**NEO 的技术实现**

共识机制：DBFT

DBFT 全称为 Delegated Byzantine Fault Tolerant，是一种通过代理投票来实现大规模节点参与共识的拜占庭容错型共识机制。NEO 管理代币的持有者通过投票，可以选出其所支持的记账人。随后由被选出的记账人团体通过 BFT 算法，来达成共识并生成新的区块。投票在 NEO 网络持续实时进行，而非按照固定任期。

DBFT 对由 n 个共识节点组成的共识系统，提供 f=⌊(n-1)/3⌋ 的容错能力，这种容错能力同时包含安全性和可用性，可以抵抗一般性故障和拜占庭故障，并适用于任何网络环境。DBFT 具有良好的最终性，一个确认即最终确认，区块无法被分叉，交易也不会发生撤销或回滚。

在 NEO 的 DBFT 共识机制下，每 15~20 秒生成一个区块，交易吞吐量实测可达到约 1000tps，在公有链中性能优秀。通过适当优化，有能力到达 10000TPS，可以支持大规模的商业化应用。

DBFT 结合数字身份技术，使得记账人可以是实名的个人或机构。从而使得冻结、撤销、继承、找回、司法判决过户等非常规操作成为可能。这有利于合规性金融资产在 NEO 网络中的登记发行。NEO 网络计划在必要的时候支持此类操作

dBFT共识机制是众多为了解决这一问题中的一种方案。在这个机制当中，存在两个参与者，一个是专业记账的“记账节点”，一个是系统当中的普通用户。普通用户基于持有权益的比例来投票决定记账节点，当需要通过一项共识时，在这些记账节点中随机推选出一名发言人拟定方案，然后由其他记账节点根据拜占庭容错算法，即少数服从多数的原则进行表态，如果超过66%的节点表示同意发言人方案，则共识达成；否则，重新推选发言人，重复投票过程。

NEO采用了一种更加类似于EOS的DPOS共识机制，**改良拜占庭将军算法——DBFT**，全称为 Delegated Byzantine FaultTolerant，是一种通过代理投票来实现大规模节点参与共识的拜占庭容错型共识机制。

DBFT和DPOS一样，都是由选出来的超级节点（记账节点）进行协作记账，在弱中心化的情况下，实现较高的效率。区别在于具体的算法和节点的选择。

DBFT的算法大致原理是这样。参与记账的是超级节点，普通节点可以看到共识过程，并同步账本信息，但不参与记账。总共n个超级节点分为一个议长和n-1个议员，议长会轮流当选。每次记账时，先有议长发起区块提案（拟记账的区块内容），一旦有至少（2n+1)/3个记账节点（议长加议员）同意了这个提案，那么这个提案就成为最终发布的区块，并且该区块是不可逆的，所有里面的交易都是百分之百确认的。

（这里说明一下，在POW的机制，理论上所有的交易都最多只是无限接近100%，而永远达到100%。在类似BitShare的DPOS的机制下，大约需要45秒才能进入不可逆状态。而EOS的DPOS+BFT机制，一个区块可以在一秒内进入不可逆状态。）

如果在一定时间内还未达成一致的提案，或者发现有非法交易的话，可以由其他记账节点重新发起提案，并且当至少[(2n+1)/3]个记账节点同意后，就形成最终确定的区块了。

作者：区块链朋克

链接：https://www.zhihu.com/question/49448972/answer/352978330

来源：知乎

著作权归作者所有。商业转载请联系作者获得授权，非商业转载请注明出处。

DBFT算法在正常情况下，也可以迅速达成共识，并且有良好的最终性（区块生成之后最终确认，区块无法被分叉，交易也不会发生撤销或回滚）。缺点在于，DBFT最多只能容忍[( −1)/3 ]个记账节点为恶意节点，而不影响整体网络环境的有效运行。理论情况下，安全性相比POW（至少50%以上才存在明显安全隐患）及DPOS+BFT（2/3以上记账节点为恶意节点才会导致不可逆的恶意区块）较低。此外，由于DBFT的算法并未经过大规模验证，整体系统的实际安全性也暂时存疑。在 NEO 的 DBFT 共识机制下，每 15~20 秒生成一个区块，根据白皮书的说法交易吞吐量可达到约 1000tps，通过适当优化，能力可达 10000TPS。和EOS虽然还差一个数量级，但比起TPS7左右的比特币已经是莫大的提高。NEO经常被质疑过于中心化，说的就是DBFT共识机制下投票选出的仅有7个共识节点。这些代理节点是通过静态选出的，并完全由项目方部署（DPOS下的记账节点是会动态变化的）。NEO的创始人达鸿飞的解释是NEO处于早期，集中有利于开发效率。随着NEO核心协议的逐步稳定，共识节点数量会有所提升，但无论如何，NEO都是一种弱中心化的共识机制。

BFT算法在实用拜占庭容错共识（PBFT）基础上进行了改进：PBFT主 要应用于存储系统，具有主从之间的区别，小蚁将其改为对等；PBFT参与共识的人为固定的、手动设计的名单，小蚁系统中加入了一些选举名单，实现了动态的 加入记账和动态的退出记账，是一个在已有论文上的改进。

这种共识机制的选择是小蚁团队认为至今为止的最优选择，但不代表未来不会有所改进，正如以太坊对于其共识机制的改进一样。

作者：张子豪  
链接：https://www.zhihu.com/question/49448972/answer/116032837  
来源：知乎  
著作权归作者所有。商业转载请联系作者获得授权，非商业转载请注明出处。

总结：

参与记账的是超级节点，普通节点可以看到共识过程，并同步账本信息，但不参与记账。总共n个超级节点分为一个议长和n-1个议员，议长会轮流当选。每次记账时，先有议长发起区块提案（拟记账的区块内容），一旦有至少（2n+1)/3个记账节点（议长加议员）同意了这个提案，那么这个提案就成为最终发布的区块，并且该区块是不可逆的，所有里面的交易都是百分之百确认的。

1. DBFT是一种弱中心化的共识机制，与dpos类似；而pos和pow是完全去中心化的；
2. DBFT的好处是，具有最终性，也就是说只要出来的区块，一定是不可逆的；而dpos、pos、pow理论上都是存在分叉导致交易会被取消的可能性。
3. DBFT的问题在于，安全性较低，只要有(n-1)/3个节点是恶意节点，就会使得系统无法正常运行；而pow、pos都可以容忍低于50%的算力（币天），Eos的dpos，也可以保证50%以下的安全性，以及只要不超过2/3的节点为恶意节点，就不会进入不可逆状态。
4. DBFT还有一个问题是，相比dpos，出块速度还是较慢，并且eos的dpos可以在一秒内进入不可逆状态（等于DBFT的最终性）。

因此综合来看，DBFT牺牲了完全的去中心化，改用弱中心化的机制，提升性能；但是相比eos的dpos+BFT的模式，似乎还是处于下风。

知道是阅读量到了质变，还是后面引用的这篇文章太好，我终于对BFT-DPOS有了更深刻的理解

POS:

出块不再由算力说了算，由节点持有的stake说了算，解决了POW算力资源被大量无用消耗，但由于无条件信任代表，节点作恶非常容易 , 比如nothing at stake攻击

DPOS：

动态产生一定数量的代表比如21个节点代表，只由这21个节点生成区块，降低了作恶的概率，同时用事后惩罚限制节点代表作恶行为，但始终不够安全(怎么惩罚，使用抵押物？抵押物需要多少？作恶导致的失败交易的损失都是可以量化的吗？如果不能量化抵押物就没有意义)，同时确认速度确实也慢(需要14个block）

BFT+DPOS：

    一个区块生产后通过BFT协议立马确认，解决了DPOS确认慢问题，需要超过1/3节点才能作恶，相比POW 51%作恶条件，这个1/3值相对来说还是偏低了点，但是这是一个折中，也只能这样了

BFT + DPOS + 小块：

    该协议是为了加快出块速度，该改进的核心思想是让同一个节点产生6个小块。这个能提升性能的核心原因是6个小块的产生没有等待确认的环节。在BFT+DPOS算法中，一个节点要创建新块，必须等上一个块被确认(或者超时)，因为它不确定上一个节点是否作恶是否会被确认，所以它只能等。但是如果上一个块时也是该节点产生的，它自然不需要等上一块确认，因为它知道上一个块是真实的，不是作恶的块，最后肯定会被确认，所以安心生产下一块。

可见DPOS+BFT+小块机制中，区块生成和确认以pipeline的方式进行，是同时进行的，加快了出块速度。

何为作恶：

    各种共识算法，节点基本都会进行数据检验功能，因而数据伪造作恶基本不可能。那真正的作恶是什么？

   主动作恶：节点主动生产两个区块，给一部分节点发送新区块A，另外一部分节点发送区块A1,这样系统就产生分叉了。

    被动作恶： 块传播因为网络延时或者中断，会导致一部分节点缺失该块的信息，进而会产生分叉，然后后续区块生产者发现分叉后，如果遵循只选择一个分支生产区块就会让分叉收敛(能收敛最后就能达成共识)，但是如果继续保持并加重这个分叉，最后就会导致整个链变成一颗森林，最后节点就永远没法达成共识, 这是被动作恶。比如POS的noting at stake攻击，当系统出现分叉时，生产者基于自己的利益会在两个分叉区块上各自生成一个区块(这样不管哪个分叉获胜，自己都能获得收益)，这其实间接助长和加重了分叉

    因而要阻止这些作恶行为，有三种方法:

    1)机制保证，让生产者没有时间来生成2个区块(POW)

    2)利益保证，生成两个区块最后肯定没用(BFT算法, 大家最后只会认可一个快)，生产者获取不了什么好处，自然没必要生产两个区块

    3）事后惩罚，比如DPOS，这种方法其实没啥用，因为事后发现作恶节点后，系统没法回溯区块，只能惩罚作恶节点，但是这个作恶损失是不确定的(前面提到过)，起不到惩罚作用，且抵押物模式是一种限制，不利于生态发展

引用原文

区块链中最重要的便是共识算法，比特币使用的是POW（Proof of Work，工作量证明），以太币使用的POS（Proof of Stake，股权证明）而EOS使用的是BFT-DPOS。

什么是BFT-DPOS呢？即拜占庭容错式的委任权益证明。

要想明白BFT-DPOS的运行机制，首先就要先明白什么是DPOS。

由于POW在比特币的共识算法中极大地消耗了算法的资源。而且会有算法集中的问题，所以在2014年的时候Dan Larimer提出了一个相较于POW来说更加高效，轻便的共识机制即DPOS。该共识机制一边能让网络成本小型化，另一方面有回复语每个持股人一定的投票权。

这些超级节点呢能够：供相关计算资源和网络资源，保证节点的正常运行;当轮到某超级节点拥有出块权时，超级节点收集该时段内的所有交易，并对交易验证后打包成区块广播至其他超级节点，其他节点验证后把区块添加到自己的数据库中。这种共识机制采用随机的见证人出块顺序，出块速度为 3 秒，交易不可逆需要45秒。为什么需要 45 秒呢？因为 DPoS 下，见证人生产一个新区块，才表示他对之前的整条区块链进行了确认，表明这个见证人认可目前的整条链。而一个交易要达到不可逆状态，需要 三分之二以上的见证人确认，在 EOS 里就是 14 个见证人。DPoS共识算法也有极强的抗分叉能力，因为区块添加到一条区块链分叉的速率与拥有该共识的超级节点比例是相关的。当一个超级节点设法在两条分叉上同时生产区块时，EOS的持有者会在下一轮投票中将该超级节点删掉，并且EOS社区会给予相关恶意节点一定的惩罚。因此，在一般情况下，使用DPoS的EOS都是很难经历分叉的。

其次，我们还要明白BFT所代表的的意义。

拜占庭容错技术（Byzantine Fault Tolerance，BFT）是一类分布式计算领域的容错技术。拜占庭假设是对现实世界的模型化，由于硬件错误、网络拥塞或中断以及遭到恶意攻击等原因，计算机和网络可能出现不可预料的行为。拜占庭容错技术被设计用来处理这些异常行为，并满足所要解决的问题的规范要求。

拜占庭容错技术来源于拜占庭将军问题。拜占庭将军问题是Leslie Lamport在20世纪80年代提出的一个假象问题。拜占庭是东罗马帝国的首都，由于时拜占庭罗马帝国国土辽阔，每支军队的驻地分隔很远，将军们只能靠信使传递消息发生战争时，将军们必须制订统一的行动计划。然而，这些将军中有叛徒，叛徒希望通过影响统一行动计划的制定与传播，破坏忠诚的将军们一致的行动计划。因此，将军们必须有一个预定的方法协议，使所有忠诚的将军能够达成一致，而且少数几个叛徒不能使忠诚的将军做出错误的计划。也就是说，拜占庭将军问题的实质就是要寻找一个方法，使得将军们能在一个有叛徒的非信任环境中建立对战斗计划的共识。在分布式系统中，特别是在区块链网络环境中，也和拜占庭将军的环境类似，有运行正常的服务器（类似忠诚的拜占庭将军），有故障的服务器还有破坏者的服务器（类似叛变的拜占庭将军）。共识算法的核心是在正常的节点间形成对网络状态的共识。

简单形容就是：通过在一群数量有限的节点中，使用轮换或者其他算法来筛选出某个节点作为主节点。并且赋予该节点出块的权利。在主节点是将该时段的交易打包成区块后用自己的私钥对该区块签名，并将其广播到所有节点。当主节点收到至少三分之二的不同节点的签名区块后，则该区块完成了所有节点的验证成为不可逆区块串联到区块链中。

BFT与DPOS二者相结合就诞生了BFT—DPOS共识算法。

、为了挖掘 EOS 系统的性能，Daniel Larimer 在以上基础上又进行了修改。首先，他将出块速度由 3 秒 缩短至 0.5 秒，理论上这样可以极大提升系统性能，但带来了网络延迟问题：0.5 秒的确认时间会导致下一个出块者还没有收到上一个出块者的区块，就该生产下一个区块了，那么下一个出块者会忽略上一个区块，导致区块链分叉（相同区块高度有两个区块）。比如：中国见证人后面可能就是美国见证人，中美网络延迟有时高达 300ms，很有可能到时美国见证人没有收到中国见证人的区块时，就该出块了，那么中国见证人的区块就会被略过。

为解决这个问题，Daniel Larimer 将原先的随机出块顺序改为由见证人商议后确定的出块顺序，这样网络连接延迟较低的见证人之间就可以相邻出块。比如：日本的见证人后面是中国的见证人，再后面是俄罗斯的见证人，再后面是英国的见证人，再后面是美国的见证人。这样可以大大降低见证人之间的网络延迟。使得 0.5 秒的出块速度有了理论上的可能。

为了保证万无一失，不让任何一个见证人因为网络延迟的意外而被跳过，Daniel Larimer 让每个见证人连续生产 6 个区块，也就是每个见证人还是负责 3 秒的区块生产，但是由最初的只生产 1 个变成生产 6 个。最恶劣的情况下，6 个区块中，最后一个或两个有可能因为网络延迟或其他意外被下一个见证人略过，但 6 个区块中的前几个会有足够的时间传递给下一个见证人。

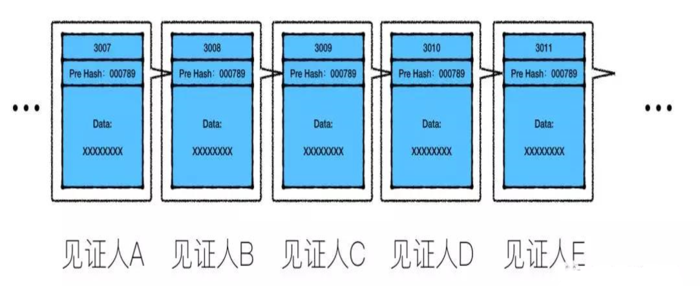
再来讨论 BFT-DPoS 的交易确认时间问题：每个区块生产后立即进行全网广播，区块生产者一边等待 0.5 秒生产下一个区块，同时会接收其他见证人对于上一个区块的确认结果。新区块的生产和旧区块确认的接收同时进行。大部分的情况下，交易会在 1 秒之内确认（不可逆）。这其中包括了 0.5 秒的区块生产，和要求其他见证人确认的时间

