区块链中隐私的保护

Protection of data and privacy in blockchain

CONTENT

壹

区块链介绍

What is blockchain

- 区块链是去中心化的账本
- 什么是去中心化：与中心化相对，去中心化说明分布全球的各个节点都保存这一模一样的信息
- 什么是账本：账本记录着交易信息的列表。

总结一下，区块链由区块组成链表，每一个区块都包含了交易信息。全球所有节点的保存的区块链都是相同的。

贰

区块链中交易的验证

Consensus mechanism

- POW共识机制:

当区块链当中的某一个节点接受的某个账户发出来的交易时，节点接受这笔交易，并把它插入到pendingTransaction列表当中，这个列表存放的是所有等待验证的交易。接着所有矿工会开始竞争解决一个数学难题，谁先解决这个难题，谁就获得打包这些交易成区块的权力。当矿工打包这些交易之前，会对交易进行验证（验证sender是否有效且sender的钱是否大于等于他转出的钱，验证转出的地址是否有效）。

Consensus mechanism

- 数学难题:

◆ 比特币工作量证明中的数学难题如下：

SHA256（ SHA256 （Version + HashPreBlock + Merkle_root +

Timstamp + Bits + Nonce））<= 目标值

➢ Nonce：矿工不断尝试的随机数，小于TargetHash的Nonce就是答案。

➢ 难度数：目标哈希值，每隔2016个区块从新计算一次，难度值的计算方法如下。

新难度值 = 旧难度值 *（过去2016个区块花费的时长 / 20160 分钟）

目标值 = 最大目标值 / 难度值

Consensus mechanism

- 隐私的泄漏: 这里我们可以看到，在验证交易有效性的时候，我们必须要知道交易里面的内容，比如：交易的发起者，交易的接受者，交易的金额等。也就是说目前以太坊上的所有交易都是公开的。
- 那么假设这么一种情况，我在以太坊上的账户是A。正常情况下，别人是没法通过A获取到我的身份的。假如有一天，我嘴贱跟夏威说，我在3点整转了404.4个以太币给刘丽锋。夏威回去后立马查了一下以太坊在3点的所有转移价值为404.4的交易，发现只有一笔。这一下，夏威不仅知道了A是我的账户，还知道了刘丽锋的账户B。
- 以后我的所有在以太坊的转账都可以被夏威查到，刘丽锋在以太坊上的转账也能被夏威看到。

