

[19] 中华人民共和国国家知识产权局



[12] 发明专利说明书

专利号 ZL 200510115512.3

[51] Int. Cl.

G06F 7/72 (2006.01)

H04L 9/32 (2006.01)

[45] 授权公告日 2008 年 8 月 27 日

[11] 授权公告号 CN 100414492C

[22] 申请日 2005.11.4

[21] 申请号 200510115512.3

[73] 专利权人 北京浦奥得数码技术有限公司

地址 100016 北京市朝阳区酒仙桥路 14
号 51 号楼二层 AE26-28 室

[72] 发明人 范欣欣 王育民 詹 阳 姜正涛
谭示崇 田海博 袁素春 于松亮

[56] 参考文献

JP2003-216026A 2003.7.30

CN1280726A 2001.1.17

JP2001-265218A 2001.9.28

A Scalable Dual - Field Elliptic Curve Cryptographic Processor. Akashi satoh and Kohji Takanobu. IEEE TRANSACTIONS ON COMPUTERS, Vol. 52 No. 4. 2003

审查员 王 荣

[74] 专利代理机构 北京银龙知识产权代理有限公司

代理人 郝庆芬

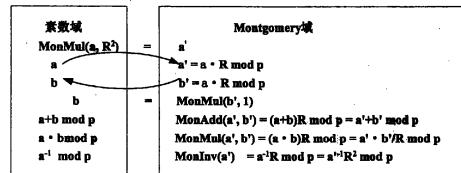
权利要求书 3 页 说明书 22 页 附图 2 页

[54] 发明名称

一种椭圆曲线密码系统及实现方法

[57] 摘要

一种椭圆曲线密码系统及实现方法，包括有限域算法模块，用于利用 Montgomery 算术实现大素数域中的运算；点加与倍点算法模块，实现椭圆曲线上 的点加运算与倍点运算；标量乘算法模块，用于 实现所述椭圆曲线密码系统的标量乘运算；DH 密 钥协商算法模块，用于调用标量乘算法模块，完成 密钥协商；数字签名与验证模块，用于调用标量乘 算法模块，完成对消息的数字签名与验证过程。其 提供了一种适合硬件实现的椭圆曲线密码系统及 实现方法，无需显式执行耗时的模归约运算，使其同 时适合软件与硬件实现。



1. 一种椭圆曲线密码系统，包括有限域算法模块，用于利用 Montgomery 算术实现大素数域中的运算；其特征在于，

还包括点加与倍点算法模块，用于在利用 Montgomery 算术实现有限域算法的基础上实现椭圆曲线上的点加运算与倍点运算，其调用有限域算法模块，输入参数为 Montgomery 域中元素，因而运算结果依然在 Montgomery 域中。

2. 根据权利要求 1 所述的椭圆曲线密码系统，其特征在于，所述的有限域算法模块包括模加运算模块、模减运算模块、模乘运算模块、模平方运算模块、模归约运算模块、求逆运算模块。

3. 根据权利要求 2 所述的椭圆曲线密码系统，其特征在于，所述的模乘运算模块为 CIOS—Montgomery 乘法算法模块。

4. 根据权利要求 3 所述的椭圆曲线密码系统，其特征在于，所述的模平方运算模块为 SRCIL—Montgomery 平方算法模块。

5. 根据权利要求 4 所述的椭圆曲线密码系统，其特征在于，所述的求逆运算模块为指数求逆算法模块。

6. 根据权利要求 1~5 任一项所述的椭圆曲线密码系统，其特征在于，所述点加运算与倍点运算为优化点加与倍点算法运算。

7. 根据权利要求 6 所述的椭圆曲线密码系统，其特征在于，还包括标量乘算法模块，用于实现所述椭圆曲线密码系统的核心运算：标量乘运算，其输入参数为素数域中元素，运算前转化为 Montgomery 域中，运算结果转化回素数域中。

8. 根据权利要求 7 所述的椭圆曲线密码系统，其特征在于，所述标量乘运算采用 NAF 随机标量乘算法。

9. 根据权利要求 8 所述的椭圆曲线密码系统，其特征在于，还包括 DH 密钥协商算法模块，用于调用标量乘算法模块，完成密钥协商，其输入参数为素数域中元素，运算前转化为 Montgomery 域中，运算结果转化回素数域中。

10. 根据权利要求 9 所述的椭圆曲线密码系统，其特征在于，还包括数字签名与验证模块，用于调用标量乘算法模块，完成对消息的数字签名与验证过程，其输入参数为素数域中元素，运算前转化为 Montgomery 域中，运算结果转化回素数域中。

11. 一种椭圆曲线密码系统的实现方法，其特征在于，包括下列步骤：
调用有限域算法模块，该模块利用 Montgomery 算术实现椭圆曲线密码系统中的大素数域中的运算；

调用点加与倍点算法模块，该模块在利用 Montgomery 算术实现有限域算法的基础上实现椭圆曲线上的点加运算与倍点运算；

所述点加与倍点算法模块调用有限域算法模块，输入参数为 Montgomery 域中元素，因而运算结果依然在 Montgomery 域中。

12. 根据权利要求 11 所述的实现方法，其特征在于，所述有限域算法模块利用 Montgomery 算术实现椭圆曲线密码系统中的大素数域中的模加运算、模减运算、模乘运算、模平方运算、模归约运算、求逆运算。

13. 根据权利要求 12 所述的实现方法，其特征在于，所述模乘运算为 CIOS·Montgomery 乘法算法。

14. 根据权利要求 13 所述的实现方法，其特征在于，所述的模平方运算为 SRCII·Montgomery 平方算法。

15. 根据权利要求 14 所述的实现方法，其特征在于，所述的求逆运算为指数求逆算法模块。

16. 根据权利要求 11~15 任一项所述的实现方法，其特征在于，所述点加运算与倍点运算为优化点加与倍点算法运算。

17. 根据权利要求 16 所述的实现方法，其特征在于，还包括下列步骤：调用标量乘算法模块，实现标量乘运算，其输入参数为素数域中元素，运算前转化为 Montgomery 域中，运算结果转化回素数域中。

18. 根据权利要求 17 所述的实现方法，其特征在于，所述标量乘运算采用 NAF 随机标量乘算法。

19. 根据权利要求 18 所述的实现方法，其特征在于，还包括下列步骤：调用 DH 密钥协商算法模块，通过标量乘算法模块，完成密钥协商，其

输入参数为素数域中元素，运算前转化为 Montgomery 域中，运算结果转化回素数域中。

20. 根据权利要求 19 所述的实现方法，其特征在于，还包括下列步骤：调用数字签名与验证模块，通过标量乘算法模块，完成对消息的数字签名与验证过程，其输入参数为素数域中元素，运算前转化为 Montgomery 域中，运算结果转化回素数域中。

一种椭圆曲线密码系统及实现方法

技术领域

本发明涉及电子设备的数字内容保护系统，特别是涉及一种椭圆曲线密码系统及实现方法。

背景技术

随着信息技术的不断发展和应用，电子信息的安全性问题变得越来越重要。而密码学作为信息安全的核心，在信息安全中扮演着极为重要的角色。密码学的发展中有两个重要的里程碑：1949年Shanon的“保密通信中的数学理论”和1976年Diffie和Hellman的“密码学的新方向”。Shanon的理论是对密码学的研究变成具有一定理论系统的科学；“密码学的新方向”一文提出了公开密钥密码的思想，开创了公钥密码学的新纪元。公钥密码提出后，立刻受到了人们的普遍关注。从1976年以来，人们提出了大量公钥密码体制的实现方案。所有这些方案的安全性都是基于求解某个数学难题的，截至目前为止，有如下三类既具有一定安全性又易于实现的数学难题：

