国密算法学习笔记

1.概述

国密算法是国家商用密码算法的简称。自 2012 年以来，国家密码管理局以《中华人民共和国密码行业标准》的方式，陆续公布了 SM2/SM3/SM4 等密码算法标准及其应用规范。其中“SM”代表“商密”，即用于商用的、不涉及国家秘密的密码技术。[1]

国密规范标准文件列表：http://www.gmbz.org.cn/main/bzlb.html

2.SM2

2.1. 简介

SM2 为基于椭圆曲线密码的公钥密码算法标准，包含数字签名、密钥交换和公钥加密，用于替换 RSA/Diffie-Hellman/ECDSA/ECDH 等国际算法。其

私钥长度：32字节。

公钥长度：SM2非压缩公钥格式字节串长度为65字节，压缩格式长度为33字节，若公钥y坐标最后一位为0，则首字节为`0x02`，否则为`0x03`。非压缩格式公钥首字节为`0x04`。

签名长度：64字节。

2.2. 椭圆曲线及参数

2.2.1 有限域

有限域 F_q，其中 q 是一个奇素数或是 2 的幂：

1. （素域）当 q 为奇素数时，$q = p$，表示为 F_p，其中 $p > 2^{191}$
2. （二元扩域）当 q 为 2 的幂，$q = 2^m$，表示为 F_{2^m}，其中 $m > 192$ 且为素数

2.2.2 椭圆曲线

1. F_p 上的椭圆曲线

   方程为：$y^2 = x^3 + ax + b$，$a, b \in F_p$，且$(4a^3 + 27b^2) mod\, p \neq 0$

   椭圆曲线的定义为：$E(F_p) = \{(x, y)|x, y \in F_p, y^2 = x^3 + ax + b\} \cup \{O\}$，O 是无穷远点

   椭圆曲线 $E(F_p)$ 上的点的数目称为椭圆曲线 $E(F_p)$ 的阶

2. F_{2^m} 上的椭圆曲线

   方程为：$y^2 + xy = x^3 + ax^2 + b$，$a, b \in F_p$，且$b \neq 0$

   椭圆曲线的定义为：$E(F_{2^m}) = \{(x, y)|x, y \in F_{2^m}, y^2 + xy = x^3 + ax^2 + b\} \cup \{O\}$，O 是无穷远点

   椭圆曲线 $E(F_{2^m})$ 上的点的数目称为椭圆曲线 $E(F_{2^m})$ 的阶

2.2.3 椭圆曲线参数

1. F_p 上的椭圆曲线

   image-20200221225055660

2. F_{2^m} 上的椭圆曲线

   image-20200221225110737

2.3 密钥对生成和公钥验证算法

2.3.1 密钥对生成

1. 输入：一个有效的 F_q 上的椭圆曲线系统参数
2. 步骤：

   a. 生成随机数 $d \in [1, n-2]$

   b. G 为基点，计算 $P(x_p, y_p)$ 点，$P = G^d$

   c. 生成密钥对是 (d, P)，其中 d 是私钥，P 是公钥

3. 输出：密钥对 (d, P)

2.3.2 公钥验证

1. 输入：

a. 一个有效的 F_q 上的椭圆曲线系统参数

b. 公钥 $P(x_p, y_p)$ 。

2. 步骤（验证公钥是否有效本质上就是在验证 P 点是否在椭圆曲线上）：

a. 验证 P 点不是无穷远点

b. 验证坐标 x_p 和 y_p 是否是区间 $[0, p-1]$ 内。

c. 验证 $y_p^2 = x_p^3 + ax_p + b(mod\,p)$ 或 $y_p^3 + x_p y_p = x_p^3 + ax_p^3 + b$

d. 验证 $p^n = O$

3. 输出：若以上验证均成功，则输出"有效"，否则为无效

2.4 数字签名算法

2.4.1. 用户身份杂凑值

1. 参数信息：

   假设用户 A 的身份标识为 ID_A，长度为 $ENTL_A$。

   椭圆曲线方程的参数为 a，b，基点 G 的坐标为 (X_G, Y_G)，用户 A 的公钥 P_A 的坐标为 (X_A, Y_A)。

2. 步骤：

   用户 A 的~~杂凑值~~ （Z值只是用来计算杂凑值的一个运算因子，并不是杂凑值）
   $Z_A = H_{256}(ENTL_A||ID_A||a||b||x_G||y_G||x_A||y_A)$

