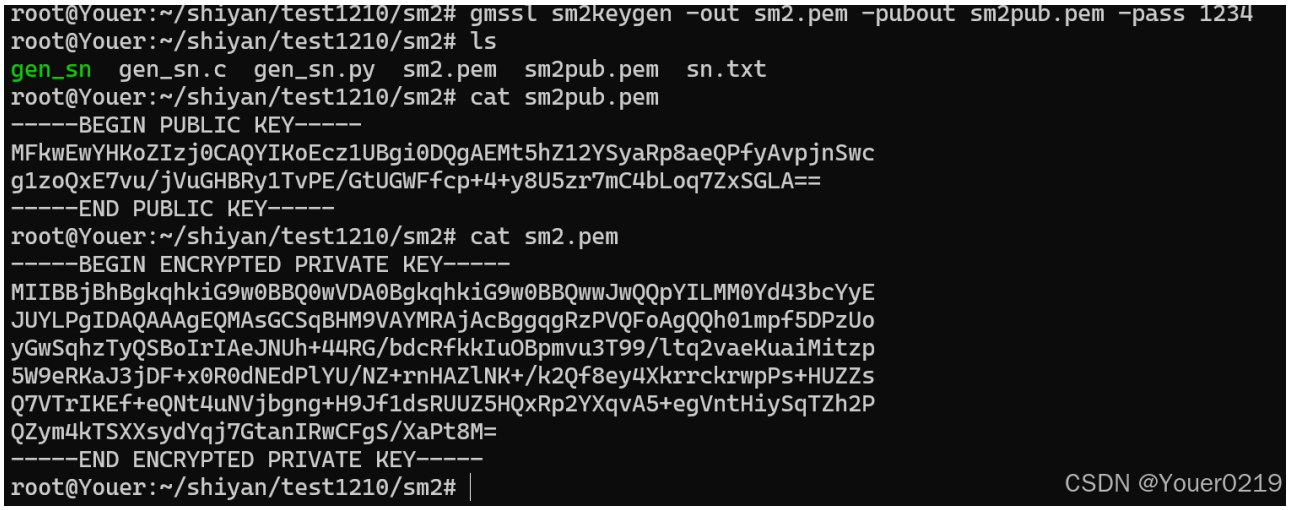
```
20221421
20221422
20221423
20221424
20221425
20221426
20221427
20221428
20221428
```

```
root@Youer:~/shiyan/test1210/sm2# nano gen_sn.c
root@Youer:~/shiyan/test1210/sm2# gcc -o gen_sn  gen_sn.c
root@Youer:~/shiyan/test1210/sm2# ls
gen_sn  gen_sn.c  gen_sn.py  sn.txt
root@Youer:~/shiyan/test1210/sm2# ./gen_sn
root@Youer:~/shiyan/test1210/sm2# ls
gen_sn  gen_sn.c  gen_sn.py  sn.txt
root@Youer:~/shiyan/test1210/sm2# cat sn.txt
20221414
20221402
20221403
20221404
20221405
20221406
20221407
20221408
20221409
20221410
20221411
20221412
20221413
20221415
20221416
20221417
20221418
20221419
20221420
20221421
20221422
20221423
20221424
20221425
20221426
20221427
20221428
20221428
root@Youer:~/shiyan/test1210/sm2#
```


使用 GmSSL 命令产生一对公私钥对

- 过程

```
root@Youer:~/shiyan/test1210/sm2# gmssl sm2keygen -out sm2.pem -pubout sm2pub.pem
-pass ''
root@Youer:~/shiyan/test1210/sm2# ls
gen_sn  gen_sn.c  gen_sn.py  sm2.pem  sm2pub.pem  sn.txt
root@Youer:~/shiyan/test1210/sm2# cat sm2pub.pem
-----BEGIN PUBLIC KEY-----
MFkwEwYHKoZIzj0CAQYIKoEcz1UBgi0DQgAEMt5hZ12YSyaRp8aeQPfyAvpjnSwc
g1zoQxE7vu/jVuGHBRy1TvPE/GtUGWFfcp+4+y8U5zr7mC4bLoq7ZxSGLA==
-----END PUBLIC KEY-----
root@Youer:~/shiyan/test1210/sm2# cat sm2.pem
-----BEGIN ENCRYPTED PRIVATE KEY-----
MIIBBjBhBgkqhkiG9w0BBQ0wVDA0BgkqhkiG9w0BBQwwJwQQpYILMM0Yd43bcYyE
JUYLPgIDAQAAAgEQMAsGCSqBHM9VAYMRAjAcBggqgRzPVQFoAgQQh01mpf5DPzUo
yGwSqhzTyQSBoIrIAeJNUh+44RG/bdcRfkkIuOBpmvu3T99/ltq2vaeKuaiMitzp
5W9eRKaJ3jDF+x0R0dNEdPlYU/NZ+rnHAZlNK+/k2Qf8ey4XkrrckrwpPs+HUZZs
Q7VTrIKEf+eQNt4uNVjbgng+H9Jf1dsRUUZ5HQxRp2YXqvA5+egVntHiySqTZh2P
QZym4kTSXXsydYqj7GtanIRwCFgS/XaPt8M=
-----END ENCRYPTED PRIVATE KEY-----
```

