(19)中华人民共和国国家知识产权局

(12)发明专利申请

(10)申请公布号 CN 104539423 A
(43)申请公布日 2015.04.22

(21)申请号 201410772127.5

(22)申请日 2014.12.16

(71)申请人 熊荣华
地址 100094 北京市海淀区遗光寺8号院北区6号楼西702号

(72)发明人 熊荣华

(51)Int.Cl.
H04L 9/32(2006.01)
H04L 9/30(2006.01)

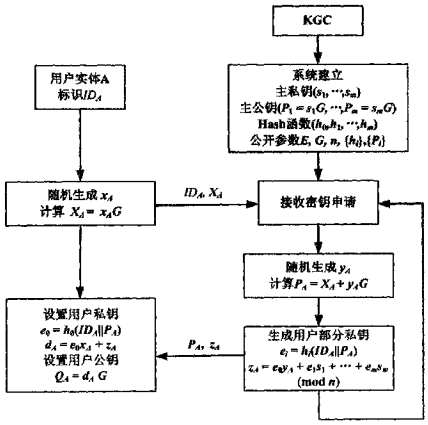
权利要求书2页 说明书7页 附图2页

(54)发明名称

一种无双线性对运算的无证书公钥密码体制的实现方法

(57)摘要

一种无双线性对运算的无证书公钥密码体制的实现方法，属于信息安全领域，用于解决用户密钥的生成、使用和对用户公钥的认证问题。在本发明中，首先由用户设置私密值并计算临时公钥，再由密钥生成中心为用户生成另一部分密钥并对两部分进行绑定，最后由用户合成自己的实际公私钥对。本发明克服了普通无证书密码体制中可能存在的公钥被替换、签名被伪造的缺陷，且用户对私钥具有完全控制权，密钥可以撤销和重新生成，用户签名具有不可否认性。本发明采用标准椭圆曲线的公钥密码算法，不使用双线性对运算，计算效率高，占用资源少，安全性强，在应用于签名、认证和密钥协商时，可脱离密钥生成中心运行。本发明可满足大规模系统和低功耗设备的身份鉴别、通信保密和抗抵赖应用需求。



CN 104539423 A

1. 一种无双线性对运算的无证书公钥密码体制的实现方法，其特征在于：所述实现方法包括系统建立、用户密钥生成和用户密钥使用的方法，用于构造基于常规椭圆曲线、不使用双线性对运算的无证书公钥密码体制。

2. 根据权利要求 1 所述系统建立方法，其特征在于：所述方法假设 E：$y^2 = x^3+ax+b$ 为有限域 F_q 上的椭圆曲线，n 为素数，$m \geq 1$ 为正整数，G 是 E 上的一个 n 阶基点，$h_0()$，$h_1()$，...，$h_m()$ 是一组 $\{0,1\}^* \rightarrow [0, n-1]$ 的 HASH 函数，KGC 随机选择 m 个私密值 s_1，…，$s_m \in Z_n^*$ 作为系统主私钥，计算系统主公钥，$P_1 = s_1G$，…，$P_m = s_mG$，KGC 保密 s_1，…，s_m，公开系统参数 $(E, n, G, \{h_i()\}_{i=1}^m, \{P_i\}_{i=1}^m)$。

3. 根据权利要求 1 所述用户密钥生成方法，其特征在于：所述方法包括以下步骤：

步骤一：标识为 ID 的用户实体随机选择私密值 $x \in Z_n^*$，计算 $X = xG$，发送 (ID, X) 至 KGC；

步骤二：KGC 在收到 (ID,X) 后，检验用户标识和身份的合法性，随机选择 $y \in Z_n^*$，计算 $P = X+yG$，$e_i = h_i(ID \| P)$，$i = 1, ..., m$（注：符号 ‖ 表示数据的串接，下同），生成用户的部分私钥 $z = e_0y+e_1s_1+\cdots+e_ms_m \pmod n$，如果 $e_0 = 0$ 或者 e_1，…，e_m 全为 0，则重新生成 y，最后回送 (P, z) 至用户，并将 P 作为用户的部分公钥公开发布；

步骤三：用户收到 (P,z) 后，计算 $e_i = h_i(ID \| P)$，$i = 0,1,...,m$，$d = e_0x+z \pmod n$，$Q = dG$，并验证 $Q = e_0P+e_1P_1+\cdots+e_mP_m$ 是否成立，若成立，则设置 d 为用户的私钥，P 为用户的部分公钥，Q 为用户的实际公钥。

4. 根据权利要求 1 所述用户密钥使用方法，其特征在于：所述无证书公钥密码体制生成的用户密钥适合于各种常规椭圆曲线上的标准公钥密码算法，标识为 ID 的用户实体在使用私钥时，可以直接使用私钥 d 进行密码运算，而不分开使用私密值 x 和部分私钥 z，公众（即其他用户）可以通过用户标识 ID 和用户部分公钥 P 计算 $e_i = h_i(ID \| P)$，再按公式 $Q = e_0P+e_1P_1+\cdots+e_mP_m$ 计算用户的实际公钥，然后使用实际公钥 Q 进行相关的密码运算。