1. 基于大数分解问题(IFP)的公钥密码系统，其中包括RSA体制和Rabin体制；
2. 基于有限域上离散对数问题(DLP)的公钥密码体制，其中主要包括ElGamal类加密体制和签名方案，Diffie-Hellman密钥交换方案，Schnorr签名方案和Nyberg-Ruppel签名方案等；
3. 基于椭圆曲线离散对数问题(ECDLP)的公钥密码体制，其中主要包括椭圆曲线型的Diffie-Hellman密钥交换方案，椭圆曲线型的MQV密钥交换方案；椭圆曲线型的数字签名算法。

将椭圆曲线用于密码算法，于1985年由Koblitz和Victor Miller分别独立地提出。它问世以来一直是密码分析学的研究对象。现在，在商业和政府的用途中，椭圆曲线密码系统(ECC)都被认为是安全的。根据已知的密码分析学知识，椭圆曲线密码系统相比于传统的密码系统来说提供了更高的安全性。

例如，相比基于 *RSA* 和 *ELGamal* 的加密与数字签名算法和基于 *Diffie-Hellman* 密钥协商算法来说，椭圆曲线密码系统具有更短的密钥和更有效的算法。而这两方面的优势使得椭圆曲线密码系统比起传统的密码系统来说更实用，且能广泛用于存储量和计算量受限的环境下，如移动电话和个人数字助理。同样的原因使得椭圆曲线密码系统也倍受高要求系统的青睐，如安全网络设备（这些设备经常用到公钥运算）。

椭圆曲线密码系统（*ECC*）相对于 *RSA* 和 *DSA* 等系统吸引人的最主要原因是其安全性是基于有限域上的椭圆曲线上的点群中的离散对数问题 *ECDLP*，*ECDLP* 是比因子分解问题更难的问题，许多密码专家认为它是指数级的难度，即解决其数学问题——椭圆曲线离散对数问题（*ECDLP*）的已知最好的算法也要用完全指数时间。与之相比，*RSA* 和 *DSA* 等其它公钥密码系统所基于的数学问题——因数分解问题（*IFP*）和离散对数问题（*DLP*）都是亚指数时间算法，与 *RSA* 和 *DSA* 等系统相比，椭圆曲线密码系统具有如下更好的优势：

1. 安全性更高

密码算法的安全性能通过该算法的抗攻击强度来反映。表1.1描述了椭圆曲线密码系统和其它几种公钥密码算法抗攻击强度的比较。我们可以看到，与其他公钥算法相比，椭圆曲线密码系统抗攻击性具有绝对的优势。如160 bit 的椭圆曲线密码系统可提供与1024 bit 的 *RSA*、*DSA* 相当的安全强度，而210 bit 的椭圆曲线密码系统则与2048 bit *RSA*、*DSA* 具有相同的安全强度。

表1.1 *ECC* 与 *RSA/DSA* 抗攻击性能比较

<i>RSA/DSA</i> 密 钥长度(比特)	<i>ECC</i> 密钥长 度(比特)	<i>RSA</i> 和 <i>ECC</i> 密 钥长度比率	所需工作量 <i>MIPS</i> 年
512	106	5:1	10^4
768	132	6:1	10^8
1024	160	7:1	10^{11}
2048	210	10:1	10^{20}
21,000	600	35:1	10^{78}

其中 *MIPS* 年表示每秒钟能运行一百万次指令的计算机运行一年的工作量

2. 计算量小，处理速度快

虽然在 *RSA* 中可以通过选取较小的公钥(如 3)的方法提高公钥处理的速度，既提高加密和签名验证的速度，使其在加密和签名验证上与椭圆曲线密码系统有可比性，但在私钥的处理速度上(解密和签名)椭圆曲线密码系统比 *RSA*、*DSA* 要快。表 1.2 描述了椭圆曲线密码系统与 *RSA*、*DSA* 在密钥生成，签名和验证速度的比较。

表1.2 ECC 与RSA/DSA计算速度比较

	<i>ECDSA</i> $GF(2^n)$ standard	<i>ECDSA</i> $GF(2^n)$ improved	<i>ECDSA</i> $GF(p)$	<i>RSA</i>	<i>DSA</i>
key generation	13.0	11.7	5.5	1s	22.7
signature	13.3	11.3	6.3	43.3	23.6
verification	68	60	26	0.65	28.3

其中，*ECC* 算法为 191 bit，*RSA* / *DSA* 为 1024 bit。随着安全强度的增加，*ECC* 比 *RSA* / *DSA* 运算的速度提高的更快。

3. 存储空间少

椭圆曲线密码系统的密钥尺寸和系统参数与 *RSA* / *DSA* 相比要小得多，意味着它所占的存储空间要少得多。这对于密码算法在 *IC* 卡和无线环境中的应用具有特别重要的意义。

4. 带宽要求低

当对长消息进行加解密时，三类密码系统有相同的带宽要求，但应用于短消息时，椭圆曲线密码系统带宽要求要低得多。带宽要求低使椭圆曲线密码系统在无线网络中具有广阔的应用前景。

由上面的优点我们可以看到，随着计算能力的提高需要密钥长度的增加，椭圆曲线密码系统相对其它公钥密码系统更吸引我们的关注。其每比特更高的安全性所带来的优点包括：更高的速度，更低的能量消耗，节约带宽，提高存储效率。这些优点在一些对于带宽、处理器能力或存储有限制的应用中显得尤为重要。

发明内容

本发明的目的在于提供一种椭圆曲线密码系统及实现方法，为高层认证协议的实现提供底层模块。

为实现本发明的目的而提供的一种椭圆曲线密码系统，包括：（一）有限域算法模块，用于利用 Montgomery 算术实现大素数域中的运算；还包括点加与倍点算法模块，用于在利用 Montgomery 算术实现有限域算法的基础上实现椭圆曲线上的点加运算与倍点运算，其调用有限域算法模块，输入参数为 Montgomery 域中元素，因而运算结果依然在 Montgomery 域中。

所述的有限域算法模块包括模加运算模块、模减运算模块、模乘运算模块、模平方运算模块、模归约运算模块、求逆运算模块。

所述的模乘运算模块为 CIOS—Montgomery 乘法算法模块。

所述的模平方运算模块为 SRCIL—Montgomery 平方算法模块。

所述的求逆运算模块为指数求逆算法模块。

所述点加运算与倍点运算为优化点加与倍点算法运算。

本发明的椭圆曲线密码系统还包括：

（二）标量乘算法模块，用于实现所述椭圆曲线密码系统的核心运算：标量乘运算，其输入参数为素数域中元素，运算前转化为 Montgomery 域中，运算结果转化回素数域中。