- 

使用 GmSSL 编程对sn.txt进行加密解密，提交代码或代码链接，以及编译运行过程（文本或截图）（10 分）

- 加密过程与代码
  - 加密过程

```
root@Youer:~/shiyan/test1210/sm2# ./sm2_encrypt sm2pub.pem sn.txt
en_output.bin
Encryption successful, output written to en_output.bin
root@Youer:~/shiyan/test1210/sm2# xxd en_output.bin
00000000: 3082 0166 0220 4eb6 91c1 2ef7 6343 00f7  0..f. N.....cC..
00000010: 0a3c 3a6b 7671 963a 47bd 4146 a46a c731  .<:kvq.:G.AF.j.1
00000020: a824 7391 494b 0221 00d4 5e7b 3ec6 bf9e  .$s.IK.!..^{>...
00000030: 7bd2 8f68 5c2b c5bb 75c6 1bfa 9bbc f24e  {..h\+..u......N
00000040: a125 235b 038d f1c5 9c04 2025 b9f2 9d36  .%#[...... %...6
00000050: 3cdb bf2b 581b 477d 2d86 c52b 5fee c24e  <..+X.G}-..+_..N
00000060: 9b81 2730 574d 2095 314c 6204 81fc 504e  ..'0WM .1Lb...PN
00000070: 35fc e7eb 3fab ce0d 3187 11ed 9f91 95b4  5...?...1.......
```

```
00000080: 54e0 9818 79d3 81ef 54ad 34da 78fe 2e66  T...y...T.4.x..f
00000090: e3b4 fdf1 a205 b213 d91f e93a e390 e80e  ...........:....
000000a0: 42b6 a856 2d4f 9fe4 41a2 595c 2804 7df7  B..V-O..A.Y\(.}.
000000b0: 9c96 eceb fe6f 182f 8be5 88ce 1365 a055  .....o./.....e.U
000000c0: ea42 331c 9f52 0023 9cf9 78a9 137f 9f88  .B3..R.#..x.....
000000d0: cac6 1821 85b8 b03e 908e f2b5 1ebc 4240  ...!...>......B@
000000e0: 2ca9 6f9a 8690 74cc 6ae6 6208 d371 243f  ,.o...t.j.b..q$?
000000f0: fd95 5307 7966 3776 c651 4f3b 1055 57a7  ..S.yf7v.QO;.UW.
00000100: f695 e9d6 9823 6882 cd5b 57d0 71f6 7078  .....#h..[W.q.px
00000110: bf7a cd67 3641 e1d3 fc97 4570 d775 b64f  .z.g6A....Ep.u.O
00000120: 4cf4 de55 aec7 2992 414e d5fa c0a6 78cf  L..U..).AN....x.
00000130: 37a6 9a11 cd1f 9144 e06f 5146 54e6 8e72  7......D.oQFT..r
00000140: 9d2b f7cf b7af d9aa 6f07 2cb9 fa86 4241  .+......o.,...BA
00000150: 5ae1 3fec 4921 ef31 5975 26e5 7ab2 ba35  Z.?.I!.1Yu&.z..5
00000160: 10ab 60cd d386 5400 8fe5                 ..`...T...
```

- 加密代码

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <gmssl/sm2.h>
#include <gmssl/pem.h>