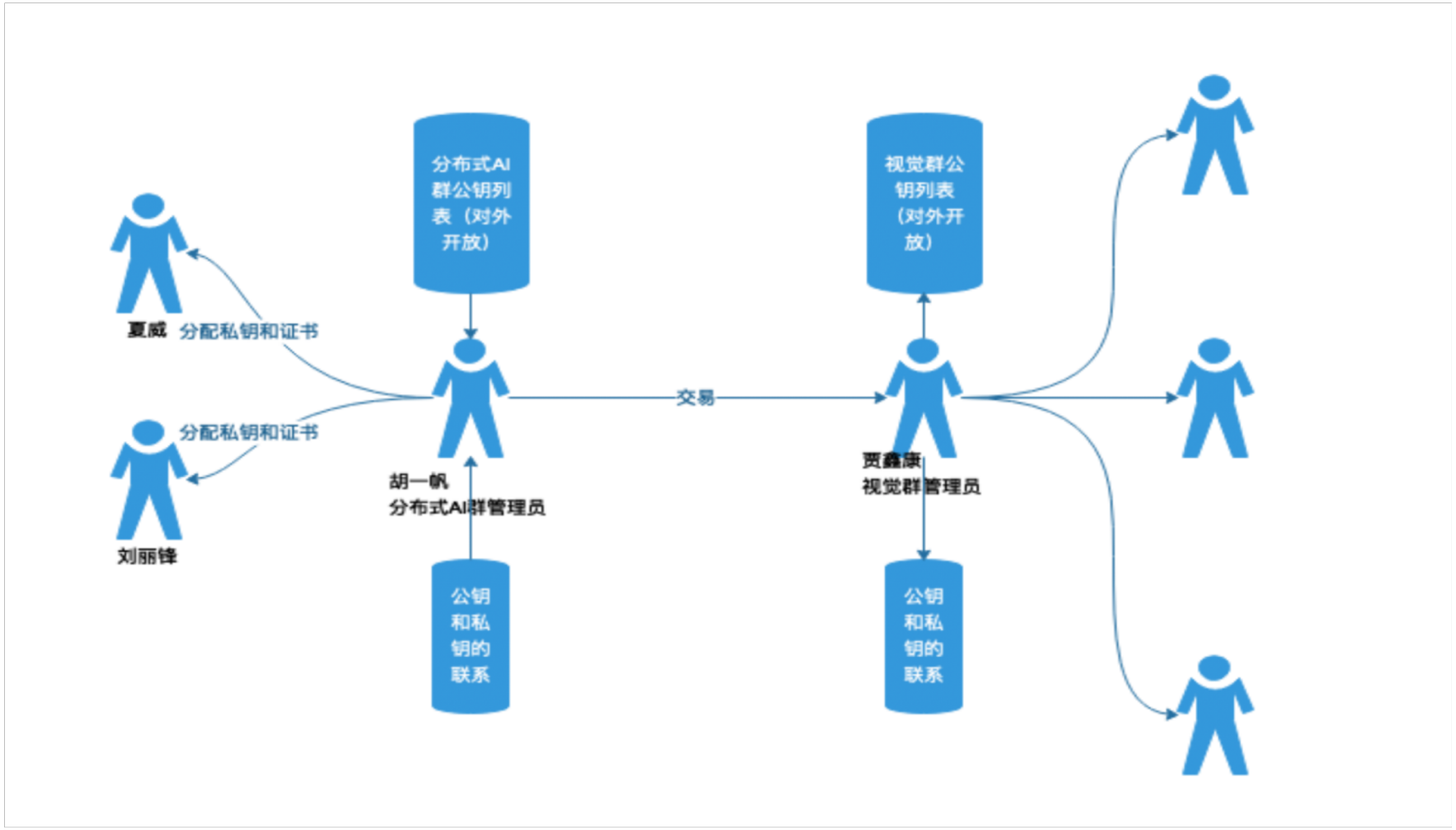
叁

群签名和环签名

群签名（Group signature）

- 群签名，群签名可以有效地防止交易发送者和交易接受者的地址的泄漏
- 群签名的工作原理：假设有两个群。分别命名为群A和群B。群A的管理员给每一个新加入的群成员分配私钥和公钥证书。所有群成员的公钥组成群管理员的群公钥列表，群管理员只暴露群公钥给外部。当群A的某一个群成员想要给群B发起交易（注意：这里的交易只有金额）时，先对这笔交易用自己的私钥进行签名，然后发给群B。群B的某一个接受者用群A的群公钥打开了交易的签名，说明这笔交易确实来自于群A，但是却不知道是群A的哪一个人发的。
- 假如说群A的管理员想查看一下那笔发给群B的交易是谁签名的，因为管理员知道每一个群成员的公钥，因此当他用某一把公钥解开了签名，那么他就知道签名的认识谁，这也是对交易进行**监管**的方法之一。

