目录

大白话椭圆曲线加密算法

作者： 虞双齐

版本记录

日期	版本	作者	内容
2020 年 04 月 03 日	V1	虞双齐	初版
云共享版： https://kdocs.cn/l/spNSpCEr5?f=101			

你应该听过 ECC，ECDH 或者 ECDSA。ECC 是椭圆曲线加密算法（Elliptic curve cryptography）的简称，后面两个是基于它的算法实现。

在数字货币加密技术中，不得不谈 ECC，它是数字货币的安全基石。本文不涉及 ECC 中复杂的数学知识，笔者将努力使用简单通俗的语言来解释 ECC 是如何提供与保障加密安全的。

整篇文章，先讲解所涉及的理论基础知识，然后讲解 ECC 的定义，再通过实例来讲解 ECC 的加密解密原理和 ECDSA 的签名原理。对理论基础不了解的读者，请务必掌握理论基础后再继续往下看，不推荐跳读。

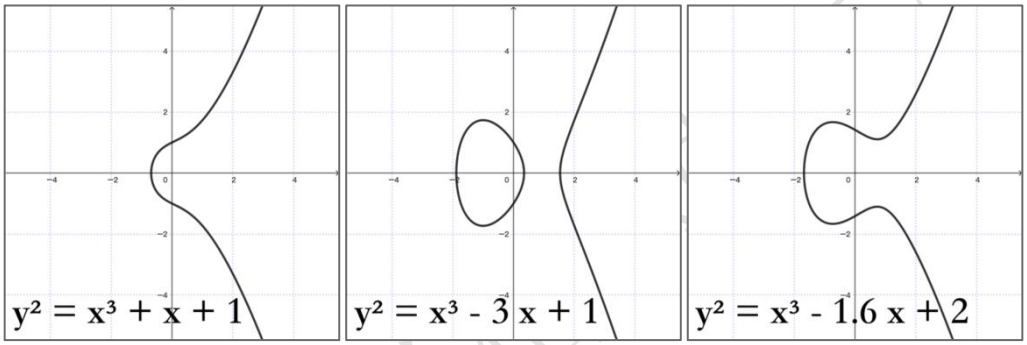
1.椭圆曲线理论基础

1.1. 定义

什么是椭圆曲线？在数学上，它是下方方程所有点的集合。

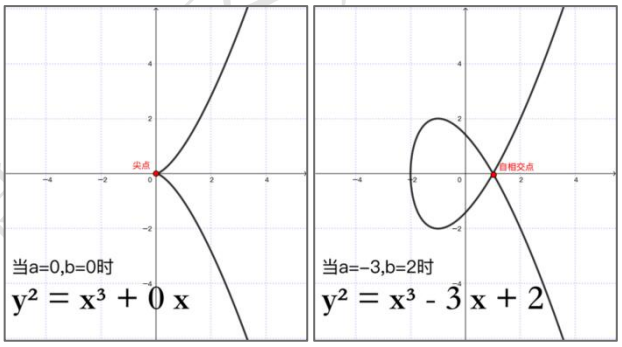
$$y^2 = x^3 + ax + b,\ 4a^3 + 27b^2 \neq 0$$

下图是不同 a,b 得到的不同图形的椭圆曲线。可以看到椭圆曲线的形状，并非椭圆的。只是因为椭圆曲线的描述方程，类似于计算一个椭圆周长的方程，故得此名。



随着 a,b 的不同(点击查看演示动图)，椭圆曲线也会在平面上呈现不同的形状。辨识度很高，可以看到椭圆曲线始终是关于 x 轴对称的。

方程后面的判别式 $4a^3 + 27b^2 \neq 0$ 是来保障曲线的光滑（非奇异），也就是说在曲线里面不能有尖点、自相交点和孤立点。从而保证椭圆曲线上所有的点都有切线。