3. 参数说明：

   - $ENTL_A$：为2个字节标识的ID的比特长度。
   - ID_A：为用户身份标识。无特殊约定的情况下，用户身份标识ID的长度为16字节，其默认值从左到右依次为：`0x31,0x32,0x33,0x34,0x35,0x36,0x37,0x38,0x31,0x32,0x33,0x34,0x35,0x36,0x37,0x38`。
   - a, b：为系统曲线参数。
   - x_G, y_G：为基点；
   - x_A, y_A：为用户的公钥。

4. 预处理2是指使用Z值和待签名消息，通过SM3运算得到杂凑值H的过程。杂凑值用于SM2数字签名。

   - 输入：
     - Z_A：字节串，预处理2的输入。
     - M：字节串，待签名消息。
   - 输出：
     - H：字节串，杂凑值。
   - 计算公式：
     - $H = SM3(Z_A||M)$

2.4.2. 签名算法

1. 输入：

   a. 待签名的消息 M

   b. 用户 A 信息的杂凑值 Z_A

   c. 用户 A 的私钥 d_A

2. 步骤：

   a. 置 $\overline{M} = Z_A||M$

   b. 使用 SM3 计算杂凑值 $e = H_v(\overline{M})$

   c. 生成随机数 $k \in [1, n-2]$

   d. 计算椭圆曲线点 $G^k = (x_1, y_1)$

   e. 计算 $r = (e + x_1) mod\, n$，若 $r = 0$ 或 $r + k = n$，返回步骤 c

   f. 计算 $s = (\frac{k - r \cdot d_A}{1 + d_A}) mod\, n$，若 $s = 0$，返回步骤 c

3. 输出：

   签名消息为 (r, s)

2.4.3. 验签算法

1. 输入：

   a. 待签名消息 M'

   b. M' 的签名消息 (r', s')

   c. 用户 A 信息的杂凑值 Z_A

   d. 用户 A 的公钥 P_A

2. 步骤：

   a. 验证 $r' \in [1, n-1]$

   b. 验证 $s' \in [1, n-1]$

   c. 置 $\overline{M'} = Z_A||M'$

   d. 使用 SM3 计算杂凑值 $e = H_v(\overline{M'})$

   e. 计算 $t = (r' + s') mod\, n$，验证 $t! = 0$

   f. 计算 $(x', y') = G^{s'} + P_A^t$

   g. 计算 $R = (e' + x_1') mod\, n$，验证 $R = r'$

   > 化简：
   >
   > $G^{s'} + P_A^t = G^{s'} + G^{(d_A \ mod\ n) \cdot t'} = G^{(s' + d_A \cdot t') mod\ n} = G^{(s' + d_A \cdot (r' + s')) mod\ n}$
   >
   > $= G^{(s'(1 + d_A) + d_A \cdot r') mod\ n} = G^{(\frac{k - r' \cdot d_A}{1 + d_A}(1 + d_A) + d_A \cdot r') mod\ n}$
   >
   > $= G^{(k - r' \cdot d_A + r' \cdot d_A) mod\ n} = G^k$

3. 输出：

   若步骤 a，b，e，g 都验证通过，则输出验证成功，否则输出验证失败

2.4.4. 数字签名算法实现

1. `ec_param_new`初始化椭圆曲线，参数p,a,b是确定一条椭圆曲线的参数，n为基点G的阶。p是素数，一般是指有限域中F_P元素的个数。a,b确定一条椭圆曲线。对于基点 $G=(x_G,y_G)\in E(F_p)$，$G!=0$，对于参数n。椭圆曲线的定义为：$E(F_p)=\{(x,y)|x,y\in F_p,y^2=x^3+ax+b\}\cup\{O\}$，O 是无穷远点椭圆曲线 $E(F_p)$ 上的点的数目称为椭圆曲线 $E(F_p)$ 的阶。
2. `ec_param_init`初始化椭圆曲线参数。
3. `sm2_ec_key_new`和`sm2_ec_key_init`新建一个key和key的初始化，根据私钥d和基点G通过`xy_ecpoint_mul_bignum`函数生成私钥。
4. 初始化签名信息