使用上述BFT-DPoS协议就可以使得EOS的出块间隔从原来的3秒降低到500毫秒，这也使得跨链通信的时延大大缩短，单位时间内可确认的交易数量大大提升。笔者相信如果这样的机制在EOSIO1.0的正式版本中成功实现，那无疑是区块链技术向支持百万级别用户的目标迈出的巨大一步。

---------------------

**传统 DPoS**

DPoS（拜占庭容错的委托股权证明），对于PoS机制的加密货币，每个节点都可以创建区块，并按照个人的持股比例获得“利息”。DPoS是由被社区选举的可信帐户来创建区块。为了成为正式受托人，用户要去社区拉票，获得足够多用户的信任。用户根据自己持有的加密货币数量占总量的百分比来投票。DPoS机制类似于股份制公司，普通股民进不了董事会，要投票选举代表（受托人）代他们做决策。  
DPOS使用随机的见证人出块顺序，出块速度为 3 秒，21个节点需要14个见证人，2/3 以上的见证人确认的交易，就是不可逆的交易了，所以交易不可逆需要45秒。

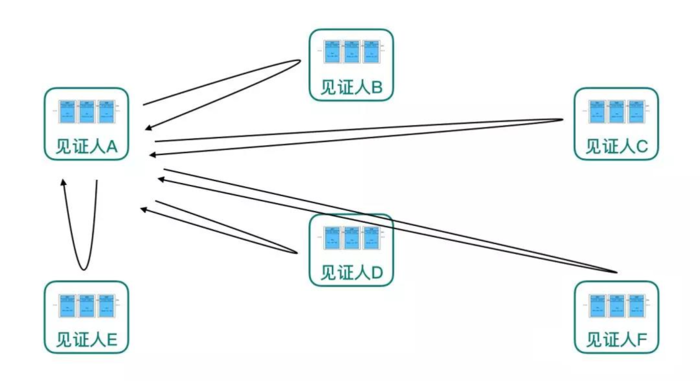


图片1.png

见证人出块顺序，出块速度为 3 秒，每个出块者生产一个区块。 2/3 以上的见证人确认的交易， 21个节点需要14个见证人，需要15\*3s=45s，此时交易就不可逆了。

**拜占庭容错（BFT）**

借鉴 PBFT（Practical Byzantine Fault Tolerance，拜占庭容错算法）的机制。在传统 DPoS 共识机制中，我们让每个见证人在出块时向全网广播这个区块，但即使其他见证人收到了目前的新区块，也无法对新区块进行确认，需要等待轮到自己出块时，才能通过生产区块来确认之前的区块。  
见证人出块时向全网广播，其他见证人收到新区块后，立即对此区块进行验证，并将验证签名完成的区块立即返回出块见证人，不需等待其他见证人自己出块时再确认。



图片2.png

出块见证人生产了一个区块，并全网广播，然后陆续收到了其他见证人对此区块的确认，在收到 2/3 见证人确认的瞬间，区块（包括其中的交易）就不可逆了。交易确认时间大大缩短，从 45 秒缩短至 3 秒左右（主要为等待生产区块的时间）。

**BFT-DPoS共识**

Daniel Larimer 在上述基础上又进行了修改。将出块速度由 3 秒 缩短至 0.5 秒，理论上这样可以极大提升系统性能，但带来了网络延迟问题：0.5 秒的确认时间会导致下一个出块者还没有收到上一个出块者的区块，就该生产下一个区块了，那么下一个出块者会忽略上一个区块，导致区块链分叉（相同区块高度有两个区块）。  
比如：中国见证人后面可能就是美国见证人，中美网络延迟有时高达 300ms，很有可能到时美国见证人没有收到中国见证人的区块时，就该出块了，那么中国见证人的区块就会被略过。



图片3.png

为解决这个问题，Daniel Larimer 将原先的随机出块顺序改为由见证人商议后确定的出块顺序，这样网络连接延迟较低的见证人之间就可以相邻出块。每个见证人连续生产 6 个区块，也就是每个见证人还是负责 3 秒的区块生产，但是由最初的只生产 1 个变成生产 6 个。最恶劣的情况下，6 个区块中，最后一个或两个有可能因为网络延迟或其他意外被下一个见证人略过，但 6 个区块中的前几个会有足够的时间传递给下一个见证人。  
交易确认时间问题：每个区块生产后立即进行全网广播，区块生产者一边等待 0.5 秒生产下一个区块，同时会接收其他见证人对于上一个区块的确认结果。新区块的生产和旧区块确认的接收同时进行。大部分的情况下，交易会在 1 秒之内确认（不可逆）。这其中包括了 0.5 秒的区块生产，和要求其他见证人确认的时间。  
分叉问题：所有节点都不会自动转移到分叉链上，因为分叉链上没有区块生产者可以满足上面所说的15/21法则。即使多数见证人想分叉区块链，也只能以相同的速度（0.5秒）与主链竞争，就算主链只剩下一个见证人，分叉链也永远不会追上主链，保证了系统的稳定。

**BFT-DPOS共识机制总结**

21个超级节点（主力见证人节点） + 100个备选见证人节点；  
0.5秒出块时间 + 1秒全网确认；  
每个主力见证人节点通过协商方式确定各自出块顺序，并且每轮产生6个区块以减少网络延时的影响，见证人间按顺序处理交易，可尽量减少地理影响；  
当21个主力见证人的15个确认交易后，交易即不可逆转；  
当达到不可逆转状态后，就无法分叉。

**BFT-DPOS共识机制缺点**

1.不是完全去中心化，可能会有多个中心之间共同串通而损害整个社区利益的行为。  
2.依赖于投票机制  
投票制度其实有以下问题，首先有可能最后投票的参与度会很低，影响投票结果。其次也会可能有这种情况，例如用户把币都存在了交易所，交易所有可能会代替他们去投票，但是用户并不是很在意到底交易所会把票投向何处。也就是说有时候代币持有者的兴趣点和用户的是可能不完全一样的。

●共识机制为区块链中最核心要素，不同公链拥有不同的解决方案。其中EOS在之前采用DPOS时，出块时间是3秒。加上BFT之后，出块时间可以提高至0.5秒，并且BFT机制对整个区块链容错能力、稳定性的提升有着重大意义。

●文中用比较清楚的语言针对BM在上周上传了关于DPOS+BFT混合共识的视频做一些延展性的解读。

●由于视频掺杂大量的口语，并且需要一定的基础知识，许多读者仍然搞不清楚其技术原理。在这里对BM的视频，让大家更加明白DPOS+BFT到底是什么，为什么能达到高性能和拜占庭容错（BFT）。

●BM上传的视频大家可以在文末找到，包含中文字幕。

●注意，技术文阅读并理解本文需要一定的基础知识。

 什么是BFT？

BFT的全称是Byzantine fault tolerance，即拜占庭容错。是解决拜占庭将军问题中，当存在叛徒时，叛徒用尽了各种手段来破坏时，将军们仍然能够达成达成共识。所以叫拜占庭容错，容错的意思就是容纳节点出现错误（或者叛徒），网络仍然能达成一致的行动，正常运作。

所谓DPOS+BFT共识，是说EOS的DPOS共识能够有足够稳健的拜占庭容错能力，能够容纳网络中一定的叛徒（或者出错）节点。

BM的视频里大致分为两块内容：

**1、「DPOS的拜占庭容错能力」**

**2、「EOS的DPOS如何提高出块速度和性能」**

一、DPOS的拜占庭容错能力

DPOS的特殊性，也是奠定拜占庭容错能力的基础框架，是它的算力节点是固定21个人，并且由大型的机构运营节点，其信息也相对透明，例如运营节点的地点、运营的情况等等。并且DPOS的算力节点是固定出块顺序的，固定地从A到B到C······。

在这种情况下，其实DPOS是拜占庭容错的特殊解，如何理解特殊解？原来的拜占庭容错（POW工作量证明），解决的是不限数量、随机广播同步的算力节点的容错能力，DPOS解决的拜占庭容错从两个维度降低了难度：

1、节点数量固定只有21个。并且节点信息透明。

2、固定出块顺序。每个节点跟接力棒一样，一个个往下接力出块。每个节点不能还没轮到它出块的时候，就出块。都是必须轮到再出块。如果出现出块故障，会跳过这个节点。（在下文详细讨论）

在POW或者其他的POS共识里，节点不限、随机出块顺序的问题，就变成只要解决「固定数量和固定出块顺序情况下的拜占庭问题」，其难度就大大降低。

在这种情况下，DPOS是如何解决拜占庭错误？

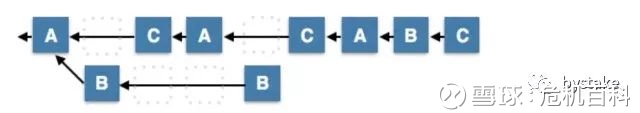
我们假设有三个节点A、B、C，出块顺序是A到比B到C。实际上EOS有21个节点，但是举例三个节点足以说明问题了。

1.1 分叉情况

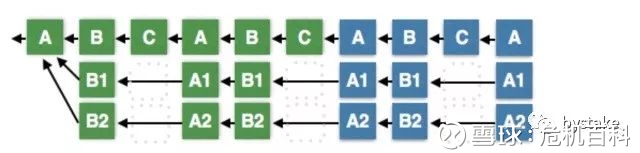
现在，B决定分叉了。在轮到B出块时候，它不再承认C和A的块，就自己出块。我们在这里假设出块时间为3s，这是比特股的出块速度。

那么B分叉出去的的链，每9秒，才能出一个块，而C、A是6s出一个块。因为即使分叉，在DPOS的机制下，还是只能等到A\C都出块后，才轮到B出块。

在这种情况下，分叉出块的速度永远追不到原来链的速度。而共识只承认最长的链。



以上是少数节点分叉，如果是三分之二的节点决定分叉呢？其实原理也是一样的。如下图。



当三分之二的节点决定分叉的时候，最后一个诚实的少数节点决定了最快、最长的那条链。分叉的B和A节点，追不上由C承认的链的增长速度。

（大毛注：这里有前提没写清楚，是作恶节点利用无利害关系(nothing at stake)多出块的情况，视频里面举了很多作恶的例子，这只举了两种。至于真的3分之2作恶，不出老链块，那么确实是会攻击成功的，也是bft2/3投票的必然结果，不过要14个节点联合也是有一定难度的）

1.2 最后不可逆块

在DPOS的共识里，最长的那条链才是真正的链。但是如何确认这条链是最长的？这里就需要「最后不可逆块」的概念。

最后不可逆块，顾名思义，就是最后那块，不能在修改的块。DPOS规定，这个区块被三分之二的节点确认，就是不可逆块。如果被最新出块的三分之二节点确认，就是最后不可逆块。

如图，绿色B就是最后不可逆块，因为被C和A确认了。



通过最后不可逆块，就能确认这条链是不是由三分之二节点签名的最长的那条链。

总结：由上文可知DPOS可以有效地防范拜占庭作恶，有着稳健的拜占庭容错。本文只列举了主要几种作恶情况，还有许多作恶情况DPOS都可以防范，不一一列举。

1.3 交易作为权益证明TaPOS

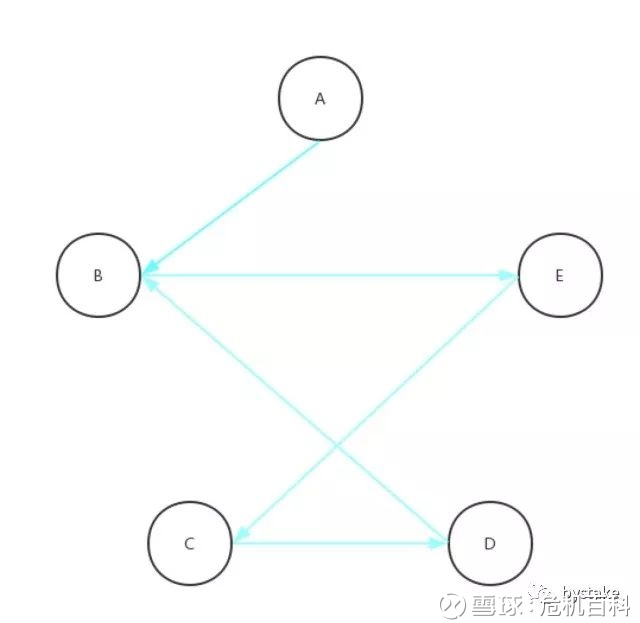
交易作为权益证明TaPOS的全称是transaction as proof of stake。有点类似比特币的「区块头」概念，即有每个最后不可逆块有一个哈希值，是来代表前一个块的数据，而前一个块又有前前块的哈希值，环环相扣。

当块数越多后，这条链就很难被替代。因为你修改一个块，所有的TaPOS值就对不上了。

二、EOS的DPOS如何提高出块速度和性能

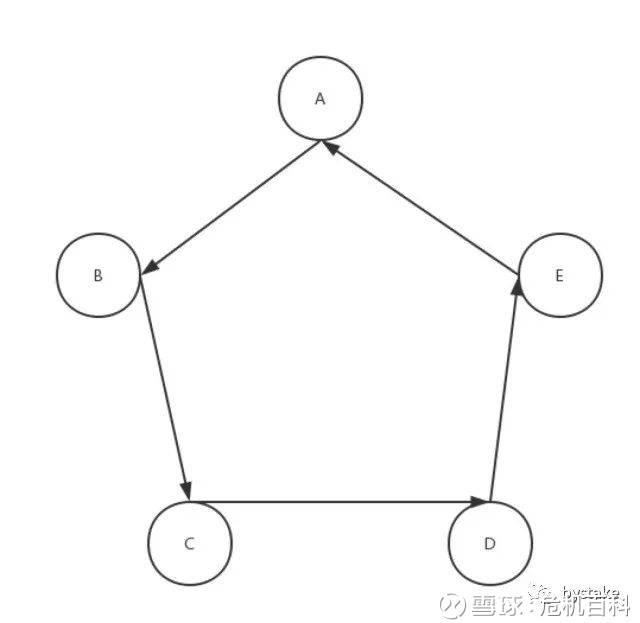
在上文提到，比特股的出块速度是3s，而EOS可以达到500ms。EOS能提高出块速度是因为定向广播。

例如在比特股和STEEM里，其广播是随机的，谁先收到块就可以接力生成新的块，那么，可能的出块顺序如下图：



因为是随机广播，所以会诞生很多同步一轮的区块路径不是最短路径。

而在EOS的机制下，节点是定向广播的。21节点的位置是透明的，会选择最短路径来规定广播顺序。如图：



在最短路径的广播顺序下，生产出最后不可逆块的速度便变快了。

如果其中有一个节点出现故障没有出块，会迅速轮到下一个区块来出产。所以当某些节点出现故障的时候，不会影响其他节点的出块速度。至于跳过节点可能产生的作恶行为的防范，上文已经提到防范的方式了

如以上提到的，每个分片和DS委员会需要分别在微块和终块上跑一个共识协议。在这一块，我们将展示在每一个分片和DS委员会中定义的共识协议的共识层。在讨论中，我们将分片和DS委员会代指为共识组。

##### 实用拜占庭容错

ZILLIQA共识协议的核心依赖于实用拜占庭容错(PBFT)协议。然而我们通过在PBFT中使用EC-Schnorr多签名来提升效率。EC-Schnorr多重签名的使用将正常情况下的通信延迟从O(n\*n)降低为O(n)，并将签名大小从O(n)减小到O(1)，其中n是共识组的大小。 在这个部分，我们提供PBFT的概述。

在PBFT中，共识组内的所有节点都按顺序排列，它有一个主节点（或领导者），其他节点称为备份节点。 每轮PBFT都有三个阶段，如下所述：

* 预准备阶段: 在这个阶段，领导者宣布该小组将应该达成一致共识的下一个记录（在我们的案例中是TX-Block）
* 准备阶段：在接收到预先准备消息后，每个节点验证其正确性并将准备消息多播给所有其他节点
* 提交阶段：在收到超过2/3\*n准备消息时，节点向组播组发送提交消息。最后，节点等待超过2/3\*n的提交消息，以确保有足够数量的节点做出相同的决定。 因此，所有诚实的节点都接受相同的有效记录。

