✅ Report on the application of this deduce technique in Ethereum with ECDSA

Elliptic Curve Signatures

（椭圆曲线数字签名算法）是一种基于椭圆曲线密码学 (ECC)的密码安全**数字签名方案。ECDSA**依赖于有限域上**的椭圆曲线循环群的数学计算以及ECDLP 问题**（椭圆曲线离散对数问题）的难度。**ECDSA 签名/验证**算法依赖于 EC 点乘法，其工作原理如下所述。对于相同的安全级别，ECDSA 密钥和签名比 RSA 中的要短。256 位 ECDSA 签名与 3072 位 RSA 签名具有相同的安全强度。

ECDSA在经典 Weierstrass 形式的有限域上使用加密**椭圆曲线**(EC)。这些曲线由它们的**EC 域参数描述**，由各种加密标准指定，例如**SECG：SEC 2**和**Brainpool (RFC 5639)**。

密码学中的椭圆曲线需要定义：

> **Generator point G**, used for scalar multiplication on the curve (multiply integer by EC point)
>
> **Order *n*** of the subgroup of EC points, generated by **G**, which defines the length of the private keys (e.g. 256 bits)

不同于传统的离散对数问题和大整数分解问题，椭圆曲线密码的单位比特强度要高于其他公钥体制。

- 签名通过使用私钥**privKey**和消息哈希**h**的椭圆曲线变换将随机点**R**（仅由其 x 坐标表示）编码为数字s，这证明消息签名者知道私钥**privKey** . 由于**ECDLP问题**的难度，签名{ **r , s**}不能泄露私钥。
- **签名验证**使用公钥**pubKey**和消息哈希**h**将签名中的证明编号**s**解码回其原始点**R** ，并将恢复的R的 x 坐标与签名中的**r**值进行比较。

Key Generation

密钥对由两部分组成：

private key (integer): **privKey**

public key (EC point): **pubKey** = **privKey** * **G**

私钥：private key 在[0,n-1]的范围内**随机生成**

公钥：public key是在椭圆曲线上计算出的一个点，即**私钥 private * 生成元 G**

ECDSA Sign

签名算法：以input message：***msg*** 和 私钥***private*** **key** 作为输入，输出签名signature，对应着一对整数**{r,s}**

> ECDSA签名算法是基于 **ElGamal signature scheme** 而工作的。

1）使用hash function(eg.SM3, SHA256) 计算msg的hash值。eg. h = hash(msg)

2）安全生成一个在[1,...,n-1]范围内的随机数k

> 在确定性 ECDSA 的情况下，值 k 是从 h privKey 派生的 HMAC

3）计算随机点 **R** = **k** * **G** ,它的X坐标就是r = R.x

4）计算签名证明:

$$s = k^{-1} * (h + r * privKey) (mod\ n)$$

其中，求逆的运算得到的结果是一个整数，其满足：

$$k * k^{-1} \equiv 1 (mod\ n)$$

5）返回签名值：**{r,s}**

ECDSA Verify Signature

验签算法：以msg和签名{r,s}作为输入，输出值是一个boolean的值：valid or invalid。

思路：使用公钥恢复点R'并检查它是否与签名过程中随机生成的点R相同。

1）使用签名期间使用的相同加密哈希函数计算消息hash： **h = hash(msg)**

2）计算签名证明的模逆：

$$s1 = s^{-1} (mod\ n)$$

3）恢复签名过程中使用的随机点：

$$R^{'} = (h * s1) * G + (r * s1) * publicKey$$

4）从***R'***获取其 x 坐标：

$$r' = R'.x$$

5）比较**r' == r**来得出验证签名的结果。

以太坊简介

区块链

区块链：具有**共享状态的密码性安全交易的单机**

> 一个 状态机 是指可以读取一系列的输入，然后根据这些输入，会转换成一个新的状态出来的东西。

以太坊的状态有百万个交易。这些交易都被“组团”到一个区块中。一个区块包含了一系列的交易，每个区块都与它的前一个区块链接起来。

为了让一个状态转换成下一个状态，交易必须是有效的。

验证交易的过程就是**挖矿**，计算机用计算资源创建一个包含有效交易的区块。

任何矿工都可以创建并验证区块。

工作量证明(proof of work)

一个矿工必须要比其他矿工更快的提供出这个“证明”，从而让一个区块添加到主链上。通过矿工提供的一个数学机制的“证明”来证实每个区块。

奖励：每次矿工证明了一个新区块，那么就会产生一个新的以太币并被奖励给矿工。

出现分岔区块时候，遵循最长链原则。

账户

以太坊的全局"共享状态"是有很多小对象（账户）来组成的，这些账户可以通过消息传递架构来与对方进行交互。每个账户都有一个与之关联的状态(state)和一个20字节的地址(address)。在以太坊中一个地址是160位的标识符，用来识别账户的。

账户状态

1. nonce：如果账户是一个外部拥有账户，nonce代表从此账户地址发送的交易序号。如果账户是一个合约账户，nonce代表此账户创建的合约序号
2. balance： 此地址拥有Wei的数量。1Ether=10^18Wei
3. storageRoot： Merkle Patricia树的根节点Hash值（我们后面在解释Merkle树）。Merkle树会将此账户存储内容的Hash值进行编码，默认是空值
4. codeHash：此账户EVM（以太坊虚拟机，后面细说）代码的hash值。对于合约账户，就是被Hash的代码并作为codeHash保存。对于外部拥有账户，codeHash域是一个空字符串的Hash值

世界状态

以太坊的全局状态就是由账户地址和账户状态的一个映射组成。这个映射被保存在一个叫做Merkle Patricia树的数据结构中。

> Merkle Tree（也被叫做Merkle trie）是一种由一系列节点组成的二叉树，这些节点包括：
>
> 1. 在树的最底层的包含了源数据的大量叶子节点
> 2. 一系列的中间的节点，这些节点是两个子节点的Hash值
> 3. 一个根节点，同样是两个子节点的Hash值，代表着整棵树

费用

gas就是用来衡量在一个具体计算中要求的费用单位。gas price就是你愿意在每个gas上花费Ether的数量，以"gwei"进行衡量。"Wei"是Ether的最小单位，1Ether表示10^18Wei. 1gwei是1,000,000,000 Wei。

区块

所有的交易都被组成一个"块"。一个区块链包含了一系列这样的链在一起区块。

在以太坊中，一个区块包含：

1. 区块头
2. 关于包含在此区块中交易集的信息
3. 与当前块的ommers相关的一系列其他区块头

交易收据

自于被包含在交易收据中的日志信息存储在头中。就像你在商店买东西时收到的收据一样，以太坊为每笔交易都产生一个收据。像你期望的那样，每个收据包含关于交易的特定信息。这些收据包含着：

1. 区块序号
2. 区块Hash
3. 交易Hash
4. 当前交易使用了的gas
5. 在当前交易执行完之后当前块使用的累计gas
6. 执行当前交易时创建的日志

ECDSA算法在以太坊中的应用

以太网中，选择相同k值，得到两个签名，通过固定的k生称固定的r，从而可以恢复出私钥privateKey。

> 同一个椭圆曲线，生成元G相同，对两个哈希值分别得到对应的签名，通过对s1和s2作差，即可以得到k。接着按照求私钥的方式，选取一个签名和k，r，G即可恢复。

参考文档

1. ECDSA：椭圆曲线签名
2. ECDSA签名算法介绍
3. 以太坊工作原理