#define SM2_CIPHERTEXT_SIZE 1024  // 根据需要调整大小

void print_usage() {
    printf("Usage: sm2_encrypt <public_key.pem> <input_file>
<output_file>\n");
}

int main(int argc, char *argv[]) {
    if (argc != 4) {
        print_usage();
        return 1;
    }

    const char *public_key_file = argv[1];
    const char *input_file = argv[2];
    const char *output_file = argv[3];

    // Load public key
    SM2_KEY sm2_key;
    FILE *fp = fopen(public_key_file, "r");
    if (!fp) {
        perror("Failed to open public key file");
        return 1;
    }

    // Use the correct function to read the public key
    if (sm2_public_key_info_from_pem(&sm2_key, fp) != 1) {
        fprintf(stderr, "Failed to read public key from PEM\n");
```

```
        fclose(fp);
        return 1;
    }
    fclose(fp);

    // Read input file
    FILE *in_fp = fopen(input_file, "rb");
    if (!in_fp) {
        perror("Failed to open input file");
        return 1;
    }

    fseek(in_fp, 0, SEEK_END);
    long input_len = ftell(in_fp);
    fseek(in_fp, 0, SEEK_SET);
    unsigned char *input_data = malloc(input_len);
    if (fread(input_data, 1, input_len, in_fp) != input_len) {
        perror("Failed to read input file");
        free(input_data);
        fclose(in_fp);
        return 1;
    }
    fclose(in_fp);

    // Encrypt data
    unsigned char ciphertext[SM2_CIPHERTEXT_SIZE];
    size_t ciphertext_len = sizeof(ciphertext);

    if (sm2_encrypt(&sm2_key, input_data, input_len, ciphertext,
&ciphertext_len) != 1) {
        fprintf(stderr, "SM2 encryption failed\n");
        free(input_data);
        return 1;
    }

    free(input_data);

    // Write output file
    FILE *out_fp = fopen(output_file, "wb");
    if (!out_fp) {
        perror("Failed to open output file");
        return 1;
    }

    if (fwrite(ciphertext, 1, ciphertext_len, out_fp) !=
ciphertext_len) {
        perror("Failed to write output file");
        fclose(out_fp);
        return 1;
    }
    fclose(out_fp);

    printf("Encryption successful, output written to %s\n",
output_file);
```

```
    return 0;
}
```

- 解密过程与代码
  - 解密过程

```
root@Youer:~/shiyan/test1210/sm2# ./sm2_decrypt
Usage: sm2_decrypt <private_key.pem> <input_file> <output_file>
root@Youer:~/shiyan/test1210/sm2# ./sm2_decrypt  sm2.pem  en_output.bin
de_output.txt
Decryption successful, output written to de_output.txt
root@Youer:~/shiyan/test1210/sm2# cat de_output.txt
20221414
20221402
20221403
20221404
20221405
20221406
20221407
20221408
20221409
20221410
20221411
20221412
20221413
20221415
20221416
20221417
20221418
20221419
20221420
20221421
20221422
20221423
20221424
20221425
20221426
20221427
20221428
20221428
```

  - 解密代码

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <gmssl/sm2.h>
#include <gmssl/pem.h>
```

```c
#define SM2_CIPHERTEXT_SIZE 1024  // 根据需要调整大小

void print_usage() {
    printf("Usage: sm2_decrypt <private_key.pem> <input_file>
<output_file>\n");
}

int main(int argc, char *argv[]) {
    if (argc != 4) {
        print_usage();
        return 1;
    }

    const char *private_key_file = argv[1];
    const char *input_file = argv[2];
    const char *output_file = argv[3];

    // Load private key
    SM2_KEY sm2_key;
    FILE *fp = fopen(private_key_file, "r");
    if (!fp) {
        perror("Failed to open private key file");
        return 1;
    }

    // 使用 PEM 格式加载私钥
    if (sm2_private_key_info_from_pem(&sm2_key, fp) != 1) {
        fprintf(stderr, "Failed to read private key from PEM\n");
        fclose(fp);
        return 1;
    }
    fclose(fp);

    // Read input file (ciphertext)
    FILE *in_fp = fopen(input_file, "rb");
    if (!in_fp) {
        perror("Failed to open input file");
        return 1;
    }

    fseek(in_fp, 0, SEEK_END);
    long input_len = ftell(in_fp);
    fseek(in_fp, 0, SEEK_SET);
    unsigned char *ciphertext = malloc(input_len);
    if (fread(ciphertext, 1, input_len, in_fp) != input_len) {
        perror("Failed to read input file");
        free(ciphertext);
        fclose(in_fp);
        return 1;
    }
    fclose(in_fp);

    // 解密数据
    unsigned char decrypted[SM2_CIPHERTEXT_SIZE];  // 根据需要调整大小
```

```
    size_t decrypted_len = sizeof(decrypted);

    if (sm2_decrypt(&sm2_key, ciphertext, input_len, decrypted,
&decrypted_len) != 1) {
        fprintf(stderr, "SM2 decryption failed\n");
        free(ciphertext);
        return 1;
    }

    free(ciphertext);

    // Write output file
    FILE *out_fp = fopen(output_file, "wb");
    if (!out_fp) {
        perror("Failed to open output file");
        return 1;
    }

    if (fwrite(decrypted, 1, decrypted_len, out_fp) != decrypted_len) {
        perror("Failed to write output file");
        fclose(out_fp);
        return 1;
    }
    fclose(out_fp);

    printf("Decryption successful, output written to %s\n",
output_file);
    return 0;
}
```