上方曲线中他们的判别式均为 0。左曲线带尖点；右曲线曲线自相交。他们都不是有效的椭圆曲线。

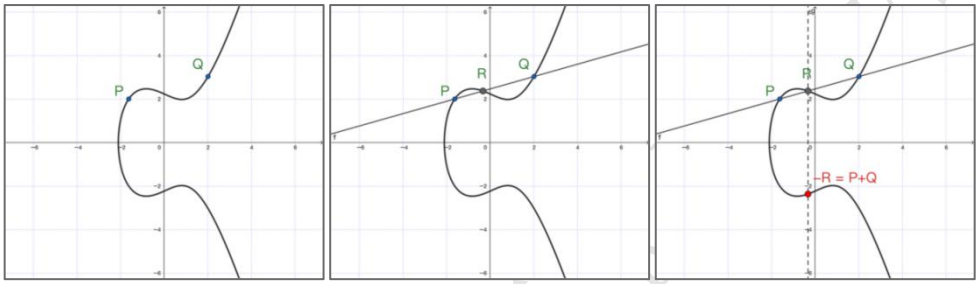
1.2. 四则运算

不同于代数几何中的四则运算，在椭圆曲线中有定义自有四则运算。

1.2.1.1. 加法运算

已知椭圆曲线上的点 P 和 Q，求 P+Q。

首先在曲线上标出点 P 和点 Q。再做经过点 P 和 Q 的直线，与曲线相交于第三点 R。继续做点 R 关于 X 轴的对称点-R。这个 -R 点就是点 P 和 Q 相加得到的点，记做 P+Q= -R。

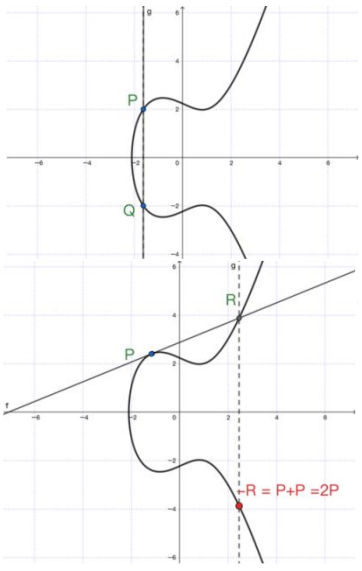


因椭圆曲线是关于 X 轴对称的，-R 是点 R 的对称点， 点 -R 必然也在椭圆曲线上，换句话说 P+Q 也在椭圆曲线上。

点 R 的存在对于 -R 至关重要。在大多数情况下直线一定将和椭圆曲线相交于第三点。如果直线和椭圆曲线没有第三个交叉点时，即点 R 不存在时，就无法按上面方法求 P+Q。以下情况中，点 R 不存在。

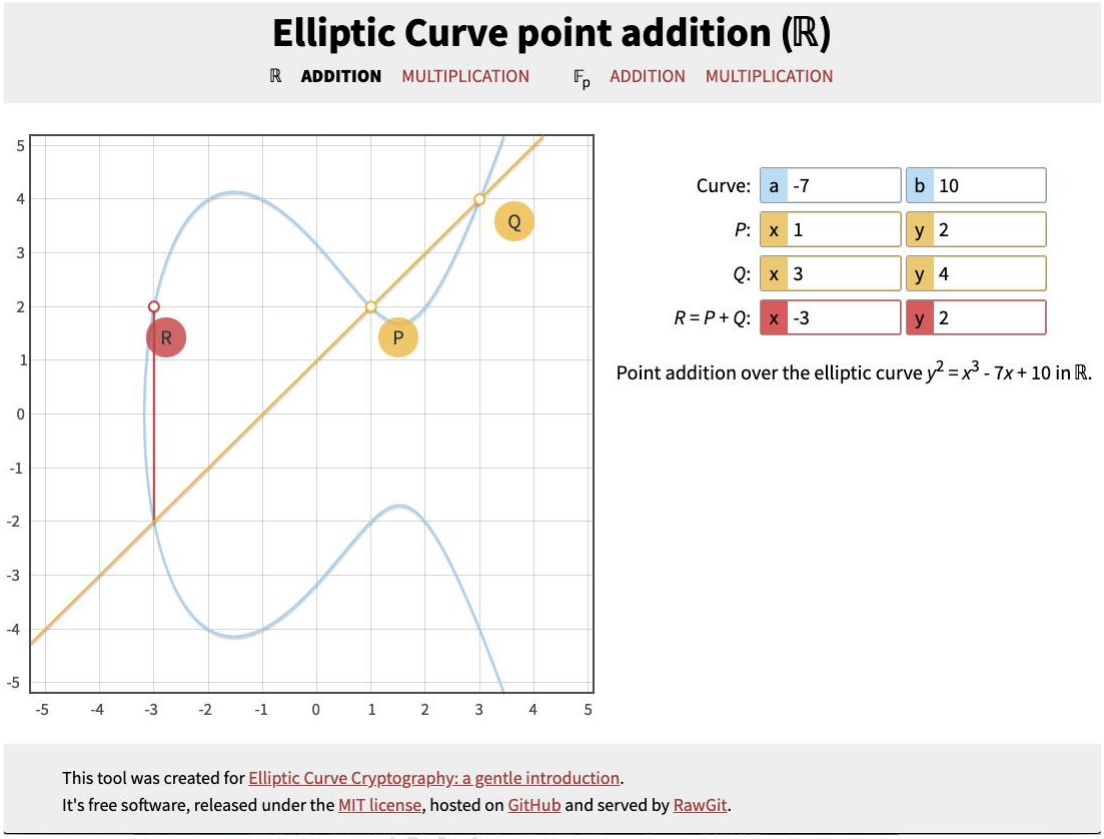
当 P = -Q 时，直线将于 Y 轴平行，没有第三个交点。假设 P+Q=O，此时 P +Q =P+(-P)=O。移项得到 P + O = P。