5. 根据权利要求 1 所述用户密钥生成方法，其特征在于：所述方法包括以下简化和变形：

(1) 取 $m > 1$，$h()$ 为一个 $\{0,1\}^* \rightarrow [1, 2^m-1]$ 的 HASH 函数，x 和 y 的定义如前，计算 $e = h(ID \| P)$，将 e 按二进制比特展开，记为 $e = (e_1, e_2, \cdots, e_m)_2$，其中 $e_i \in \{0,1\}$，$i = 1, ..., m$，再计算 $z = y+e_1s_1+\cdots+e_ms_m \pmod n$，设置用户的实际私钥为 $d = x+z \pmod n$，用户的部分公钥为 $P = xG+yG$，用户的实际公钥为 $Q = P+e_1P_1+\cdots+e_mP_m$；

(2) 取 l，N 为正整数，$m \leq 2^N$，$h()$ 为一个 $\{0,1\}^* \rightarrow [1, 2^{lN}-1]$ 的 HASH 函数，x 和 y 的定义如前，计算 $e = h(ID \| P)$，将 e 按二进制比特展开，每连续 N 比特组成一个字，共组成 l 个字，记作 $e = (w_1, w_2, \cdots, w_l)_N$，再令 $e_i = (w_i \bmod m)+1$，则 $e_i \in [1, m]$，$i = 1, \cdots, l$，再计算 $z = y+s_{e1}+\cdots+s_{el} \pmod n$，设置用户的实际私钥为 $d = x+z \pmod n$，用户的部分公钥为 $P = xG+yG$，用户的实际公钥为 $Q = P+P_{e1}+\cdots+P_{el}$；

(3) 取 N 为正整数，满足 $mN \leq \log_2(n)$，$h()$ 为一个 $\{0,1\}^* \rightarrow [1, 2^{mN}-1]$ 的 HASH 函数，x 和 y 的定义如前，令 $e = h(ID \| P)$，将 e 按二进制比特展开，每连续 N 比特组成一个字，共组成 m 个字，记作 $e = (e_1, e_2, \cdots, e_m)_N$，再计算 $z = y+e_1s_1+\cdots+e_ms_m \pmod n$，设置用户的实际私钥为 $d = x+z \pmod n$，用户的部分公钥为 $P = xG+yG$，用户的实际公钥为 $Q = P+e_1P_1+\cdots$

$+e_mP_m$。

一种无双线性对运算的无证书公钥密码体制的实现方法

技术领域

[0001] 本发明属于信息安全领域，特别涉及一种基于椭圆曲线并且不使用双线性对运算的无证书公钥密钥体制的实现方法。

背景技术

[0002] 公钥密码体制需要解决密码算法、密钥生成和密钥分配等问题，最关键的是要解决对用户公钥的认证问题。根据公钥认证方法的不同，常见的公钥密码体制有以下三种：

[0003] 基于证书的公钥密码体制：PKI(Public Key Infrastructure)；

[0004] 基于标识的公钥密码体制：IBC(Identity Based Cryptograph)；

[0005] 无证书公钥密码体制：CLPKC(Certificateless Public key Cryptography)。

[0006] PKI是一种公钥基础设施，它采用证书权威机构(Certification Authority，CA)颁发证书的形式来建立用户身份与其所拥有的公钥之间的联系，CA的数字签名可保证用户公钥的真实性。然而证书的管理过程需要很多的计算开销和存储开销。为了免除对用户证书的管理，1984年Shamir提出了一种基于身份的公钥密码(IBC)体制，在这个体制中，用户的公钥是由与用户身份相关的比特串构成，由用户标识唯一确定，而私钥则由信任权威机构生成。这种体制消除了对用户证书的依赖，简化了密钥的管理过程，但也存在3个弱点：1)用户私钥必须由信任权威机构生成，存在密钥托管问题，用户签名不具有唯一性和不可否认性；2)用户密钥无法撤销；3)使用椭圆曲线构造基于身份的密码系统时必须使用双线性配对运算，计算复杂，效率低下。2003年，Al-Riyami等人首次提出无证书公钥密码体制(CLPKC)，成为密码学领域的研究热点。无证书密码体制介于PKI公钥密码体制和基于身份的公钥密码体制之间，兼有二者的优点。在该体制中，用户的密钥由可信中心和用户自己分别独立生成，用户对私钥具有完全控制权；用户公钥可通过用户标识和用户部分公钥计算得出，不需要使用公钥证书；这样既减少了传统PKI中的证书管理问题，又消除了基于身份密码体制中的密钥托管问题，提高了系统的运行效率，减少了系统的复杂性。在Al-Riyami提出的无证书密码体制或其它改进版本中，密钥分配一般经过三个步骤：首先是可信中心初始化，设置系统参数，其次是可信中心为用户生成部分私钥，最后由用户加入私密值合成用户的实际私钥和实际公钥。因此用户密钥是由可信中心和用户自己分别独立生成的，用户私钥对外界是完全保密的，消除了密钥托管问题。从该体制设想提出至今，国内外学者已提出了数十种无证书公钥密码方案，但大部分方案都将系统生成和用户生成的两部分密钥分开使用，或者由用户将两个部分合成后使用，从而使它们容易遭受公钥替换攻击和伪造签名攻击，存在安全缺陷，造成部分方案被攻破。另一方面，无证书密码体制建立在普通离散对数计算难题(DLP)和椭圆曲线离散对数计算难题(ECDLP)之上，大多数实现方案都使用了双线性对运算，运算效率较低，降低了无证书密码体制的应用优势。有的实现方案算法比较特殊，仅仅只适合一种密码运算，有的只能用于签名，有的只能用于加密，不能形成通用完整的无证书公钥密码体系。因此，创设一种可实现无双线性对运算的无证书公钥密码体制的方法是本发明的主要宗旨。