所述标量乘运算采用 NAF 随机标量乘算法。

（三）DH 密钥协商算法模块，用于调用标量乘算法模块，完成密钥协商，其输入参数为素数域中元素，运算前转化为 Montgomery 域中，运算结果转化回素数域中。

（四）数字签名与验证模块，用于调用标量乘算法模块，完成对消息的数字签名与验证过程，其输入参数为素数域中元素，运算前转化为 Montgomery 域中，运算结果转化回素数域中。

本发明还提供一种椭圆曲线密码系统的实现方法，包括下列步骤：

（一）调用有限域算法模块，该模块利用 Montgomery 算术实现椭圆曲线密码系统中的大素数域中的运算；

调用点加与倍点算法模块，该模块在利用 Montgomery 算术实现有限域算法的基础上实现椭圆曲线上的点加运算与倍点运算；所述点加与倍点算法

模块调用有限域算法模块，输入参数为 Montgomery 域中元素，因而运算结果依然在 Montgomery 域中。

所述有限域算法模块利用 Montgomery 算术实现椭圆曲线密码系统中的大素数域中的模加运算、模减运算、模乘运算、模平方运算、模归约运算、求逆运算。

所述模乘运算模块为 CIOS—Montgomery 乘法算法。

所述的模平方运算为 SRCII.—Montgomery 平方算法。

所述的求逆运算模块为指数求逆算法模块。

所述点加运算与倍点运算为优化点加与倍点算法运算。

本发明的椭圆曲线密码系统实现方法，还包括下列步骤：

(二) 调用标量乘算法模块，实现标量乘运算，其输入参数为素数域中元素，运算前转化为 Montgomery 域中，运算结果转化回素数域中。

所述标量乘运算采用 NAF 随机标量乘算法。

(三) 调用 DH 密钥协商算法模块，通过标量乘算法模块，完成密钥协商，其输入参数为素数域中元素，运算前转化为 Montgomery 域中，运算结果转化回素数域中。

(四) 调用数字签名与验证模块，通过标量乘算法模块，完成对消息的数字签名与验证过程，其输入参数为素数域中元素，运算前转化为 Montgomery 域中，运算结果转化回素数域中。

本发明的有益效果是：本发明提供了一种适合硬件实现的椭圆曲线密码系统及实现方法，其采用了 Montgomery 算术来实现大素数域中的各种运算，实现椭圆曲线密码系统的所需所有算术运算能够在 Montgomery 域中进行：模加/模减、模乘以及求逆运算，无需显式执行耗时的模归约运算，使其同时适合软件与硬件实现。

附图说明

图 1 为椭圆曲线密码系统层次体系结构图；

图 2 为素数域与 Montgomery 域元素之间的转换关系。

具体实施方式

下面结合附图 1、2 进一步详细说明本发明的椭圆曲线密码系统及方法。

椭圆曲线的定义：

一条椭圆曲线是在射影平面上满足方程：

$$Y^2Z + a_1XYZ + a_3YZ^2 = X^3 + a_2X^2Z + a_4X^2 + a_6Z^3 \quad [3-1]$$

的所有点的集合，且曲线上的每个点都是非奇异或者光滑的。

其中， $Y^2Z + a_1XYZ + a_3YZ^2 = X^3 + a_2X^2Z + a_4X^2 + a_6Z^3$ 是维尔斯特拉斯方程 (Weierstrass, Karl Theodor Wilhelm Weierstrass, 1815-1897)，是一个齐次方程。

所谓“非奇异”或“光滑”的，在数学中是指曲线上任意一点的偏导数 $F_x(x,y,z)$, $F_y(x,y,z)$, $F_z(x,y,z)$ 不能同时为 0。

设椭圆曲线上有一个无穷远点 $O\infty$ (0:1:0)，因为这个点满足方程[3-1]，

$x=X/Z$, $y=Y/Z$ 代入方程[3-1]得到：

$$Y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6 \quad [3-2]$$

其中， (x,y) 为普通平面直角坐标系上的坐标。

也就是说满足方程[3-2]的光滑曲线加上一个无穷远点 $O\infty$ ，组成了椭圆曲线。

椭圆曲线是连续的，并不适合用于加密，因此，一般地，要把适合加密的椭圆曲线定义在有限域上。

有限域是一种只由有限个元素组成的域。

给出一个有限域 F_p ，这个域只有有限个元素，即 F_p 中只有 p (p 为素数) 个元素 $0, 1, 2, \dots, p-2, p-1$ 。

定义 F_p 的加法 $(a+b)$ 法则是 $a+b \equiv c \pmod{p}$ ；即， $(a+b) \div p$ 的余数 和 $c \div p$ 的余数相同；

F_p 的乘法 $(a \times b)$ 法则是 $a \times b \equiv c \pmod{p}$ ；

F_p 的除法 $(a \div b)$ 法则是 $a/b \equiv c \pmod{p}$ ；即 $a \times b^{-1} \equiv c \pmod{p}$ ；(b^{-1} 也是一个 0 到 $p-1$ 之间的整数，但满足 $b \times b^{-1} \equiv 1 \pmod{p}$)。

F_p 的单位元是 1，零元是 0。

但是，并不是所有的有限域上的椭圆曲线都适合加密，其中对于定义在大素数域上的椭圆曲线是适合于加密的椭圆曲线，大素数域上的椭圆曲线可通过同构映射将一般的曲线方程变换为特别简单的形式： $y^2 = x^3 + ax + b$ ，其中曲线参数 $a, b \in F_p$ 并且满足 $4a^3 + 27b^2 \neq 0 \pmod{p}$ 。

因此，满足下列方程的所有点(x,y)，再加上无穷远点 $O\infty$ ，构成一条定义在大素数域 F_p 上的椭圆曲线。

$$Y^2=x^3+ax+b \pmod{p}$$

其中 x, y 属于 0 到 $p-1$ 间的大素数，并将这条椭圆曲线记为 $E_p(a,b)$ 。

公开密钥算法总是要基于一个数学上的难题。比如 RSA 密码系统依据的是：给定两个大素数 p, q 很容易相乘得到 n ，而对 n 进行因式分解却相对困难。

考虑如下等式：

$K=kG$ [其中 K, G 为 $E_p(a,b)$ 上的点， k 为小于 n (n 是点 G 的阶) 的整数，不难发现，给定 k 和 G ，根据加法法则，计算 K 很容易；但给定 K 和 G ，求 k 就相当困难了。]

这就是椭圆曲线密码系统基于的数学难题。把点 G 称为基点(base point)， k ($k < n$, n 为基点 G 的阶) 称为私有密钥(private key)， K 称为公开密钥(public key)。

本发明是采用了 Montgomery 算数来实现大素数域中的各种运算，从而实现椭圆曲线密码的系统和方法。

如图 2 所示 Montgomery 算数由 Peter Montgomery 提出，其核心运算是 Montgomery 乘法，由于 Montgomery 乘法既不需要计算及其耗时的除法也不需要利用商估值技术，因此它简化了模归约运算。

从数学的角度来看，Montgomery 域与素数域 $GF(p)$ 是同构的， $GF(p)$ 中的每一个元素在 Montgomery 域都有一个唯一与之对应的元素。元素 $a \in GF(p)$ 在 Montgomery 域中表示为 $a' = a \cdot R \pmod{p}$ ，其中 R 称为 Montgomery 常数(R 必须大于 p)。Montgomery 常数 R 与大素数域的特征必须互素，即 $\gcd(R, p)=1$ 。通常选取 R 是 2 的幂次： $R = 2^m$ ，其中 m 反映了硬件的规模，较佳为 $R = 2^{192}$ 。