PBFT依靠正确的领导者开始每个阶段，并在足够多节点存在时继续进行。 如果领导是拜占庭，它可能会拖延整个共识协议。 为了应对这一挑战，PBFT提供了视图更改协议来使用另一个取代拜占庭领袖。 如果节点在有限的时间内没有看到任何进展，他们可以独立宣布改变领导者的愿望。 如果超过2/3\*n个节点的法定人数决定领导者有问题，那么在已知计划中的下一个领导者就会接管。

由于在准备/提交阶段每个节点的多播，正常情况下PBFT的通信复杂度为O（n\*n）

##### 提高效率

经典的PBFT使用消息认证码（MAC）进行节点之间的认证通信。 由于MAC需要在每两个节点之间共享密钥，所以一个共识组中的节点可以在同一个记录上达成一致，其中每个节点的通信复杂度为O（n\*n）。 由于二次复杂性，当委员会有20多个节点时，PBFT变得不切实际。

为了提高效率，我们使用来源于ByzCoin的想法：

* 我们用数字签名替换MAC来有效地减少O（n）的通信开销。
* 与此同时，为了让其他节点能够验证协议，一种典型的方法是从诚实的多数收集签名并将它们附加到协议中，从而导致协商规模与协商组的大小成线性关系。 为了改善这一点，我们使用EC-Schnorr多重签名来将几个签名聚合成O（1） - 大小多重签名

然而，我们不能直接在PBFT设置中使用经典的EC-Schnorr多重签名方案。 这是因为在古典设置中，所有签名者都同意签署给定的消息，并且签名只有在所有签名者都签名后才有效。 在PBFT设置中，我们只需要在共识组中超过2/3\*n个节点签署消息。 所需的主要修改之一是为参与签名过程的签名者维护位图B. 如果第i个节点参与该过程，则B [i] = 1，否则为0。位图由领导者构建。 位图可以被任何验证者用来验证签名。 最终的协议留在附录B中。

##### Zilliqa共识

在ZILLIQA中，我们使用PBFT作为基础共识协议，并采用两轮EC-Schnorr多重签名来替换PBFT中的准备阶段和提交阶段。 下面将解释对PBFT阶段的不同修改。

* 预准备阶段: 与标准PBFT中一样，领导者将TX-Block或声明（由领导者签名）分发给共识组中的所有节点。
* 准备阶段：所有诚实的节点检查TX块的有效性，并且领导者收集来自超过2/3\*n个节点的响应。 这保证领导者提出的陈述是安全的并且与以前的所有历史一致。 签名是使用EC-Schnorr多重签名生成的。 领导者还构建签署TX块的节点的位图
* 提交阶段：为了确保超过2/3\*n的节点知道超过2/3\*n节点验证了TX-Block的事实。我们进行第二轮EC-Schnorr多重签名。 正在签署的声明是上一轮生成的多重签名。

在三个阶段结束时，就领导者提出的TX-Block将达成共识。

##### 领导者改变

在我们的共识协议中，如果领导者是诚实的，它可以不断的推动共识小组中的节点就新的交易达成协议。 但是，如果领导是拜占庭，它可以有意地延迟或丢弃来自诚实节点的消息，并减慢协议。 为了惩罚这些恶意领导者，我们的协议会定期更改每个分片的领导和DS委员会。 这可以防止拜占庭领袖在无限期的时间内拖延共识协议。 由于所有节点都是有序的，下一个领导者将以循环方式选择。

事实上，每一个微块后分片的领导者都会改变，并且在每个最后一个区块之后DS委员会的领导者也会更改。 让我们假设共识组的大小为n，那么在一个DS-epoch时期内，我们允许的最终块的最大值为n，每个最终块最多在1个分片聚合1个微块。

Algorand是权益证明(POS)的一个升级，彻底消除区块链分叉的可能性，可以在一小段时间内确认交易，Algorand的核心使用称为BA⋆的拜占庭协议，同时扩展到许多用户。即使一些用户是恶意的，网络被临时分区，Algorand也确保用户从未对已确认的交易有不同意见。在Algorand的BA⋆协议中，除了私钥之外，用户不会保留任何私有状态。

两种共识：

最终共识： BA⋆（）将不会在这一轮的任何其他块达成共识。

暂定共识：其他用户可能已经在不同的块上达成了暂定共识（只要没有用户达成最终的共识）。

这个算法每一轮的总流程，分为两个大部分——块提议和BA⋆。块提议中首先在普通用户中通过加密抽签选择出这一轮的委员会成员（委员会成员会在每一轮通讯进行替换），然后在委员会成员中再次通过加密抽签选择出这一轮的提议者，每个提议者提议一个块。然后连同哈希和证明（一种随机号码，数字签名可以很容易对其进行验证）一起传播到网络上。之后，这一轮的所有委员会成员接到消息后，通过BA⋆协议对块达成共识（暂定或最终共识）。如果这一轮是暂定共识，则只有在当后续块（后几轮的块）有达成最终共识的情况下才能确认前几轮的暂定共识块。

加密抽签。用来创建和验证区块链（如若是在一个拥有数百万用户的系统中，那么参与者的数量可能达到数千人）。这种抽签的方法基于前一个区块，选择过程是自动并且完全随机的。用户被选中的概率基于他的权重。关键技术是使用可验证的随机函数（VRFs）,以私人和非交互方式随机选择用户。允许用户私下检查是否被选择参与拜占庭协议。

每个用户在每一轮都要进行加密抽签。在输入字符串（种子+角色）上，VRFsk返回两个值：散列和证明。然后用为该角色选择的用户的预期数量τ除上所有用户的权重和，得到一个概率p。j参数表示此用户被选择了多少次。 被选择j次意味着用户以j个不同的“子用户”的身份参与。一开始j初始化为0，然后循环确定此用户的多少子用户被选择，抽签算法将区间[0,1）划分为连续区间的形式，如果hash / 2hashlen（其中hashlen是散列的比特长度）属于间隔，则用户恰好具有j个选定的子用户。

要进行加密抽签就要先获得本轮的种子seed。每一轮都会产生一个新的种子。假设现在为第r轮，在r轮的块提议期间，通过这个公式，为第r+1轮块提议计算种子seed r+1。这个种子（和相应的VRF证明π）被包含在每个提出的块中，所以一旦Algorand在r轮的块上达成一致，在r+1轮开始时，每个人都知道seed r+1。 如果块不包含有效的种子（例如，因为该块是由恶意用户提出的并且包括无效的交易），则用户将整个提出的块看作是空的，并且使用这个公式来计算第r轮的相关种子。引导种子选择的seed0的值可以在算法一开始时通过初始参与者（在声明他们的公钥之后）用分布式随机数随机选择生成。

由第一个公式可知，在选择第r+1轮的种子之前要选择每个用户的密钥sk u（r轮使用的）。

块提议。抽签确保随机选择一小部分用户，并通过其权重为每个所选用户提供优先级，可以在用户之间进行比较。由于抽签是随机的，所以可能会有多个子用户选择提出块，用户通过比较子用户优先级，采用有最高优先级的子用户的那个块，子用户的最高优先级是块的优先级。 然后用户通过网络传播这个块，以及其优先级和证明。其他用户等待一定的时间来接收块。如果用户没有收到块提议，则会用空白块初始化BA⋆。

拜占庭协议BA⋆：确保区块链不会分叉。块提议不保证所有用户都收到相同的块。为了在一个单独的块上达成共识，Algorand使用BA⋆。每个用户使用他们接收的最高优先级块初始化BA⋆。

BA⋆的执行由两个阶段组成。第一阶段，将在任意一个块的散列上达成一致的问题转换为在两个值达成一致，这两个值是特定建议的块散列，或者是空块的散列。 程序里是用Reduction这个函数。在第二阶段，BA⋆就这些选择之一达成一致：要么同意一个建议的区块，要么同意一个空的区块。程序里是用BinaryBA⋆这个函数。

首先Reduction这个函数，主要分为两步，step1，每个委员会成员对由BA⋆ main函数传递给Reduction（）的块的散列进行投票。Step2，委员会成员对第一步至少得到T·τ票的散列进行投票，或者如果没有散列得到足够的票数，则默认空块的散列。 Reduce（）确保至多有一个非空块可以被Reduce（）返回给所有诚实用户。

这里的投票是有两个主要函数——发送选票和计票。

发送选票函数，检查在给定的轮和步骤中用户是否被选择为委员会成员，这一步用之前的加密抽签函数实现。如果被选中，则用户就会传播一个包含值（要投给的块的哈希）的签名消息传递。

计票函数，从这个缓冲区中读取属于当前轮和步骤的消息。它通过调用每个消息的这个函数来处理投票， 这个函数是用来确保投票的有效性的，它不仅返回消息中包含的值，还返回与该值关联的投票数。在计票函数while循环里，只要一个值超过T·τ票，就会返回该值。如果在分配的λ时间窗口内没有收到足够的消息，则会产生超时并返回TIMEOUT。

第二阶段，BinaryBA⋆这个函数。前面说了，是二选一。每个委员会成员都为一个值投了一票，所有的用户都对票数进行统计。在BinaryBA⋆（）的每个步骤中，如果一个用户看到了某个值超过了T·τ票，那么在下一步中将投票给相同的值。 然而，如果没有任何值获得足够的选票，BinaryBA⋆（）选择下一个投票。

在一般情况下，当网络强同步且块提议者是诚实时，BinaryBA⋆（）将为大多数用户启动相同的block\_hash，并将在第一步达成共识，因为大多数委员会成员为相同 block\_hash值投票。程序里有显示。

如果网络不是强同步的，BinaryBA⋆（）可能在两个不同的块上返回共识。【例如，假设在BinaryBA⋆（）的第一步中，所有的用户投票给block\_hash，但是只有一个诚实的用户A接收这些投票。 在这种情况下，A将在block\_hash上返回共识，但所有其他用户将转到下一步。 现在，其他用户再次给block\_hash投票，因为CountVotes（）返回TIMEOUT。 但是，让我们假设网络将所有这些投票放弃。 最后，用户在第三步给投票empty\_hash，网络表现良好，所有投票都被传递。 因此，用户将继续投票给empty\_hash，直到下一个迭代循环，此时他们在empty\_hash上达成共识。】通过引入最终和暂定共识的概念来解决这个问题。BA⋆算法main函数最后是判断建立的是最终共识还是暂定共识。

尽管开发者正在竞相开发一种这样的系统，不过最近一位杰出学者的设计可能会成为这场长期探索中取得的一个标志性进展。这种设计被称为“[algorand](https://arxiv.org/abs/1607.01341)"，其创建者是来自麻省理工学院（MIT）的Silvio Micali教授。

Micali是一位密码学家和计算机理论学家，他因在伪随机数和零知识证明领域的成就而出名。（零知识证明是zk-SNARKS的基础，匿名区块链项目[Zcash](http://www.8btc.com/zcash)使用的就是零知识证明）。Micali还是图灵奖的获得者（该奖项有“**计算机界的诺贝尔奖**”之称）。

不过，除了拥有光彩照人的奖励证书，Mical的技术同样非常有前途。Algorand是权益证明（POS）机制的一种变形，POS使用了密码技术来随机选择那些负责将下一个区块（或交易）添加到区块链中的参与者。

Micali在2013年第一次开始了解比特币，他解释说比特币的协议设计令他印象深刻，但也渴望对这种协议中的一些他认为需要更多研究的地方进行改进。

Micali告诉Coindesk说：

“我真的对比特币协议非常着迷。但我同样要说的是，“如果让我来设计比特币的共识机制，我会使用一种完全不同的方式。”

如果成功，Micali相信他的系统能够轻松处理上百万节点——这将为区块链目前面临的最大的问题提供一种解决方案。

**靠自己的运气**

对于比特币来说，矿工需要通过竞争来解决一种加密谜团。获胜的矿工可以获得下一个区块，并赢得区块奖励。

但是比特币的工作量证明（PoW）机制（同样又被称为“中本聪共识”）会导致极高的能源消耗。一些人说这还可能导致比特币挖矿的中心化，也就是说可能会导致只有少数大型实体才能获得新的比特币。

为了实现这种分配方式的民主化，algorand使用了一种被Micali称为“加密抽签”的方法来选择参与者来创建和验证区块（若是在一个拥有百万级用户的系统中，参与者的数量也许可以达到数千）。

尽管大多数POS系统都依赖于随机性，不过algorand是不同的，因为你可以通过自己的计算机进行抽签，从而靠自己的运气来决定能否成为一名幸运的参与者。这种抽签的方法基于前一个区块，选择过程是自动的（不涉及到信息交换）并且完全随机。

Micali的这种想法来自于古雅典时代，当时的官员是通过一种抽签流程来选出的。（本质上就是把一些人的名字放到一个大帽子中，然后随机抽出几个。）

通过使用密码抽签，理论上algorand可扩展需求，同时还能为安全性和速度带来好处。

**一种公平且民主的系统**

因为Algorand对计算的要求几乎微不足道，任何人都能在自己的笔记本后台运行这种系统。尽管比特币存在用户分级（进行交易的消费者和搜索区块的矿工），Algorand并没有这样的区别。

Algorand的愿景就是所有人都拥有相同网络访问权限。

与其他POS系统相似，你被选中的概率基于你持有或预留的币的数量（我们把它称为algos），你拥有的algos越多，那么你被选中的概率就越大。

一旦你知道自己被选中成为申请人（proposer），那么就可以创建一个区块，然后连同一种哈希证明（一种随机号码，数字签名可以很容易对其进行验证）一起广播到网络上，本质上就是告诉别人，“这是我的区块，这是我赢得创建这个区块的权利的证明。”

拥有最小哈希证明的proposer（仍旧是随机抽取）将会成为下一个候选区块的创建者。

Algorand的下一步就是验证候选区块以及——在出现一个区块proposer提交两个或更多区块时——确保区块链不会分叉。

为此，Micali选择了一种用了几十年的协议——拜占庭协议。

**向分叉说拜拜**

中本聪共识的一个副产品就是网络分叉的可能性，当两个矿工几乎同时解决网络谜团时，就可能会发生这种分叉。

因此，用户一般会等待大约30分钟（三个区块时间）来合理地确保一笔交易已经被通过。

Micali说：

“而目前你必须要应对分叉，这就造成了一些心理上的焦虑，因为区块虽不是最终的，而人需要一种确定性。”

Algorand处理这种模糊性的方式就是对那个具有很小分叉可能性的区块达成共识。这种系统利用了一种拜占庭共识的修改版本。

[拜占庭协议](http://www.8btc.com/bitcoin-and-the-byzantine-generals-problem)于20世纪80年代被构想出来，这种协议为分布式系统（没有可信任的节点）提供了一种获得共识的方法。在这种设计中，这种系统可以容忍高达三分之一的参与者反对该系统。

拜占庭协议拥有两种属性：如果所有的参与者开始以相同的值开始，那么他们就是对这个值达成了一致共识。如果参与者以不同的值开始，那么所有诚实的参与者（遵循该协议的人）将会在某一个值上达成一致。在区块链上，这些值便是候选区块，而参与者便是交易验证者。

然而，传统的拜占庭协议存在一个问题，那就是参与者之间需要进行多轮强烈的通讯，致使难以对系统进行扩展。

Micali说：

“我无法在一百万或一千万用户之间运行拜占庭协议。这个数量实在太大。”

为了弥补，他开发了一个修改版的拜占庭协议。

**参与者可更换性**

在Algorand中，一小部分参与者可以代表整个系统来运行拜占庭协议。这可以让协议运行速度更快，随着越来越多的参与者在每一步中被替代，那么就可以在一种敌对的环境下使该系统安全。

**Micali的拜占庭协议的原理**就是：币持有者在第一轮中自己选择成为验证者（verifier)或矿工。这些验证者将他们的信息和凭据一起发送到该网络。

这时候这些验证者已经把自己暴露了出来，那些资源丰富的对手可以轻易贿赂他们。不过这无关紧要，因为一旦信息被公开，就不可能在收回。

“这个对手将无可奈何，就像政府无法收回维基泄密公开的消息一样。政府可以逮捕泄密者，把他送进监狱，但这个消息已经被传播到网络上。”