发明内容

[0007] 本发明综合了 PKI 公钥密码体制、基于身份的公钥密码体制和无证书密码体制的优点，公开了一种新的基于椭圆曲线的、不使用双线对运算的无证书公钥密码体制的具体实现方法。本发明采用新的密钥分配方式，对用户密钥进行有效管理。本发明不需要进行复杂的证书管理，不使用双线性对运算，用户密钥可以撤销。在这种新的无证书公钥密码体制的基础上，可构造基于身份的签名 (Identity Based Signature，IBS) 方案、基于身份的认证 (Identity Based Identificaton，IBI) 协议和基于身份的密钥协商 (Identity Based Authentication Key Exchange，IBAKE) 协议，以及用于普通的数字签名、公钥加密、密钥交换和签密等，与现有椭圆曲线密码算法完全兼容，具有运算效率高、占用资源少、安全性强的明显优势，适合于具有海量用户的超大规模系统应用。

[0008] 本发明所述无证书公钥密码体制由可信的密钥生成中心 (Key Generation Center，简称 KGC) 和用户实体组成。密钥分配方案与 Al-Riyami 提出的无证书体制原始版本有所不同，用户私密值的设置在 KGC 为用户生成部分私钥之前，而不是在 KGC 为用户生成部分私钥之后，而且，KGC 为用户生成部分私钥时，以签名的方式将用户标识和用户部分公钥绑定在一起，可以消除替换公钥攻击、假冒身份攻击和伪造签名攻击。进一步，本发明将系统生成和用户生成的两部分密钥合成为一个密钥对使用，密码运算时不需要使用双线性对运算，可以使用现有基于椭圆曲线的标准公钥密钥算法。

[0009] 本发明所述无证书公钥密码体制的实现方法建立在常规椭圆曲线之上，主要内容包括系统建立、用户密钥生成、用户密钥使用和密码算法应用等。

[0010] 1. 系统建立

[0011] 本发明所述系统建立按以下步骤完成。

[0012] 设 $E: y^2 = x^3 + ax + b$ 为有限域 F_q 上的椭圆曲线，n 为素数，$m \geq 1$ 为正整数，G 是 E 上的一个 n 阶基点，$h_0()$，$h_1()$，...，$h_m()$ 是一组 $\{0,1\}^* \rightarrow [1, n-1]$ 的 HASH 函数。

[0013] KGC 随机选择 m 个私密值 $s_1, \ldots, s_m \in Z_n^*$ 作为系统主私钥，计算系统主公钥：$P_1 = s_1G, \ldots, P_m = s_mG$。KGC 保密 $s_1, \ldots, s_m$，公开系统参数 $(E, n, G, \{h_i()\}_{i=1}^m, \{P_i\}_{i=1}^m)$。

[0014] 2. 用户密钥生成

[0015] 本发明所述用户密钥生成通过以下步骤完成。

[0016] (1) 标识为 ID 的用户实体随机选择私密值 $x \in Z_n^*$，计算 $X = xG$。发送 (ID，X) 至 KGC。

[0017] (2)KGC 在收到 (ID，X) 后，检验用户 ID 和身份的合法性。随机选择 $y \in Z_n^*$，计算：$P = X + yG$，$e_i = h_i(ID||P)$，$i = 0, 1, \ldots, m$，如果 $e_0 = 0$ 或者 $e_1, \ldots, e_m$ 全为 0，则重新选择 y。最后生成用户部分私钥 $z = e_0y + e_1s_1 + \cdots + e_ms_m \pmod n$，回送 (P，z) 至用户，并将 P 作为用户的部分公钥公开发布。为保证系统安全，KGC 应保证对于不同的用户选择不同的 y 和不同的 P。

[0018] (注：符号 || 表示数据的串接，下同。)

[0019] (3) 用户收到 (P，z) 后，计算 $e_i = h_i(ID||P)$，$d = e_0x + z \pmod n$，$Q = dG$，并验证 $Q = e_0P + e_1P_1 + \cdots + e_mP_m$ 是否成立。若成立，则设置 d 为用户的私钥，P 为用户的部分公钥，Q

为用户的实际公钥。

[0020] 记 $r = x+y$，本发明所述用户私钥可以表示为 $d = e_0r+e_1s_1+\cdots+e_ms_m \pmod n$，所述用户部分公钥可以表示为 $P = rG$，所述用户实际公钥可以表示为 $Q = e_0P+e_1P_1+\cdots+e_mP_m$，并且满足 $Q = dG$。