这意味着点 P 加上一个点 O 后还是自身点 P。我们把这个点 O 成为加法的单位元，曲线上的任意点与单位元相加不会改变其值。实际上点 O 表示平行线的相交点，称为无穷远点 O∞。可以理解为所有平行于 y 轴的直线交于 O 点。



当 P=Q 时，只能以点 P 作为切点做一条关于椭圆曲线的切线，切线将与椭圆曲线相加与另一点，记做点 R。同样，做点 R 的关于 X 轴的对称点-R。这个 -R 点就是点 P 和 P 相加得到的点，记做 P+P=2P=-R。

在图像上，我们可以非常便捷的找到 P+Q 点。而在数学上，该如何计算呢？可以通过代数算法来计算 P+Q，但这里不展开讲解，本文将借助网页工具《Elliptic Curve point addition (ℝ)》来进行求解。



1.2.1.2.减法运算

已知椭圆曲线上的点 P 和 Q，求 P-Q。

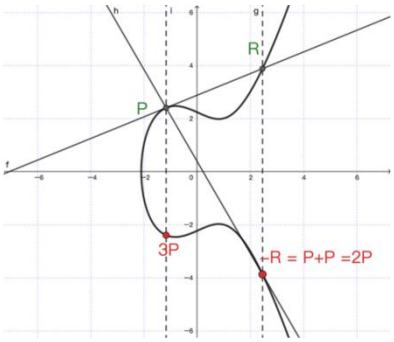
因 P -Q = P +(-Q)，求 P-Q 就是点 P 与点-Q 的加法运算。而点-Q 就是点 Q 对称点，在运算中称-Q 是 Q 的负元，Q(x,y)的负元是(x,-y)。综上，可得 P-Q 的计算公式如下：

$$P(x_1, y_1) - Q(x_2, y_2) = (x_1, y_1) + (x_2, -y_2)$$

1.2.1.3.乘法运算

已知椭圆曲线上的点 P 和数值 n，求 nP。

前面已计算出 P+P，同理可以继续计算出 P+P+P=2P+P。



重复 n 次加法运算就可以得到 n 次点 P 相加的结果，记为 nP 。我们把这个过程叫做 P 的自累。

$$nP = \underbrace{P + P + \cdots + P}_{n \text{ times}}$$

这就是乘法运算，比如 3P=P+P+P，通过进行 3 次加法运算就可以得到 3P 的值。

思考：如果 n 相当大，如何提高加法运算效率？

1.2.1.4.除法运算

已知椭圆曲线上的点 P 和数值 r，求 $r^{-1}P$。

注意，这里的 r^{-1} 并不是表示 r 的负一次方。在椭圆曲线中有如下定义。

$$r \cdot r^{-1} = O\infty$$

当已知 r 和单位元 0 的值即可求解出 r^{-1}，从而使用乘法求解 $r^{-1}P$。

假设已知椭圆曲线上的点 P(1, 4)和 r=4, 椭圆曲线单位元为 8。则有 $4 \cdot r^{-1}=8$, 得 $r^{-1}=2$, $r^{-1}P=2P$。