2.5 密钥交换算法

密钥交换是指在用户 A，B 之间进行密钥交换协商的过程。用各自的私钥和对方的公钥来商定一只只有他们知道的秘密密钥。这个共享的秘密密钥通常用在某个对称密码学算法中，该密钥交换协议能够用于密钥管理和协商。

2.5.1 密钥交换算法

1. 相关参数

   a. 用户 A 的私钥 d_A，公钥 P_A

   b. 用户 B 的私钥 d_B，公钥 P_B

   c. 用户 A 的杂凑值 $Z_A=H_{256}(ENTL_A||ID_A||a||b||x_G||y_G||x_A||y_A)$

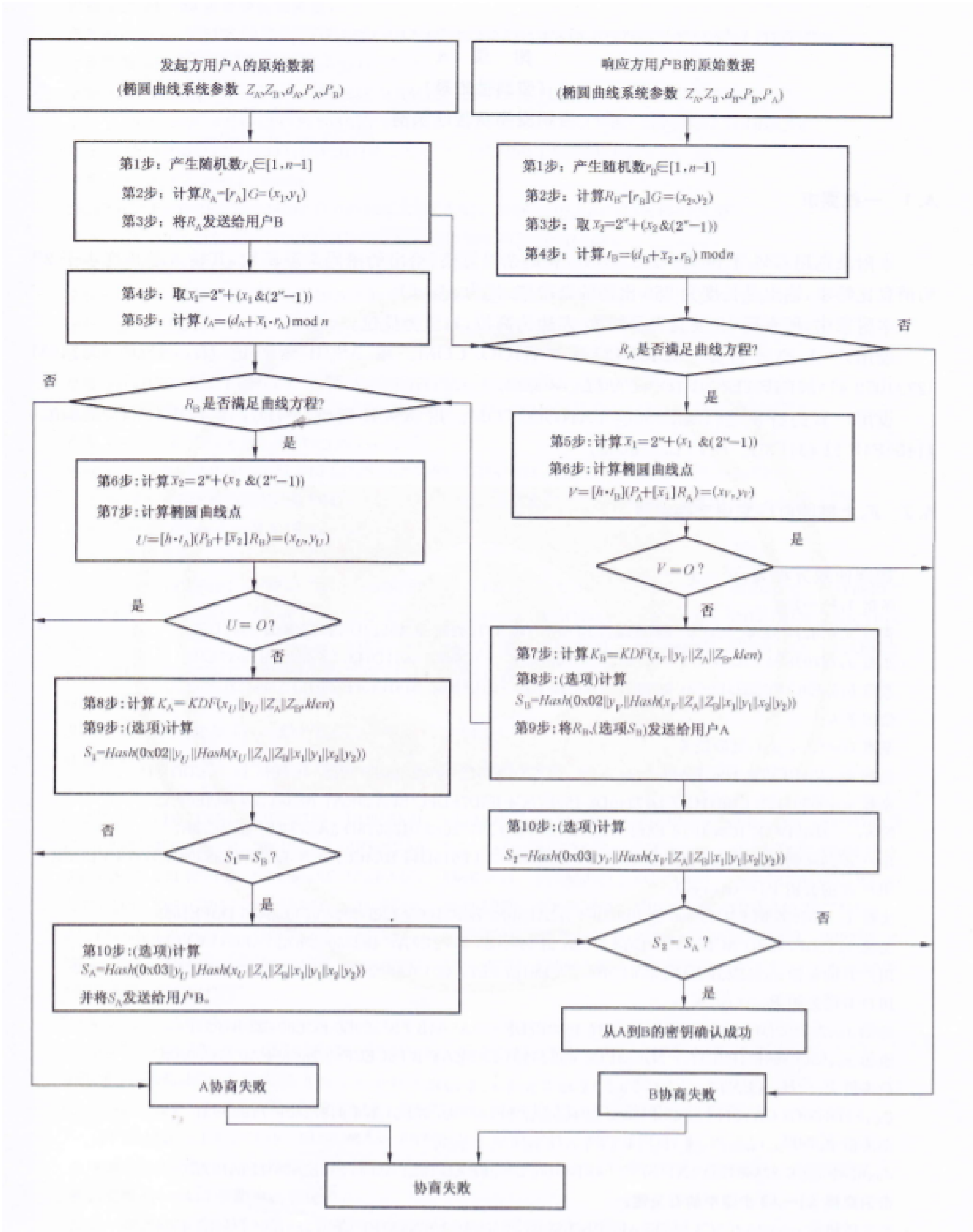
   d. 用户 B 的杂凑值 $Z_B=H_{256}(ENTL_B||ID_B||a||b||x_G||y_G||x_B||y_B)$

