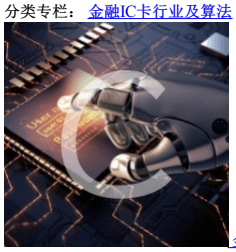
谈谈PBOC3.0中使用的国密SM2算法

转载

特立独行的猫a 于 2018-04-20 11:10:26 发布 1113 收藏 1
分类专栏： 金融IC卡行业及算法



金融IC卡行业及算法 专栏收录该内容
16 篇文章 1 订阅
订阅专栏

作者:小马

一 知识准备

SM2是国密局推出的一种他们自己说具有自主知识产权的非对称商用密码算法。本身是基于ECC椭圆曲线算法的，所以要讲sm2, 先要弄懂ECC。

完全理解ECC算法需要一定的数学功底，因为涉及到射影平面坐标系，齐次方程求解, 曲线的运算规则等概念。这里不做过多的数学分析(主要是我自己也没有完全整明白)。想要深入了解ECC的我推荐网名为ZMWorm 的大牛在多年前写的<<椭圆曲线ECC加密算法入门介绍>>。此人是早年看雪论坛中的一个版主，对算法和密码学很有研究。

本篇的主旨还是希望能以简单通俗的语言，讲清楚PBOC3.0认证过程中，所用到的SM2的相关概念，包括它的实现，使用等。

1 椭圆曲线到底是什么样的

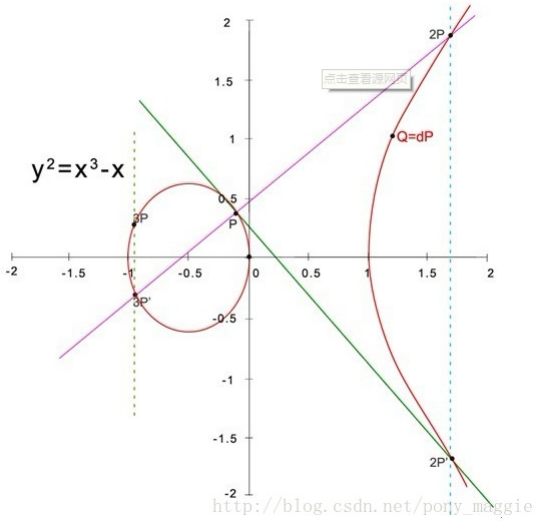


图1

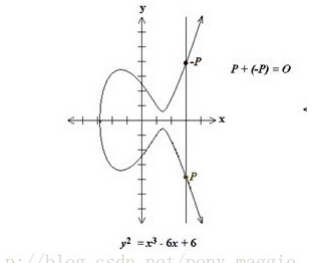


图2

上面是两个不同椭圆曲线在坐标系中的几何表示, 不过这个坐标系不是二维坐标系，而是射影坐标系。可以用空间思维想像一下(但是注意不是三维坐标系), 打个比方，你晚上站在一个路灯前面，地上有你的影子，你本身是在一个二维坐标系(把你想像成一个纸片)，和你的影子一起构成一个射影坐标系。

曲线的每一个点, 用三个参量表示, (X,Y,Z)。我们知道在二维坐标系里的每个图形都遵循一个方程，比如直接的二元一次方程是y=kx+b, 圆的方程是(x－a)2+(y－b)2=r2, 椭圆曲线在射影坐标系里也有自己的定义:

$$Y^2Z+a_1XYZ+a_3YZ^2=X^3+a_2X^2Z+a_4XZ^2+a_6Z^3$$

所有椭圆曲线上的点都满足上述方程，a1,a2,a3,a4,a6是系数，决定曲线的形状和位置。

二维坐标和射影坐标有一个对应关系，即x=X/Z, y=Y/Z, 这样就可以把上面的方程转成普通的二维坐标系方程:

$$y^2+a_1xy+a_3y= x^3+a_2x^2+a_4x+a_6$$

2 离散的椭圆曲线

上面的坐标系都是基于实数的，椭圆曲线看起来都是平滑的，如果我们限制曲线的点都必须是整数，曲线就变成离散的了，如图3所示:

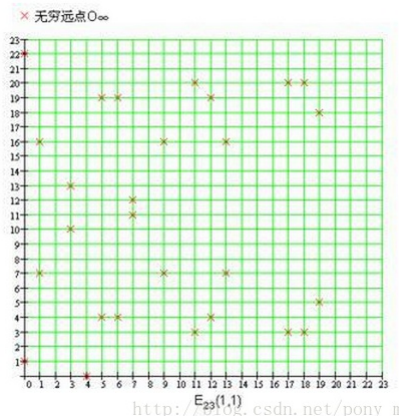


图3

再进一步限制，要求整数必须大于0, 小于某个大整数P, 这样就形成了一个有限域F_p.然后我们在这个有限域里定义一些点与点之间的加减乘除运算规则，比如A点加B点得到C点(记做A+B≡C (mod p))，或者A点乘以n得到K点(记做A×n≡K (mod p))。至于具体规则细节可以不用关心，只要知道有这样的操作即可。

选一条曲线，比如

$y^2=x^3+ax+b$

把它定义在F_p上, 要求a,b满足：

$4a^3+27b^2\neq 0 \pmod p$

我们把这样的曲线记为Ep(a,b)

加解密是基于这样的数学难题，K=kG，其中 K,G为Ep(a,b)上的点，k是整数，小于G点(注意区分，不是我们平常说的那个意思)的阶(不用关心什么是点的阶)。给定k和G，计算K很容易；但给定K和G，求k就困难了。这样，G就叫做基点，k是私钥，K是公钥。

最后总结。描述一条Fp上的椭圆曲线，有六个参量:
T=(p,a,b,G,n,h)。

p 、a 、b 用来确定一条椭圆曲线，
G为基点，
n为点G的阶，
h 是椭圆曲线上所有点的个数m与n相除的整数部分）

知识准备阶段知道这么多就可以了。

二 SM2在PBOC认证中的使用

1 签名和验签的原理

前面提到根据系数的不同，ECC曲线可以有很多，SM2使用其中一种，这就表明它的曲线方程,以及前面说到的六个参量都是固定的。根据国密局给出的规范定义如下：

```
y2=x3+ax+b

p=FFFFFFFE FFFFFFFFFFFFFFFF FFFFFFFF FFFFFFFF 00000000 FFFFFFFF FFFFFFFF
a=FFFFFFFE FFFFFFFFFFFFFFFF FFFFFFFF FFFFFFFF 00000000 FFFFFFFF FFFFFFFC
b=28E9FA9E 9D9F5E344D5A9E4B CF6509A7 F39789F5 15AB8F92 DDBCBD41 4D940E93
n=FFFFFFFE FFFFFFFFFFFFFFFF FFFFFFFF 7203DF6B 21C6052B 53BBF409 39D54123
Gx=32C4AE2C 1F198119 5F990446 6A39C994 8FE30BBF F2660BE1 715A4589 334C74C7
Gy=BC3736A2 F4F6779C 59BDCEE3 6B692153 D0A9877C C62A4740 02DF32E5 2139F0A0
```

这里容易引起一个误解，会认为参数都固定了，公私钥是不是只能有一对？当然不是，注意前面提到的K=kG的模型，K才是公钥，所以公钥其实是曲线在离散坐标系中，满足条件的一个曲线上的点。可以有很多个。另外，从这几个参量可以获知PBOC 3.0的公钥长度都是256位。

基于这种离散椭圆曲线原理的SM2算法一般有三种用法，签名验签，加解密，密钥交换。PBOC 3.0中的脱机数据认证只用到签名验签的功能。终端关心的是如何验签，卡片则要考虑如何实现生成签名。

2 基于openssl实现sm2

这里给出一个基于openssl的sm2实现，如果不了解openssl，可以先搜索一下相关知识，这里不讲解。openssl已经实现ECC算法接口，也就是核心已经有了，实现sm2其实并不难，关键是理解它里面各种接口如何使用。下面就分析一个终端验签的sm2实现。另外需要说明这里给出的只是代码片段，仅供理解用。