因此即使对手成功贿赂了验证者，但还是晚了一步。因为这时候，下一轮的通讯的新参与者已经被挑选好，这一过程将会再持续8轮，直到达成共同协议。

一旦协议达成，足够数量的参与者将会在拜占庭协议的最后一步中认证该区块，然后区块被广播到网络上，所有的用户都可以将其添加到区块链中。

鉴于这个系统中的唯一延迟是基于区块在网络上的传播速度，Micali将他的区块大小设定在了1MB。他认为当网络变得更快时就可以提升区块大小且没有任何安全风险。

**新的世界秩序**

Micali并不认为Algorand将会取代比特币。他认为不同的系统可以同时存在。

即便是今天，社会上仍旧存在以物换物的交易模式，所以Micali认为比特币没有理由将来会不再存在。但是他确实强烈地认为比特币的能源浪费是没有必要的。

他说：

“不知怎么地有些人比喻当你挖金子的时候也在浪费能源。事实上，黄金开采过程中的能源浪费并不意味着我们在破坏这个星球，而是我们的先辈破坏的。”

他还指出Algorand旨在作为所有类型的区块链系统的一种共识协议，而不仅仅是加密货币。

不过，Algorand还只是一种理论的协议。

Micali表示他正在商讨一些技术问题，并希望很快能够进行测试。

Algorand是一种新的加密机制，可以在一小段时间内确认交易延迟，同时扩展到许多用户。 Algorand确保用户从未对已确认的交易有不同意见，即使一些用户是恶意的，网络被临时分区。相比之下，现有的加密货币允许临时分叉，因此需要很长时间，大约一个小时，以高信任确认交易。

 Algorand使用新的拜占庭协议（BA）协议在下一组交易中达成用户的共识。为了将共识扩大到许多用户，Algorand使用基于可验证随机函数的新颖机制，允许用户私下检查是否被选择参与拜占庭协议（BA）协议以同意下一组交易，并包括其选择证明在他们的网络消息。在Algorand的BA协议中，除了私钥之外，用户不会保留任何私有状态，允许Algorand在发送消息后立即替换参与者。这样可以减轻他们的身份被揭露后对所选参与者的有针对性的攻击。

我们实施Algorand并评估其在1000个EC2虚拟机上的性能，模拟最多50万用户。实验结果表明，Algorand在一分钟内确认交易，达到125倍比特币的吞吐量，并且几乎不会对更多的用户进行扩展【并且几乎不会导致对更多的用户进行罚款】。

**一、             介绍**

比特币等加密货币可以使新的应用程序（如智能合同和公平协议）能够简化货币转换，并可以避免受信任的集中管理机构来管理交易。 然而，目前建议在交易延迟和交易信任之间取得平衡。 例如，在比特币交易得到确认的高度信任上需要大约一个小时的等待时间。 另一方面，要求低延迟的应用程序不能确定他们的交易将得到确认，并且必须相信付款人不会双重支付。

双重支付是任何加密货币面临的核心问题，持有$1的对手给予$1到两个不同的用户。加密数字货币通过在交易的有序日志（“区块链”）达成共识来阻止双重支付。 由于开放的环境，达成共识是困难的：由于任何人都可以参与，对手可以创建任意数量的假名（“Sybils”），这使得依赖传统的共识协议，需要一小部分诚实用户是不可行的。

比特币和其他加密数字货币使用工作证明（PoW）解决了这个问题，用户必须重复计算散列来增加块链，并且最长的链被认为是权威的。 PoW确保对手不会通过创建假名获得任何优势。 然而，PoW允许分叉的可能性，其中两个不同的块链具有相同的长度，并且两者都不取代另一个。 减轻分叉需要两个不幸的牺牲：用一个块增长链的时间必须相当高（例如，比特币10分钟），并且应用程序必须等待几个块（在比特币中推荐是6个区块），以确保其交易保留在权威链上。 结果是，确定比特币交易大约需要一个小时。

本文介绍了Algorand，一种新的加密机制，旨在一分钟确认交易。 Algorand的核心使用称为BA⋆的拜占庭协议，可以扩展到许多用户，这使得Algorand能够在低延迟的新块上达成共识，并且没有分叉的可能性。 使BA⋆适合Algorand的关键技术是使用可验证的随机函数（VRFs）,以私人和非交互方式随机选择用户。 BA⋆以前在高级研讨会上提出，陈和Micali的技术报告描述了Algorand的早期版本。

Algorand面临三个挑战。 首先，Algorand必须避免Sybil攻击，对手创建许多假名来影响拜占庭协议。 第二，BA⋆必须扩展到数百万用户，远远高于最先进的拜占庭协议的规模。 最后，Algorand必须对拒绝服务攻击具有弹性，即使对手断开了一些用户，也可以继续运行。

Algorand使用几种技术来解决这些挑战，如下。

**加权用户。**为了防止Sybil攻击，Algorand为每个用户分配一个权重。 只要用户的加权分数（大于2/3的常数）是诚实的，BA⋆就保证共识。 在Algorand，我们会根据用户账户中的金额来衡量用户。 因此，只要超过一部分（超过2/3）的钱是诚实用户所有，Algorand可以避免分叉和双重支付。

**委员会共识。**BA⋆通过选择一个委员会来实现可扩展性，该委员会是从整个用户组中随机选择的一小组代表，以运行其协议的每一步。 所有其他用户观察协议消息，这允许他们学习商定的块。 BA⋆根据用户的权重，在所有用户之间随机选择委员。 这样，Algorand就可以确保足够的部分委员会成员诚实。 然而，依靠一个委员会就可能对选定的委员会成员进行有针对性的攻击。

**加密抽签。**为防止对手针对委员会成员，BA⋆以私人和非互动的方式选择委员会成员。 这意味着系统中的每个用户可以通过从区块链中一个计算函数（VRF）计算其私钥和公共信息来独立地确定它们是否被选择在委员会中。 如果函数指示用户被选中，则返回一个短字符串，证明该用户的委员会成员资格给其他用户，用户可以将其包括在他的网络消息中。 由于成员资格选择是非互动的，因此在该用户开始参与BA⋆之前，对手不知道目标是哪个用户。

**参与者更换。**最后，一旦该成员在BA⋆发送消息，对手可能会针对一个委员会成员。BA⋆通过要求委员会发言一次来减轻这次袭击。 因此，一旦一个委员发了他的信息（将他的身份暴露给对手），委员会成员与BA⋆无关。 BA⋆通过避免任何私人状态（用户的私钥除外）来实现此属性，这使得所有用户同样有能力参与和被选举为拜占庭协议每个步骤的新的委员会成员。

我们实现了Algorand和BA⋆的原型，并用它来经验性地评估Algorand的性能。 在1,000个Amazon EC2 VM上运行的实验结果表明，Algorand可以在〜22秒内为50,000用户确认1 MByte的交易块，Algorand的延时在扩展到50万用户时几乎保持不变，Algorand达到了125倍的比特币交易吞吐量，即使在存在主动恶意用户的情况下，Algorand也能实现可接受的延迟。

**二、             相关工作**

**POW。**比特币是主要的加密货币，使用工作证明（POW）来确保每个人都同意已批准的交易; 这种方法通常被称为“Nakamoto共识”。比特币必须平衡时间长度来计算新的块，伴随着浪费算力的可能性，并且设置参数以平均每10分钟生成一个新的块。 尽管如此，由于可能出现分叉，广泛的建议是，在考虑交易确认之前，用户等待块扩展至少六块。 这意味着比特币的交易需要一小时的时间才能确认。 许多后续的加密货币采用比特币的工作证明方式，并继承其限制。 分叉的可能性也使新用户难以安全地引导：隔离用户网络的对手可以说服用户使用块链的特定分支。

通过依靠拜占庭协议，Algorand消除了分叉的可能性，并且避免了对采矿策略的理解。 因此，交易大约一分钟确认。 为了使拜占庭协议对Sybil的攻击很有效，Algorand关联权重和用户账户上持有的资金。 过去曾提出过其他技术，以抵制以拜占庭协议为基础的加密货币的Sybil攻击，包括让参与者提交安全存款并惩罚那些偏离协议的人。

**拜占庭共识。**已经使用拜占庭协议来复制一小组服务器，例如PBFT。后续工作显示了如何使拜占庭的容错性能良好，并扩展到数十台服务器。在此设置中使用的拜占庭式容错协议的一个缺点是需要一套固定的服务器来提前确定;允许任何人加入服务器组，这将在协议上为Sybil攻击打开一个缺口。这些协议也不能扩展到Algorand所目标的大量用户。 BA⋆是一种拜占庭共识协议，不依赖于固定的服务器组，避免了对知名服务器攻击的可能性。通过用户的货币余额对用户进行加权，只要诚实用户所持有的金额的比例至少保持在大于2/3的一个常数，BA⋆允许用户加入加密机制，并不会有Sybil攻击的风险。 BA⋆的设计还允许它使用VRFs来公平选择一个随机委员会来扩展到许多用户（例如，我们的评估中显示的50万）。

大多数拜占庭共识协议要求超过2/3的服务器是诚实的，Algorand的BA⋆继承了这一限制（以诚实用户持有的2/3的钱的形式）。 BFT2F表明，有可能实现“fork \* -consensus”，只有一半以上的服务器是诚实的，但是fork \* -consensus将允许对手在两个分支区块链中双重支附，这是Algorand避免的。

Honey Badger展示了如何使用拜占庭容错来构建一个加密机制。 具体来说，Honey Badger指定一套服务器来负责在一组批准的交易上达成共识。 这允许Honey Badger在5分钟内达成共识，并使用10 MByte块和104个参与服务器实现了数据吞吐量200 KB /秒附加到总帐。 这种设计的一个缺点是加密不再分散; 当系统首次配置时，选择了一组固定的服务器。 固定式服务器在危害服务器或将其与网络断开连接的目标攻击方面也存在问题。 Algorand可以实现更好的性能（在一分钟内确认交易，达到类似的吞吐量），而无需提前选择一组固定的服务器。

Bitcoin-NG建议使用Nakamoto共识来选举领导者，然后让领导人发布交易块，从而在比特币确认交易的延迟时间方面提升了一个数量级。Hybrid共识改进了使用Nakamoto共识的方法，定期选择一组参与者（例如，每天），并在所选参与者之间运行拜占庭协议以确认交易，直到选择新服务器。 这样可以提高性能，超过标准的Nakamoto共识（比如比特币）。 例如，在拜占庭协议中，ByzCoin提供大约35秒的延迟，并且以8 MB的块大小和1000个参与者附加到数据吞吐量为230KBytes /秒的总帐上。虽然Hybrid共识使得拜占庭服务器组动态，但由于POW共识来使服务器组达成一致，因此有分叉的可能性。 这个问题不会在Algorand中出现。

Pass和Shi的论文承认Hybrid共识设计仅在“轻度适应”对手方面是安全的，这些对手在一天内（参与者选择间隔）不能危害所选择的服务器，并明确提出处理完全适应对手的开放问题。 Algorand的BA⋆通过立即更换任何选定的委员会成员来明确地解决这个公开问题。 因此，Algorand不容易受到目标的妥协【targeted compromises】或针对性的DoS攻击。

Stellar采用一种替代方法在加密货币中使用拜占庭共识，其中每个用户可以信任法定数目的其他用户，形成信任层次结构。只要所有交易共享至少一个过渡性信任的用户群体，就可以确保一致性，并且这些用户中有很多人是诚实的。 Algorand避免了这种假设，这意味着用户在配置客户端软件时不必做出复杂的信任决策。

**POS。**Algorand根据其在系统中的币值按比例分配权重，灵感来自于POS方法，建议作为POW的替换或改进。 然而，在Algorand使用币值作为权重和许多POS加密货币间存在一个关键的区别。 在许多POS加密货币中，恶意领导者（组装新的块）可以在网络中创建一个分支，但如果被捕获（例如，由于新块的两个版本都用他的密钥签名），领导者会失去他的钱。然而，Algorand的权重只是为了确保攻击者不能通过使用假名来扩大他的权力; 只要攻击者控制的币值不到1/3，Algorand就可以保证分叉的概率是可以忽略不计的。 Algorand可能会扩展到“检测和惩罚”恶意用户，但这并不需要防止分叉或双重支付。

POS避免了POW的计算开销，因此可以减少交易确认时间。 然而，在实践中实现POS是有挑战性的。 由于生成块中没有涉及工作【算力】，恶意领导者可以发布一个块，然后为了孤立用户介绍一些其他块。 攻击者也可能会将他们的信用分成几个“用户”，他们可能被选为领导者，以便坏领导者被抓时最小化惩罚。 因此，一些POS加密货币要求主密钥周期性地签署总帐的正确分支，以减轻分叉。 这引起了对货币的重大信任担忧，并且在过去也引起了意外的分叉。 Algorand通过避免分叉来回答这个挑战，即使领导者是恶意的。

Ouroboros是最近提出的实现POS的提案。 为了安全起见，Ouroboros假定诚实用户可以在一些有限的延迟内进行通信（即强同步网络）。 此外，它选择一些用户参与joint-coin-flipping协议，并且假定大多数用户在对手处于重大时期（如一天）时是清廉的。 相反，Algorand假设对手可能会暂时完全控制网络，并立即有针对性的败坏用户。

**树和DAG代替链。**GHOST，SPECTER和Meshcashare最近提出的通过用树或有向非循环图（DAG）结构替换基础链结构化总帐，并解决这些数据结构中分叉的冲突来提高比特币的吞吐量。 这些协议依赖于使用POW的Nakamoto共识。 通过仔细设计树/ DAG的分支之间的选择规则，它们能够显着提高吞吐量。 相反，Algorand集中在消除分叉; 在未来的工作中，探索树或DAG结构是否可以同样增加Algorand的吞吐量可能是有趣的。

**三、             目标和假设**

Algorand允许用户同意交易的有序日志，并且针对日志达成两个目标：

**安全目标**。以绝对的概率，所有用户同意相同的交易。 更准确地说，如果一个诚实的用户接受交易A（即它出现在日志中），则其他诚实用户接受的任何未来交易将出现在已经包含A的日志中。 即使对于与网络断开连接的隔离用户（例如，通过Eclipse攻击）也是如此。

**活力目标**。除了安全性之外，Algorand还在下面我们描述的关于网络可达性的另外的假设下有进度（即允许将新交易添加到日志中）。 Algorand旨在，在大约一分钟内就一系列新交易达成共识。

**假设**。 Algorand制定标准的加密假设，如公钥签名和散列函数。 Algorand假定诚实用户运行无bug软件。 如前所述，Algorand假设诚实用户持有的部分资金高于某个阈值h（一个大于2/3的常数），但对手可以参与Algorand并拥有一些资金。我们认为这个假设是合理的，因为这意味着为了成功攻击Algorand，攻击者必须投入大量的财力。 Algorand假设对手可能会败坏目标用户，但对手不能败坏大量持有相当一部分资金的用户（即，诚实用户持有钱的数量，不妥协的用户所持有的资金数额必须保持在阈值以上）。

为了实现活力，Algorand做出了一个“强烈的同步”假设，大多数诚实的用户（例如95％）可以发送大多数其他诚实用户（例如，95％）在已知时间限制内接收的消息。这个假设允许对手控制一些诚实用户的网络，但不允许对手大规模地操纵网络，并且不允许网络分区。

Algorand以“弱同步”假设实现安全性：网络可以是长时间但有限的时间段（例如，最多1天或1周）的异步（即完全由对手控制）网络。在异步时段之后，Algorand为了确保安全性，网络必须在一段相当长的时间内（例如，几个小时或一天）强烈同步。 更正式地，弱同步假设是，在每个长度为b（考虑b为一天或一周）的时期，必须有一个强烈同步的长度s <b（几个小时的s）。

最后，Algorand假定所有用户（例如，使用NTP）松动地同步时钟，以便在弱同步之后恢复活动。 具体来说，时钟必须足够接近，以便大多数诚实用户大致在同一时间启动恢复协议。如果时钟不同步，则恢复协议不成功。

