**SM2算法实现**

**Date:** 2019.01.22-2019.02.22

**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。

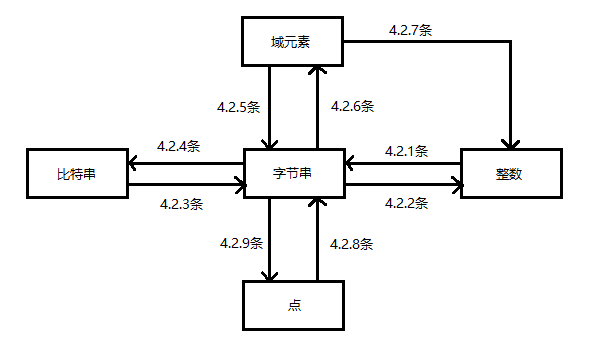
SM2算法的实现包含两个部分，算法准备，基于SM2的数字签名算法，基于SM2的密钥交换协议和基于SM2的公钥加密算法。

**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。

二元阔域加法是比特串的异或，二元阔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。

1. 用随机数发生器产生整数 （ 为椭圆曲线基点的阶）
2.  为基点，计算 
3. 密钥对是 ，其中  为私钥， 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 。

输出：若对于给定的系统参数， 是有效的，则输出“有效”，否则输出“无效”。

1. 验证  不是无穷远点 ；
2. 验证  的坐标是域  中的元素；
3. 验证  的坐标满足椭圆曲线方程；
4. 验证 ；
5. 若通过所有验证则输出“有效”，否则输出”无效“。

**3.2 基于SM2的数字签名算法**

在数字签名算法中，除以上所提到的算法，还需两个辅助算法，即密码杂凑算法和随机数发生器。此外，还需要用户的其他信息，称为Z。

在本算法中，密码杂凑算法使用的国际标准SHA256算法。随机数发生器使用的是randint()随机数发生器。

**3.2.1 数字签名的生成算法**

**Step1.** 置

**Step2.** 计算

**Step3.** 用随机数发生器产生随机数

**Step4.** 计算椭圆曲线点

**Step5.** 计算，若或则返回A3

**Step6.** 计算，若则返回A3

**Step7.** 消息M 的签名为

**3.2.2 数字签名的验证算法**

**Step1.** 检验是否成立，若不成立则验证不通过

**Step2.** 检验是否成立，若不成立则验证不通过

**Step3.** 置

**Step4.** 计算

**Step5.** 计算， 若，则验证不通过

**Step6.** 计算椭圆曲线点

**Step7.** 计算，检验是否成立，若成立则验证通过；否则验证不通过

**3.2.3 签名验证**



**3.3 基于SM2的密钥交换协议**

在密钥交换协议中，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。此外，还需要用户的其他信息，称为Z。