函数接口

```
int SM2_Verify(BYTE* Px,BYTE* Py, BYTE* DataIn,DWORD DataLen, BYTE* sigrs)
```

前两个参数是K的坐标值，也就是公钥。Datain和datalen分别是明文数据和其长度，最后一个参数是待验证的签名。

曲线的参数常量定义如下：

```

static const char *group_p ="FFFFFFFEFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF00000000FFFFFFFFFFFFFFFF";
static const char *group_a ="FFFFFFFEFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF00000000FFFFFFFFFFFFFFFC";
static const char *group_b ="28E9FA9E9D9F5E344D5A9E4BCF6509A7F39789F515AB8F92DDBCBD414D940E93";
static const char *group_Gx ="32C4AE2C1F1981195F9904466A39C9948FE30BBFF2660BE1715A4589334C74C7";
static const char *group_Gy ="BC3736A2F4F6779C59BDCEE36B692153D0A9877CC62A474002DF32E52139F0A0";
static const char *group_n = "FFFFFFFEFFFFFFFFFFFFFFFFFFFFFFFF7203DF6B21C6052B53BBF40939D54123";
static const char *ENTL_ID ="00803132333435363738313233343536373738";
#define SM2_KEY_LENGTH 32 //256位曲线

```

ENTL_ID是pboc规范中指定的用于SM3产生摘要的报文头数据。

```
strcpy(szBuff, ENTL_ID);
strcat(szBuff,group_a);
strcat(szBuff,group_b);
strcat(szBuff,group_Gx);
strcat(szBuff,group_Gy);
AscToBcd(szDataForDigest,(unsigned char *)szBuff, nLen);

      ….
```

```
9.
10. SM3(szDataForDigest,nLen+SM2_KEY_LENGTH*2, digestZA);
11.
12. memcpy(szDataForDigest,digestZA, ECC_LENGTH);
13. memcpy(szDataForDigest+ECC_LENGTH,DataIn, DataLen);
14.
15. SM3(szDataForDigest,DataLen+ECC_LENGTH, digestH);
```

第一步，对上述数据用sm3做摘要，摘要的结果与数据拼接后再做摘要。

```
[cpp] view plain copy
1. p = BN_new();
2. a = BN_new();
3. b = BN_new();
4. group = EC_GROUP_new(EC_GFp_mont_method());
5.
6. BN_hex2bn(&p, group_p))
7. BN_hex2bn(&a, group_a))
8. BN_hex2bn(&b, group_b))
```

这里是把定义的曲线常量转换成大数表式，这样才能使用openssl中的接口。

Group是ECC中的曲线组，它是ECC算法的核心，为什么这么说呢？ 因为这个group里的所有字段就确定了曲线的所有信息， 后面会看到，这里只是用EC_GROUP_new生成一个空的group，然后由p,a,b等参数来填充group，再以这个group为基础去生成曲线上的点。

```
[cpp] view plain copy
1. if (!EC_GROUP_set_curve_GFp(group, p, a, b,ctx))
2. {
3.                     gotoerr_process;
4. }
5.
6. P = EC_POINT_new(group);
7. Q = EC_POINT_new(group);
8. R = EC_POINT_new(group);
9. if (!P || !Q || !R)
10. {
11.         gotoerr_process;
12. }
```

这一段就确定了group的所有信息，并且根据group生成三个曲线上的点(点一定在曲线上，这个很重要)。

```
[cpp] view plain copy
1. <span style="white-space:pre">    </span>x = BN_new();
2.         y= BN_new();
3.         z= BN_new();
4.         if(!x || !y || !z)
5.         {
6.                     gotoerr_process;
7.         }
8.
9.         //Gx
10.        if(!BN_hex2bn(&x, group_Gx))
11.        {
12.                    gotoerr_process;
13.        }
14.
15.        if(!EC_POINT_set_compressed_coordinates_GFp(group, P, x, 0, ctx))
16.        {
17.                    gotoerr_process;
18.        }
19.
20.        if(!BN_hex2bn(&z, group_n))
21.        {
22.                    gotoerr_process;
23.        }
24.        if(!EC_GROUP_set_generator(group, P, z, BN_value_one()))
25.        {
26.                    gotoerr_process;
27.        }
28.
29.        if(!EC_POINT_get_affine_coordinates_GFp(group, P, x, y, ctx))
30.        {
```

```
31.                     gotoerr_process;
32.             }
```