利用 Montgomery 算数完成素数域运算需要将操作数由素数域转化到 Montgomery 域，转化可利用 Montgomery 乘法完成： $a' = MonMul(a, R^2) = a \cdot R^2 / R \pmod{p} = a \cdot R \pmod{p}$ 。

在系统中通过硬件实现 Montgomery 乘法算法，则这种转化在硬件实现时不再需要其它的运算，因此也不需要附加的硬件资源，转化时 Montgomery

常数 $R = 2^m \bmod p$ 已经被预算算，即计算 R^2 需要一些花费，但对于每个模 p 仅需计算一次。

由 Montgomery 域转化到素数域也可利用 Montgomery 乘法完成：
 $a = \text{MonMul}(a', 1) = a' \cdot 1 / R \bmod p = a \cdot R \cdot 1 / R \bmod p = a \bmod p$ ；
 b 与 a 的运算方式是一样的，即对 b 有下式成立 $b = \text{MonMul}(b', 1) = b' \cdot 1 / R \bmod p = b \cdot R \cdot 1 / R \bmod p = b \bmod p$ 。当在 Montgomery 域中执行大量算数运算时，这种转化的花费可以忽略，因为只需在所有运算开始和结束时进行一次。本发明中的椭圆曲线密码系统运算开始时所有的操作数被转化到 Montgomery 域中接着在 Montgomery 域完成所有椭圆曲线上的点运算，最后运算结果被转化到大素数域。

这样，本发明利用大整数的 Montgomery 表示可以有效地实现素数域中的模运算，而且在 Montgomery 域中无需显式执行耗时的模归约运算，实现椭圆曲线密码系统的所需的所有算数运算能够在 Montgomery 域中进行：模加/模减、模乘、求逆运算、模平方运算以及模归约运算。

模加运算： $a+b \bmod p = \text{MonAdd}(a', b') = (a+b)R \bmod p = a'+b' \bmod p$

模乘运算： $a \cdot b \bmod p = \text{MonMul}(a', b') = (a \cdot b)R \bmod p = a' \cdot b' / R \bmod p$

求逆运算： $a^{-1} \bmod p = \text{MonInv}(a') = a^{-1}R \bmod p = a'^{-1}R^2 \bmod p$

模减运算： $a-b \bmod p = \text{MonSub}(a', b') = (a-b)R \bmod p = a'-b' \bmod p$

模平方运算： $a^2 \bmod p = \text{MonSq}(a'^2) = (a^2)R \bmod p = a'^2 / R \bmod p$

模归约运算： $r = cR^{-1} \bmod p$

其中， MonMul ：模乘运算； MonAdd ：模加运算； MonInv ：求逆运算；
 MonSub ：模减运算； MonSq ：模平方运算

下面结合图 1 详细说明本发明实施例的椭圆曲线密码系统：

如图 1 所示，本实施例的椭圆曲线密码系统包括：

有限域算法模块，用于利用 Montgomery 算数实现大素数域中的加法运算、乘法运算、平方运算以及求逆运算。

这里的大素数域就是指的一般的素数域 F_p ， p 是大于 2 的素数。

所述的有限域算法模块包括模加运算模块、模减运算模块、模乘运算模块、模平方运算模块、模归约运算模块、求逆运算模块。

所述的模乘运算模块为 CIOS—Montgomery 乘法算法模块（Coarsely

Integrated Operand Scanning)，该算法模块占用的资源最少，速度最快。

CIOS—Montgomery 乘法算法的原理是：该算法集成了乘法与归约步骤。具体地说，此算法并没有直接计算 a 与 b 的乘积然后对该乘积进行归约，而是对乘法和归约在外层循环中交替进行，这样做是由于在第 i 次外层循环中归约步骤所用到的值 m 仅依赖于 $s[i]$ 的值，而在乘法的第 i 次循环中 $s[i]$ 的值已经计算完毕。

所述的模平方运算模块为 SRCIL—Montgomery 平方算法模块 (Squaring Reduction with Inner Loop)，该算法速度最快。SRCIL—Montgomery 平方算法原理是：该算法去除了一般 Montgomery 平方算法中的冗余部分并最大化算法软件并行的能力，其基本思想是使用了一个单独的循环计算 $a_i \times a_i$ 、去掉了进位的延迟并且通过改变循环的结构去掉冗余的部分。

所述的求逆运算模块为指数求逆算法模块，用费马小定理实现有限域中的求逆运算，即 $b = a^{p-2} \pmod{p}$ ，其能够利用硬件电路板上的 Montgomery 乘法器资源，该算法模块在有 Montgomery 乘法器的硬件电路板上的执行时间约为二元扩展欧几里得算法执行时间的 70%。

本实施例中的椭圆曲线密码系统，还包括：

点加与倍点算法模块，用于在利用 Montgomery 算数实现有限域算法的基础上实现椭圆曲线上的点加运算与倍点运算，其采用优化点加与倍点算法实现群运算，调用有限域算法模块，输入参数为 Montgomery 域中元素，因而运算结果依然在 Montgomery 域中。

标量乘算法模块，用于实现所述椭圆曲线密码系统的核心运算：标量乘运算，其输入参数为素数域中元素，运算前转化为 Montgomery 域中，运算结果转化回素数域中。所述标量乘模块中的运算采用 NAF 随机标量乘算法。

NAF 随机点标量乘算法原理是：NAF 随机点标量乘算法首先对标量进行 NAF 编码，即标量 k 的 NAF 表示为 $\sum_{i=0}^l k_i 2^i$ ，其中 $k_i \in \{0, \pm 1\}$ 并且相邻的 k_i 中至少有一个为 0。通过这种编码方式可以有效地计算标量乘，NAF 标量乘算法期望的运行时间约为 $(m/3)A + mD$ ，其中 $m = [\log_2 p]$ ， A 表示点加运算， D 表示倍点运算。

DH 密钥协商算法模块，用于调用标量乘算法模块，完成密钥协商。其输

入参数为素数域中元素，运算前转化为 Montgomery 域中，运算结果转化回素数域中。

EC-Diffie-Hellman 密钥协商是从一个主体的私钥和另一个主体的公钥导出一个共享的秘密值，此处两个主体具有相同的 EC 域参数。如果双方能够正确的执行完该协议，则他们将得到相同的结果。该算法可被一些方案调用以产生一个共享的秘密钥，其中，所输入的密钥是有效的。

数字签名与验证模块，用于调用标量乘算法模块，完成对消息的数字签名与验证过程。其输入参数为素数域中元素，运算前转化为 Montgomery 域中，运算结果转化回素数域中。

椭圆曲线数字签名与验证算法(Elliptic Curve Digital Signature Algorithm, ECDSA)是类似于 DSA(Digital Signature Algorithm)的一种基于椭圆曲线的数字签名与验证算法，其不象一般的离散对数问题和整数分解问题，椭圆曲线离散对数问题没有亚指数算法，正由于这点，使得采用椭圆曲线离散对数的算法的每比特强度在实质上更强了。