[0021] 3. 用户密钥使用

[0022] (1) 用户私钥使用

[0023] 本发明所述用户在使用私钥时，与普通椭圆曲线上公钥密码算法相同，可直接使用私钥 d 进行密码运算，而不用分开使用私密值 x 和部分私钥 z。

[0024] (2) 用户公钥使用

[0025] 本发明所述公钥依赖方（即公众用户）通过用户标识 ID 和用户部分公钥 P 计算 $e_i = h_i(ID||P)$，再使用系统公钥计算用户的实际公钥，计算公式如下：

[0026] $Q = e_0P+e_1P_1+\cdots+e_mP_m$。

[0027] 由于本发明所述用户私密值 x 是用户独立生成的，用户的实际私钥 d 对其他人是不可知的，即使对 KGC 也是如此，因此用户对私钥具有完全控制权。

[0028] 本发明还包括一种特殊情况，如果在上述生成过程中，令 x 为 0，则用户的实际私钥是 $d = z$。在这种情况下，KGC 需要用安全的方式将 z 回送用户，并且用户对私钥没有完全控制权。

[0029] 4. 密码算法应用

[0030] 采用本发明所述无证书公钥密码体制实现方法生成的用户密钥对 (Q, d) 实际上就是常规椭圆曲线上的公私钥对，所以适用于所有关于椭圆曲线上的标准公钥密码算法，如 ECDSA 签名算法，Schnorr 签名算法，国家 SM2 标准签名算法、公钥加密算法和密钥协商算法等，在使用这些公钥密码算法时，不需要使用双线性对运算。

[0031] 如果用户在签名、认证和密钥协商时，将自己的部分公钥作为签名结果、认证凭据和密钥协商数据的一部分提交，接收者则可直接计算用户的实际公钥而不依赖密钥生成中心的支持。因此，除公钥加密外，基于本发明所述无证书公钥密码体制的签名、认证和密钥协商均可转换为基于身份的签名 (IBS)、基于身份的认证 (IBI) 和基于身份的密钥协商 (IBAKE)。

[0032] 下面以签名算法为例进行说明：

[0033] 设签名者为用户 A，其标识为 ID_A，部分公钥为 P_A，用私钥 d_A 对消息 M 的标准签名记为 $\mathbf{SIGN}_{d_A}(M)$，则 $\sigma = (\mathbf{SIGN}_{d_A}(M),\ P_A)$ 就是该用户对消息 M 的标识签名。

[0034] 其他用户收到该用户的标识 ID_A、消息 M 和签名 σ 后，首先根据用户 ID_A、用户部分公钥 P_A 和系统公钥 $\{P_i\}$ 计算该用户的实际公钥 Q_A，再用公钥 Q_A 按标准签名算法验证对 M 的签名 $\mathbf{SIGN}_{d_A}(M)$。这个验证过程完全由验证者独立完成，不需要第三方协助，因此等同于一种基于标识的签名算法。

[0035] 公钥加密算法和密钥协商算法可用类似的方法处理。

[0036] 5. 安全性

[0037] 本发明所述无证书公钥密码体制实现方法的安全性基于以下两个假设：

[0038] a)KGC 是可信的（类似于 PKI 体制的 CA）。

[0039] b) 解椭圆曲线上离散对数难题 (ECDLP) 在计算上是不可行的。

[0040] (1) 系统密钥安全性

[0041] 首先，根据 ECDLP 假设，攻击者不能从系统公钥 P_i 求出系统私钥 s_i。

[0042] 抗共谋攻击：即抵抗所有用户合谋对系统密钥的攻击。

[0043] 假定攻击者获取了一组用户 ID、私钥、以及对应的部分公钥：

[0044] $ID_i, d_i, PP_i, i = 1,2,\ldots, tt << n$

[0045] 其中，$d_i = e_{i,0}r_i + e_{i,1}s_1 + \cdots + e_{i,m}s_m \pmod n$，$e_{i,j} = h_j(ID_i || PP_i)$，$PP_i = r_iG$，希望从中求出系统私钥 $s_1, \ldots, s_m$。

[0046] 在这组式子中，$s_1, \ldots, s_m, r_1, \ldots, r_t$ 对攻击者是未知的，根据用户密钥生成规则，r_i 随机生成且互不相同。将 t 个方程联立组成含 t+m 个未知元的线性方程组：

[0047] $\{e_{i,0}r_i + e_{i,1}s_1 + \cdots + e_{i,m}s_m = d_i \pmod n \mid i = 1,2,\ldots, t\}$

[0048] 这个方程组的解空间至少是 m 维的，由于 $m \geq 1$，得到真实解 $s_1, \ldots, s_m$ 的概率为 $1/n^m$，比求解 ECDLP 还要困难。因此，从任意多组用户密钥推导系统私钥 $s_1, \ldots, s_m$ 是计算上不可行的，这说明本发明所述系统密钥能够抵抗共谋攻击。

[0049] (2) 用户密钥安全性

[0050] 本发明所述用户部分公钥与用户私钥组成的对 (P, d) 能够映射到所述系统私钥对用户 ID 的一个 Schnorr 签名。文献证明，在随机预言模型下，Schnorr 签名算法是安全的。事实上，目前还没有发现对 Schnorr 签名算法的有效攻击方法，已有的攻击方法基本上都等同于解椭圆曲线上离散对数难题。因此本发明所述用户密钥生成算法是安全的，所述用户部分公钥与私钥组成的对 (P, d) 是不可伪造的。