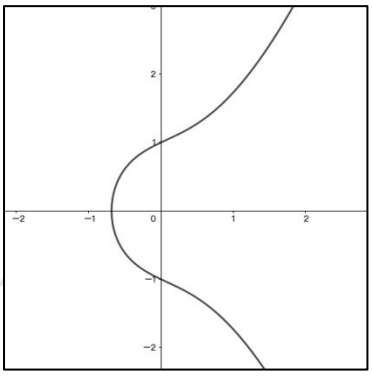
1.3. 有限域 GF(p)

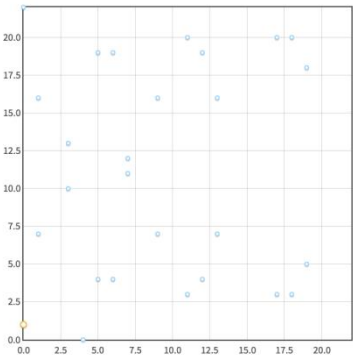
有限域是指有限个元素的域,记做 GF(p),其中 p 是质数，有{0,1,2,3...,p-1}共 p 个元素。例如 GF(7)={0 ,1,2 ,3 ,4 ,5 ,6 }。

GF(p)中的四则运算都是基于取模运算的，就是小学学过的带余除法的余数。例如：12除以 7，商为 1，余数为 5，记做 12 mod 7 =5 。

如果两个不同的数 a,b 除以同一个数 p 之后余数相同，称之为 a,b 模 p 同余，记 a mod p = b mod p。为了简便，我们把“mod p”统一写到最后面，并且为了防止与 “a=b”混淆，使用恒等符号“≡”，记 a ≡ b (mod p)。

而椭圆曲线（如 $y^2=x^3+x+1$）是定义在实数域上的，实数是连续的，导致了椭圆曲线的连续（右图）。这种连续性，使它并不合适用于安全加密。所以，可以把椭圆曲线定义在有限域上，使得曲线变成离散的点。下图是椭圆曲线取 p 为 2 所形成的离散的点。





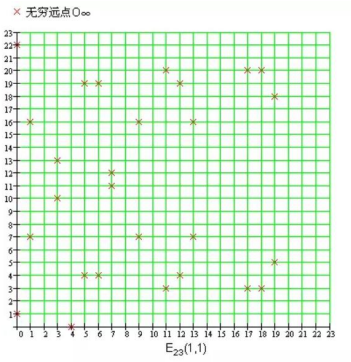
1.4. 有限域椭圆曲线

如前面所述，在有限域 F_p 上的椭圆曲线，可用于加密。按下定义有限域椭圆曲线：

1. 选择满足条件 $4a^3+27b^2 \neq 0 \pmod p$，且小于 p 的两个非负整数 a、b。
2. 满足方程 $y^2=x^3+ax+b \pmod p$ 的所有点(x,y)，再加上无穷远点 O∞，构成一条椭圆曲线。其中 x,y 属于 0 到 p-1 间的整数。

我们把这条椭圆曲线记为 $E_p(a,b)$，比如 $E_{23}(1,1)$ 方程如下，它的离散点如右图所示。

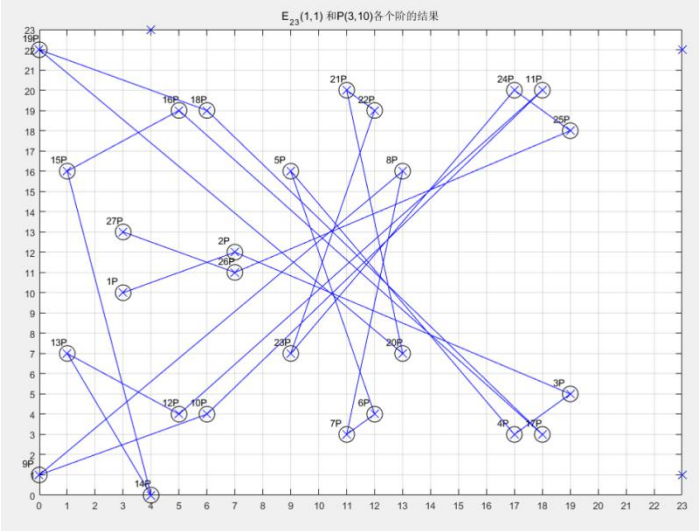
$$E_{23}(1,1) : y^2 \equiv x^3 + x + 1 \pmod{23}$$