ECDSA 的主要参数包括：定义在有限域 $GF(p)$ 上的椭圆曲线 E ， E 上的 $GF(p)$ -有理点的个数# $E(GF(p))$ 可被一个大素数 n 整除，一个基点 $G \in E(GF(p))$ ，可记为： $D=(p, a, b, G, n, h), (d, Q)$, Hash 函数 H 。

将 $D=(p, a, b, G, n, h)$, Hash 函数 H ， Q 公开， d 保密。

在本实施例的椭圆曲线密码系统中，首先需要实现底层有限域中的各种基本的运算包括：加法运算、乘法运算、平方运算以及求逆运算；其次在有限域算法库的基础上实现椭圆曲线上的点加运算与倍点运算；最后实现椭圆曲线密码系统的核心运算：标量乘运算；通过调用标量乘算法模块，完成密钥协商；通过调用标量乘算法模块，完成对消息的数字签名与验证过程。

下面结合本发明的椭圆曲线密码系统，进一步详细描述本发明的椭圆曲线密码系统的实现方法，包括下列步骤：

(一) 利用 Montgomery 算数实现大素数域中的加法运算、乘法运算、平方运算以及求逆运算。

这里的大素数域就是指的一般的素数域 F_p ， p 是大于 2 的素数。

◆模加运算：

modadd 算法:

输入: 整数 $a, b \in [0, p-1]$, $a = (a_{t-1}, a_{t-2}, \dots, a_1, a_0)$, $b = (b_{t-1}, b_{t-2}, \dots, b_1, b_0)$ 。

输出: $c = a + b \bmod p$ 。

1. $c_0 \leftarrow Add(a_0, b_0)$, //低 32 比特相加

2. For i from 1 to t-1 do: $c_i \leftarrow Add(a_i, b_i) + carry(a_{i-1}, b_{i-1})$. //a 与 b 进行带进位

的 32 比特加法

3. If $c \geq p$, then $c \leftarrow c - p$. //如果 $carry(a_{t-1}, b_{t-1}) \neq 0$, 则最后的运算结果有进位执行减 p 运算

4. Return(c). //返回运算结果

其中, $Add(a_i, b_i) = (a_i + b_i) \bmod 2^{32}$, $carry(a_i, b_i) = (a_i + b_i) / 2^{32}$

◆模减运算:**modsub 算法:**

减运算和加运算是非常类似的, 不同的是它要用到借位。

输入: 整数 $a, b \in [0, p-1]$, $a = (a_{t-1}, a_{t-2}, \dots, a_1, a_0)$, $b = (b_{t-1}, b_{t-2}, \dots, b_1, b_0)$ 。

输出: $c = a - b \bmod p$ 。

1. $borrow \leftarrow 0$. //初始化将借位 borrow 赋予 0

2. For i from 0 to t-1 do: //a 与 b 进行带进位的 32 比特加法

2.1. $c_i \leftarrow (a_i - b_i - borrow) \bmod 2^{32}$ //32 比特减法运算结果

2.2. If $a_i - b_i - borrow \geq 0$, then $borrow \leftarrow 0$; otherwise $borrow \leftarrow 1$. //32

比特减法结果为正则借位 borrow 为 0, 否则借位 borrow 为 1

3. If $borrow > 0$, then $c \leftarrow c + p$. //如果运算最后还有借位则执行加 p 运算

4. Return(c). //返回运算结果

由于在 modadd 中用到了 $a - b (a \geq b)$, 这只是 modsub 的一个特殊情况, 只须将上面算法中的第 3 步去掉即可。

◆模乘运算:

Montgomery 乘法有多种实现方式, 本实施例采用了最为有效的 CIOS—Montgomery 乘法算法 (Coarsely Integrated Operand Scanning), 该算法占用的资源最少, 速度最快。

输入： 整数 $a, b \in [0, p-1]$ ， $a = (a_{t-1}, a_{t-2}, \dots, a_1, a_0)$ ， $b = (b_{t-1}, b_{t-2}, \dots, b_1, b_0)$ 。

输出： $c = abR^{-1} \bmod p$

1. For i from 0 to t-1 do: $s[i] \leftarrow 0$. //初始化存储运算结果的数组

$s6 \leftarrow 0, s7 \leftarrow 0$ //初始化临时变量

2. For i from 0 to t-1 do:

2.1. $C \leftarrow 0$. //初始化临时变量

2.2. For j from 0 to t-1 do: //a 与 b 进行 32 比特乘法

计算 $(C, S) = s[j] + a_j b_i + C$, //32 比特乘法的中间结果

令 $t[i] \leftarrow S$. //存储 32 比特进位

2.3. $(C, S) \leftarrow s6 + C, s6 \leftarrow S, s7 \leftarrow C$. //存储最高位的运算结果

2.4. $C \leftarrow 0, m = s[0] * p[0] \bmod 2^{32}, (C, S) = s[0] + m * p[0]$. //对最低 32 比特进行归约

2.5. For j form 1 to t-1 do: //逐段归约

计算 $(C, S) = s[j] + m * p[j] + C$, //32 比特中间结果归约

令 $s[j-1] \leftarrow S$. //存储 32 比特进位

2.6. $(C, S) \leftarrow s6 + C, s[t-1] \leftarrow S, s6 \leftarrow s7 + C$ //存储归约结果

3. If $s6 \neq 0$ //有进位

3.1 $C = 1$; //初始化临时变量

3.2 For i from 0 to t-1 do: //逐段进行进位纠正

计算 $(C, S) = s[i] + \sim p[i] + C$, //进位纠正

令 $s[i] \leftarrow S$. //存储 32 比特进位

4. Return($(s_{t-1}, \dots, s_1, s_0)$) //返回运算结果

这里 C, S 都是 32 比特的字，(C, S) 是 C, S 的连接，它是 64 比特；

$p'[0] = -p[0]^{-1} \bmod 2^{32}$, 其中 p[0] 是 192 比特素数 p 的最低 32 比特(最低的字)。

◆模平方运算:

Montgomery 平方也有多种实现方式, 本实施例采用了最为有效的 SRCIL —Montgomery 平方算法 (Squaring Reduction with Inner Loop), 该算法速度最快。

输入: 整数 $a, b \in [0, p - 1]$, $a = (a_{t-1}, a_{t-2}, \dots, a_1, a_0)$ 。

输出: $c = a^2 R^{-1} \bmod p$

1. For i from 0 to t-1 do: $(s[2i+1], s[2i]) \leftarrow a_i \times a_i$; //计算第 i 段相乘结果
2. For i from 0 to t-1 do: //对第 1 步运算结果进行归约
 - 2.1 $m = s[i] \times p[0] \bmod 2^{32}$; //计算每段归约值
 - 2.2 For j from 0 to i do: //逐段归约

$$(C, s[i+j]) = s[i+j] + m \times p[j] + C; //32 比特中间结果归约$$
 - 2.3 $C_1 = 0; C_2 = 0$; //初始化临时变量
 - 2.4 For j from i+1 to t-1 do: //计算 a_i 与 a_j 的乘积并进行归约
 - 2.4.1 $s_{long} = 2 \times C_1 + C + s[i+j]$; //存储中间结果
 - 2.4.2 $(C_1, S) = a_i \times a_j$; //计算 a_i 与 a_j 的乘积
 - 2.4.3 $(C, s[i+j]) = s_{long} + 2 \times S$; //与低段运算结果的进位相加
 - 2.4.4 $(C_2, s[i+j]) = m \times p[j] + s[i+j] + C_2$. //32 比特中间段归约
 - 2.5 $(prevcar, s[i+t]) = C + 2 \times C_1 + C_2 + s[i+t] + prevcar$; //存储最高 32 比特
3. $s[2t] = s[2t] + prevcar$; //存储最高位的进位
4. For i from 0 to t-1 do: $s[i] \leftarrow s[i+t]$ //将运算结果移至低 t 个单元
5. If $s[2t] \neq 0$ //若进位不等于 0
 - 1.1 $C = 1$; //初始化临时变量
 - 1.2 For i from 0 to t-1 do: //逐段进行进位纠正