[0051] (3) 算法安全性

[0052] 本发明所述无证书公钥密码体制实现方法采用标准的签名算法、加密算法、身份认证协议和密钥协商协议，它们的安全性已经得到公认。

[0053] (4) 用户公钥的自证性

[0054] 在本发明所述无证书公钥密码体制实现方法中，当使用系统公钥从用户标识 ID 和部分公钥 P 计算用户的实际公钥 Q 时，实际上是对系统私钥签名 (P, d) 的一个验证。如果验证正确，即 Q 计算正确，则 Q 必是用户 ID 对应的实际公钥。如果由于某种原因（如攻击者伪造了一个用户部分公钥）使 Q 计算错误，则随后使用 Q 所作的验签或公钥加密都不会得到预期的结果，原因是攻击者不能伪造出与 Q 对应的私钥。这说明本发明所述用户密钥具有一种自认证性，当使用系统公钥从用户标识 ID 和部分公钥 P 计算用户的实际公钥 Q 时，自动隐含了对用户公钥的认证，只有标识为 ID 的用户才具有与 P 和 Q 对应的私钥。

[0055] (5) 用户密钥不存在托管

[0056] 在本发明所述的无证书公钥密码体制实现方法中，用户的私钥对 KGC 也是未知的，因此不存在用户密钥的托管问题。原因是用户密钥由用户和 KGC 共同生成，用户私钥 $d = e_0x + z \pmod n$，其中 x 对 KGC 是未知的，e_0, z, Q 和 X 对 KGC 是已知的，并且 $Q = dG = e_0xG + zG = e_0X + zG$，如果 KGC 能够知道用户的私钥 d，则 KGC 可通过计算 $\boldsymbol{x = e_0^{-1}(d - z) \pmod n}$，得出 $\boldsymbol{xG = e_0^{-1}(d - z)G = e_0^{-1}(Q - zG) = X}$，从而推出 KGC 可解 X = xG 这一 ECDLP 难题，与安全性假设矛盾。

[0057] 6. 高效性

[0058] 本发明所述的无证书公钥密码体制的实现方法，不使用双线性对运算，不使用数字证书，计算效率高，占用资源少，整体效率达到了一个新的高度。

[0059] 根据测算，在密码安全强度相当的基础上，如果采用超奇异椭圆曲线上的双线性对运算实现 CLPKC，一次双线性对运算耗费的计算时间是一次多倍点运算的 10-20 倍，即使在最新的 BN 曲线上经过优化了的双线性对运算，一次双线性对运算耗费的计算时间是一次多倍点运算的 8-10 倍。本发明所述的无证书公钥密码体制，不使用复杂的双线性对运算，仅需要少量的多倍点运算和点加运算，计算用户实际私钥降低到 1/10 个多倍点计算量，签名 / 验签、加密 / 解密等运算也采用标准的密码算法，计算量达到最小。因此，本发明所述无证书公钥密码体制是高效性的。

[0060] 本发明所述无证书公钥密码体制实现方法中，KGC 需要保存系统私钥和系统参数。每生成一个用户密钥，系统只需要保存用户的标识和部分公钥以及随机数 y。由此可见 KGC 的存储开销相当小，因此发明所述无证书公钥密码体制适合于大规模海量用户的系统应用。

[0061] 本发明所述无证书公钥密码体制实现方法中，用户只需要保存系统参数、用户私钥、用户公钥和部分公钥，存储开销非常小，节省了存储资源和网络带宽。用户端密码设备只需要支持普通椭圆曲线密码算法，能够使用现有的芯片。

[0062] 本发明所述无证书公钥密码体制的实现方法，适用于标准的椭圆曲线签名算法（如 ECDSA、SM2、Schnorr 等算法），具有签名格式简单、签名数据短的优点，加上签名者的用户标识和部分公钥，一个数字签名只占用相当于 2 个椭圆曲线点的存储空间。例如，当有限域 F_q 和 n 的规模为 256bit 时，如果用户标识长度不超过 32 字节，则签名与用户标识及用户部分公钥的总长度不超过 128 字节。因此，采用本发明生成的无证书签名有利于签名的传递和验证。

附图说明

[0063] 图 1 为本发明的密钥生成流程。

[0064] 图 2 以签名为例说明本发明的密钥使用流程。

具体实施方式

[0065] 本发明所述无证书公钥密码体制的实现方法，基于常规椭圆曲线、不使用双线性对运算，用户密钥由 KGC 和用户自身独立生成的两部分合成，合成过程由 KGC 完成。下面详细描述本发明的具体实施方式及其简化和变形。应理解，下述实施例是用于说明本发明而不是限制本发明的保护范围。

[0066] 实施例一

[0067] 阶段一：系统建立

[0068] 系统建立由 KGC 完成。

[0069] KGC 选择有限域 F_q 上的安全椭圆曲线 E：$y^2 = x^3+ax+b$，取 E 上的一个 n 阶点 G 作为基点，其中 n 为素数。再选取正整数 $m \geq 1$，和一组 $\{0,1\}^* \rightarrow [1,n-1]$ 的 HASH 函数 $h_0()$，$h_1()$，...，$h_m()$。一般选择 F_q 为素数域，且 q 和 n 的比特数在 192 以上，例如可选择国家 SM2