1.5. 有限域椭圆曲线点的阶

如果椭圆曲线 $E_p(a,b)$ 上一点 P，存在最小的正整数 n 使得 $nP \equiv O\infty$,则将 n 称为 P 的阶。

若 n 不存在，则 P 是无限阶的。下图是 E23(1,1)和 P(3,10)各个阶的结果。从图可以明显看出，点的分布和顺序都是杂乱无章的。



计算可得 27P=-P=(3,13)，所以 28P=27P+P=-P+P=O∞，因此 $E_{23}(1,1)$ 上点 P(3,10)的阶为 29。

2. ECC 椭圆曲线加密算法

2.1. 定义

椭圆曲线加密算法，描述的是一条 F_p 的椭圆曲线，记为：T =(p,a,b,G,n,h)。

其中：

1. p,a,b 用于确定一条有限域椭圆曲线 $E_p(a,b)$；
2. G 为基点；
3. n 为 G 点的阶数；
4. h 是椭圆曲线上所有点的个数 m 与 n 相除的整数部分。

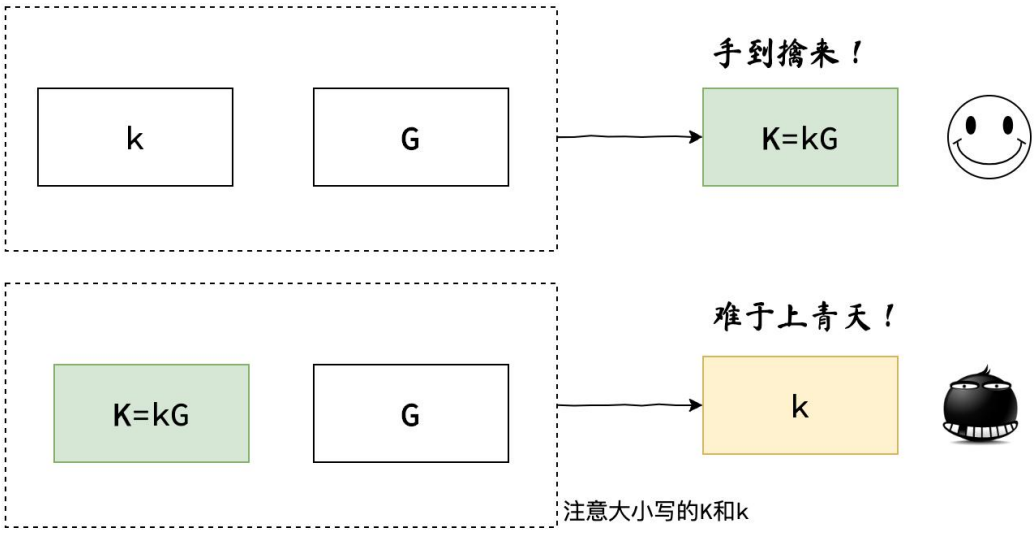
这几个参数取值，直接影响到加密的安全性。参数值一般满足以下条件：

1. p 当然越大越安全。但是越大，计算速度会变慢，200 位左右可以满足一般安全要求;
2. $p \neq nh$;
3. $pt \neq 1 \pmod n$, $t \in [1,20)$;
4. $4a^3+27b^2 \neq 0 \pmod p$;
5. n 为素数；

6. h≤4。

2.2.安全性

如果已知 k 和 G，根据乘法法则求 K 是很容易的，但反过来，仅已知 K 和 G，求 k 是非常困难的。

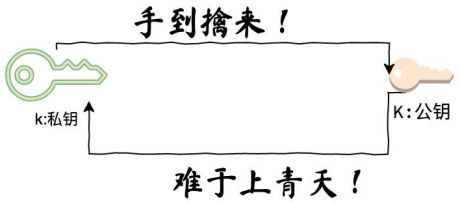


在普通代数里，比如说你有一个数 10，然后有人告诉你，他是用 5 做的乘法得到的 10，你就知道另一个乘数肯定是 2。 但是在 ECC 里中，想求 k 是非常困难的。

为了寻找 k ，你需要暴力枚举，直到找到了符合条件的点。以目前人类文明的计算能力这是不可能的，尤其当你使用了非常大的数值 n。

实际应用中的 ECC 原则上会把 k 取得相当大，n 也相当大，要把 n 个解点逐个计算出来是不可能的。这个逆向寻找 k 的过程的困难就是离散对数问题。这个问题的难度保证了 ECC 的安全性。