计算 $(C, S) = s[i] + \sim p[i] + C$; //进位纠正

令 $s[i] \leftarrow S$. //存储 32 比特进位
6. Return($(s_{t-1}, \dots, s_1, s_0)$) //返回运算结果

其中， C, C_1, C_2, S 都是 32 比特的字， (C, S) 是 C, S 的连接，它是 64 比特； $p'[0] = p[0]^{-1} \bmod 2^{32}$ ，其中 $p[0]$ 是 192 比特素数 p 的最低 32 比特(最低的字)。

◆ Montgomery 模归约算法：

输入： $c = (c_{t-1}, \dots, c_1, c_0)$

输出： $r = cR^{-1} \bmod p$.

1. For i from 0 to t-1 do: //归约

 1.1 $C = 0$; //初始化临时变量

 1.2 $m = c_i * p[0]$; //计算每段归约值

 1.3 For j from 0 to t-1 do: //逐段归约

 计算 $(C, S) = c_{i+j} + m * p[j] + C$; //32 比特中间结果归约

 1.4 $(prevcar, s[i+t]) = C + s[i+t] + prevcar$; //存储最高 32 比特

2. For i from 0 to t-1 do: $c[i] \leftarrow c[i+t]$ //将运算结果移至低 t 个单元

3. If $prevcar \neq 0$ //若进位不等于 0

 3.1 $C = 1$; //初始化临时变量

 3.2 For i from 0 to t-1 do: //逐段进行进位纠正

 计算 $(C, S) = c[i] + \sim p[i] + C$; //进位纠正

 令 $s[i] \leftarrow S$ //存储 32 比特进位

4. Return($(r_{t-1}, \dots, r_1, r_0)$) //返回运算结果

◆ Montgomery 求逆算法：

考虑到利用硬件电路板上的 Montgomery 乘法器资源，因此本实施例考虑用费马小定理实现有限域中的求逆运算，即 $b = a^{p-2} \pmod{p}$ 。该算法在有 Montgomery 乘法器的硬件电路板上的执行时间约为二元扩展欧几里得(Euclid)算法执行时间的 70%。

输入： 整数 $a \in [0, p-1]$ ， $a = (a_{t-1}, a_{t-2}, \dots, a_1, a_0)$

输出: $b = a^{-1} \bmod p$.

1. $\bar{a} = a \cdot R \bmod p$; //将 a 变换到 Montgomery 域
2. $\bar{x} = 1 \cdot R \bmod p$; //将 1 变换到 Montgomery 域
3. For i from j-1 down to 0 do: //计算模指数
 - 3.1 $\bar{x} = MontMult(\bar{x}, \bar{x})$; //模平方运算
 - 3.2 If $e_i = 1$ then $\bar{x} = MontMult(\bar{x}, \bar{a})$;
 //如果 p-2 的当前比特是 1 则执行模乘运算
4. Return $b = MontMult(\bar{x}, 1)$. //返回运算结果至素数域

◆ 除法:

有限域中的除法运算是乘法和求逆的组合，本领域普通技术人员能够根据本实施例中关于乘法与求逆的描述，实现本发明的除法运算，因此，在本实施例中不再详细描述。

(二) 在利用 Montgomery 算数实现有限域算法的基础上实现椭圆曲线上的点加运算与倍点运算，本实施例采用优化的点加与倍点算法实现群运算，调用有限域算法模块，输入参数为 Montgomery 域中元素，因而运算结果依然在 Montgomery 域中。

对于群运算，本实施例对 IEEE P1363 标准中的点加以及倍点算法进行了优化，采用了与 IEEE P1363 标准(IEEE P1363 标准的参考文献：IEEE std 1363-2000 : Standard specifications for public-key cryptography, 2000, standards.ieee.org/catalog/oils/busarch.html)中不同的运算顺序，从而使得算法所需要的临时变量最少，优化算法如下：

◆ 点加(elliptic_add)

GF(p)上椭圆曲线 $y^2 = x^3 + ax + b$ 的点加公式的 Modified—Jacobian 坐标形式是：

$$P = (X_1, Y_1, Z_1, aZ_1^4), Q = (X_2, Y_2, Z_2, aZ_2^4), \text{且 } P + Q = (X_3, Y_3, Z_3, aZ_3^4)$$

这里 $U_1 = X_1 Z_2^2$, $U_2 = X_2 Z_1^2$, $S_1 = Y_1 Z_2^3$, $S_2 = Y_2 Z_1^3$, $H = U_2 - U_1$, $T = S_2 - S_1$,

$$X_3 = -H^3 - 2U_1 H^2 + T^2, \quad Y_3 = -S_1 H^3 + T(U_1 H^2 - X_3), \quad Z_3 = Z_1 Z_2 H,$$

$$aZ_3^4 = a(Z_3)^4.$$

其算法如下：

输入： $p, a, b, Q_{\text{in}} = (X, Y, Z), P = (X_2, Y_2)$.

输出： $Q_{\text{out}} = (X, Y, Z, aZ^4) = Q_{\text{in}} + P$.

1. If ($P == O$) //判断是否点 P 是无穷远点

$aZ^4 = a * Z^4$ if necessary //需要时计算 aZ^4 的值

return Q_{out} //返回点加结果

2. If ($Z == 0$) //判断是否点 Q_{in} 是无穷远点

$Q_{\text{out}} = P$ //点加结果为 P

return Q_{out} //返回点加结果

3. $aZ^4 = Z^2$ //计算 Z^2

4. $T_1 = X_2 * aZ^4$ //计算 $U_2 = X_2 Z^2$

5. $T_1 = T_1 - X$ //计算 $H = U_2 - U_1$

6. $aZ^4 = Z * aZ^4$ //计算 Z^3

7. $aZ^4 = Y_2 * aZ^4$ //计算 $Y_2 = Y_2 Z^3$

8. $aZ^4 = aZ^4 - Y$ //计算 $T = S_2 - S_1$

9. $Z = Z * T_1$ //计算 $Z_3 = ZH$

10. If ($T_1 == 0$) //如果 $U_2 = U_1$

If ($aZ^4 == 0$) //如果 $P = Q_{\text{in}}$

$Q_{\text{out}} = P$ //初始化 Q_{out}

Double (Q_{out}) //此时计算结果为 $2P$

return Q_{out} //返回点加结果

```

else          //此时 P=-Qin

Z=0          //此时计算结果为无穷远点

aZ4=0      //此时计算结果为无穷远点

return Qout //返回点加结果

11. T2= T12      //计算 H2

12. T1=T1*T2    //计算 H3

13. Y=T1*Y        //计算 S1H3

14. T2=X*T2      //计算 U1H2

15. X=(aZ4)2    //计算 T2

16. X=X-T1        //计算 T2-H3

17. X=X-T2        //计算 T2-U1H2-H3

18. X=X-T2        //计算 X3=T2-2U1H2-H3

19. T2=T2-X      //计算 U1H2-X3

20. T2=aZ4*T2  //计算 T(U1H2-X3)

21. Y=T2-Y        //计算 Y3=T(U1H2-X3)-S1H3

22. aZ4=Z2       //计算 Z2

23. aZ4=(aZ4)2 //计算 Z4

24. If(a==p-3) //如果 a=p-3

    aZ4=0-3aZ4 //计算 -3Z4

else

    aZ4=a*aZ4 //计算 aZ4