**四、             概述**

Algorand要求每个用户拥有公钥。 Algorand维护一个交易记录，称为区块链。每个交易是一个付款，由一个用户的公钥签收【signed】，用于将钱转移到另一个用户的公钥。 Algorand在异步回合中增长了区块链，类似于Bitcoin。在每一轮中，包含一组交易和指向前一个块的指针的新块被附加到区块链。 在本文的其余部分，我们指的是用户计算机上运行的Algorand软件。

Algorand用户通过Gossip【gossip】协议进行通信。用户使用Gossip协议提交新交易。 每个用户收集他们听到的待处理交易块，以防它们被选择提出下一个块，如图1所示。Algorand使用BA⋆在其中一个等待块上达成共识。

BA⋆步骤执行，通过相同的Gossip协议进行通信，并产生一个新的协商块。 BA⋆可以产生两种共识：最终达成共识，暂定共识。 如果一个用户达成最终的共识，这意味着在同一回合中达成最终共识或暂定共识的任何其他用户必须同意相同的块值（不管是否保持强同步假设）。 这确保了Algorand的安全性，因为这意味着所有未来的交易将被链接到这个最终的块（并且过渡到其前辈）。 因此，Algorand在交易的块（或任何后续块）达成最终共识时确认一个交易。 另一方面，暂定共识意味着其他用户可能已经在不同的方面达成了暂定共识（只要没有用户达成最终的共识）。用户只有在当后续块达成最终共识的情况下才能从暂定块中确认交易。

BA⋆在两种情况下产生了暂定共识。 首先，如果网络强同步，对手可能会以很小的可能性导致BA⋆在一个块上达成暂定共识。 在这种情况下，BA⋆将不会在两个不同的区块达成共识，但根本无法确认网络是强同步的。 Algorand最终（在几轮中）对后继块达成最终共识，压倒性概率，从而确认这些较早的交易。

第二种情况是，网络只是弱同步（即完全由对手控制，对手可以控制多长时间有一个上限）。 在这种情况下，BA⋆可以在两个不同的块上达成暂定共识，形成多个分支。 这可以反过来阻止BA⋆再次达成共识，因为用户对之前的块意见不统一，被分成不同组。 为了恢复活力，Algorand定期调用BA⋆以在从目前开始分叉应该被用方面达成共识。 一旦网络恢复了强同步，这将允许Algorand选择一个分叉，然后在该分叉的后续块上达成最终共识。

我们现在描述Algorand的组件如何组合在一起。

**Gossip协议**。Algorand实现了一个Gossip网络（类似于比特币），其中每个用户选择一个小随机的对等体来传播消息。 为了确保消息不能被伪造，每个消息都由其原始发件人的私钥签名; 其他用户在中继前检查签名是否有效。 为避免转发环路，用户不会中继相同的消息两次。 Algorand通过TCP实现Gossip，并根据他们拥有多少钱来衡量对等选择，以减轻污染攻击。

**块提议**（见六）。 所有Algorand用户都执行加密抽签，以确定它们是否在给定轮次中被选中提出一个块。 我们在§5中描述抽签，但在高水平上，抽签确保随机选择一小部分用户，并通过其帐户余额进行权衡，并为每个所选用户提供优先级，可以在用户之间进行比较，是所选用户优先级的一个证明。 由于抽签是随机的，所以可能会有多个用户选择提出一个块，优先级决定了每个人应该采用哪个块。 被选中的用户通过gossip协议分配其待处理交易的块，以及其优先级和证明。 为了确保用户以高概率聚合在一个块上，块提议的优先次序基于提出用户的优先级，用户等待一定的时间来接收块。

**使用BA⋆协议**（见七）。块提议不保证所有用户都收到相同的块，Algorand的安全性不依赖于块提议协议。为了在一个单独的块上达成共识，Algorand使用BA⋆。每个用户使用他们接收的最高优先级块初始化BA⋆。 BA⋆以重复的步骤执行，如图2所示。每个步骤都以抽签（见五）开始，所有用户检查在该步骤中是否被选为委员会成员。然后，委员们播放一个包含选择证明的消息。这些步骤重复，直到在BA ⋆的某些步骤中，委员会的足够用户达成共识。 （每个用户的步骤都不会同步;每个用户一旦观察到上一步结束就检查选择。）如前所述，BA⋆的一个重要特征是委员会成员不会保留私有状态，除了私钥，所以委员会成员可以在每一步之后重置，以减轻对他们的有针对性的攻击。

**效率**。 当网络强同步时，BA⋆保证如果所有诚实用户从相同的初始块开始（即，最高优先级块提议者是诚实的），则BA⋆建立与该块最终共识，并且精确地终止于与用户间的4个互动步骤。 在相同的网络条件下，在最糟糕的情况下，一个特别幸运的对手，所有诚实用户在预期的13个步骤中就在下一个块达成共识，如附录C所分析。

**五、             加密抽签**

加密抽签是根据每个用户权重来选择用户的随机子集的算法; 即给定一组权重wi和所有用户的权重W =∑i w i ,，用户i被选择的概率与wi/ W成正比。 抽签算法中的随机性来自一个众所周知的随机种子（seed）， 我们稍后描述如何选择这个种子。 为了让用户证明他们被选中，抽签要求每个用户都有一个公钥/私钥对，（pk i，ski）。

抽签是使用可验证的随机函数（VRFs）来实现的。 非正式地，在任何输入字符串x上，VRFsk（x）返回两个值：散列和证明。 散列是由sk和x唯一确定的hashlen-bit-long值，但对于不知道sk的任何人来说，是无法区分的。证明π使得任何知道pk的人都能够检查hash确实对应于x，而不必知道sk。 为了安全起见，即使攻击者选择了pk和sk，我们也要求VRF提供这些属性。

**5.1 选择程序**

使用VRFs，Algorand实现算法1中所示的加密抽签。抽签需要一个角色（role）参数来区分用户可以选择的不同角色; 例如，用户可能在某个回合中被选择提出一个块，或者在BA⋆的某个步骤他们被选择为委员会的成员。 Algorand指定了一个阈值τ，用于确定为该角色选择的用户的预期数量。

抽签选择用户权重按比例是非常重要的。 否则，分拣不会抵御“假名”攻击。 一个微妙的可能情况是用户可以通过抽签选择不止一次（例如，因为它们具有高权重）。 抽签通过返回j参数来解决这个问题，该参数表示用户被选择了多少次。被选择j次意味着用户以j个不同的“子用户”的身份参与。

如果用户按照他们的资金比例来选择用户，我们将每单位的Algorand视为一个不同的“子用户”。如果用户i拥有wi（整数）单位的Algorand，则模拟用户（i，j），其中j∈{1，...。 wi}表示i拥有第j个单位的货币，并且以概率p =τ/ W来选择用户i，其中W是Algorand中货币单位的总量。

如算法1所示，用户通过计算⟨hash,π⟩ ← VRFsk (seed||role)执行抽签，其中sk是用户的密钥。 伪随机hash决定了选择多少个子用户，如下所述。 在w（用户的权重）子用户中选择恰好k的概率遵循二项分布，。由于B（k 1; n 1，p）+ B（k 2; n 2，p）= B（k 1 + k 2; n 1+n 2，p），所以在Sybils之间划分用户的权重（货币）不会影响在他/她的控制下选择的子用户的数量。

为了确定有多少用户的w子用户被选择，抽签算法将区间[0,1）划分为连续区间的形式Ij= 。如果hash / 2 hashlen（其中hashlen是散列的比特长度）属于间隔I j中，则用户恰好具有j个选定的子用户。 选择的子用户数可以使用证明π（来自VRF输出）进行公开验证。

抽签提供了两个重要的属性。 首先，给定一个随机种子，VRF输出一个伪随机哈希值，它基本上是均匀分布在0和2 hashlen − 1之间的。因此，用户是根据它们的权重随机选择的。 其次，一个不知道ski的对手不能猜测用户i 被选择了多少次，或者究竟用户i是否被选中了（更准确地说，敌人根据权重随机猜测是猜不到的）。

在算法2中显示了用于验证抽签证明的伪代码，遵循相同的结构以检查是否选择了该用户（用户的公钥的权重是从总账中获得的）。 该函数返回所选子用户的数量（如果用户没有被选中，则返回0）。

**5.2 选择种子**

抽签需要随机选择并公开的种子。 对于Algorand来说，每一轮都需要一个在该轮内众所周知的种子，但不能被对手控制。 否则，攻击者可能能够选择有利于选择腐败用户的种子。

在Algorand的每一轮都会产生一个新的种子。 Algorand第r回合产生的种子是由前一回合r-1的种子通过VRF来确定的。更具体地说，在回合r-1的块提议阶段期间，为第r轮块提议选择的每个用户u也计算提出的种子⟨seedr，π⟩←VRF sku（seedr-1|| r）。Algorand要求在决定该轮种子之前先让u选好sku（§5.3）。 这确保即使u是恶意的，seedr的结果是伪随机的。

这个种子（和相应的VRF证明π）被包含在每个提出的块中，所以一旦Algorand在r-1轮的块上达成一致，在r轮开始时，每个人都知道seedr。 如果块不包含有效的种子（例如，因为该块是由恶意用户提出的并且包括无效的交易），则用户将整个提出的块看作是空的，并且使用加密哈希函数H（我们假设它是 随机预言）来计算第r轮的相关种子，seedr= H（seedr-1 || r）。 引导种子选择的seed0的值可以在Algorand开始时通过初始参与者（在声明他们的公钥之后）用分布式随机数随机选择生成。

为了限制攻击者操纵抽签的能力，从而操纵不同委员会的用户选择，选择种子（交给算法1和算法2）每R轮被刷新一次。也就是说，在第r轮中，Algorand用seedr-1-（r mod R）调用抽签函数，

**5.3 在seed之前选择sku**

计算seedr需要在该轮选择使用的种子（即seedr-1-（r mod R））之前选择每个用户的（r-1轮）密钥sku。当网络不是高度同步的时候，攻击者完全控制链路，可以因此终止块提议并强制用户同意空块，这样可以计算未来选择的种子。为了减轻这种攻击，Algorand依赖弱同步假设（在每个长度为b的周期内，必须有一个长度为s <b的强同步周期）。每当Algorand为第r轮执行加密抽签时，它会检查包含在商定块中第r-1-（r mod R）轮的时间戳，并使用来自块r - 1 - （r mod R）之前b时间创建的最后的块中的keys（和相关的权重）。强同步时间长度的下限s应该允许创建足够多的块，以压倒性的概率确保至少一个块是诚实的。这确保了只要s适当大，选择密钥sku的对手u就不能预测第r轮的种子。

这个回溯期限b可以通过下面的权衡（trade-off）：一个大的 b减轻了攻击者能够打破弱同步性假设的风险，但是它增加了用户将货币转移给其他人的机会，因此如果系统的安全性破坏，就没有任何损失。 俗称“没有利害关系”问题， 一种可能的方式是避免我们在Algorand探索这种权衡，即将用户的当前最低余额和来自回溯块的用户余额作为用户的权重。

附录A正式分析了在网络强连接时Algorand需要在s周期间创建的块的数量。 我们证明，为了确保小概率的失败F，在1 / F中块的数量是对数的，这使得我们能够以合理的低数量的所需块来获得高安全性。

**六、             Block proposal（块提议）**

为了确保每一轮都提出一个区块，Algorand将区块提议角色的抽签阈值τproposer设置为大于1（尽管Algorand将在这些提议的区块中的至多一个上达成共识）。 附录B证明，选择τproposer= 26确保以非常高的概率（例如1-10-11）选择合理数量的提议者（至少一个，并且不超过70作为合理的上限）。

**尽量减少不必要的块传输。** 选择几个提议者的风险是每个人都会传播自己的提议块。 对于一个大块（比如1兆字节）来说，这可能会导致很大的通信成本。 为了降低这个代价，使用抽签散列来优先考虑块提议：为用户i选择的每个子用户1，...，j，块提议的优先级是通过哈希（可验证的随机）VRF输出的散列与子用户索引连接获得的。 所有块提议者选择的子用户的最高优先级是块的优先级。

Algorand用户放弃有关该用户迄今为止看不到最高优先级的块的消息。 Algorand还传两种消息：一种仅包含所选块提议者的优先级和证明（来自抽签），另一个包含整个块，其中还包括提议者的抽签哈希和证明。 第一种消息很小（约200字节），并通过gossip网络迅速传播。 这些消息使大多数用户能够了解谁是最高优先级的提议者，并因此迅速放弃其他提议的块。

**等待块建议。**每个用户都必须等待一定的时间才能通过gossip协议接收块提议。 选择这个时间间隔不会影响Algorand的安全保证，但是对性能很重要。 等待很短的时间将意味着没有收到建议。 如果用户没有收到块提议，他或她用空白块初始化BA⋆，如果有许多用户这样做，Algorand会在空白块上达成一致。 另一方面，等待时间过长会收到所有块建议，但也会不必要地增加确认延迟。

为了确定等待块提议的适当时间，我们考虑用户可能发现自己的合理情况。当用户开始等待第r轮块提议时，他们可能是在第 r – 1轮中第一批达到共识的用户之一。由于该用户完成r-1轮，所以足够多的用户在该轮的BA⋆的最后一步发送消息，因此，大部分网络最多落后于该用户一步。 因此，用户必须以某种方式等待其他人从r-1完成BA⋆的最后一步。此时，r轮中恰好具有最高优先级的一些提议者将传播他们的优先级和证明消息，并且用户必须以某种方式等待接收信息。 然后，用户可以简单地等待，直到他们收到对应于最高优先级证明的块（具有超时λBLOCK，大约一分钟，之后用户将回退到空块）。

用户不可能为上述情况的前两个步骤准确地等待正确的金额。 因此，Algorand估计了这些量（λSTEPVAR，不同的用户完成BA⋆的最后一步需要多长时间，λPRIORITY，传播优先级和证明消息的时间），并等待λSTEPVAR+ λPRIORITY时间来确定最高优先级。 §10通过实验表明，这些参数是保守（适当）的，每个5秒。 如上所述，即使这些估计不准确，Algorand也将保持安全。

**恶意提议者。**即使某些块提议者是恶意的，最坏的情况是他们欺骗不同的Algorand用户使用不同的块来初始化BA⋆。这可能导致Algorand在空白区块达成共识，并可能采取额外措施。 然而，事实证明，即使这种情况是相对不可能的。 特别是，如果对手不是一轮中最高优先级的提议者，那么优先级最高的提议者将向所有用户传送一致的版本。 如果对手是一轮中最高优先级的提议者，他们可以提出空白块，从而防止实际交易被确认。 然而，这种情况发生的概率至多为1 −h，根据Algorand的假设，至少h> 2/3的加权用户是诚实的。

**七、             BA⋆**

BA⋆的执行由两个阶段组成。 在第一阶段，BA⋆减少了同意一个块就两个选项之一达成一致的问题。 在第二阶段，BA⋆就这些选择之一达成一致：要么同意一个建议的区块，要么同意一个空的区块。

每个阶段由几个互动步骤组成; 第一个阶段总是需要两个步骤，第二个阶段如果最高优先级的方案提出者是诚实的（向所有用户发送相同的块），则采取两个步骤，正如我们在分析中显示的预计11步，在最坏的情况下，一个恶意的最高优先级提议者在每个步骤中与大部分委员参与者勾结。

在每一步中，每个委员会成员都为一个值投了一票，所有的用户都对票数进行统计。 获得某个值的投票数超过某个阈值，用户将在下一步（如果被选为委员会成员）中为该投票值投票。 如果用户没有获得足够的某个值的投票，他们超时，他们选择下一步的投票是由步骤号确定的。

在一般情况下，当网络强烈同步且优先级最高的块提议者是诚实的时，BA⋆通过最后一步确认在同一轮中不存在任何其他相互认同的块，从而达成最终共识。 否则，如果由于可能的网络不同步而无法确认其他块的不存在，BA⋆可以宣布暂定共识。