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。

**3.3.1 密钥交换协议**

设用户A和B协商获得密钥数据的长度为比特，用户A为发起方，用户B为响应方。 用户A和B双方为了获得相同的密钥，应实现如下运算步骤：（记）

**用户A：**

A1. 用随机数发生器产生随机数

A2. 计算椭圆曲线点

A3. 将发送给用户B

**用户B：**

B1. 用随机数发生器产生随机数

B2. 计算椭圆曲线点

B3. 计算

B4. 计算

B5. 验证是否满足椭圆曲线方程，若不满足则协商失败；若满足，计算

B6. 计算椭圆曲线点。若是无穷远点，则B协商失败

B7. 计算

B8. （选项）计算

B9. 将、(选项)发送给用户A

**用户A：**

A4. 计算

A5. 计算

A6.1 验证是否满足椭圆曲线方程，若不满足则协商失败；若满足，计算

A7.1 计算椭圆曲线点。若是无穷远点，则A协商失败

A8. 计算

A9. 计算，并检验是否成立，若等式不成立则从B到A的密钥确认失败

A10. (选项)计算，并将发送给用户B

**用户B：**

B10. (选项)计算，并检验是否成立， 若等式不成立则从A到B的密钥确认失败

**3.3.2 密钥交换协议验证**





由上可知，

又，





故，

**3.4 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。为公钥，为私钥。

**3.4.1 加密算法**

**Step1.** 用随机数发生器产生随机数，

**Step2.** 计算椭圆曲线点

**Step3.** 计算椭圆曲线点，判断是否为无穷远点，若是，错误

**Step4.** 计算椭圆曲线点

**Step5.** 计算，若为全0比特串，返回step1

**Step6.** 计算

**Step7.** 计算

**Step8.** 得到密文

**3.4.2 解密算法**

**Step1.** 从中取出比特串，转换为椭圆曲线上的点

**Step2.** 计算椭圆曲线点，判断是否为无穷远点，若是，错误

**Step3.** 

**Step4.** 计算，若为全0比特串，则错误

**Step5.** 从中取出比特串，计算

**Step6.** 计算，从中取出比特串，比较和，若不等于，则错误

**Step7.** 得到明文

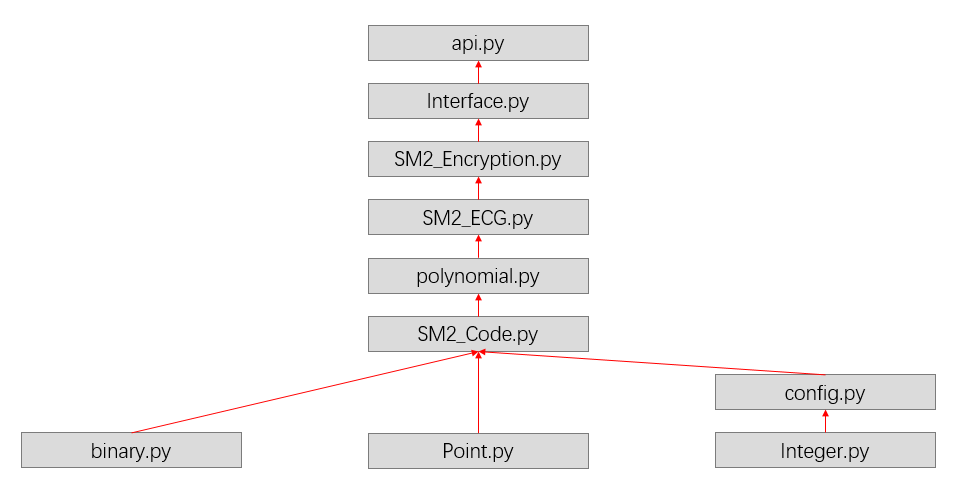
**3.4.3 加解密验证**

，其中。

由上可知，。

因此，若，则有，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。

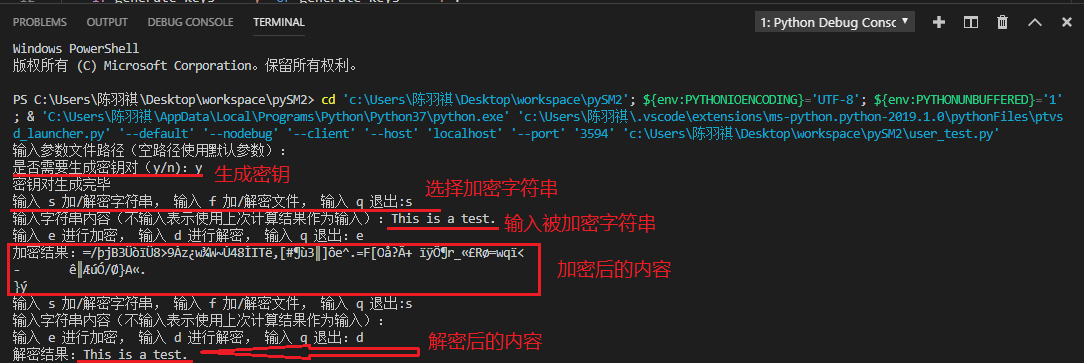
**5. 实验结果**

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> 下载。

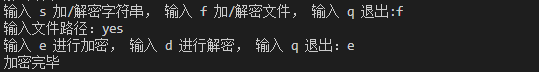
**加解密字符串**：

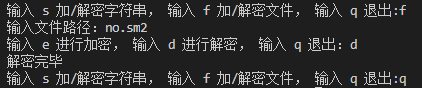
1. 首先使用内置的默认参数生成椭圆曲线和公私钥对；
2. 对输入的“This is a test.”进行加密，并输出结果；
3. 再对上一步的输出结果进行解密并输出结果。