群签名（Group signature）



群签名（Group signature）的问题

群签名的问题：

1.群签名无法对交易的值进行保护。

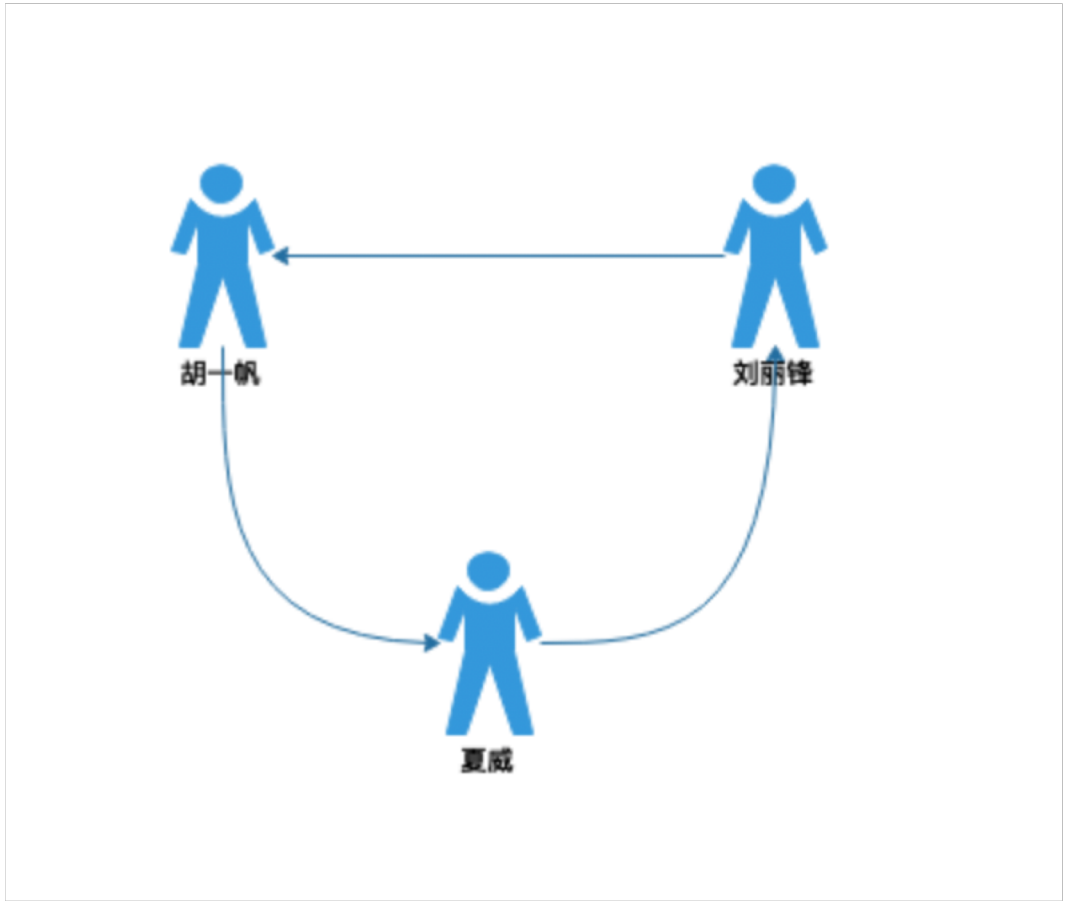
2.群签名存在致命的中心化问题。假如说群A的管理员偷偷地将公私钥列表交给了某人，那么这个人就可以伪造群A，发出交易给其他人。

3.群签名群成员越多越好，可能会导致群管理员所需要保存的群公钥列表很大。

环签名（Ring signature）

- 环签名，环签名是群签名的进化版，成功的解决了群签名中群管理员的中心化问题。
- 环签名的工作原理：群A中的所有群成员都拥有公钥和私钥。若群中小红要发送一笔交易，小红先用自己的私钥给交易签名，再按一定的随机规则在群中循环，直到根据规则选择出某个人来发送这笔交易。群B的某一个接受者用群A的群公钥打开了交易的签名，说明这笔交易确实来自于群A，但是却不知道是群A的对该交易签名的人是谁。