BA⋆设计的一个关键方面是它不保密，除了用户私钥之外。 这允许任何观察消息的用户“被动参与”协议：验证签名，计票，达成协议决定。

**7.1 BA⋆的main程序**

在算法3中示出了由Algorand调用的实现BA⋆的顶层程序。该过程采用上下文ctx，该上下文ctx从最高优先级块提案者（§6）捕获总账的当前状态，轮数和新提议的块。 Algorand负责确保块的有效性（通过检查提议块的内容，并使用一个空块，如果它是无效的，如§8所述）。 上下文由用于抽签的种子，用户权重和上一个商定的块组成。

为了提高效率，BA⋆投票表决块的散列，而不是整个块的内容。 在BA⋆算法的最后，我们使用BlockOfHash（）函数来表明，如果BA⋆还没有收到商定散列的前映像，它必须从其他用户那里获得它（并且，由于 块被商定了，许多诚实的用户在块建议期间必须得到它）。

BA⋆算法还决定是否建立最终的或暂定的共识。 我们将在讨论算法8时详细讨论这个检查。

**7.2投票**

**发送选票（算法4）。** 算法4显示了CommitteeVote（）的伪代码，它检查在给定的轮和步骤BA⋆中是否为委员会选择了用户。 CommitteeVote（）程序从算法1调用Sortition（）来检查用户是否被选择参与委员会。 如果用户在这一步选择被选中，用户就会传播一个包含值的签名消息传递给CommitteeVote（），通常是某个块的哈希值。 要将投票绑定到上下文，签名的消息包含前一个块的散列。

**计票（算法5和算法6）。** CountVotes（）程序（算法5）从incomingMsgs缓冲区中读取属于当前轮和步骤的消息。 （为了简单起见，我们的伪代码假定后台程序获取传入的投票并将它们存储到缓冲区中，由消息的轮和步骤索引）。它通过调用每个消息的ProcessMsg（）程序来处理投票（算法6）， 这确保了投票的有效性。 请注意，不需要私有状态来处理这些消息。

ProcessMsg（）不仅返回消息中包含的值，还返回与该值关联的投票数。 如果消息不是来自选定的委员会成员，则ProcessMsg（）返回零票。 如果委员会成员多次被选中（见§5），ProcessMsg（）返回的选票数量也反映了这一点。 ProcessMsg（）也返回抽签散列，我们将在后面的算法9中使用。

只要一个值超过T·τ票，CountVotes（）就会返回该值。 τ是Sortition（）为委员会选择的用户的预期数量，除了最后一步（τFINAL）之外，每个步骤（τSTEP）都是相同的。 T是预期委员会规模（T> 2/3）的一小部分，定义BA⋆的投票门槛; 这也是除了最后一步之外，每个步骤都是一样的，我们在§7.5中对它进行分析。 如果在分配的λ时间窗口内没有收到足够的消息，则CountVotes（）会产生超时（TIMEOUT）。 阈值确保了如果一个诚实用户的CountVotes（）返回一个特定的值，那么即使在弱同步假设下（参见附录C.2中的引理1），所有其他诚实用户也将返回相同的值或超时（TIMEOUT）。

**7.3减少（Reduction（））**

在算法7中所示的Reduction（）程序将在任意值（一个块的散列）上达成一致的问题转换为在两个值中的一个上达成一致：或者是特定建议的块散列，或者是空块的散列。 Turpin和Coan的两步技术启发了我们的减少。 这种减少对于确保活力非常重要。

在reduction的第一步，每个委员会成员对由BA⋆（）传递给Reduction（）的块的散列进行投票。 第二步，委员会成员对第一步至少得到T·τ票的散列进行投票，或者如果没有散列得到足够的票数，则默认空块的散列。 Reduce（）确保至多有一个非空块可以被Reduce（）返回给所有诚实用户。

在网络强同步且优先级最高的块提议者是诚实的常见情况下，大多数（例如95％）的用户将使用相同的hblock参数调用Reduction（），并且Reduction（）会将同样的hblock结果返回给大多数用户。

另一方面，如果最高优先级块提议者是不诚实的，则不同的用户可能用不同的hblock参数启动Reduction（）。 在这种情况下，没有一个单独的hblock值可能足够受欢迎，以至于通过选票的门槛。因此，Reduction（）将返回empty\_hash。

CountVotes要么TIMEOUT要么T·τ票的散列

**7.4 Binary agreement（两元的共识）**

算法8显示了BinaryBA⋆（），它在两个值之一上达成共识：要么传递给BinaryBA⋆（）的散列，要么是空块的散列。 BinaryBA⋆（）依赖于Reduce（）来确保至多一个非空块的散列被所有诚实的用户传递给BinaryBA⋆（）。

**强同步的安全性。** 在BinaryBA⋆（）的每个步骤中，如果一个用户看到了某个值超过了T·τ票，那么在下一步（如果选中）中将投票给相同的值。 然而，如果没有任何值获得足够的选票，BinaryBA⋆（）选择下一个投票，以确保在强同步网络中的共识。

具体而言，用户A可以接收来自对手的投票，这将让A观察的票超过阈值，但是对手可能不向其他用户发送相同的投票（或者可能不及时发送）。 因此，A返回一个块的共识，但其他用户在该步骤超时。 BinaryBA⋆（）选择下一步的选票在某种程度上是非常重要的，将和A已经返回的区块相匹配。 算法8遵循以下规则：每一个return语句都与一个TIMEOUT检查相结合，将下一步投票设置为可能返回的相同值。

BinaryBA⋆（）能够在下一步收集足够的选票来发扬A已经达成共识的值也是至关重要的。 如果有许多像A这样已经返回共识的用户，则BinaryBA⋆（）可能没有足够的用户推动下一个步骤中的CountVotes（）超过阈值。 为了避免这个问题，每当用户返回共识时，该用户在接下来的三个步骤中以他们达成共识的值进行投票。

在一般情况下，当网络强同步并且块提议者是诚实的，BinaryBA⋆（）将为大多数用户启动相同的block\_hash，并将在第一步达成共识，因为大多数委员会成员为相同 block\_hash值投票。

**弱同步的安全性。**如果网络不是强同步的（例如，有一个分区），BinaryBA⋆（）可能在两个不同的块上返回共识。 例如，假设在BinaryBA⋆（）的第一步中，所有的用户投票给block\_hash，但是只有一个诚实的用户A接收这些投票。 在这种情况下，A将在block\_hash上返回共识，但所有其他用户将转到下一步。 现在，其他用户再次给block\_hash投票，因为CountVotes（）返回TIMEOUT。 但是，让我们假设网络将所有这些投票放弃。 最后，用户在第三步给投票empty\_hash，网络表现良好，所有投票都被传递。 因此，用户将继续投票给empty\_hash，直到下一个迭代循环，此时他们在empty\_hash上达成共识。 这是不可取的，因为BinaryBA⋆（）在两个不同用户的不同哈希值上返回共识。

BA⋆（）通过引入最终和暂定共识的概念来解决这个问题。最终共识意味着BA⋆（）将不会在这一轮的任何其他块达成共识。 暂时共识意味着BA⋆（）无法保证安全，无论是因为网络同步还是由于恶意块提议者。

如果Binary BA⋆（）恰好在第一步中达成了V的共识，并且如果有足够的用户观察到达成了这个共识，那么BA⋆（）将V值的共识指定为“最终的”。 具体来说，BinaryBA⋆（）发出了一个特殊的FINAL步的投票，为了表明用户恰好在第一步达成了一些值的共识，BA⋆（）收集这些投票，以确定是否达成最终共识。在一个诚实的块提议者的强同步网络中，BinaryBA⋆（）将在第一步达成共识，大多数委员会成员将在BinaryBA⋆（）的特殊FINAL步投票达成共识块， 并且将会在BA⋆（）中收到的超过阈值的这种选票，从而宣布该块为最终的。 FINAL步是类似于许多拜占庭弹性协议中实施的最终确认步骤。

直观地说，这保证了安全性，因为用户的一个大阈值已经宣布了对V的共识，并且不会在同一回合投票给其它值。 在我们上面的例子中，当用户A在不同于所有其他用户的块上达成共识时，这两个块都不会被指定为最终的，因为只有一个用户（即A）在第一步观察到共识，永远不会有足够的选票将该块标记为最终。 附录C.1正式确认并证明了这一安全性。

由于BA⋆依靠一个委员会来宣布最终共识，而不是依靠所有的参与者这个事实，出现了一个微妙的问题。因此，即使一个用户观察到最终共识，控制网络的对手也许能够阻止一小部分其他用户为任意数量的步骤达成任意类型的共识（最终的或暂定的）。 这些步骤中的每个步骤都使对手有一个额外小概率就不同的值达成共识（例如空白块）。 为了限制对手这样做的总概率，BA⋆限制了允许步数的总数; 附录C.1依赖于此。 如果协议的运行超过MAXSTEPS步，BA⋆停止协商共识，并依靠§8.2中描述的恢复协议恢复活跃性。

**越来越不卡（Getting unstuck）。**剩下的一个问题是，如果诚实的用户被分成两个组A和B，并且两个组中的用户投票给不同的值（例如，我们在步骤1中，A为empty\_hash投票，B投票给block\_hash），共识会卡住。 两组都不够大，无法自己收集足够的选票，但是和对手的选票合在一起，A组足够大。在这种情况下，对手可以决定每个用户在下一步的投票。为了让用户在下一步中对empty\_hash投票，对手在超时到期之前将对手自己的票投给empty\_hash，这与A的投票放在一起，超过阈值。为了让用户对block\_hash投票，攻击者不会向该用户发送任何投票;因此，该用户的CountVotes（）将返回TIMEOUT，并且根据BinaryBA⋆（）算法，用户将为下一步的投票选择block\_hash。这样，攻击者可以在下一步将用户分成两组，并且无限期地继续这个攻击。

上面描述的攻击要求攻击者知道用户从CountVotes（）接收到TIMEOUT后如何投票。BinaryBA⋆（）的第三步是为了通过推动接受基于随机“普通硬币”的block\_hash或empty\_hash来避免这种攻击，，这意味着对于所有用户来说主要是相同的二元。 虽然这听起来可能是循环的，但用户无需就这个普通硬币达成正式的共识。 只要有足够的用户观察到相同的硬币位，并且该位在该步骤之前并未被攻击者知道，则BinaryBA⋆（）将在概率1/2（即， 攻击者猜测错误的概率）的下一次迭代循环中达到共识。 通过重复这些步骤，共识的概率很快接近1。

为了实现这个硬币，我们利用所有消息附带的基于VRF的委员会成员哈希。 每个用户将普通硬币设置为在此步骤中观察到的最低散列值的最低有效位，如算法9所示。如果用户得到多票（即选择了他们的几个子用户），则CommonCoin（） 考虑来自该用户的多个散列，通过将该用户的抽签散列与子用户索引进行散列。 注意散列是随机的（因为它们是通过散列伪随机VRF输出产生的），所以它们的最低有效位也是随机的。只有在CountVotes（）超时的情况下才使用普通硬币，为所有选票在网络中传播提供了足够的时间。 如果哈希值最低的委员会成员是诚实的，那么接收到他的消息的所有用户都观察到相同的硬币。

如果一个恶意的委员会成员恰好拥有最低的哈希值，那么他可能只把它发送给一些用户。 这可能会导致用户观察到不同的硬币值，从而无助于达成共识。 然而，由于抽签散列是伪随机的，所以诚实用户具有最低散列的概率是h（诚实用户持有的钱的一小部分），因此至少有h>2/3的概率，最低抽签散列持有者将是诚实的，这导致在每个循环迭代中概率为1/2·h> 1/3的共识。 这使得附录C.3可以表明，在强同步的情况下，BA⋆不会以极大的概率超过MAXSTEPS。

**7.5 委员会规模**

在Algorand中权重诚实用户的h> 2/3部分必须转化为“足够诚实”的BA⋆委员会。 BA⋆有两个参数可供选择：控制预期委员会规模的τ和控制达成共识（T·τ）所需票数的T。 为了活跃度，我们希望T越小越好，但T越小，τ就越大，以确保对手不会有机会获得足够的选票。 由于更大的委员会转化为更高的带宽成本，所以我们选择两个不同的参数集：FINAL步的TFINAL和τFINAL，这确保了不管强同步的压倒性安全概率，而所有其他步骤的TSTEP和τSTEP 在生存性，安全性和性能之间取得合理的平衡。

为了对τSTEP和TSTEP进行精确的约束，我们用g表示诚实委员人数，用b表示恶意人员的数量; 在期望中，b +g=τSTEP，但是b +g可以变化，因为它是通过抽签来选择的。 为了保证活跃性，如附录C.2所证明，BA⋆需要1/2g+ b≤TSTEP·τSTEP和g> TSTEP·τSTEP。

由于委员会成员如何选择的概率性质，一些步骤中的b和g总是有一些小的机会不能满足上述的约束条件，BA⋆的目标是使这个概率可以忽略不计。 图3绘制了作为h的函数满足两个约束条件所需的预期委员会规模τSTEP，将违反安全的概率限制在5×10-9; 附录B更详细地描述了这个计算。这个图显示了一个权衡：对诚实用户持有的部分货币的假设（h）越小，委员会规模就越大。结果表明，当h接近2/3时，委员会规模迅速增长。 然而，在h = 80％时，τSTEP = 2 000可以确保这些约束保持1 - 5×10-9的概率（使用TSTEP = 0.685）。

TFINAL和τFINAL的约束是由弱同步下的安全证明决定的; 附录C.1显示TFINAL= 0.74时τFINAL = 10,000就足够了。

有这些参数，即使优先级最低的块提议者是恶意的（提出不同的块），BA也能确保安全。 附录C提供了弱同步下的BA⋆安全性（§C.1），强同步下的BA⋆活跃性（§C.2），还有BA⋆效率（§C.3）。

**八、             ALGORAND**

到目前为止，我们已经描述了在原型上建Algorand和要求Algorand解决本节讨论的一些更高级别的问题。

**8.1 块格式**

Algorand的块包含一系列交易，以及BA⋆所需的元数据。 具体来说，元数据由轮数，提议者的基于VRF的种子（第6节），总账中前一个块的散列以及指示何时提出块的时间戳组成。 块中的交易清单逻辑上转换为每个用户公钥的权重集合（根据该密钥的货币余额），以及所有未偿还货币的总权重。

一旦用户收到来自最高优先级提议者的块，用户在将块传递给BA⋆之前确认块内容。 特别是，用户检查所有交易是否有效; 种子是有效的; 前面的块散列是正确的; 该块的轮数是正确的; 并且时间戳大于前一个块的时间戳，也近似于当前（比如在一个小时内）。如果其中任何一个不正确，用户将空白块传递给BA⋆。

**8.2 安全性和活跃性**

在很大程度上，Algorand依靠BA⋆在总账中达成共识。 Algorand仅在出现在最后一个区块或最后一个区块的前身时才确认交易。 最后的区块保证在同一轮中没有其他区块可以达成共识。 这意味着所有的最终块相互之间是完全有序的，因为（1）块形成一个线性链，并且（2）在链中的任何给定位置上只能有一个最终块。 换句话说，给定两个最后的块，其中一个（具有较小轮数r 1的那个）必须是另一个（具有较高轮数r 2的）的前驱，因为在r 1轮中必须有r 2块的前驱，和安全条件，保证r 1块是唯一可能的这种块。

剩下的问题是，如果网络不是强同步，BA⋆可能创建分支（即不同的用户在不同的块上达成共识）。 这并不违反安全性，因为BA⋆在这种情况下会返回暂定共识。 然而，分叉的确影响活力：不同的分叉上的用户将具有不同的ctx.last\_block值，这意味着他们不会计数相互的选票。 因此，至少有一个分叉（也可能是所有的分叉）没有足够的参与者来超过投票的阈值，BA⋆将无法在该分支的任何块上达成共识。

为了解决这些分叉，Algorand定期提出一个所有用户都应该同意的分支，并使用BA⋆就所有用户是否应该切换到这个分支达成共识。 ——为了确定可能的分叉集合，Algorand用户被动地监视所有BA⋆投票（即，甚至投票prev\_hash值与当前用户链不匹配），并跟踪所有的分叉。 然后，用户使用松散同步时钟来停止常规块处理，并在每个时间间隔（例如，每小时）启动恢复协议，这将提出这些分叉之一作为每个人都应该同意的分支。