标准规定的椭圆曲线参数。在 SM2 标准中，q 和 n 的比特数都是 256。

[0070] KGC 随机选择 m 个私密值$s_1, \ldots, s_m \in Z_n^*$作为系统主私钥，计算系统主公钥：$P_1 = s_1G, \ldots, P_m = s_mG$。KGC 保密 $s_1, \ldots, s_m$，公开系统参数$(E, n, G, \{h_i()\}_{i=1}^m, \{P_i\}_{i=1}^m)$。

[0071] 阶段二：用户密钥生成

[0072] 用户密钥生成由 KGC 和用户共同完成。

[0073] (1) 标识为 ID 的用户实体随机选择私密值$x \in Z_n^*$，计算 X = xG，发送 (ID，X) 至 KGC。

[0074] (2)KGC在收到 (ID,X) 后，检验用户 ID和身份的合法性，为用户生成另一部分私钥 z。KGC 随机选择$y \in Z_n^*$，计算：$P = X+yG$，$e_i = h_i(ID||P)$，$i = 0,1,\ldots,m$，$z = e_0y+e_1s_1+\cdots+e_ms_m \pmod n$。如果 $e_0 = 0$ 或者 $e_1, \ldots, e_m$全为 0，则重新生成 y。最后 KGC 回送 (P，z) 至用户，并将 P 作为用户的部分公钥公开发布，。

[0075] 为保证系统安全，KGC 可通过数据库列表查询的方式保证对于不同的用户选择不同的 y 和不同的 P。

[0076] (3) 用户收到 (P，z) 后，计算 $e_i = h_i(ID||P)$，$i = 0,1,\ldots,m$，$d = e_0x+z \pmod n$，$Q = dG$，并验证 $Q = e_0P+e_1P_1+\cdots+e_mP_m$是否成立。若成立，则设置 d 为用户私钥，P 为用户部分公钥，Q 为用户实际公钥。

[0077] 阶段三：用户密钥使用

[0078] (1) 用户私钥使用

[0079] 用户在使用私钥时，与普通公钥密码算法相同，可直接使用私钥 d 进行密码运算。

[0080] (2) 用户公钥使用

[0081] 用户公钥依赖方（即公众用户）通过用户标识 ID 和用户部分公钥 P 计算 $e_i = h_i(ID||P)$，再通过系统公钥 $\{P_i\}$ 计算用户的实际公钥 $Q = e_0P+e_1P_1$十$\cdots+e_mP_m$。

[0082] 在本实施例中，计算用户的实际公钥需要进行 m+1 次多倍点运算和 m 次点加运算。存储 m 个系统主公钥需要 m×O(G) 字节空间，其中 O(G) 表示基点 G 的字节数。在 m = 1 的特殊情况下，计算用户实际公钥的公式变为 $Q = e_0P+e_1P_1$，根据文献资料显示，存在一种快速算法，使这种类型的计算量仅相当于 1.17 个多倍点的计算量，存储 1 个系统主公钥需要 O(G) 字节空间。例如，当 q 和 n 的比特数都是 256 时，存储 1 个系统主公钥只需要 64 字节空间。

[0083] 实施例二

[0084] 选取椭圆曲线如实施例一。取 m > 1，h() 为一个 $\{0,1\}^* \rightarrow [1,2^m-1]$ 的 HASH 函数，系统公开参数为$(E, n, G, h(), \{P_i\}_{i=1}^m)$。在用户密钥生成阶段，x 和 y 的定义如实施例一，令 e = h(ID||P)，将 e 按二进制展开，记为 $e = (e_1, e_2, \ldots, e_m)_2$，其中 $e_i \in \{0,1\}$，i = 1，...，m。最后生成的用户私钥为 $d = x+y+e_1s_1+\cdots+e_ms_m \pmod n$，用户的部分公钥为 P = xG+yG，用户的实际公钥为 $Q = P+e_1P_1+\cdots+e_mP_m$。

[0085] 当 $e_i = 0$ 时，计算 Q 的式子中的第 i 项将不出现，因此计算用户实际公钥平均只需要进行 m/2 次点加运算。当 $m < \log_2(n)$ 时，计算 Q 花费的时间比 1 次多倍点运算花费的时间要少得多，因此采用这种方式能够获得较高的效率。为保证安全，实际应用中一般要求 $m \geqslant 128$。例如当 n 的比特数为 256 时，一次多倍点计算平均需要 255 次双倍点运算和 128

次点加运算。取 m = 128，本实施例计算用户的实际公钥平均需要 64 个点加运算，是一个多倍点计算量的 1/6，而存储 128 个系统主公钥仅需要 8K 字节空间。本实施例计算用户实际公钥的计算量是实施例一的 1/(6m)。