   e. $w=\lceil(\lceil log_2(n)\rceil/2)\rceil-1$

   f. h 为 n 的余因子

2. 步骤
   1. **用户 A**：产生随机数 $r_A\in[1,n-1]$，并计算 $R_A=G^{r_A}=(x_1,y_1)$，将 R_A 发送给 B。
   2. **用户 A**：计算 $\overline{x_1}=2^w+(x_1\ \&\ (2^w-1))$，代入计算 $t_A=(d_A+\overline{x_1}\cdot r_A)mod\ n$
   3. **用户 B**：产生随机数 $r_B\in[1,n-1]$，并计算 $R_B=G^{r_B}=(x_2,y_2)$
   4. **用户 B**：计算 $\overline{x_2}=2^w+(x_2\ \&\ (2^w-1))$，代入计算 $t_B=(d_B+\overline{x_2}\cdot r_B)mod\ n$
   5. **用户 B**：验证 R_A 是否满足曲线的方程，如果不满足，则协商失败，终止（参考 2.3.2 公钥验证）。
   6. **用户 B**：计算 $\overline{x_1}=2^w+(x_1\ \&\ (2^w-1))$，代入计算 $V=(P_A+R_A^{\overline{x_1}})^{h\cdot t_B}=(x_V,y_V)$，验证 $V\ ?=0$，如果是，则协商失败，终止。
   7. **用户 B**：计算 $K_B=KDF(x_V||y_V||Z_A||Z_B,klen)$，代入计算哈希值 $S_B=Hash(0x02||y_V||Hash(x_V||Z_A||Z_B||x_1||y_1||x_2||y_2))$，将 R_B 和 S_B 发送给用户 A
   8. **用户 A**：验证 R_B 是否满足曲线的方程，如果不满足，则协商失败，终止（参考 2.3.2 公钥验证）。
   9. **用户 A**：计算 $\overline{x_2}=2^w+(x_2\ \&\ (2^w-1))$，代入计算 $U=(P_B+R_B^{\overline{x_2}})^{h\cdot t_A}=(x_U,y_U)$，验证 $U\ ?=0$，如果是，则协商失败，终止。
   10. **用户 A**：计算 $K_A=KDF(x_U||y_U||Z_A||Z_B,klen)$，代入计算哈希值 $S_1=Hash(0x02||y_U||Hash(x_U||Z_A||Z_B||x_1||y_1||x_2||y_2))$
   11. **用户 A**：验证 $S_1\ ?=S_B$，若不一致，则协商失败，终止。
   12. **用户 A**：计算 $S_A=Hash(0x03||y_V||Hash(x_V||Z_A||Z_B||x_1||y_1||x_2||y_2))$，并将 S_A 发送给用户 B。
   13. **用户 B**：计算 $S_2=Hash(0x03||y_U||Hash(x_U||Z_A||Z_B||x_1||y_1||x_2||y_2))$，验证 $S_2\ ?=S_A$，若不一致，则协商失败，终止。
   14. 协商成功，K_A，K_B 即为协商的密钥，而后续的 S_1，S_2，S_A，S_B 验证则为可选项。

> 推导 $K_A = K_B$：
>
> $V = (P_A + R_A^{\overline{x_1}})^{h \cdot t_B} = (G^{d_A} + (G^{r_A})^{\overline{x_2}})^{h \cdot t_B} = (G^{d_A + \overline{x_2} \cdot r_A})^{h \cdot t_B} = (x_V, y_V)$
>
> $U = (P_B + R_B^{\overline{x_2}})^{h \cdot t_A} = (G^{d_B} + (G^{r_B})^{\overline{x_2}})^{h \cdot t_A} = (G^{d_B + \overline{x_2} \cdot r_B})^{h \cdot t_A} = G^{h \cdot t_A \cdot t_B} = (x_U, y_U)$