使用 GmSSL 编程对sn.txt进行签名验签，提交代码或代码链接，以及编译运行过程（文本或截图）（10 分）

- 签名

```
root@Youer:~/shiyan/test1210/sm2# nano sm2_sign.c
root@Youer:~/shiyan/test1210/sm2# gcc -o sm2_sign sm2_sign.c -lgmssl
root@Youer:~/shiyan/test1210/sm2# ./sm2_sign
Usage: ./sm2_sign <private_key.pem> <input.txt> <signature.sig>
root@Youer:~/shiyan/test1210/sm2# ls
de_output.txt  gen_sn    sm2.pem      sm2_decrypt.c  sm2_encrypt.c  sm2_gen.c
sm2_sign.c  sn.txt
en_output.bin  gen_sn.c  sm2_decrypt  sm2_encrypt    sm2_gen        sm2_sign
sm2pub.pem
root@Youer:~/shiyan/test1210/sm2# touch signature.bin
root@Youer:~/shiyan/test1210/sm2# ./sm2_sign sm2.pem  sn.txt signature.bin
Signature generated and saved to signature.bin
root@Youer:~/shiyan/test1210/sm2# xxd signature.bin
00000000: 3046 0221 0095 868e 70bb f29e eddc abe1  0F.!....p.......
00000010: 6f5c 5338 0a72 dc62 3a62 31ab 39af cb97  o\S8.r.b:b1.9...
00000020: 397e 034c 5e02 2100 cb0f a036 6ecf fe6d  9~.L^.!....6n..m
```

```
00000030: 3900 124b 4314 d3ea 4321 2744 9a6b 1901  9..KC...C!'D.k..
00000040: 993b 6a1c ea17 c7bc                      .;j.....
```

- 签名代码

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <gmssl/sm2.h>
#include <gmssl/pem.h>
#include <gmssl/error.h>
#include <gmssl/sm3.h>

int main(int argc, char **argv)
{
    if (argc != 4) {
        printf("Usage: %s <private_key.pem> <input.txt> <signature.sig>\n",
argv[0]);
        return 1;
    }

    const char *private_key_file = argv[1];
    const char *input_file = argv[2];
    const char *signature_file = argv[3];

    SM2_KEY sm2_key;
    FILE *key_fp = NULL;
    FILE *input_fp = NULL;
    FILE *sig_fp = NULL;
    unsigned char dgst[32];
    unsigned char sig[SM2_MAX_SIGNATURE_SIZE];
    size_t siglen;
    unsigned char buffer[1024];
    size_t len;

    // 加载私钥
    key_fp = fopen(private_key_file, "r");
    if (!key_fp) {
        perror("Failed to open private key file");
        return 1;
    }

    // 使用 PEM 格式加载私钥
    if (sm2_private_key_info_from_pem(&sm2_key, key_fp) != 1) {
        fprintf(stderr, "Failed to read private key from PEM\n");
        fclose(key_fp);
        return 1;
    }
    fclose(key_fp);

    // 读取输入文件并计算SM3哈希
    input_fp = fopen(input_file, "r");
```

```
    if (!input_fp) {
        fprintf(stderr, "Error opening input file: %s\n", input_file);
        return 1;
    }