这里的 k 就是私钥(private key),K 是公钥(public key)。通过私钥计算公钥非常容易，但是希望通过公钥逆向推导私钥是非常困难的。



非常困难，不代表比可能。科学家们已经尝试通过量子计算机暴力枚举，寻找 k 值。但不需要过于担心，现有的量子计算机还是一个婴儿，科学们也在研究抗量子计算机破解的加密算法。

2.3.加密解密原理与演示

在 ECC 之前，已经有诸如 RSA 等非对称加密算法，但是 ECC 是一种更加优越的非对称加密算法。

假设情报员葛二蛋需要加密传递一份情报给延安总部。如果葛二蛋掌握了椭圆曲线加密算法，则他可以这样加密传递情报。

第一部分：延安总部告诉葛二蛋使用哪个公钥加密情报。

1. 延安总部选定一条椭圆曲线 $E_p(a,b)$ 和一个基点 G。假设选定 $E_{23}(4,20)$，基点 G(13,23)，G 的阶数 n=37。
2. 使用私钥 k，生成公钥 K=kG。假设 k=25，则 K=25G=(14,6)。
3. 将椭圆曲线 $E_{23}(4,20)$、n=37、K(14,6)、G(4,20)传递给葛二蛋。

第二部分：葛二蛋使用公钥 K 加密情报。

1. 葛二蛋将待传递的情报进行编码。假设编码结果为 3;
2. 在椭圆曲线 $E_{23}(4,20)$ 上，当 x=3 时到，y=28。把这点 M（3，28）当做情报编码在曲线上的映射;
3. 生成一个随机数 r （r<n），假设 r=6;
4. 计算点 C1= M + rK 和 C2=rG;
   ① C1= M+6K= M+6*25G=M+150G=（3,28）+(27,27)=(6,12);
   ② C2=rG=6G=(5,7);
5. 将点 C1(6,12)和 C2(5,7)传递给延安总部;

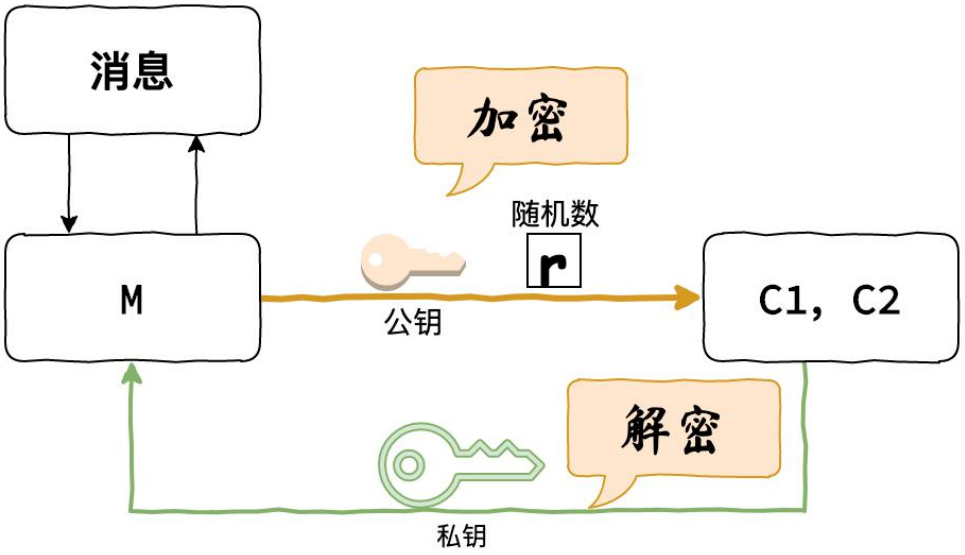
第三部分：延安总部解密情报。

延安总部收到 C1 和 C2 后，计算 C1-kC2，计算结果应该就是情报在曲线上的映射点 M。

C1 - kC2= C1- 25C2= (6,12) - 25(5,7)=(6,12) - (27,27)= (6,12) + (27,2)=（3，28）。

延安总部能解密出情报的原因是因为：

$$\begin{aligned} C1 - kC2 &= M + rK - krG \\ &= M + r(K - kG) \\ &= M + r(kG - kG) \\ &= M \end{aligned}$$



