

# SM2 算法实现

Date: 2019.01.22-2019.02.25

Name: Chen Yuqi / Wang Lulu

## 1. 实验目的

掌握椭圆曲线密码算法 SM2 的算法原理

掌握 SM2 的算法流程和实现方法

## 2. 实验环境

Python 3.7.1 和 Python 3.6.4

## 3. SM2 算法详述

SM2 算法是由国家密码管理局发布的椭圆曲线公钥密码算法。

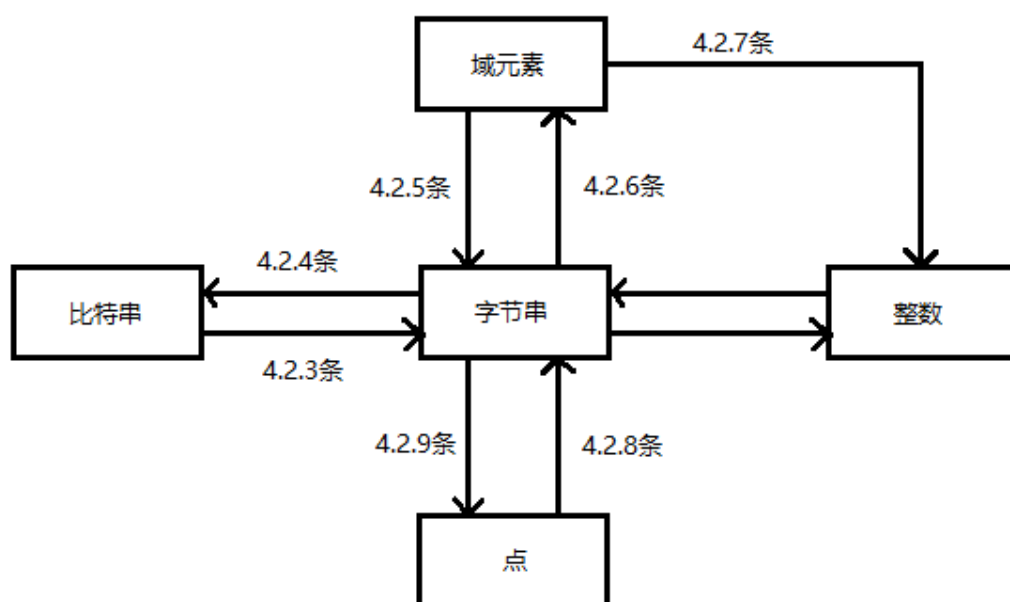
SM2 算法的实现包含两个部分，算法准备和基于 SM2 的公钥加密算法。

### 3.1 算法准备

在这一部分中，实现了 SM2 中所需要的一般技术，以帮助实现基于 SM2 公钥加密算法的密码机制。实现的内容包括数据类型的转换，有限域及椭圆曲线群的运算，椭圆曲线系统参数的验证以及密钥对的生成与公钥的验证。

#### 3.1.1 数据类型的转换

在 SM2 的算法实现中，数据类型包括比特串、字节串、域元素、椭圆曲线上的点和整数，因此会涉及到不同数据类型之间的转换。具体转换情况如下图所示：



除此之外，在转换过程中还涉及到字节串的三种表示形式，分别为压缩表示形式、未压缩表示形式和混合表示形式。表现形式可以为自定义选择，因此在接下来的实现过程中默认字节的表示形式为混合表示形式。

### 3.1.2 有限域及椭圆曲线群的运算

在 SM2 的算法实现中，运用到有限域的运算（有限域加法及有限域乘法）以及椭圆曲线群的运算（椭圆曲线加法）。

有限域加法是整数的模  $p$  加法，有限域乘法是整数的模  $p$  乘法。

二元域加法是比特串的异或，二元域的乘法是比特串模约化多项式的乘法。

椭圆曲线群加法是点与点相加，多倍点运算是同一个点的多次相加。

### 3.1.3 椭圆曲线系统参数的验证

由于椭圆曲线系统的安全性不依赖于系统参数的保密，因此，SM2 算法的实现不涉及到椭圆曲线系统参数的生成，但是却需要验证系统参数，本部分即介绍椭圆曲线系统参数的验证方法。

素域上椭圆曲线的系统参数包括域的规模，有限域中的两个元素和基点，基点的阶。每个参数都有其取值范围，因此验证方法即判断每个参数是否满足其取值条件。

二元扩域上椭圆曲线的系统参数包括域的规模，约化多项式，有限域中的两个元素和基点，基点的阶。每个参数都有其取值范围，因此验证方法即判断每个参数是否满足其取值条件。

### 3.1.4 密钥对的生成与公钥的验证

#### 密钥对的生成

输入：一个有效的椭圆曲线系统参数集合。

输出：与椭圆曲线系统参数相关的一个密钥对  $(d, P)$ 。

- 用随机数发生器产生整数  $d \in [1, n - 2]$  ( $n$  为椭圆曲线基点的阶)
- $G$  为基点，计算  $P = [d] G$
- 密钥对是  $(d, P)$ ，其中  $d$  为私钥， $P$  为公钥。