2.5.2 密钥交换算法实现

2.6 非对称加解密算法

2.6.1 密钥派生算法（KDF）

1. 杂凑函数

a. 密码杂凑函数 $H_v()$，其中 v 是杂凑值的长度

2. 输入：

a. 比特串 Z

b. 派生出来的密钥长度 $klen$

3. 步骤：

a. 设置计数器 ct=1

b. for $i = 1; k < \lceil klen/v \rceil; i + +\{$

$H_{a_i} = H_v(Z||ct)$

$ct + +$

}

c. 若 $klen/v$ 为整数，令 $Ha!_{\lceil klen/v \rceil} = Ha_{\lceil klen/v \rceil}$；

否则，$Ha!_{\lceil klen/v \rceil} = Ha_{\lceil klen/v \rceil}[0, klen - (v \times \lfloor klen/v \rfloor) - 1]$

d. 令 $K = Ha_1||Ha_2||\ldots||Ha_{\lceil klen/v \rceil - 1}||Ha!_{\lceil klen/v \rceil}$

4. 输出

密钥 K

2.6.2 加密算法

1. 输入：

a. 待加密的信息 M，长度为 $klen$

b. 用户 B 的公钥 P_B

2. 步骤：

a. 生成随机数 $k \in [1, n-1]$

b. 计算 $C_1 = G^k = (x_1, y_1)$

c. 计算 $S = P_B^h$，若 $S = O$，则报错（h 为 n 的余因子）

d. 计算 $P_B^k = (x_2, y_2)$

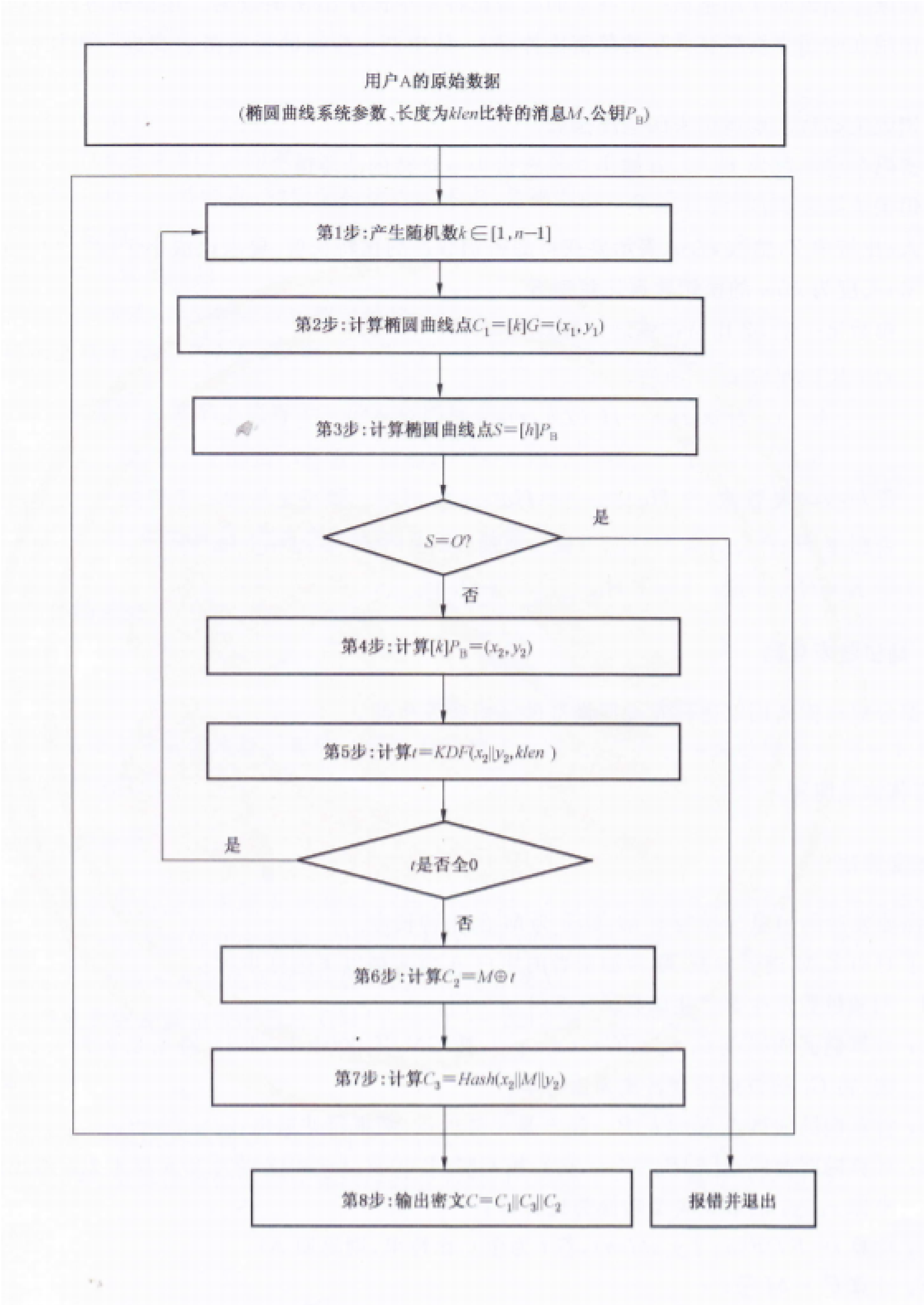
e. 派生密钥 $t = KDF(x_2||y_2, klen)$，若 t 为全 0 的比特串，返回步骤 a

