项目说明 -- ECDSA在以太坊中应用报告

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

ECDSA介绍

ref:

> ECDSA签名算法介绍
>
> Introduction to ECC

ECDSA全称椭圆曲线数字签名算法，是使用椭圆曲线密码（ECC）对数字签名算法（DSA）的模拟。与普通的离散对数问题（DLP）和大数分解问题（IFP）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。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。

了解了前置的数学知识后，我们直接关注ECDSA方案的主体部分。

参数准备:

- 构造有限域，选取了大素数

  P=0xFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFEFFFFFC2F

  =2^256 − 2^32 − 2^9 − 2^8 − 2^7 − 2^6 − 2^4 − 1
- 选择椭圆曲线：$y^2 = x^3 + Ax + B, A = 0, B = 7$
- 选定基点
  $G = (0x79BE667EF9DCBBAC55A06295CE870B07029BFCDB2DCE28D959F2815B16F81798,$
  $0x483ADA7726A3C4655DA4FBFC0E1108A8FD17B448A68554199C47D08FFB10D4B8)$

  $N * G = I$，N为G的阶，
  $N = 0xFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFEBAAEDCE6AF48A03BBFD25E8CD0364141$

秘钥生成:

- 随机生成一个256bit的数d作为私钥 sk，然后再ECC上计算公钥$vk = d * G$

签名过程:

- 有消息M，私钥sk
- 在 $\mathbb{F}_n^*$ 中随机选取k，计算$R = k * G = (r_x, r_y)$
- 计算$e = hash(M)$，这里bitcoin用的是double hash
- 计算$s = k^{-1}(e + r_x d) mod N$
- 输出签名$sig = (r_x, s)$

验证过程:

- 使用公钥P、消息m或消息的hash值e，验证签名$signature = (r_x, s)$
- 计算$R' = s^{-1}(eG + r_x P) = (r'_x, r'_y)$
- 如果 $r_x = r'_x$ 则通过验证

以太坊介绍

ref：https://zhuanlan.zhihu.com/p/203134170

一些前置的基础知识，比如区块链就不再赘述，直接观察以太坊的结构。以太坊主要由几个部分组成：

1. 账户(accounts)
2. 状态(state)
3. 损耗和费用(gas and fees)
4. 交易(transactions)
5. 区块(blocks)
6. 交易执行(transaction execution)
7. 挖矿(mining)
8. 工作量证明(proof of work)

账户(accounts)

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，用来识别账户。

账户有外部账户和合约账户两种。一个外部拥有的账户可以通过创建和用自己的私钥来对交易进行签名，来发送消息给另一个外部账户或合约账户。**两个外部账户之间**传送的消息只是一个简单的价值转移。**外部账户到合约账户**的消息会激活合约账户的代码，允许它执行各种动作（比如转移代币，写入内部存储，挖出一个新代币，执行一些运算，创建一个新的合约等等）。

合约账户不可以自己发起一个交易，只有在接收到一个交易之后(从一个外部账户或另一个合约账户接)，为了响应此交易而触发一个交易。

账户状态

任何账户状态都有四个组成部分：

1. nonce：外部账户代表从此账户地址发送的交易序号。合约账户代表此账户创建的合约序号
2. balance： 地址拥有Wei的数量。1Ether=10^18Wei
3. storageRoot： Merkle Patricia树的根节点Hash值
4. codeHash：此账户EVM（以太坊虚拟机）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Merkle Priticia Tree中。区块链就是一群节点来维持的。主要有全节点和轻节点，全节点就是整颗Merkle Tree，轻节点仅下载链的头，从创世纪块到当前块的头，不执行任何的交易或检索任何相关联的状态。

任何节点想要验证一些数据都可以通过Merkle证明来进行验证，Merkle 证明的组成：

1. 一块需要验证的数据
2. 树的根节点Hash
3. 一个"分支"（从 chunk到根这个路径上所有的hash值）

这和传统Merkle Tree的验证路径是一样的。

Gas和费用

以太坊网络上的交易而产生的每一次计算，都会产生费用，这个费用是以称之为”gas”的来支付。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。gas不仅仅是用来支付计算这一步的费用，而且也用来支付存储的费用。存储的总费用与所使用的32位字节的最小倍数成比例。

交易和信息

以太坊是一个基于交易的状态机。换句话说，在两个不同账户之间发生的交易才让以太坊全球状态从一个状态转换成另一个状态。一个交易就是被外部拥有账户生成的加密签名的一段指令，序列化，然后提交给区块链。

有两种类型的交易：消息通信和合约创建(也就是交易产生一个新的以太坊合约)。所有交易都包含如下部分：

1. nonce：发送者发送交易数的计数
2. gasPrice：发送者愿意支付执行交易所需的每个gas的Wei数量
3. gasLimit：发送者愿意为执行交易支付gas数量的最大值。
4. to：接收者的地址。
5. value：从发送者转移到接收者的Wei数量。
6. v,r,s：用于产生标识交易发生者的**签名**
7. init（只有在合约创建交易中存在）：用来初始化新合约账户的EVM代码片段。
8. data（可选域，只有在消息通信中存在）：消息通话中的输入数据(也就是参数)。

交易执行

所有的交易必须都要符合最基础的一系列要求，包括：

- 交易必须是正确格式化的RLP。”RLP”代表Recursive Length Prefix，它是一种数据格式，用来编码二进制数据嵌套数组。以太坊就是使用RLP格式序列化对象。
- **有效的交易签名。**
- 有效的交易序号。
- 交易的gas limit 一定要等于或者大于交易使用的intrinsic gas

其他

以太坊还包括合约创建，消息通信，执行模式等等，但是与我们的ECDSA关系不大了。

ECDSA在以太坊中应用

在以太网中，如果在签名时使用同一个私钥，选择了相同的K值，输出了两个签名，那么我们就可以使用课上介绍的方法，反推以太坊账户的私钥。

- 对于一个固定的椭圆曲线，G是相同的，R = kG。
- 如果有对消息hash值 e_1 和 e_2 的签名$sig(r, s_1)$和$sig(r, s_2)$，
- 我们就能推$s_1 - s_2 = k^{-1}(e_1 + rd) - k^{-1}(e_2 + rd) = k^{-1}(e_1 - e_2)$，然后得到$k = \frac{e_1 - e_1}{s_1 - s_2}$
- 然后可以计算私钥$sk = dG = \frac{s_1 * k - e_1}{r} * G$