#### 公钥的验证

输入：一个有效的椭圆曲线系统参数集合和一个相关的公钥  $P$ 。

输出：若对于给定的系统参数， $P$  是有效的，则输出“有效”，否则输出“无效”。

- 验证  $P$  不是无穷远点  $O$ ；
- 验证  $P$  的坐标是域  $F_q$  中的元素；
- 验证  $P$  的坐标满足椭圆曲线方程；
- 验证  $[n] P = O$ ；
- 若通过所有验证则输出“有效”，否则输出“无效”。

## 3.2 基于 SM2 的公钥加密算法

在公钥加密算法中，除以上所提到的算法，还需三个辅助算法，即密码杂凑算法，密钥派生函数和随机数发生器。

在本算法中，密码杂凑算法使用的国际标准 SHA256 算法。密钥派生函数可以从一个共享的秘密比特串中派生出密钥数据，在输入共享的秘密比特串和所需密钥长度后，得到所需长度的密钥比特串，记为 KDF 函数。随机数发生器使用的是 randint() 随机数发生器。

接下来，详细介绍加解密过程。PB 为公钥，dB 为私钥。

### 3.2.1 加密算法

- Step1.** 用随机数发生器产生随机数  $k$ ， $k$  在区间  $[1, n-1]$  上
- Step2.** 计算椭圆曲线点  $C1 = [k]G = (x1, y1)$
- Step3.** 计算椭圆曲线点  $S = [h]PB$ ，判断  $S$  是否为无穷远点，若是，错误
- Step4.** 计算椭圆曲线点  $[k]PB = (x2, y2)$
- Step5.** 计算  $t = \text{KDF}(x2 \parallel y2, \text{klen})$ ，若  $t$  为全 0 比特串，返回 step1
- Step6.** 计算  $C2 = M$  异或  $t$
- Step7.** 计算  $C3 = \text{Hash}(x2 \parallel M \parallel y2)$
- Step8.** 得到密文  $C = C1 \parallel C2 \parallel C3$

### 3.2.2 解密算法

- Step1.** 从  $C$  中取出比特串  $C1$ ，转换为椭圆曲线上的点
- Step2.** 计算椭圆曲线点  $S = [h]PB$ ，判断  $S$  是否为无穷远点，若是，错误
- Step3.**  $[dB]C1 = (x2, y2)$
- Step4.** 计算  $t = \text{KDF}(x2 \parallel y2, \text{klen})$ ，若  $t$  为全 0 比特串，则错误
- Step5.** 从  $C$  中取出比特串  $C2$ ，计算  $M' = C2$  异或  $t$
- Step6.** 计算  $u = \text{Hash}(x2 \parallel M' \parallel y2)$ ，从  $C$  中取出比特串  $C3$ ，比较  $u$  和  $C3$ ，若  $u$  不等于  $C3$ ，则错误
- Step7.** 得到明文  $M'$