恢复协议开始于用户使用块提议机制（§6）提出一个分支。 具体而言，如果用户被选择为“分叉提议者”，则用户提出一个空的块，其前驱散列是用户目前观察到的最长分支（通过块的数量）。 每个用户都等待最高优先级的分叉提议，就像块提议机制一样。 每个用户通过确保块的父指针是一个与该用户看到的最长链一样长的链，来验证建议的块。 选择最长的分叉可以确保这个分叉包含所有的最终块。 最后，用户调用BA⋆在这个块上达成共识，通过在建议块中找到的轮数。

为了让BA⋆在其中一个分叉上达成共识，所有的Algorand用户必须使用相同的种子和用户权重。 这意味着Algorand必须在发生任何可能的分叉之前使用用户权重和种子。为此，Algorand依赖于弱同步的假设，即在每一个长度为b的时期（以b为1天），必定有一个长度为s <b的强同步期（把s看成几个小时）。 在这个假设下，使用块时间戳，Algorand将时间量化为b-long时间段（天数），并从倒数第二个完整的b-long时间段找到最近的块。 然后，Algorand使用来自该块的种子，并使用来自至少b-long时间之前同意的最后块的用户权重（§5.3）。

Algorand使用从来自倒数第二个b长的时间段的块中的种子，因为最近的b-long时期可能还有一个未解决的分叉。 这样的分叉将阻止用户同意恢复中使用的种子和权重。 然而，只要Algorand在最近的b-long期间的s-long强同步期内能够恢复，所有用户将在倒数第二个时间段（只要它们的时钟大致同步）内同意同一个块。

为了确保Algorand在s -long同步期间从一个分支（即，大多数诚实的用户切换到相同的分支）恢复，Algorand用户反复尝试在分支上达成共识（每次为了产生一个不同的提议者集合和委员会成员，将散列函数应用于种子），直到他们达成共识。 由于通过假设，Algorand是在一个强同步时期内运作的，所以BA⋆在这种情况下是否返回“最终”或“暂定”的共识并不重要。 当Algorand在强同步时期外恢复时，我们不能确保在s时间内恢复。

**8.3 Bootstrapping**

**引导系统。** 要部署Algorand，必须为所有用户提供一个共同的创世纪块，以及最初的加密抽签种子。 在初始参与者集合的公钥和权重是公知的之后，使用分布式随机数生成来确定在创世纪块中指定的seed0的值。

**引导新用户。** 加入系统的用户需要了解系统的当前状态，该状态被定义为BA⋆共识结果，链的结果。 为了帮助用户赶上，Algorand为BA⋆（包括空白块）同意的每个块生成证书。 该证书是来自BinaryBA⋆（）最后一步（不包括FINAL步）的票数的总和，足以让任何用户通过处理这些票数（即，必须至少有⌊TSTEP·τSTEP⌋+ 1票）达到相同的结论。 重要的是，用户必须像在算法6中一样检查抽签散列和证明，并且证书中的所有消息都用于相同的Algorand 轮和BA⋆步骤。

证书允许新用户验证以前的块。 用户从创世纪块开始按顺序验证块。 这确保用户知道在任何给定的轮中验证抽签证明的正确权重。 用户还可以要求证明一个块的安全的证书; 这只是FINAL步简单的选票收集。 由于最终块是完全有序的，用户只需要检查最近块的安全性。

使用证书造成的一个潜在风险是对手可以提供证书，似乎表明BA⋆在大量步骤后完成。 这使对手有机会找到一个BA⋆步数（直到MAXSTEPS），在这个步数中，对手控制了超过选定委员会成员的阈值（然后用他们的私钥创建一个签名的证书）。 我们把委员会的规模设定得足够大，以确保攻击者找到这样的步数的可能性微乎其微。 当τSTEP > 1 000时，每一步攻击的概率小于2-166，让这样的攻击不可实行。

**存储。** 块历史记录和匹配证书允许新用户追上，而对于已经与当前总账保持同步的用户不是必需的。 因此，Algorand在用户之间分发证书并块存储。 对于N个碎片，用户存储其轮数等于其公钥模N的块/证书。

**8.4 通讯**

**传播块和中继消息。** Algorand的块提议协议（§6）假设选定的用户可以在对手可以学习用户的身份并他们进行针对性的DoS攻击之前传播新的区块。 在实际中，Algorand的块大于最大包大小，因此来自选定块提议者的一些包将不可避免地被发送。 一个特别快的对手可以利用这个优势立即开始发送多个数据包，DOS任何用户，假定用户是一个块提议者。

从形式上来说，这意味着Algorand的活跃性保证在实践上略有不同：只要攻击者无法在受害者通过TCP连接（几秒钟）发送块的时间内有针对性的DoS攻击，Algorand就可以确保活跃性，而不是面对立刻有针对性的DoS攻击提供活力。我们认为这并不重要，有如此快速的反应时间的对手也可能对网络有广泛的控制，从而可以防止Algorand节点之间的通信。另一种方法可能是依靠Tor使对手很难快速断开用户连接。

为了避免对手发送垃圾信息和压倒Algorand的gossip网络，Algorand节点必须在中继消息之前对消息进行验证。 具体来说，Algorand节点应该使用算法6来验证每个消息，避免中继多于一个的消息，每个消息⟨round, step⟩由一个给定的公钥签名。

**可扩展性。**每个用户的通信成本取决于委员会的预期大小和通过τPROPOSER，τSTEP和τFINAL（与用户数量无关）Algorand设置的块提议者的数量。 随着更多的用户加入，在gossip网络中传播消息的时间会更长。 Algorand的gossip网络形成一个随机的网络图（每个用户连接到随机的对等点）。 我们的理论分析表明，几乎所有用户都将成为图中一个连接组件的一部分，传播时间随着该组件的直径而增长，这是用户数量的对数。 实验证实，Algorand的表现只受到更多用户的轻微影响（§10）。

由于我们的随机图使用固定数量的对等点，所以一个潜在的问题是它可能包含断开的元件。 但是，只有一小部分用户可能会在断开连接的组件中死亡，这对BA⋆没有任何问题。 而且，Algorand每轮替换传播同伴，这有助于用户在前一轮可能断开链接的情况下恢复。

**九、             实现**

我们用C ++实现了一个Algorand的原型，由大约5,000行代码组成。 我们使用Boost ASIO库进行网络连接。 签名和VRFs是通过Curve 25519实现的，我们使用SHA-256作为哈希函数。 我们使用Goldberg等[27：§4]中概述的VRF。

在我们的实现中，每个用户连接到4个随机对等点，接受来自其他对等体的传入连接，并将消息传播给他们所有人。 这给了我们平均8个对等点。 我们目前为每个用户提供一个“地址簿”文件，列出每个用户的公钥的IP地址和端口号。 在真实世界的调度中，我们想象用户可以传播这些信息，用他们的密钥签名，或者通过公共公告牌发布信息。 在我们的原型实现中，这个gossip协议的天真设计可能会受到Sybil攻击的影响，因为它不能阻止对手以大量的身份加入gossip网络。 我们将实施防Sybil 的gossip网络的问题留给未来的工作。

我们的实现和§7中显示的伪代码之间的一个区别在于BinaryBA⋆（）函数。 算法8中的伪码在达成共识后，在接下来的3个步骤中进行投票。 为了提高效率，我们的实施代替回到前面的3个步骤，可能在未来的一步中达成共识。 这个逻辑产生相同的结果，但是在伪代码中更难表达。

图4显示了我们的Algorand原型中的参数; 我们通过实验验证了第10节中的超时参数。 h = 80％意味着对手需要控制Algorand货币的20％才能创造一个分叉。 以此类推，在美国，最高的0.1％的人拥有大约20％的财富，所以最富有的30万人将不得不串通起来创造分叉。

λPRIORITY应该足够大，以允许块提议者传播他们的优先级和证明。 比特币网络中消息传播的测量表明，传播1 KB到90％的比特币对等网络大约需要1秒。 我们保守地将λPRIORITY设置为5秒。

λBLOCK确保即使块提议者不发送块，Algorand也可以取得进展。 我们的实验（第10节）显示，约10秒钟就可以传播一个1MB的块。 我们保守地将λBLOCK设置为一分钟。

λSTEP应该足够高，以允许用户从委员会成员那里接收消息，但是如果没有足够多的委员会成员的意见，则足够低以允许Algorand取得进展（转向下一步）。 我们保守地将λSTEP设置为20秒。 我们将λSTEPVAR ，BA⋆完成时间的估计方差，设置为10秒。

**十、             评估**

我们的定量评估回答以下问题：

•Algorand在确认交易方面可达到的延迟是多少？随着用户数量的增长，它如何扩展（scale）？（§10.1）

•Algorand在每秒交易方面的吞吐量可达到多少？（§10.2）

•Algorand的CPU，带宽和存储成本是多少（§10.3）

•当用户行为不当时，Algorand如何执行（§10.4）

•Algorand是否选择合理的超时参数？（§10.5）

为了回答这些问题，我们使用1,000 m4.2xlarge虚拟机（VMs）在Amazon的EC2上部署我们的Algorand原型，每个虚拟机具有8个内核和高达1 Gbps的网络吞吐量。为了用大量的用户来衡量Algorand的性能，我们在同一个虚拟机上运行多个Algorand用户（每个用户都是一个进程）。默认情况下，我们每个虚拟机运行50个用户，用户建提议1兆字节的块。要模拟商品网络链接，我们将每个Algorand进程的带宽限制为20 Mbps。为了模拟网络延迟，我们使用城市间延迟和抖动测量，并将每台机器分配到全球20个主要城市之一;同一城市内的延迟可以忽略不计。我们为每个用户分配相同的资金;货币的平均分配最大化了用户需要处理的消息的数量。本节其余部分的图形绘制了Algorand完成整个轮的时间，并包括所有用户的最小值，中值，最大值，第25和第75百分位 times。

**10.1 延迟**

图5显示了用户数从5,000改变到50,000（通过将活动虚拟机数量从100改为1,000）的结果。 结果表明，Algorand可以在一分钟内确认交易，并且随着用户数量的增长，延迟几乎不变。 （由于τFINAL = 10,000，完成FINAL步所花费的时间会增加，直到系统中有10,000个用户;在此之前，用户不止一次被选中，以较高的权重发送较少的选票。）

为了确定Algorand是否继续扩展到更多用户，我们运行了一个每个虚拟机500个Algorand用户进程的实验。 这个配置运行有两个瓶颈：CPU时间和带宽。 大部分的CPU时间花在验证签名和VRFs上。 为了缓解我们的实验设置中的这个瓶颈，对于这个实验，我们用相同持续时间的睡眠来代替验证。 我们无法缓解带宽瓶颈，因为每个虚拟机的网络接口都是最大的;我们用增加λSTEP到1分钟来代替。

图6显示了这个实验的结果，将用户数量从50,000增加到500,000（通过将虚拟机的数量从100改变到1,000）。 由于带宽瓶颈，本实验中的延迟比图5中的延迟大4倍，即使是相同数量的用户也是如此。 然而，扩展性能仍然大致持平至50万用户，这表明Algorand的扩展很好。

**10.2 吞吐量**

在以下一组实验中，我们在1,000个虚拟机（每台计算机50个用户）上部署了50,000个用户。 图7显示了不同块大小的结果。 这个数字将Algorand轮分成三部分。 在图表底部的块提议（§6）是用户获得提议块所花费的时间。 小块的块提议时间主要由λPRIORITY+λSTEPVAR等待时间控制。 对于大块，传播大块内容的时间占主导地位。 BA⋆除了FINAL步之外，在图表的中间，是BA⋆达到FINAL步所花费的时间。 最后，BA⋆的FINAL步，在图表顶端，就是BA⋆完成FINAL步所花费的时间。 我们分开FINAL步，因为为了吞吐量的目的，可以在下一轮流水线化（尽管我们的原型没有这样做）。

结果显示，Algorand的约定时间（即BA⋆）与块大小无关，即使对于大块也保持相同（12秒）。 在下一轮Algorand中，通过流水线化需要大约6秒时间的FINAL步来进一步提高吞吐量。 运行BA⋆的固定时间和块传播时间（块的大小）的线性增长表明，增加块大小允许分摊为了提交更多数据而运行BA⋆所花费的时间，因此达到最大化网络能力的吞吐量。

在最低延迟时间内，Algorand在大约22秒内提交了一个2 MB的块，这意味着它可以每小时提交327 MB的交易。 比较而言，比特币每10分钟提交1 MB的块，这意味着每小时可以提交6 MB的交易。 随着Algorand的块大小的增长，Algorand以延迟增加为代价实现了更高的吞吐量。 例如，对于10 MB的块大小，Algorand每小时提交约为750 MB的交易量，这是比特币的125倍。

**10.3 运行Algorand的成本**

运行Algorand的用户将承担CPU，网络和存储成本。 运行Algorand的CPU成本适中; 当每个虚拟机运行50个用户时，8核虚拟机的CPU使用率约为40％（其中大部分用于验证签名和VRFs），这意味着每个Algorand进程使用核心的6.5％。 就带宽而言，在我们的实验中，每个用户有1 MB的块，50,000个用户使用大约10 Mbit / sec（根据经验计算为发送的数据总量除以实验的持续时间）。 我们注意到，每个用户的通信成本与运行Algorand的用户的数量无关，因为用户拥有预期的固定数量的传播消息的邻居，并且共识协议中的消息数量取决于委员会的规模（而不是总用户数）。

在存储成本方面，Algorand存储了块证书，以向新用户证明已经提交了块。 这个存储成本除了块本身之外没有了。 每个块证书是300K字节，与块大小无关; 对于1 MB的块，这将是~30％的存储开销。 用户间的分块存储（§8.3）可以按比例降低存储成本。 例如，以10为模的分片将要求每个用户平均为附加到总账的每1MB块存储130KB。

**10.4 行为不当的用户**

Algorand的安全性由BA⋆（§7）保证，但通过实验证明这将需要测试所有可能的攻击者策略，这是不可行的。但是，要通过实验表明我们的Algorand原型处理恶意用户，我们选择一个特定的攻击策略。我们强制优先级最高的块提议者对所提议的块进行模糊处理：即，提议者将块的一个版本发送给他一半的对等点，将另一个版本发送给其他人（注意，作为优化，如果用户在块提议步骤完成之前从最高优先级块提议者那里接收到的块的版本冲突，他丢弃两个提议并且以空块开始BA⋆）。被选为BA⋆委员会成员的恶意用户对这两个块投票。图8显示了Algorand的性能如何受到恶意用户权重部分的影响。结果显示，至少根据经验对于这次特殊的攻击来说，Algorand并没有受到太大的影响。

**10.5 超时参数**

上述结果证实BA⋆步在λSTEP（20秒）内完成，BA⋆完成次数的第25和第75百分位之间的差在λSTEPVAR（5秒）以下，并且该块在λBLOCK （1分钟）内传播。 我们分别衡量传播块提议者的优先级和证明所需的时间; 它始终在1秒左右，远低于λPRIORITY（5秒），证实了Decker和Wattenhofer的测量结果。

**十一、     未来工作**

本文侧重于实现交易的共识机制，并解决相关的可扩展性和安全性问题。 在设计无许可的加密货币时，仍然存在一些未解决的问题：

**激励。**为了鼓励Algorand用户参与，即在选择时上网，并支付运营Algorand的网络成本，系统可能需要包括激励，可能以奖励机制的形式。 设计和分析一个激励机制包括许多挑战，如确保用户不会有不正当的激励（例如，拒绝投票），恶意用户不能“钻空子”获得比遵守协议的用户更多的奖励（例如 ，通过影响种子选择）。

**加盟成本。** 要加入Algorand，新用户可以使用随附的证书获取所有现有的块，这可以包含大量的数据。 其他加密货币也面临类似的问题，但由于Algorand的吞吐量相对较高，这可能会带来可扩展性挑战。

**转发安全。** 攻击者可能会随着时间的推移而试图腐败用户，因为委员会成员的身份在发送消息之后就会暴露出来。 如果攻击者设法获得足够的用户密钥，他可以构造一个伪造的证书来创建一个分支。 一种解决方案是用户在发送签名的消息之前忘记签名密钥（并提前提交一系列签名密钥，假定使用基于身份的加密方法）。