[0086] 实施例三

[0087] 选取椭圆曲线如实施例一。取 l，N 为正整数，$m \leqslant 2^N$，h() 为一个 $\{0,1\}^* \rightarrow [1, 2^{lN}-1]$ 的 HASH 函数，系统公开参数为$(\boldsymbol{E}, \boldsymbol{n}, \boldsymbol{G}, \boldsymbol{h}(), \{\boldsymbol{P_i}\}_{\boldsymbol{i=1}}^{\boldsymbol{m}})$。在用户密钥生成阶段，x 和 y 的定义如实施例一，令 e = h(ID||P)，将 e 按二进制比特展开，每连续 N 比特组成一个字，共组成 l 个字，记作 $e = (w_1, w_2, \ldots, w_l)_N$，再令 $e_i = w_i \pmod m + 1$，则 $e_i \in [1, m]$，i = 1，...，l。最后生成的用户私钥为 $d = x+y+s_{e1}+\cdots+s_{el} \pmod n$，用户的部分公钥为 P = xG+yG，用户实际的公钥为 $Q = P+P_{e1}+\cdots+P_{el}$。

[0088] 在本实施例中，计算用户实际公钥只需要进行 l 次点加运算。为保证安全，要求组合数$C_{m-l+1}^{l} \geq 2^{128}$。例如当 n 的比特数为 256 时，取 N = 8，l = 32，m = 128 可满足要求。在这种情况下，lN = 256，计算用户的实际公钥只需要 32 次点加运算，比计算一个多倍点要快 12 倍，而存储 128 个系统主公钥仅需要 8K 字节空间，因此获得了比实施例二更高的效率。

[0089] 实施例四

[0090] 选取椭圆曲线如实施例一。取 N 为正整数，满足 $mN \leqslant \log_2(n)$，h() 为一个 $\{0,1\}^* \rightarrow [1, 2^{mN}-1]$ 的 HASH 函数，系统公开参数为$(\boldsymbol{E}, \boldsymbol{n}, \boldsymbol{G}, \boldsymbol{h}(), \{\boldsymbol{P_i}\}_{\boldsymbol{i=1}}^{\boldsymbol{m}})$。在用户密钥生成阶段，x 和 y 的定义如实施例一，令 e = h(ID||P)，将 e 按二进制比特展开，每连续 N 比特组成一个字，共组成 m 个字，记作 $e = (e_1, e_2, \ldots, e_m)_N$。最后生成的用户私钥为 $d = x+y+e_1s_1+\cdots+e_ms_m \pmod n$，用户的部分公钥为 P = xG+yG，用户的实际公钥为 $Q = P+e_1P_1+\cdots+e_mP_m$。

[0091] 在本实施例中，虽然计算用户实际公钥需要进行 m 次多倍点运算，但 e_i 是比较小的整数，计算效率较高。例如当 n 的比特数为 256 时，取 N = 128，m = 2，在这种情况下，计算用户的实际公钥比计算一个普通多倍点要快 1/4，而存储 2 个系统主公钥仅需要 128 字节空间。本实施例在计算效率和存储空间上均占有优势，在时间和空间上达到了相对平衡，是本发明的一种优选实施例。

[0092] 上述实施例仅用于说明本发明的实质精神和技术构思，以便于本发明的使用和实施，并不能以此限制本发明的保护范围。凡根据本发明精神实质进行的等效变换和修饰，都应涵盖在本发明的保护范围之内。