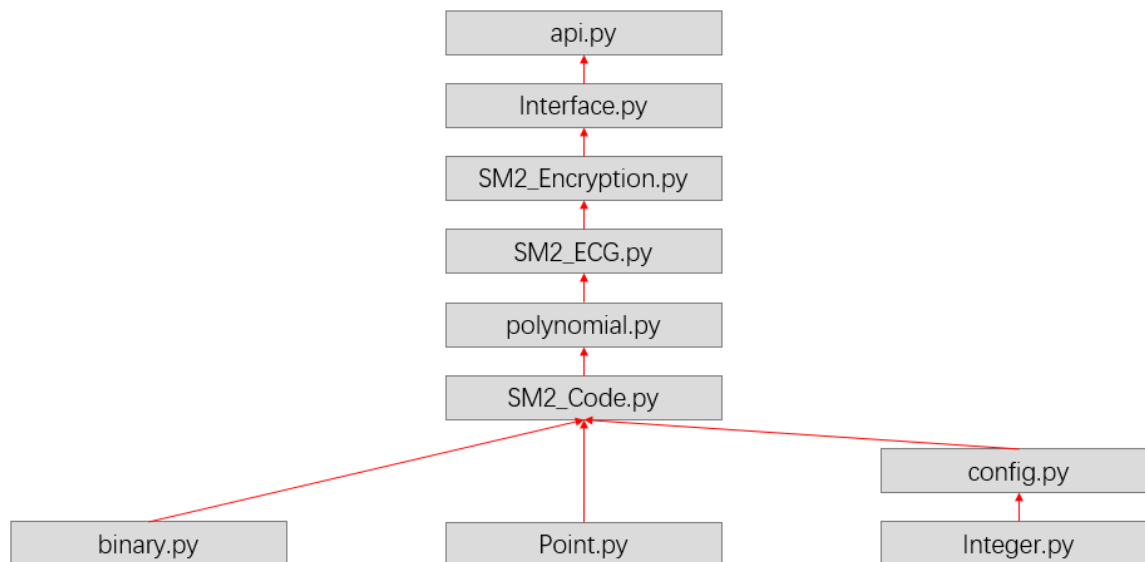
### 3.2.3 加解密验证

$[dB]C1 = [dB][k]G = [k]PB = (x2, y2)$ ，其中  $PB = [dB]G$ 。

由上可知， $M' = M$ 。

因此，若  $M' = M$ ，则有  $u = C3$ ，即解密过程的 step6 用来验证解密是否正确。

## 4. 代码实现



## 文件结构

文件结构如图所示

binary 进行比特串处理，Point 定义 点 数据类型，Integer 中包括快速模指数算法，Miller - Rabin 检测算法等数论算法。

config 存放椭圆曲线系统参数，提供设置和读取系统参数的接口。

SM2\_Code 实现比特串、字节串、整数、域元素、椭圆曲线点等数据类型之间的转换。

polynomial 实现比特串运算，包括加法、乘法、带余除法、取模。

SM2\_ECG 实现有限域和椭圆曲线群的运算以及 SM2 公钥生成和验证算法；有限域运算包括加法、求加法逆元、乘法、求乘法逆元、除法、快速幂运算；椭圆曲线群运算包括点相加、求二倍点、求多倍点。

SM2\_Encryption 实现 SM2 加解密算法，并通过 Interface 提供加解密字节数据的接口。

api 通过使用 Interface 的接口，实现加解密字符串和文件的功能供用户使用。

具体代码可从 <https://github.com/L1v1T/pySM2> 下载。

## 5. 实验结果

我们编写了用户测试代码来进行测试。

代码可从 [https://github.com/L1v1T/pySM2/blob/master/user\\_test.py](https://github.com/L1v1T/pySM2/blob/master/user_test.py) 下载。

**加解密字符串：**

- a) 首先使用内置的默认参数生成椭圆曲线和公私钥对；

- 对输入的“This is a test.”进行加密，并输出结果；
- 再对上一步的输出结果进行解密并输出结果。

```

Windows PowerShell
版权所有 (C) Microsoft Corporation。保留所有权利。

PS C:\Users\陈羽祺\Desktop\workspace\pySM2> cd 'c:\Users\陈羽祺\Desktop\workspace\pySM2'; ${env:PYPATH}=${env:PYPATH}; & "C:\Users\陈羽祺\AppData\Local\Programs\Python\Python37\python.exe" "c:\Users\陈羽祺\.vscode\extensions\ms-python.python-2019.1.0\pythonFiles\ptvsd_launcher.py" "--default" "--nodebug" "--client" "--host" "localhost" "--port" "3594" "c:\Users\陈羽祺\Desktop\workspace\pySM2\user_test.py"
输入参数文件路径 (空路径使用默认参数):
是否需要生成密钥对 (y/n): y 生成密钥
密钥对生成完毕
输入 s 加/解密字符串, 输入 f 加/解密文件, 输入 q 退出:s
输入字符串内容 (不输入表示使用上次计算结果作为输入): This is a test. 输入被加密字符串
输入 e 进行加密, 输入 d 进行解密, 输入 q 退出: e
加密结果: =/bJ83U6IU8>9Az;w0W-U48IITE,[#9u3]]0e^.=F[0á?A+ iY0Gj_«ER0=wqI<
- e||Kú0/0}A«. 加密后的内容
输入 s 加/解密字符串, 输入 f 加/解密文件, 输入 q 退出:s
输入字符串内容 (不输入表示使用上次计算结果作为输入):
输入 e 进行加密, 输入 d 进行解密, 输入 q 退出: d
解密结果: This is a test. 解密后的内容
  
```

### 加解密文件:

- 加密文件 yes , 生成 yes.sm2;
- 将 yes.sm2 重命名为 no.sm2;
- 解密 no.sm2, 生成解密文件 no。

```

输入 s 加/解密字符串, 输入 f 加/解密文件, 输入 q 退出:f
输入文件路径: yes
输入 e 进行加密, 输入 d 进行解密, 输入 q 退出: e
加密完毕
  