在这个加解密过程中，如果有一个偷窥者 H，他只能看到 Ep(a,b)、K、G、 C1、C2 。

通过 K= kG 求 k 或者通过 C2=rG 求 r 是相对困难的。因此 H 无法得到 A、B 件传送的明文消息。

需要强调的是，如果延安总部需要加密发送指令给葛二蛋，也可以采用同样的方式进行，唯一不同的是延安总部需要使用葛二蛋提供的公钥进行加密。只要双方各自的私钥不泄露，那么他们间的情报交流是安全的，即使落在别人手里，也无法解密。

2.4.ECDSA 数字签名原理

有一次，延安总部需要紧急通知所有情报员，“国共合作已宣告破裂，请务必保护和隐藏好自己”。情况紧急，延安总部如果逐个给所有情报员发送加密通知，则效率低下。延安总部可以将此通知刊登到报纸上，这样一早起来各地的情报员就可以看到。

可问题是，情报员怎么确信该通知真的出自延安总部呢？这里，我们就需要用到数字签名技术。

ECC 是加密算法，无法直接用于数字签名。在 ECC 之前就已经有数字签名（digital signature）技术，如 DSA。

椭圆曲线签名算法（ECDSA）就是使用 ECC 对 DSA 数字签名算法的模拟，最终签名可得到两个值 r 和 s。验证签名只需要求解一个值，判断值是否与 r 相同，如果相同，则说明是有效签名。

ECDSA 签名时，已知 Ep(a,b)、基点 G、G 的阶数 n。

第一部分：生成摘要

要签名的内容，实际上并非待签名数据的明文，而是对数据的哈希进行签名。

记做 $h = \mathrm{Hash}(\mathrm{message})$。

第二部分：生成签名

1. 生成随机数 k，k 是小于 n 的正整数。
2. 计算 $R_{(x,y)} = kG$
3. 计算 $r = x \bmod n$,如果 x mod n 为 0，则回到步骤 1 重试随机数。
4. 计算 $s = k^{-1}(h + rd_A) \bmod n$,其中 d_A 为私钥。这里的 k^{-1} 并不是 k 的-1 次方，而是表示 k 的逆元。在 ECC 中有 a•a-1 (mod n)=O∞=1
5. 签名结果就是 r,s。公开签名数据 message 和 r 和 s 值。

第三部分：验证签名

1. 计算 $u_1 = s^{-1}h \bmod n = \frac{h}{s} \bmod n$
2. 计算 $u_2 = s^{-1}r \bmod n = \frac{r}{s} \bmod n$
3. 计算点 R，$R = u_1G + u_2K$，其中 K 为签名者的公钥。
4. 判断点 R 是否等于 rG。如果两者相同，则说明签名合法。

验证中求解的 R 的计算过程如下：

$$R = \frac{h}{s}G + \frac{r}{s}K = \frac{hG + rK}{s}$$

为什么说要求解 R 就可以呢？这是因为：

$$\begin{aligned} R &= \frac{h}{s}G + \frac{r}{s}K \\ &= \frac{h}{s}G + \frac{r}{s}d_AG \\ &= \frac{h + rd_A}{s}G \\ &= \frac{h + rd_A}{\frac{h+rd_A}{k}}G \\ &= rG \end{aligned}$$

这里的关键是在于在签名时引入了随机数 k，提供了安全性。即使对于同一消息，只要改变随机数 k,所得到的签名也会随之变化。

2.5.签名演示

现在，延安总部将使用 ECDSA 签名通知刊登到报纸上。假设已知信息如下：

$$E_{29}(4,20): y^2 = x^3 + 4x + 20$$
$$n = 37,\ G = (2,6)$$
$$h = \mathrm{Hash}(m) = 88$$

- **延安总部签名**

1. 假设延安总部的私钥是 7，则 K=kG=7(2,6)=(3,28);
2. 生成随机数 k =11;
3. 计算 P=kG=11(2,6)=(16,2);
4. 计算 r= 16 mod 37 = 16;
5. 计算 s= k^{-1}(h+rk) mod 37 = k^{-1} (88+16*7) mod 37 = 200 k^{-1} mod 37。因 k•k-1 (mod 37)=11•k-1 (mod 37)=1,得 k-1 为 27。因此 s= 5400 mod 37 =35;
6. 在报纸刊登通知和签名结果(r,s);