环签名（Ring signature）



环签名（Ring signature）的问题

- 环签名不能够隐藏交易的值。
- 环签名无法引入监管。因为没有一个节点拥有群中所有人的公私钥映射表

肆

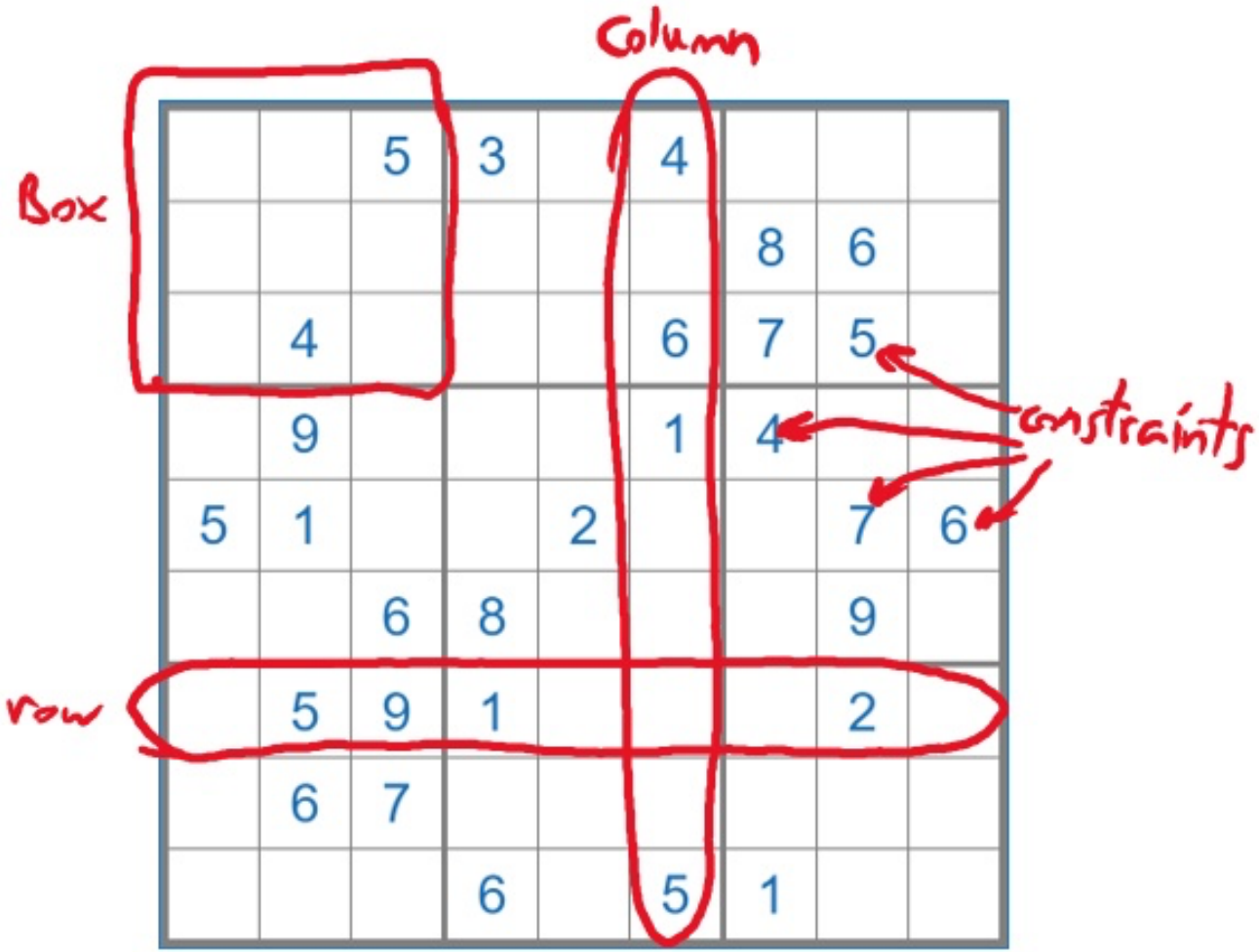
零知识证明

零知识证明（Zero knowledge proof）

- 零知识证明要解决什么问题：

零知识证明就是要在我不告诉你问题答案的情况下，让你相信我知道答案。

零知识证明（Zero knowledge proof）



零知识证明（Zero knowledge proof）

- 首先，我先把答案都填到数独的空格里面，然后背面朝上，不让夏威看见。
- 然后，我让夏威随机选择一种方式（行，列和九格）验证。例如，选择了行，则将每一行格子里的数据随机打乱然后给夏威，如果每一行都有不重复的1~9这9个数字，那么通过行验证。
- 之后，夏威可以验证其余两种方式，若其余的两种方式也通过验证
- 夏威相信我真的知道答案，但是他却不知道答案是什么。