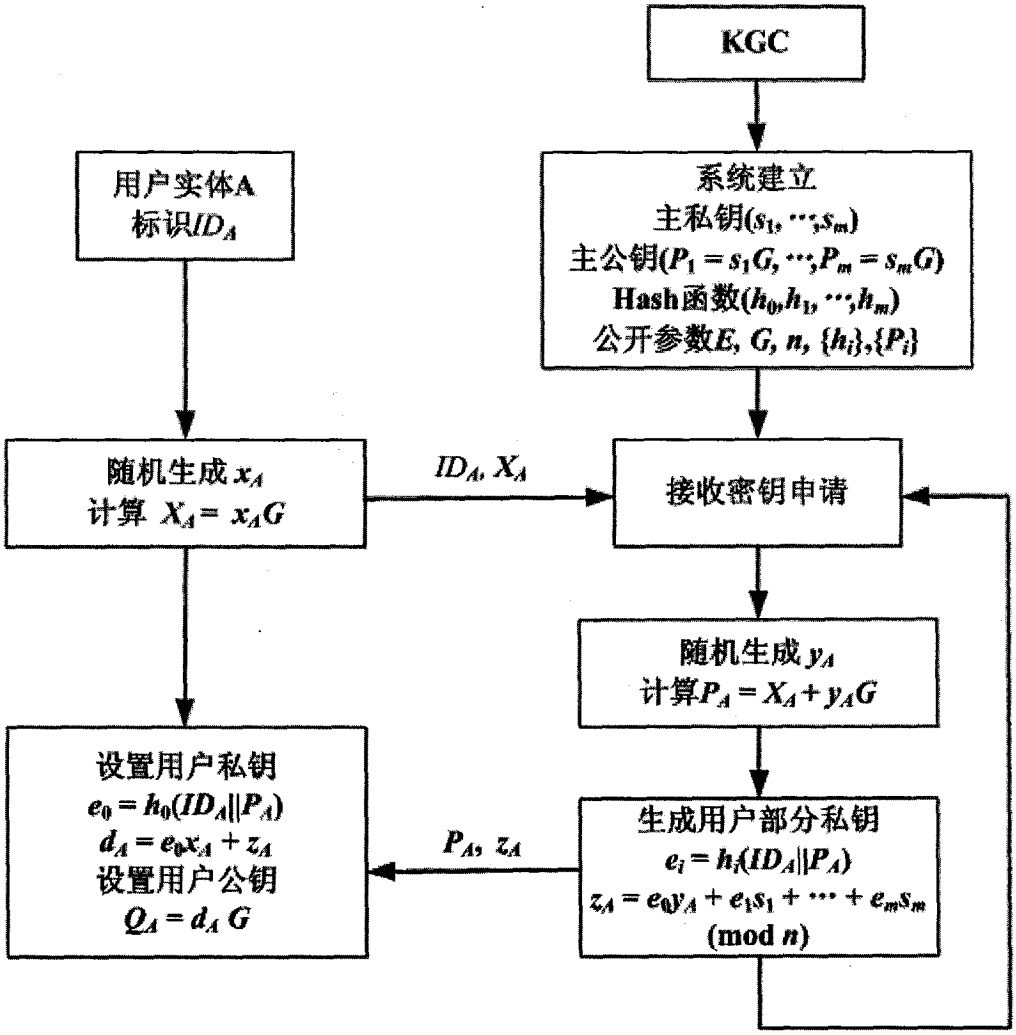

KGC
用户实体A
标识ID_A
系统建立
主私钥$(s_1, \cdots, s_m)$
主公钥$(P_1 = s_1G, \cdots, P_m = s_mG)$
Hash函数$(h_0, h_1, \cdots, h_m)$
公开参数$E, G, n, \{h_i\}, \{P_i\}$
随机生成x_A
计算 $X_A = x_AG$
ID_A, X_A
接收密钥申请
随机生成y_A
计算$P_A = X_A + y_AG$
设置用户私钥
$e_0 = h_0(ID_A \| P_A)$
$d_A = e_0x_A + z_A$
设置用户公钥
$Q_A = d_A G$
P_A, z_A
生成用户部分私钥
$e_i = h_i(ID_A \| P_A)$
$z_A = e_0y_A + e_1s_1 + \cdots + e_ms_m$
(mod n)

图 1

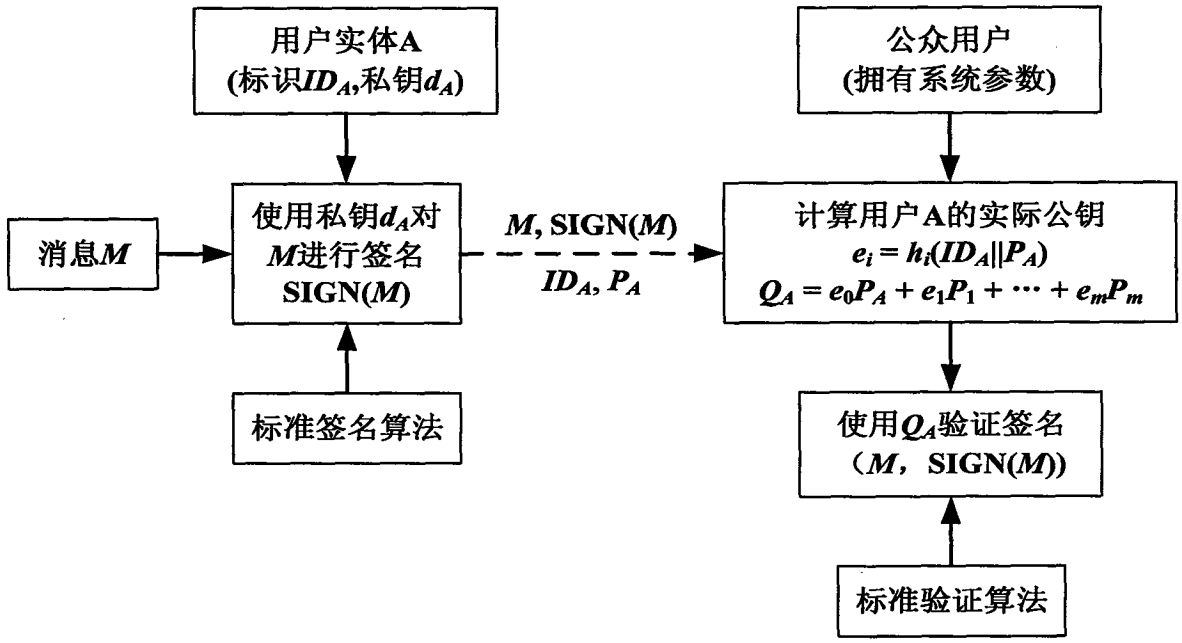

用户实体A
(标识ID_A,私钥d_A)
公众用户
(拥有系统参数)
消息M
使用私钥d_A对
M进行签名
SIGN(M)
M, SIGN(M)
ID_A, P_A
计算用户A的实际公钥
$e_i = h_i(ID_A||P_A)$
$Q_A = e_0P_A + e_1P_1 + \cdots + e_mP_m$
标准签名算法
使用Q_A验证签名
(M, SIGN(M))
标准验证算法

图 2