- **情报员验证签名**

1. 计算 u1= s^{-1} *88 mod 37 = 18*88 mod 37 =30;
2. 计算 u2 = s^{-1} *16 mod 37 = 18 *16 mod 37 =29;
3. 计算 R= 30G +29K = (3,1)+(14,23)= (16,2);
4. 计算 16 mod 37 = 16 和 r 相等，说明签名合法;

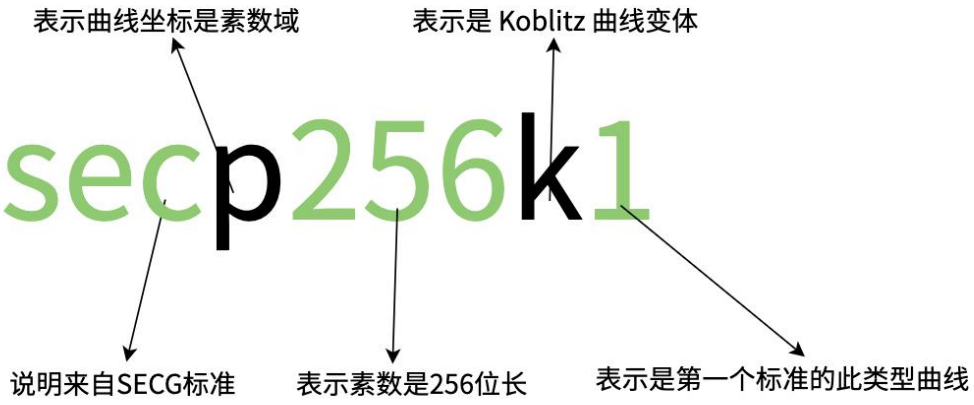
2.6.secp256k1

secp256k1 是高效密码组标准(SECG) 协会开发的一套高效的椭圆曲线签名算法标准。在比特币流行之前，secp256k1 并未真正使用过。自从比特币之后，secp2256k1 已成为数字货币中默认的数字签名算法。大多数常用的曲线具有一个随机的结构，但是 secp256k1 是为了更有效率的计算而构造了一个非随机结构。因此如果该算法的实现通过合理的优化，其计算效率可以比别的曲线快 30%以上。同时，与常用的 NIST 曲线不同，secp256k1 的常量是通过可预测的方式挑选的，这可以有效的减少防止曲线设计者安置后门的可能性。

secp256k1 是一条基于有限域的 Koblitz 椭圆曲线，T =(p,a,b,G,n,h)的各项参数定义如下：

1. p 是一个无限接近 2256 的大数;
   p = FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFE FFFFFC2F
   = 2256 - 232 - 29 - 28 - 27 - 26 - 24 - 1。
2. a=0,b=0;
3. G= 04 79BE667E F9DCBBAC 55A06295 CE870B07 029BFCDB 2DCE28D9 59F2815B 16F81798 483ADA77 26A3C465 5DA4FBFC 0E1108A8 FD17B448 A6855419 9C47D08F FB10D4B8
4. n = FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFE BAAEDCE6 AF48A03B BFD25E8C D0364141

5. h=01



图：secp256k1数据签名算法命名规则

为什么比特币要选择 secp256k1 签名算法而不是其他算法呢？比特币开发者社区曾讨论过 secp256k1 是否安全。中本聪没有明确解释，只是说道"有根据的推测"。

社区的讨论不外乎是在安全和效率上做权衡，选择一个不受任何政府控制、无后门的签名算法是比特币的首要考虑因素；其次，也需要提供计算速度，毕竟在比特币中签名与校验签名是不断在处理的事情（60%左右的 CPU 时间几乎全用在这上面），而具有可预测性、高计算效率特性的 Koblitz 曲线是不错的选择；

除椭圆曲线签名算法使用非常短的私钥和签名值外。基于安全第一，效率第二原则，secp256k1 就是一个优解。以太坊的签名算法也是采用 secp256k1。

3. 结语

椭圆曲线加密算法是一种非对称加密算法，比 RSA 具有更高的效率和更短私钥。有效解决"提高安全强度必须增加密钥长度"的工程实现问题。且已得到广泛的支持和使用。使用 ECC 作为加密算法已成为首先项。

4.关于作者

虞双齐是区块链技术 CTO，登录学院合伙人。

5.参考资料

1. 掘金《椭圆曲线加密原理与应用》
2. 掘金《椭圆曲线机密算法》
3. 博客园-ECC 椭圆曲线详解