f. 计算 $C_2 = M \oplus t$

g. 计算哈希值 $C_3 = Hash(x_2||M||y_2)$

3. 输出：

密文 $C = C_1||C_2||C_3$



2.6.3 解密算法

1. 输入：
   a. 密文 $C=C_1||C_2||C_3$
   b. 用户 b 的私钥 d_A
2. 步骤：
   a. 验证 C_1 是否满足曲线方程
   b. 计算 $S=C_1^h$，若 $S=O$，则报错
   c. 计算 $C_1^{d_B}=(x_2,y_2)$

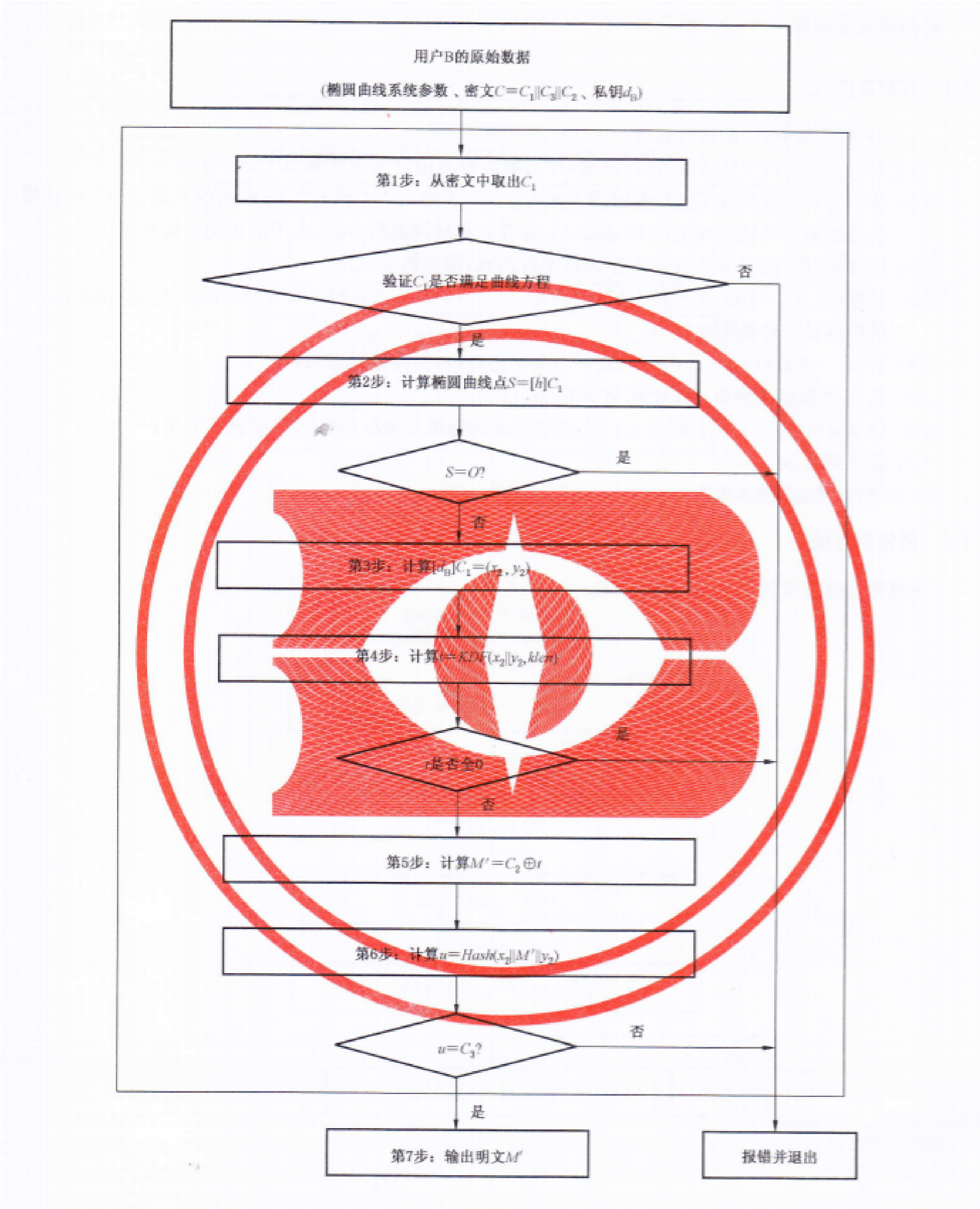
d. 计算 $t = KDF(x_2||y_2, klen)$，若 t 为全 0 的比特串，则报错