零知识证明（Zero knowledge proof）

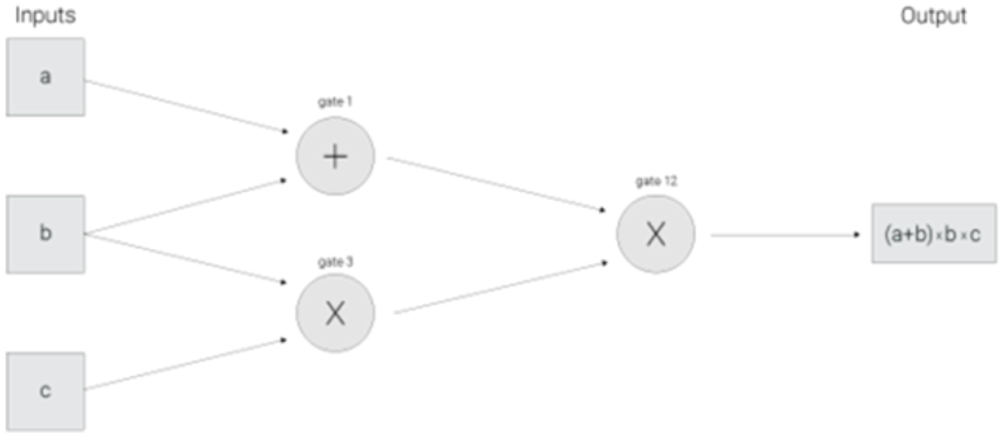
- 零知识证明在区块链上如何应用：

区块链交易信息之所以要在区块链上公开，是为了进行交易的验证。那么如果我们可以用零知识证明，使得验证节点在不知道我交易信息的情况下认可我的交易信息有效，那就可以避免在区块链上公开交易信息。

那么如何做到这一点呢？

零知识证明（Zero knowledge proof）

- 首先，我们在验证者和被验证者之间生成一串公共参数。这个公共参数在这两个对象之间共享。
- 其次，我们将验证节点的验证交易函数进行分解。将其分解成一个一个算术单元。比如一个boolean类型的算术单元为一系列与，或，非操作的组合。



- 我们将验证函数分解之后，我们就得到好多加，减，乘，除的运算单元。

零知识证明（Zero knowledge proof）

- 我们可以将每一个的算术单元视为是对交易信息的一个约束，即交易信息要满足验证函数的这一个约束。
- 之后，根据《Quadratic Span Programs and Succinct NIZKs without PCPs》这篇论文，可以将这些约束聚合成一个的多项式方程的形式。
- 被验证者紧接着通过之前那一串公共参数生成一些点（用这些点代替交易信息）。如果这些点满足验证者的多项式方程，那么说明被验证的交易是正确的。但是被验证者不能告诉验证者具体的点（零知识证明），防止点信息的泄漏。

问题就变成了我不告诉你我的点，但我要让你相信我的点满足你的多项式方程。

举个例子！

零知识证明（Zero knowledge proof）

1.假设验证函数最终转化后的多项式方程为x+y=7

2.被验证者的点为（x,y）

3.我们首先构造一个同态隐藏函数E，E要满足一下三条性质：

- 从E(x)推不出x
- 不同的x的E(x)不同
- 如果知道E(x)和E(y)那么能推出x和y做自变量的同态隐藏函数，例如：E(x+y)，E(x*y)

4.被验证者将E(x)和E(y)发给验证者，验证者根据E(x)和E(y)推出E(x+y)，如果验证者发现E(x+y)==E(7)的值，那么验证者就承认验证者的点确实满足条件，交易得到确认。

如何构造同态隐藏函数E呢？

零知识证明（Zero knowledge proof）

- 我们首先获得一个素数P
- 我们已知在集合{1,2,.....,P-1} 任意选择几个数相乘得K，（K mod P）也属于{1,2,.....,P-1}
- 我们在从集合{0,1,2,.....,P-2}中任选一个数a，从集合{1,2,.....,P-1}中任选一个数g，我们可知h= (g^a mod P)属于{1,2,.....,P-1}
- 并且有离散对数问题可知，当P比较大时，对于给定的h和g，求a的时间复杂度为O(2^n)，n为P的二进制长度，因此我们看到在目前的算利息啊，如果P的二进制长度达到上百，求解a几乎是不可能的。因此若 $E(x) = g^x$ ，在已知E(x)和g的情况下是推不出x的。
- 由上可知 $g^a \cdot g^b = g^{a+b \pmod{p-1}}.$ a,b都属于{0,1,2,....,P-2}，因此E(x)*E(y)==E(x+y)
- 综上所述 $E(x) = g^x$ 就是我们想要构造的同态隐藏函数，当然g属于{1,2,.....,P-1}，x属于{0,1,...,P-2}

零知识证明（Zero knowledge proof）

注意：

- 被验证者和验证者都知道这个同态隐藏函数
- 验证者根据公共参数生成的点要满足同态隐藏函数的条件

零知识证明可以将所有交易信息完全匿名，因此零知识证明是区块链隐私保护的中流砥柱

2018

感谢您的聆听与观看

Lorem ipsum dolor sit amet, consectetur adipisicing elit.