这一段首先是设置n到group，n前面讲过，是曲线的阶。另外就是由G点坐标x,y确定点P，这里我其实也有点不太明白，G点坐标y本来就是已知的，为什么要再通过曲线变换表式生成y，是不是想要对比前面的group信息是否正确？只是为了校验？

```
[cpp] view plain copy

1.  if ((eckey = EC_KEY_new()) == NULL)
2.          {
3.                  gotoerr_process;
4.          }
5.          if(EC_KEY_set_group(eckey, group) == 0)
6.          {
7.                  gotoerr_process;
8.          }
9.
10. EC_KEY_set_public_key(eckey, P);
11.         if(!EC_KEY_check_key(eckey))
12.         {
13.                 gotoerr_process;
14.         }
15.
16.
17.         if(SM2_do_verify(1, digestH, SM2_KEY_LENGTH, signature, sig_len, eckey) != 1)
18.         {
19.                 gotoerr_process;
20.         }
```

这段比较好理解，生成公钥eckey,并由eckey最终验签。验签的执行函数sm2_do_verify有点复杂，这里不做过多的解释(其实是我解释不清楚，哇哈哈)，在一个国外的网站上找到一个比较好的描述，拿过来用用。

For Bob to authenticate Alice's signature, he must have a copy of her public key Q_A. If he does not trust the source of Q_A, he needs to validate the key (O here indicates the identity element):

1. Check that Q_A is not equal to O and its coordinates are otherwise valid
2. Check that Q_A lies on the curve
3. Check that $nQ_A = O$

After that, Bob follows these steps:

1. Verify that r and s are integers in $[1, n-1]$. If not, the signature is invalid.
2. Calculate $e = \text{HASH}(m)$, where HASH is the same function used in the signature generation. Let z be the L_n leftmost bits of e.
3. Calculate $w = s^{-1} \pmod n$.
4. Calculate $u_1 = zw \pmod n$ and $u_2 = rw \pmod n$.
5. Calculate $(x_1, y_1) = u_1 G + u_2 Q_A$.
6. The signature is valid if $r = x_1 \pmod n$, invalid otherwise.

3 脱机数据认证使用sm2的具体流程

我假设看这篇文章的人对PBOC 2.0中基于RSA国际算法的脱机数据认证流程已经比较了解，相关概念不再过多描述，重点关注二者的差异性。

另外，简单说一下sm3,因为下面会用到。sm3是一个类似hash的杂凑算法，即给定一个输入(一般很长)，产生一个固定长度的输出(sm3是32个字节，hash是20个字节)。它有两个特点：

1 不可逆性，即无法由输出推导出输入。

2 不同的输入，产生不同的输出。

下面就拿终端SDA认证卡片来看看sm2如何在pboc中使用的。

首先，发卡行公钥等数据(还包括公钥，算法标识，有效期等信息)被CA私钥签名(注意不是加密)生成发卡行公钥证书，这个证书会个人化到ic卡中。这些数据如下图所示：

表1　由认证中心签名的发卡行公钥数据（待签名数据）

字段名	长度	描述	格式
证书格式（记录头）	1	十六进制，值为‘12’	B
发卡行标识	4	主账号最左面的3~8个数字。（在右边补上十六进制数‘F’）	Cn 8
证书失效日期	2	MMYY，在此日期后，这张证书无效	N 4
证书序列号	3	由认证中心分配给这张证书的唯一的二进制数	b
发卡行公钥签名算法标识	1	标识发卡行公钥对应的数字签名算法。SM2 算法为‘04’。	b
发卡行公钥加密算法标识	1	标识发卡行公钥对应的加密算法，暂不使用，取值‘00’。	b
发卡行公钥参数标识	1	用于标识所用的椭圆曲线。见附录 A.4	b
发卡行公钥长度	1	标识发卡行公钥字节长度	b
发卡行公钥	N_I	SM2 公钥是椭圆曲线上的一个点	b