e. 计算 $M' = C_2 \oplus t$

f. 计算 $u = Hash(x_2||M'||y_2)$，验证 $u = C_3$，若验证失败则报错

3. 输出：

明文 M'



2.6.4 加解密算法实现

3.SM3

3.1 简介

SM3 为密码哈希算法，用于替代 MD5/SHA-1/SHA-256 等国际算法。

3.2 密码杂凑算法

密码杂凑算法的输入值 m，是长度为 l 的比特消息（$l < 2^{64}$）。

3.2.1 填充

1. 消息 m 后填充一位 1
2. 再填充 k 位的 0，使得 $(l+1+k) mod\ 512 = 448$
3. 再添加 64 位比特串，内容为 l 的二进制表示
4. 填充后的完整消息为 m'

image-20200219222124881

3.2.2 迭代压缩

1. 迭代处理

   a. 将 m' 以512位为单位进行分组，$m' = B_0||B_1||......||B_{n-1}$，其中 $n = (l+k+65)/512$

   b. 从 0 到 n-1 逐个进行迭代：$V_{i+1} = CF(V_i, B_i)$，其中 $CF()$ 为压缩函数，V_0 是一个初始值 IV，为一个固定值

   V_n 就是杂凑值

3.2.3 压缩函数CF()

1. 消息扩展

   a. 将 B_i 划分成16份，$B_i = W_0||W_1||W_2||...||W_{15}$

   b. for j=16; j<=67; j++ {

   $$W_j = P_1(W_{j-16}) \oplus W_{j-9} \oplus (W_{j-3} <<< 15) \oplus (W_{j-1} <<< 7) \oplus W_{j-6}$$