```

这个算法需要 9 次域乘法、5 次域平方和 2 个临时变量。

◆倍点(elliptic_doubl)

GF(p)上椭圆曲线 $y^2 = x^3 + ax + b$ 的倍点公式的 Modified—Jacobian 坐标形式是：

$$Q_{in} = (X_1, Y_1, Z_1, aZ_1^4), \quad Q_{out} = 2Q_{in} = (X_3, Y_3, Z_3, aZ_3^4)$$

这里: $S = 4X_1Y_1^2$, $U = 8Y_1^4$, $M = 3X_1^2 + (aZ_1^4)$, $T = M^2 - 2S$,

$$X_3 = T, \quad Y_3 = M(S-T) - U, \quad Z_3 = 2Y_1Z_1, \quad aZ_3^4 = 2U(aZ_1)^4$$

其算法如下:

输入: p , $a=-3$, b , $Q_{in} = (X, Y, Z, aZ^4)$.

输出: $Q_{out} = (X, Y, Z, aZ^4) = 2Q_{in}$.

1. If ($Z == 0$) return Q_{out} //如果 Q_{in} 是无穷远点输出结果
2. $T1 = 2Y$ //计算 $2Y$
3. $Z = T1 * Z$ //计算 $Z3=2YZ$
4. $Y = Y2$ //计算 $Y2$
5. $T1 = 2X$ //计算 $2X$
6. $T1 = 2T1$ //计算 $4X$
7. $T1 = T1 * Y$ //计算 $S=4XY2$
8. $T2 = X2$ //计算 $X2$
9. $X = 2T2$ //计算 $2X2$
10. $T2 = X + T2$ //计算 $3X2$
11. $T2 = T2 + aZ4$ //计算 $M=3X2+aZ4$
12. $X = T2^2$ //计算 $M2$
13. $X = X - T1$ //计算 $M2-S$
14. $X = X - T1$ //计算 $X3=T=M2-2S$
15. $T1 = T1 - X$ //计算 $S-T$
16. $T2 = T2 * T1$ //计算 $M(S-T)$
17. $Y = 2Y$ //计算 $2Y2$
18. $Y = Y2$ //计算 $4Y4$

```

19. Y = 2Y          //计算 U=8Y4
20. T1 = 2Y         //计算 2U
21. aZ4 = T1 * aZ4 //计算 2U(aZ4)
22. Y = T2 - Y     //计算 Y3 = M(S-T)-U

```

本实施例选取的 $a = p - 3$, 上面算法仅需 4 次域乘法以及 4 次域平方。

(三) 实现所述椭圆曲线密码系统的核心运算: 标量乘运算, 其输入参数为素数域中元素, 运算前转化为 Montgomery 域中, 运算结果转化回素数域中。所述标量乘模块中的运算采用 NAF 随机标量乘算法。

对于标量乘算法, 考虑到硬件实现标量乘算法时是采用有限状态机对标量的状态进行编码的, 因此为了减少有限状态机控制逻辑的复杂性以及存储空间的大小本实施例对标量进行了 NAF 编码并采用了下面的随机点标量乘算法, 该算法无需进行预算算。

◆随机点标量乘

随机点标量乘算法对标量进行 NAF(non-adjacent form)编码, 下面将这一算法描述如下:

输入: 整数 $k = \sum_{i=0}^{l-1} b_i 2^i$, 其中 $b_i \in \{0,1\}$ 且 $b_l = b_{l+1} = 0$, 椭圆曲线上的点 P

输出: 椭圆曲线上的点 Q, $Q = kP$

/*对标量进行 NAF(non-adjacent form)编码: NAF $\sum_{i=0}^l k_i 2^i$, 其中 $k_i \in \{0,\pm 1\}$ */

1. $\alpha \leftarrow 0$ //初始化临时变量

2. For i from 0 to l do: //NAF(non-adjacent form)编码

$$\beta \leftarrow \left\lfloor \frac{b_i + b_{i+1} + \alpha}{2} \right\rfloor,$$

$k_i \leftarrow b_i + \alpha - 2\beta$, //计算 NAF 编码并存储在 k_i 中

$$\alpha \leftarrow \beta,$$

/*由标量的 NAF 表示计算标量乘*/

3. For i from l down to 0 do: //计算标量乘

3.1. $Q \leftarrow 2Q$ //倍点运算

3.2. If $k_i \neq 0$ then: // k_i 不为零时需要执行点加运算

```

If  $k_i = 1$ , then  $Q \leftarrow Q + P$ ; // $k_i=1$  时执行  $Q+P$ 
Else  $Q \leftarrow Q - P$            // $k_i=-1$  时执行  $Q+(-P)$ 
Return(Q)                  //返回标量乘运算结果

```

(四) 用于调用标量乘算法模块, 完成对消息的数字签名与验证过程。其输入参数为素数域中元素, 运算前转化为 Montgomery 域中, 运算结果转化回素数域中。

DH 密钥协商算法:

EC-Diffie-Hellman 密钥协商是从一个主体的私钥和另一个主体的公钥导出一个共享的秘密值, 此处两个主体具有相同的 EC 域参数。如果双方能够正确的执行完该协议, 则他们将得到相同的结果。该算法可被一些方案调用以产生一个共享的秘密钥, 其中, 所输入的密钥是有效的。

输入:

- EC 基本参数 q, a, b, n 和 G 以及相应的密钥 s 和 W' (对于 s 和 W' , 基本参数应该一样)
- 主体自己的私钥 s
- 另一主体的公钥 W'

其中: 私钥 s , EC 基本参数 q, a, b, r 和 G , 以及公钥 W' 是有效的;所有的密钥都和同一基本参数相关。

输出: 导出的共享秘密值 $z \in GF(q)$; 或者 “error”

操作. 共享的秘密值 z 必须按照以下步骤进行:

1. 计算椭圆曲线点 $P = s W'$.
2. 如果 $P = O$ 输出 “error” 并停止。
3. 令 $z = xP$, 即点 P 的 x 坐标。
4. 输出 z 作为共享的秘密钥。

(五) 调用标量乘算法模块, 完成对消息的数字签名与验证过程。其输入参数为素数域中元素, 运算前转化为 Montgomery 域中, 运算结果转化回素数域中。

椭圆曲线数字签名与验证算法(ECDSA):