图4

签名其实就是对这个表里的数据做一系列运算，最终生成一个64字节的数据，分别由各32字节的r和s拼接而成（r和s只是个符号而已，没有特别意思）。这64字节的签名拼接到图4后面就是发卡行公钥证书。

这里与国际算法的公钥证书就有明显的区别了。国际算法证书是密文的，发卡行公钥用rsa加密。而国密的公钥证书完全是明文。用数据举例，下面这些数据来自pboc3.0的卡片送检指南,就是检测时要求个人化到卡里的数据。

```
【发卡行公钥】：
173A31DD681C6F8FE3BA6C354AD3924A4ADFD15EB0581BC1B37A1EB1C88DA29B47155F62FCF4CCCD201B134351A049D77E81F6A6C66E9CB32664F41348DA11F

【CA哈希值】(r||s)：
3499A2A0A7FED8F74F119B416FF728BA98EF0A32A36BCCB8D0110623D466425CA44C68F8E49121D9BFA9484CAEF9B476C5EB576D1A8DD6BC4A0986AF4134ABAF

【Tag_90 】(发卡行公钥证书)：
126228000112200000020400114017 3A31DD681C6F8FE3BA6C354AD3924A4ADFD15EB0581BC1B37A1EB1C88DA29B47155F62FCF4CCCD201B134351A049D77E86A6C66E9CB32664F41348
```

可以自己的解析一下，看看是否和上面表格中的数据一致，并且都是明文。

终端在读记录阶段获取发卡行公钥证书，国际算法需要用rsa公钥解密整个证书，然后验证hash,通过后取出发卡行公钥。而国密算法，终端只要用sm2公钥验64字节的签名，通过后直接取明文发卡行公钥，所以国密的验签的动作其实就相当于国际里的rsa解密和hash对比两个动作。

接着终端进行静态数据签名的验证，情况类似，对于国密，这些数据都是明文形式，后面拼接64字节的sm2签名，这64字节是用发卡行私钥对明文数据签名得到的。终端要做的就是拿刚刚获取的发卡行公钥对这64字节数据验证即可，验证通过就表示SDA通过。

四　PBOC为什么要选择国密

首先从技术角度，实现同样的计算复杂度，ECC的计算量相对RSA较小，所以效率高。RSA现在在不断的增加模长，目前都用到了2048位。并不是说现在一定要用2048位才是安全的，只是它的安全性更高，破解难度更大，这个要综合考虑，位数高也意味着成本高。有些不差钱的大公司比如谷歌就已经未雨绸缪的把2048位用在了它们的gmail邮箱服务中。PBOC3.0 认证中目前只用到1984位，其实也是相对安全的。

不过这个观点目前还存在争议。前段时间在清华大学听了一个关于密码算法的课，清华有个教授认为sm2并不见得比rsa更高级，只是sm2的原理比rsa难理解，所以大部分人认为它会相对安全些。一旦椭圆曲线被大家研究透了，sm2的光环也可能就此褪去。当然这个也是他个人的观点。

另外一个因素，要从国家战略的角度考虑，RSA之前一直被传与美国安全局合作，在算法中加入后门，这种事是宁可信其有的。国密算法咱起码是自己研发的东西,所有的过程细节都一清二楚，不用担收后门的事情。

央行现在非常重视国产安全芯片的推进工作，前些天央行的李晓枫还公开强调未来金融IC卡芯片要国产化，国密算法是其中很关键的一步。未来国产芯片加国密算法，才会有真正自主，安全的国产金融IC卡产品。