```

```

输入 s 加/解密字符串, 输入 f 加/解密文件, 输入 q 退出:f
输入文件路径: no.sm2
输入 e 进行加密, 输入 d 进行解密, 输入 q 退出: d
解密完毕
输入 s 加/解密字符串, 输入 f 加/解密文件, 输入 q 退出:q
  
```

```

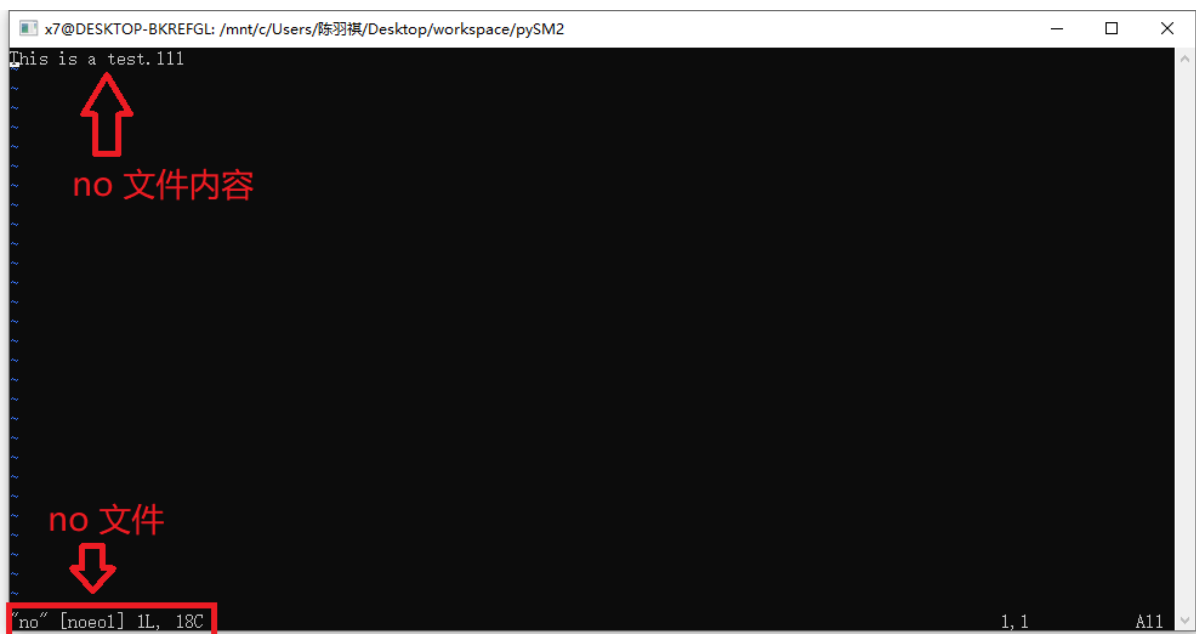
x7@DESKTOP-BKREFGL: /mnt/c/Users/陈羽祺/Desktop/workspace/pySM2
This is a test.111
yes 文件内容
yes 文件
yes" [noeol] 1L, 18C
  
```

加密前的 yes 文件



A terminal window titled 'x7@DESKTOP-BKREFGL: /mnt/c/Users/陈羽祺/Desktop/workspace/pySM2'. The terminal displays a large block of garbled, encrypted text. A red arrow points from the text 'yes.sm2(no.sm2) 文件内容' to the terminal output. At the bottom left, a red arrow points from the text 'yes.sm2 文件' to a red-bordered box containing the text 'yes.sm2" [noeol] 1L, 150C'. The bottom right of the terminal shows '1, 1' and 'A11'.

加密后生成的 yes.sm2(no.sm2)文件



A terminal window titled 'x7@DESKTOP-BKREFGL: /mnt/c/Users/陈羽祺/Desktop/workspace/pySM2'. The terminal displays the text 'This is a test.111'. A red arrow points from the text 'no 文件内容' to the terminal output. At the bottom left, a red arrow points from the text 'no 文件' to a red-bordered box containing the text 'no" [noeol] 1L, 18C'. The bottom right of the terminal shows '1, 1' and 'A11'.

no.sm2 解密后得到的 no 文件

## 6. 总结

通过这次的实验，我们清楚的了解了 SM2 的加解密原理，在编程过程中虽然出现了一些错误，但总归都解决了，通过代码的实现，使我们对算法有了更深的了解。

尽管 SM2 加解密已经完成，但是仍有很多方面可以更深一步研究，例如完成基于 SM2 的数字签名方案或者基于 SM2 的密钥交换协议，除此之外，实验完成的代码可以进一步封装来作为 SM2 算法的库文件。