report on the application of this deduce technique in Ethereum with ECDSA

SM2 在以太坊 ECDSA 中的应用

2022 年 7 月 21 日

目录

1 ECDSA 简介

椭圆曲线密码自出现以来，便受到了研究者们的关注与重视。由于它拥有其他公钥密码体制无法比拟的优势，目前在各个领域中逐渐产生了取代了经典的 RSA 公钥密码体制的趋势。随着椭圆曲线密码体制的逐步普及，椭圆曲线数字签名也成为了研究者们的关注的重点之一。椭圆曲线数字签名是椭圆曲线密码体制的重要应用之一，它将椭圆曲线密码的优势应用到了数字签名当中。

ECDSA 的安全性基于素域上的离散对数问题。它可以看作是椭圆曲线对先前基于离散对数问题（DLP）的密码系统的模拟，只是群元素由素域中的元素数换为有限域上的椭圆曲线上的点。椭圆曲线密码体制的安全性基于椭圆曲线离散对数问题（ECDLP）的难解性。椭圆曲线离散对数问题远难于离散对数问题，椭圆曲线密码系统的单位比特强度要远高于传统的离散对数系统。因此在使用较短的密钥的情况下，ECC 可以达到于 DL 系统相同的安全级别。这带来的好处就是计算参数更小，密钥更短，运算速度更快，签名也更加短小。

ECDSA 于 1999 年成为 ANSI 标准，并于 2000 年成为 IEEE 和 NIST 标准。它在 1998 年既已为 ISO 所接受，并且包含它的其他一些标准亦在 ISO 的考虑之中。与普通的离散对数问题和大数分解问题不同，椭圆曲线离散对数问题没有亚指数时间的解决方法。因此椭圆曲线密码的单位比特强度要高于其他公钥体制。

2 ESCDEA 实现原理

ECDSA 是 ECC 与 DSA 的结合，整个签名过程与 DSA 类似，所不一样的是签名中采取的算法为 ECC，最后签名出来的值也是分为 r,s。

签名过程如下：

1、选择一条椭圆曲线 Ep(a,b)，和基点 G；

2、选择私有密钥 k（k<n，n 为 G 的阶），利用基点 G 计算公开密钥 K=kG；

3、产生一个随机整数 r（r<n），计算点 R=rG；

4、将原数据和点 R 的坐标值 x,y 作为参数，计算 SHA1 做为 hash，即 Hash=SHA1(原数据,x,y)；

5、计算 s=r - Hash * k (mod n)

6、r 和 s 做为签名值，如果 r 和 s 其中一个为 0，重新从第 3 步开始执行

验证过程如下：

1、接受方在收到消息 (m) 和签名值 (r,s) 后，进行以下运算

2、计算：sG+H(m)P=(x1,y1), r1 x1 mod p。

3、验证等式：r1 =r mod p。

4、如果等式成立，接受签名，否则签名无效。

3 ESCDEA 实现基础

3.1 椭圆曲线密码 (ECC)

椭圆曲线密码是用有限域上的椭圆曲线构成的群来类比有限域的乘法群，从而得到类似于使用有限域乘法群的公钥密码体制。椭圆曲线密码体制的安全性是基于椭圆曲线上离散对数问题求解的困难性，这个问题目前还没有找到能够有效解决的亚指数时间算法。

椭圆曲线密码体制中使用的椭圆曲线为 $y^2 = x^3 + ax + b$ 方程表示的曲线，此椭圆曲线关于 X 轴对称，且满足约束条件 $4a^3 + 27b^2 \neq 0$。

我们可以基于椭圆曲线定义一个群，这个群满足以下特点：

a) 群里的元素都在椭圆曲线上

b) 椭圆上的单位元指的是无限远点

c) 椭圆上的点 p 的逆元与 P 关于 x 轴对称

d) 加法满足以下规则：对于在同一条直线上的非零点 P、Q、R 有 P+Q+R=0;

而 ECC 的实现过程可以简化为以下步骤：

选取一条椭圆曲线 E_p(a,b)：$y^2 = x^3 + ax + b$, 并选取椭圆曲线上的一个点 P 作为基点;

选定一个较大的数 k 作为算法的私钥，并根据椭圆曲线的系数乘法 Q=kP 运算得到作为算法公钥的 Q 点;

加密：选择一个随机数 r，将待加密的明文 M 生成密文 C，此处生成的 C 是一个点对，其中 C=(rP,M+rQ)

解密：由于对解密者 k 已知且 P、Q 均为公开内容，故可执行操作 M+rQ-k(rP) = M，从上述信息中恢复出 M 的信息

3.2 椭圆曲线数字签名 (ECDSA)

3.2.1 椭圆曲线数字签名生成算法

假设 Alice 希望对消息 m 进行签名，所采用的椭圆曲线参数为 D = (p , a , b , G , n , h) , 对应的密钥对为 (k , Q), 其中 Q 为公钥，k 为私钥。

Alice 将按如下步骤进行签名：

1. 产生一个随机数 d,$1 \leq d \leq n-1$

2. 计算 $dG = (x_1, y_1)$, 将 x_1 转化为整数 $x_1^{'}$

3. 计算 r=$x_1^{'}$mod n, 若, 则转向第 1 步.

4. 计算 d^{-1} mod n.

5. 计算哈希值 $H(m)$，并将得到的比特串转化为整数 e.

6. 计算 $s = d^{-1}(e + kr)$ mod n，若 s = 0, 则转向第 1 步.

7.(r,s) 即为 Alice 对消息 m 的签名.

3.2.2 椭圆曲线数字签名验证算法

为验证 Alice 对消息 m 的签名 (r,s)，Bob 需要得到 Alice 所用的椭圆曲线参数 D=(p,a,b,G,n,h) 以及 Alice 的公钥 Q。

步骤如下：

1. 验证 r 和 s 是区间 [1,n1] 上的整数.

2. 计算 H(m) 并将其转化为整数 e.

3. 计算 $w = s^{-1}$ mod n.

4. 计算 $u_1 = e * w$ mod n 以及 $u_2 = r * w$ mod n. 5. 计算 $X = (x_1, y_1) = u_1G + u_2Q$

6. 若 X=O，则拒绝签名，否则将 X 的 x 坐标 x_1 转化为整数 $x_1^{'}$，并计算 $v = x_1^{'}$ mod n.

7. 当且仅当 v=r 时，签名通过验证.

4 ESCDEA 实现步骤

如下步骤:

第一步：初始化化秘钥组，生成 ECDSA 算法的公钥和私钥

第二步：执行私钥签名，使用私钥签名，生成私钥签名

第三步：执行公钥签名，生成公钥签名

第四步：使用公钥验证私钥签名

5 总结

本质上，ECDSA 过程包括四个基本要素:

1. 参与数字签名的所有通信方都使用相同的全局域参数，用定义椭圆曲线以及曲线上的基点。

2. 签名者首先需要生成一对公钥、私钥。对于私钥，签名者选择一个随机数或者伪随机数作为签名者的私钥。使用随机数和基点，签名者计算出椭圆曲线上的另一解点，作为签名者的公钥

3. 对于待签名的消息计算其 hash 值。使用私钥、全局域参数、hash 值来产生签名、签名包括两个整数，r 和 s

4. 如果要对签名进行验证，验证者使用签名者的公钥、全局域参数、

整数 s 作为输入，并将计算得到的输出值 v 与收到的 r 进行比较。如果 v=r，则签名通过.