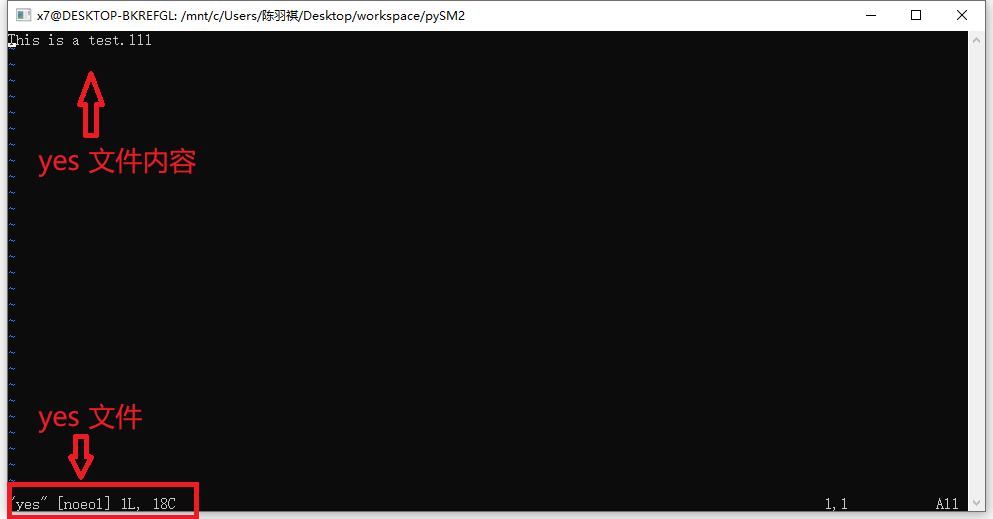


**加解密文件**：

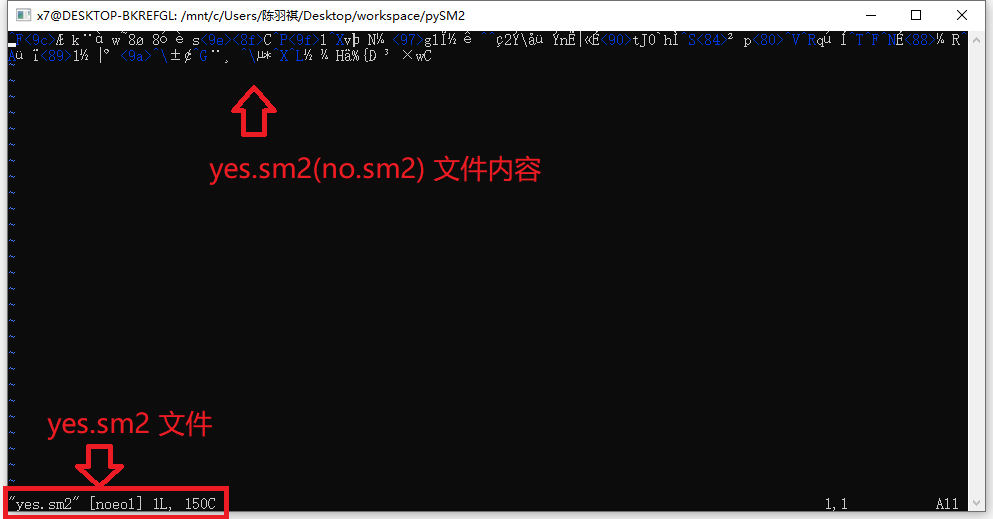
1. 加密文件 yes ，生成 yes.sm2；
2. 将 yes.sm2 重命名为 no.sm2；
3. 解密 no.sm2，生成解密文件 no。



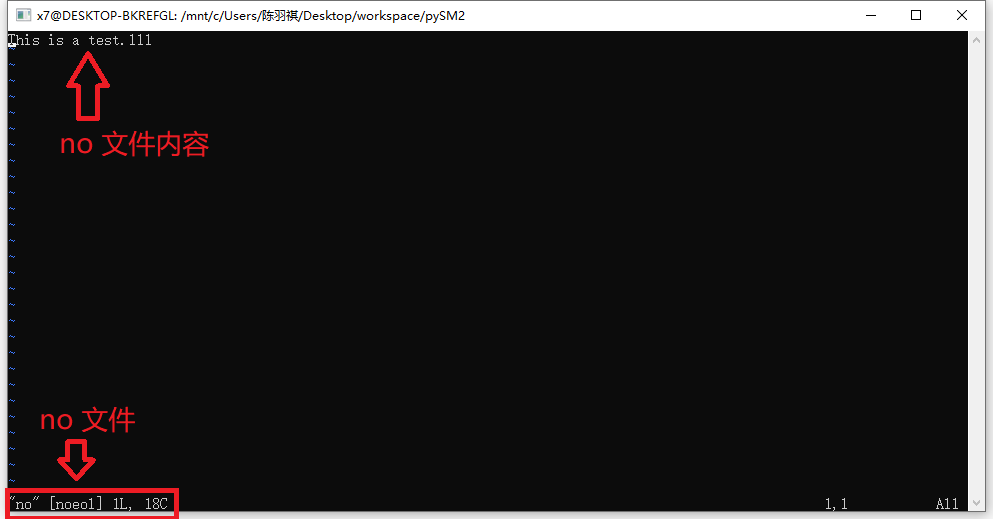




加密前的 yes 文件



加密后生成的 yes.sm2(no.sm2)文件



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。