ECDSA(Elliptic Curve Digital Signature Algorithm)是类似于 DSA (Digital Signature Algorithm) 的一种基于椭圆曲线的数字签字与验证算法。不象一般的离散对数问题和整数分解问题，椭圆曲线离散对数问题没有亚指数算法，正由于这点，使得采用椭圆曲线离散对数的算法的每比特强度在实质上更强了。

ECDSA 的主要参数包括：定义在有限域 $GF(p)$ 上的椭圆曲线 E ， E 上的 $GF(p)$ -有理点的个数 $\#E(GF(p))$ 可被一个大素数 n 整除，一个基点 $G \in E(GF(p))$ 。我们可记为： $D = (p, a, b, G, n, h), (d, Q)$, Hash 函数 H 。

将 $D = (p, a, b, G, n, h)$, Hash 函数 H , Q 公开， d 保密。

◆签名生成:

签名者 A 利用上面的参数及公私钥对如下对一消息 m 进行签名：

1. 随机或伪随机地选择一个整数 $k \in_{\mathbb{Z}_n^*}$;
2. 计算 $kG = (x_1, y_1)$, $r = x_1 \bmod n$, 如果 $r=0$, 则返回到 1;
3. 计算 $k^{-1} \bmod n$;
4. 计算 $e = H(m)$;
5. 计算 $s = k^{-1}(e + dr) \bmod n$, 如果 $s=0$, 则返回到 1。

其中， (r, s) 是 A 对消息 m 的签名。

◆签名验证:

验证者 B 如下验证 (r, s) 是 A 对消息 m 的签名：

1. 验证 r, s 是 $[1, n-1]$ 中的整数;
2. 计算 $e = H(m)$;
3. 计算 $w = s^{-1} \bmod n$;
4. 计算 $u_1 = ew \bmod n$, $u_2 = rw \bmod n$;
5. 计算 $X = u_1G + u_2Q = (x_1, y_1)$, 如果 $X=O$, 则拒绝这个签名, 否则, 计算 $v = x_1 \bmod n$, 当且仅当 $v=r$ 时接受这个签名。

本发明提供了一种适合硬件实现的椭圆曲线密码系统及实现方法，其采用了 Montgomery 算数来实现大素数域中的各种运算，实现椭圆曲线密码系统的所需所有算数运算能够在 Montgomery 域中进行：模加/模减、模乘以及求逆运算，无需显示执行耗时的模归约运算，使其同时适合软件与硬件实现。

本实施例是为了使本领域普通技术人员理解而对本发明所进行的详细描述，但本领域普通技术人员可以想到，在不脱离本发明的权利要求所涵盖的范围内还可以做出其它的变化和修改，其均在本发明的保护范围内。

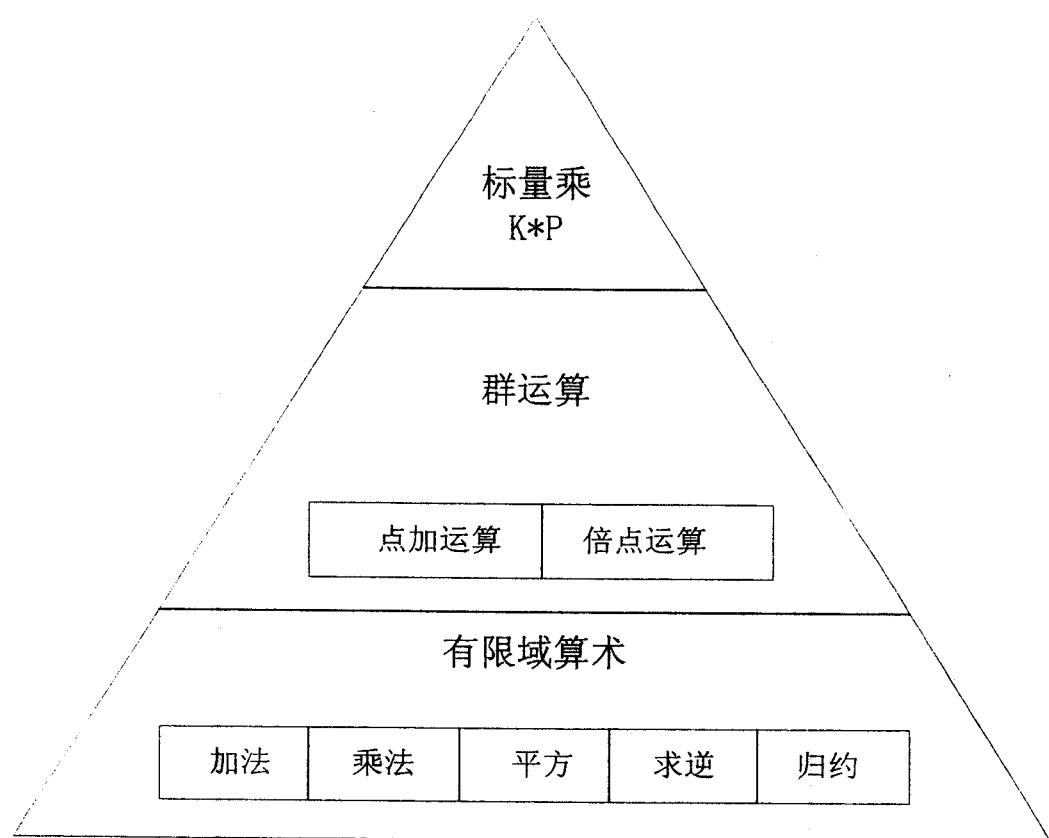


图 1

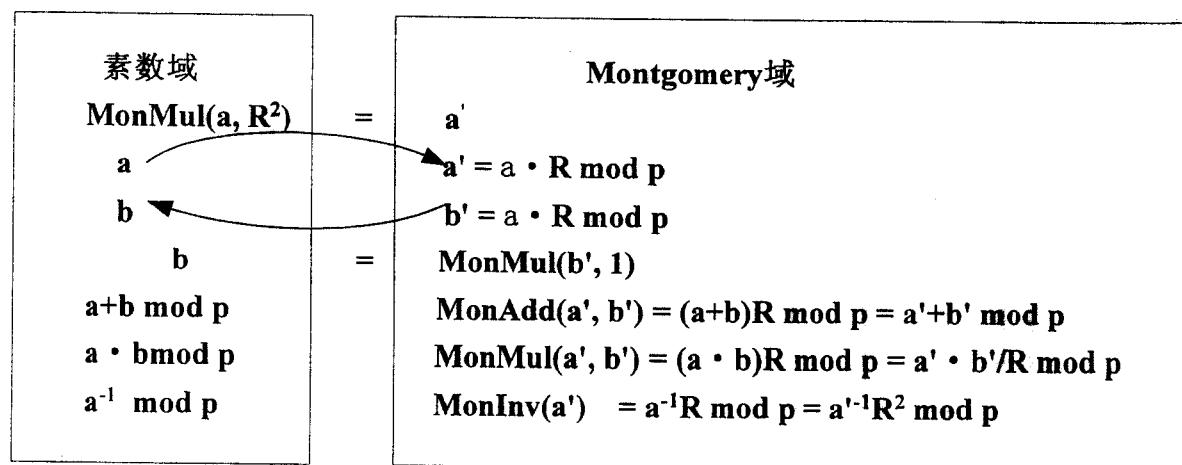


图 2