    SM3_CTX sm3_ctx;
    sm3_init(&sm3_ctx);

    while ((len = fread(buffer, 1, sizeof(buffer), input_fp)) > 0) {
        sm3_update(&sm3_ctx, buffer, len);
    }
    fclose(input_fp);

    sm3_finish(&sm3_ctx, dgst);

    // 生成签名
    siglen = sizeof(sig);  // 确保我们为签名的大小传递正确的变量
    if (sm2_sign(&sm2_key, dgst, sig, &siglen) != 1) {
        fprintf(stderr, "Error generating SM2 signature\n");
        return 1;
    }

    // 写入签名到文件
    sig_fp = fopen(signature_file, "wb");
    if (!sig_fp) {
        fprintf(stderr, "Error opening signature file: %s\n",
signature_file);
        return 1;
    }

    if (fwrite(sig, 1, siglen, sig_fp) != siglen) {
        fprintf(stderr, "Error writing signature to file: %s\n",
signature_file);
        fclose(sig_fp);
        return 1;
    }
    fclose(sig_fp);

    printf("Signature generated and saved to %s\n", signature_file);
    return 0;
}
```

- 验签

```
root@Youer:~/shiyan/test1210/sm2# nano sm2_very.c
root@Youer:~/shiyan/test1210/sm2# gcc -o sm2_very sm2_very.c -lgmssl
root@Youer:~/shiyan/test1210/sm2# ./sm2_very
Usage: ./sm2_very <public key PEM file> <input file> <signature file>
root@Youer:~/shiyan/test1210/sm2# ./sm2_very sm2pub.pem  sn.txt signature.bin
Signature is valid.
```

- 验签代码

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <gmssl/sm2.h>
#include <gmssl/sm3.h>
#include <gmssl/pem.h>
#include <gmssl/error.h>

int main(int argc, char **argv) {
if (argc != 4) {
    fprintf(stderr, "Usage: %s <public key PEM file> <input file> <signature
file>\n", argv[0]);
    return 1;
}

const char *pubkey_filename = argv[1];
const char *input_filename = argv[2];
const char *signature_filename = argv[3];

SM2_KEY key;
FILE *fp;
SM3_CTX sm3_ctx;
uint8_t dgst[32]; // SM3摘要结果大小

// 读取公钥
if (!(fp = fopen(pubkey_filename, "r"))) {
    perror("Error opening public key file"); // 使用 perror 打印系统错误信息
    return 1;
}
if (!sm2_public_key_info_from_pem(&key, fp)) {
    fprintf(stderr, "Error reading public key from PEM file %s\n",
pubkey_filename);
    fclose(fp);
    return 1;
}
fclose(fp);

// 读取输入文件并计算哈希
if (!(fp = fopen(input_filename, "rb"))) {
    perror("Error opening input file"); // 使用 perror 打印系统错误信息
    return 1;
}

sm3_init(&sm3_ctx);
unsigned char buf[1024];
size_t len;
while ((len = fread(buf, 1, sizeof(buf), fp)) > 0) {
    sm3_update(&sm3_ctx, buf, len);
}
if (ferror(fp)) {
    fprintf(stderr, "Error reading input file %s\n", input_filename);
```

```
    fclose(fp);
    return 1;
}
sm3_finish(&sm3_ctx, dgst);
fclose(fp);

// 读取签名
unsigned char signature[256];
size_t siglen;
if (!(fp = fopen(signature_filename, "rb"))) {
    perror("Error opening signature file"); // 使用 perror 打印系统错误信息
    return 1;
}
siglen = fread(signature, 1, sizeof(signature), fp);
if (ferror(fp)) {
    fprintf(stderr, "Error reading signature file %s\n",
signature_filename);
    fclose(fp);
    return 1;
}
fclose(fp);

// 确保读取的签名长度有效
if (siglen == 0) {
    fprintf(stderr, "Signature file %s is empty or could not be read.\n",
signature_filename);
    return 1;
}

// 验证签名
int verify_result = sm2_verify(&key, dgst, signature, siglen);
if (verify_result == 1) {
    printf("Signature is valid.\n");
} else {
    fprintf(stderr, "Signature is invalid. Please check the public key,
input file, and signature file.\n");
    return 1;
}

return 0;
}
```