   }

   c. for j=0 ;j++; j<=63{

   $$W'_j = W_j \oplus W_{j+4}$$

   }

2. 压缩函数

   a. 令 $V_i = ABCDEF$

   b. for j=0;j<64;j++{

   $$SS1 = (A <<< 12) + E + (T_j <<< (j\ mod\ 32)) <<< 7$$

   $$SS2 = SS1 \oplus (A <<< 12)$$

   $$TT1 = FF_j(A, B, C) + D + SS2 + W'_j$$

   $$TT2 = GG_j(E, F, G) + H + SS1 + W_j$$

   $$D = C$$

   $$C = B <<< 9$$

   $$B = A$$

   $$A = TT1$$

   $$H = G$$

   $$G = F <<< 19$$

$F = E$

$E = P_0(TT2)$

}

c. 计算 $V_{i+1} = ABCDEFGH \oplus V_i$

4.SM4

4.1 简介

SM4 为分组密码，用于替代 DES/AES 等国际算法。

SM4 密码算法是一个分组算法，该算法的分组长度为 128 比特，密钥长度为 128 比特。加密算法和密钥扩展算法都采用 32 轮非线性迭代。

分组加密（英语：**Block cipher**），又称**分块加密**或**块密码**，是一种[对称密钥算法]。它将明文分成多个等长的模块（block），使用确定的算法和[对称密钥]对每组分别加密解密。

4.2 分组加密算法

对于每一个分组长度为 128 比特的明文分组，进行加密。

1. 参数说明：

   加密密钥：$MK = (MK_0, MK_1, MK_2, MK_3)$，其中$MK_i$ 为 32 比特，MK 总共为 128 比1特。

   轮密钥：$(rk_0, rk_1, \ldots, rk_{31})$

   明文：(X_0, X_1, X_2, X_3)，其中X_i 为 32 比特，X 总共为 128 比特。

   密文：(Y_0, Y_1, Y_2, Y_3)，其中Y_i 为 32 比特，Y 总共为 128 比特。

2. 输入：

   明文：(X_0, X_1, X_2, X_3)

3. 步骤：

   a. for i=0; i<32;i++{

   $X_{i+4} = F(X_i, X_{i+1}, X_{i+2}, X_{i+3}, rk_i) = X_i \oplus T(X_{i+1} \oplus X_{i+2} \oplus X_{i+3} \oplus rk_i)$

   }

   其中 $T()$ 为转置函数，$T(A) = L(\tau(A))$，

   假设$A = (a_0, a_1, a_2, a_3)$， $\tau(A) = (Sbox(a_0), Sbox(a_1), Sbox(a_2), Sbox(a_3))$

   image-20200221214611054

   $L(A) = A \oplus (A <<< 2) \oplus (B <<< 10) \oplus (B <<< 18) \oplus (B <<< 24)$

   b. 反序求得： $(Y_0, Y_1, Y_2, Y_3) = (X_35, X_34, X_33, X_32)$

4. 输出：

   密文：(Y_0, Y_1, Y_2, Y_3)

4.3 分组解密算法

分组解密算法和加密算法相同，区别只有 32 轮迭代时，轮密钥 rk_i 的使用顺序需要颠倒。

4.3.1 密钥扩展算法

1. 输入：

   加密密钥：$MK=(MK_0,MK_1,MK_2,MK_3)$，其中MK_i 为 32 比特，MK 总共为 128 比1特。

   $FK=(FK_0,FK_1,FK_2,FK_3)$ 为固定的系统参数，
   $FK_0=(A3B1BAC6)$，$FK_1=(56AA3350)$，$FK_2=(677D9197)$，$FK_3=(B27022DC)$

   $CK=(CK_0,CK_1,......,CK_{31})$ 为固定参数，$CK_i=(ck_{i,0},ck_{i,1},ck_{i,2},ck_{i,3})$，
   $ck_{i,j}=(4i+j)\times 7(mod\ 256)$

2. 步骤：

   $(K_0,K_1,K_2,K_3)=(MK_0\oplus FK_0,MK_1\oplus FK_1,MK_2\oplus FK_2,MK_3\oplus FK_3)$

   $rk_i=K_{i+4}=K_i\oplus T'(K_{i+1}\oplus K_{i+2}\oplus K_{i+3}\oplus CK_i)$

   $T'(A)=L'(\tau(A))$

   $L'(A)=A\oplus(A<<<13)\oplus(A<<<23)$

3. 输出：

   rk_i

5.SM9

5.1 简介

SM9 为基于身份的密码算法，可以替代基于数字证书的 PKI/CA 体系。通过部署国密算法，可以降低由弱密码和错误实现带来的安全风险和部署 PKI/CA 带来的开销。

6.总结

参考文献

[1] http://gmssl.org/