**十二、     总结**

Algorand是一个新的加密货币，大约1分钟确认交易，同时几乎可以忽略不计分叉的概率。 Algorand的设计基于与BA⋆拜占庭协议相结合的加密抽签机制。 Algorand在每个步骤使用参与者替换的选定参与者来避免有针对性的攻击。 以Algorand为原型的实验结果表明，它实现了sub-minute延迟和125倍的比特币吞吐量，并且可以扩展到50万个用户。

**附录：（公式见原文）**

**A 不可预见性选种的必需块数**

**定理1.**在强同步下，攻击者可以计算出下一个k选择种子的概率随k减小。

*证明.*让h表示诚实用户的分数。当第r轮约定的块是由诚实的用户i提出时，那么在第r + 1轮的块提议者中设置优先权的seedr + 1取统一的随机值。因此，在第r + 1轮提出块的最高优先级具有概率h属于诚实用户的，并且在增加的序列中每个优先级也是如此（第二高优先级也具有概率h，等等）。如果优先级属于一个诚实的用户，那么这个用户会提出一个非空的块，它将成为第r + 1轮的唯一认证块。如果c个最高优先级属于恶意用户，那么攻击者可以使这些c用户中的任何一个提出下一个块（并且具有由所有诚实用户同意的块），或者他可以强制第r + 1个块成为空块（通过发送具有最高优先级的选择证明，然后不发送实际的块）。所以敌手对seedr + 1的值有c + 1的选择。这些选项使对手有能力影响未来的种子，并确保恶意块提议者的概率大于1-h。我们现在证明，敌手影响未来种子的能力有限。

为了证明，我们构建一棵树，每个节点表示一个用户的优先级值，并包含该用户设置的下一个选择种子（如果关联的优先级是最高的）。树中的每个节点都可以属于诚实的或恶意的用户。这棵树的根源属于最后一位诚实的用户，他们提出了一个被认可的块。根中的选择种子是seedr，它是诚实用户提出的块中的最后一个种子。根的孩子代表从左到右排序的用户，按seedr确定的优先顺序递减，并且每个子节点也包含seedr + 1的可能值。我们使用我们用于根的相同过程来定义每个子节点的子节点。每个节点的子节点定义用户，这是与所有用户的优先级的一个伪随机置换（通过在父节点的选择种子设定）采样的顺序。

我们将由诚实/恶意用户设置的块称为诚实/恶意块（分别）。我们希望在两个连续的诚实块之间限制块的期望数量。为了证明，我们给予对手给定选择种子的权力，以优先级的降序来知道用户的位置。如果根的最左边的子代表的是一个诚实的用户，那么在第r + 1轮的块将是诚实的，因此两个诚实块之间的恶意块的数目是0.如果最左边的子代是恶意的，下一个诚实的块至少是1。这发生的概率为1-h，所以至少有一个非诚实的块出现在下一个诚实块之前的概率是1-h。如果我们让f（k）表示在下一个诚实的块之前至少有k个非诚实块出现的概率，那么f（1）= 1 -h。我们现在用f（1）作为基本情况，以归纳方式计算f（k）的上界。

对于f（k + 1），如果root根有第一个诚实的孩子留下的c个恶意的孩子，那么对手对于seedr+ 1有c + 1个选项。 如果敌手能够生成至少k个恶意块，那么攻击者可以生成k + 1个恶意块，这些恶意块是从他拥有的用于该块后面的块的c + 1选项之一开始的。 c + 1个选项中至少有一个产生k个非诚实块的概率小于（c + 1）f（k）。 所以我们得到递归公式：

所以通过归纳，

由于h≥2/3，认为，

因此，对于攻击者可以连续控制k个块的失败概率f，选择k = log0.84 f就足够了。 对于f = 10-18，使用k = 238是足够的。如果我们假设一个更高的诚实率，例如h = 3/4，那么使用k = 77就足够了。

**B选择的用户数量**

我们至少有h·U个诚实的子用户，其中U是系统中最小的Algorand单元的总数。 我们假设阿尔戈兰德有大量的货币单位（即U可以任意大）。 在每一轮中，为一个给定的角色选择的子用户的数量是变化的，但是是预期选定的子用户数量τ是固定的。 要被选择的子用户的概率p由公式p =τ/U给出。 K个子用户的抽样概率是多少？

有U的(U/K)个子集，恰好由K个子用户组成。 对于每一个这样的集合，恰好该集合的子用户从U采样的概率是pK（1-p）U -K。所以K K∈{0,1...}说的子用户抽样的概率是：

这等于：

对于Algorand来说，我们可以用U来任意大，并且固定K，

和

所以对K个子用户进行抽样的概率是：

**B.1阻止提议者- Block proposers**

我们需要每轮至少选择一个提议者，但是不要选择太多的用户。 让提议者的预期数量为τproposer。 我们可以使用公式2中的公式来获得0个好提议者的概率为：

对于τpropose= 26，我们得到至少一个提议者但不超过70的概率是：

**B.2委员会成员-Committee members**

如果＃good是诚实委员的数量，#bad是恶意委员的数量，那么我们希望在预定的阈值Tstep·τstep时满足以下条件。 我们给出了每个条件被违反的概率的明确公式。

条件（1），＃good> Tstep·τstep。 当诚实委员人数≤Tstep·τstep时，违反了这个条件。 从方程式2，我们确切地有K个诚实的委员会成员的概率是 因此，违反条件的概率由下式给出：

条件（2）， 。如上所述，#bad = L的概率是：

所以＃good= K和#bad = L的概率是：

当K + 2L> 2Tstep·τstep时违反条件，所以条件（2）违反的概率由下式给出：

这个总和收敛，我们将上限定义为h，Tstep，τstep参数的特定选择。

具体而言，选择参数Tstep和τstep以确保两个条件在随机生成的委员会中以概率F保持。 在这里，F是一个参数，它表示任何一种条件的失败概率可以忽略不计，我们通常将其设置为F = 10-12或F = 10-18。

**分析图3。**对于诚实委员h（图3的x轴）的给定比例，目标是最小化预期的委员会规模，同时保持条件（1）或（2）最多不超过F的概率。如果τstep的某个值满足概率为1-F的条件，并且对于某个适当的Tstep值，则任何较大的τstep值也对于同一个步骤具有至少1-F的概率。因此，为了找到最佳的τstep，我们从τstep的任意高值开始，比如10,000，然后看看是否能够找到满足条件（1）和（2）的Tstep> 2/3。如果存在这样的阈值，则减少τstep并测试是否存在好的阈值。我们继续这个迭代过程，直到找到确保两个条件的最小委员会规模和相应的临界Tstep。

为了检查τstep的特定值是否工作，我们进行二分搜索以找到最高阶段∈（2/3,1）值，使得条件（1），＃good> Tstep·τstep以概率≤(1/2)F失败，然后检查如果对于特定的价值Tstep，条件（2）也以概率≤1/2F失败，通过联合约束，任一条件失败的概率是F，所以预期的委员会规模τstep起作用。

通过上述方法，我们找到τstep的最佳值的近似值，该最优值在τstep的真实最优值的1％以内。

**C BA⋆ ANALYSIS**

在本节中，我们将分析第7节介绍的BA⋆的安全性，活性和效率。

**C.1安全性弱同步- Safety with weaksynchrony**

**定理2.**假设参数Tfinal = 0.74，τfinal= 10,000，Tstep = 0.685，τstep= 2000（如图4所示）。修正r = 0 mod R（其中R是选择种子刷新间隔，见§5.2）。存在一个圆r <r'<r + R的概率，使BA（算法3）为一个诚实的用户返回一个块A的最终共识，对于另一个诚实的用户，一个共同的（最终的或暂定的）块 B≠A，并<10-7。

**推论3.**由于定理2和R = 103（即每隔103轮以R的整数倍开始，从同一选择种子中获得其委员会），我们期望BA⋆返回一个诚实的用户对块的最终共识 A，对另一个诚实的用户来说，对于同一回合的块B≠ A来说是一个共识（最终的或者是暂定的），每103·107= 1010轮只有一次。

**证明.** 我们接下来证明了定理2.在§5.3和附录A，我们展示了弱同步假设允许Algorand在对手知道选择种子之前选择用户的公钥（和相关的权重），确保用户对BA⋆委员会的随机抽样。 我们接下来利用这个属性非正式地表明委员会成员最后一步代表系统中的所有用户。 因此，如果足够多的用户投票决定一个区块，那么有足够多的用户在第一步完成BinaryBA⋆并且不会在将来的步骤中投票（所以没有其他区块可以被批准）。

我们设置τfinal= 10,000，并假设用户的加权诚实比例是h = 0.8。 因此，最后一步委员会中诚实委员的预期数量是0.8·1万= 8000，而预计恶意委员的数量是0.2·10000 = 2000。我们接下来表明，产生了一个round-r'方块A的7400个最后一个投票的概率，而另一个方块B也被认证为同一回合的概率是微不足道的。

设seedr-1-（r mod R）为用于选择圆r'∈[r，r + R）委员会的种子。 如果当种子发布的时候，即网络没有强烈同步的时候，发生轮r-1（r mod R），那么攻击者可以通过丢弃来自诚实用户的块提议来操纵种子（seedr-1-（r mod R））的选择。 我们首先表明，尽管攻击者有操纵种子的能力，计算种子是不可行的：（1）在r’轮的最后一步中给了i个恶意的委员会成员;（2）j恶意委员会成员在其他一些r'，其中i + 3j> 4,100。

i和j的预期值分别是0.2·1000 = 2000和0.2·2000 = 400。 因此，对于固定的一轮和一步，我们应用公式2中的公式，以找到随机种子给出的概率i + 3j> 4,100：

为了检查种子值是否满足i + 3j> 4,100，攻击者必须至少执行一次加密哈希或VRF操作。 每轮最多有MaxSteps = 150步的委员会。 对于每一轮，150个（即100个）委员会中只有2/3可以创建一个证书（只有那些在BinaryBA⋆循环中的第一步或第二步投票的人）。 所以通过联合约束，一个种子值对于一个特定的回合（以及该回合的任何一个步骤）给出i + 3j> 4,100的概率是≈2-93。 所以对手找不到这样的种子是不可能的。

请注意，我们上面的论点并不排除多于一对的圆桌决赛委员会有i个恶意的成员和一个有针对性的BinaryBA⋆委员会有j个恶意成员，其中i + 3j = 4,100。 然而，直觉上来说，如果攻击者把所有的计算能力集中在寻找一轮（具有高度恶意的最终委员会来完成A块）并且在该轮内有一个有针对性的步骤（用高度恶意的委员会来证实B块），使得i + 3j = 4,100，那么对于所有其他轮次和步骤，恶意委员会成员的数量是随机的，这使得攻击者成功进入其中一个循环/步骤对的概率可以忽略不计（<10-20）。

更准确地说，如果攻击者以i + 3j> 3850为目标的两个循环/步骤对，那么这两个对将具有i + 3j <3950。这是事实，因为i + 3j> 3850的概率大约是0.41·10 -18，而i + 3j> 3950的概率大约为0.6·10-23，所以两个事件发生的概率小于2.5·10-41。 因为对于一个特定的选择种子，R = 103个回合，每个回合有102个步骤，有 = (1/2)1010个可能的回合/步骤对。 攻击者设法瞄准多于一个回合/步骤的概率，其中i + 3j> 3950，因此在联合限制下小于2.5·10-41·0.5·1010 = 1.25·10-31 <2-93，所以它不应该发生。

总之，攻击者有两种策略：（1）以i +3j≤4100为目标，在目标轮一步进行，或者（2）以i +3j≤3950为目标的多个步/轮对。接下来分析这两个策略。

**策略1：**一轮/一对，其中i +3j≤4100。如果r’轮的最后一步委员会有i个恶意成员，那么同一轮的任何其他委员会最多只有4100个恶意成员。显然，最坏的情况是i + 3j = 4100。因此，为了证明失败的概率是10-7，我们需要证明对于i + 3j中所有的整数元组（i，j） 4100，两个委员会有足够的用户的概率小于10-7。等价地，对于区间[0,4100/3]中的每个整数值j，我们需要证明对于i = 4100-3j，两个委员会有足够的诚实成员（即最终委员会的概率为0.74·10000-i = 7400-i和另外一个委员会的0.685·2000-j = 1370-j诚实成员）小于10-7（自Tfinal = 0.74，τfinal= 1000，Tstep= 0.685，τstep= 2000）。对于每个积分值j∈[0,41003]我们做了一个单独的计算来证明期望的概率总是小于10-7。

我们接下来描述用于上限每个元组（i，j）=（4100-3j，j）的概率的方法。

固定（i，j）。 i个恶意票进入最后的A区，而这张j票是去证明另一个区块B。因此，最终委员会需要7400-i个签名，另一个委员会需要1370-j个签名。 假设A表示将投票批准A块的诚实用户集合作为最终（如果在第一步完成BinaryBA⋆并选入最终委员会），以及如果他们在一个委员会里，并且没有在第一步完成BinaryBA⋆，那么B将会投票证实另一个区块B（≠A）的诚实用户。 显然，A∩B=∅。 所以每个诚实的用户最终都会成为两套中最多一个的成员。 在7400和1370两个门槛都达到的情况下，集合A必须包括7400-i个委员会成员进行决赛，而B组必须包括1370-j个委员会成员进行另一个步骤。。

我们说，一个诚实的用户“加入”集合A，当该用户计数足够多（超过Tstep · τstep）的投票block\_hash在第一步完成BinaryBA⋆。 否则（在BinaryBA⋆的第一步之后），我们说诚实用户“加入”集合B。 如果没有很强的同步性，网络就处于攻击者的控制之下，所以攻击者可以选择用户是否加入集合A或B（通过投票给用户）。 这允许攻击者得到一些反馈：如果攻击者让用户加入集合A，则他可以观察用户是否被选入最终委员会（即查看该用户是否投票）。 以这种方式，攻击者可以让用户一个接一个地加入设置，直到产生足够多的选票来批准块A为止，然后加入剩下的诚实用户给集合B。

因此，用户以某种顺序加入集合A。假设设置的用户第一次在目标回合中可以产生7400-i个最后一步投票，那么n就是“A”的大小。 设H为诚实用户持有的Algorand货币单位总数; 为了简化我们的证明，我们假设每个用户只有一个货币单位，所以H也是用户数量（如果用户有多个单位，我们认为用户由多个“子用户”代表，每个用户有一个单位 的货币）。 对于n的固定值，至多H -n用户加入集合B。 攻击者要成功，这些H -n用户必须在BinaryBA⋆的目标步骤中为B块产生1370-j票。 设Pk表示下列概率：（1）n = k，（2）集合中剩余的H-k个用户对B块产生1370-j票。Pk由下式给出：

使用方程式2中的公式，我们可以计算Pr [H中的H-k用户产生正确的C票]。 在集合B（包括H-k用户）中为委员会选择的诚实用户的预期数量是h·2000·(H-k)/H，因此对于h = 0.8，预期的用户数量是1600·(H-k)/H。 因此，等式3中的表达式等于：

敌手成功概率 ，等于：

内和收敛，可以评估。 然而，外部总和重复了总数可能是任意大的诚实货币单位。 因此，我们评估一个上限：

以上是 从0到H的积分的上部黎曼和，其中区间[0，H]被分割成100个等份。 由于函数 正在减少k，所以这是一个上限。 评估如下：

因此，失败的概率由下式计算：

**策略2：**多个循环/步骤对，其中i +3j≤3950。我们首先表明，对手不能瞄准3循环/步骤对，其中最终委员会至少有i个恶意成员和步委员会有j个恶意成员，其中i + 3j> 3850。这是因为瞄准一循环/步骤对，其中i + 3j> 3850最多为0.41·10-18，因此以三个这样的对为目标的概率是（0.41·10-18）3<10-51。对于每个选择种子，R = 103轮，每轮最多有102个步骤，攻击者可以瞄准（即BinaryBA⋆可以结束和证实一个块的步骤）。所以我们最多有103 · 102个步骤委员会为每个选择种子设置，因此攻击者可以瞄准的循环/步骤对的数量是 攻击者成功的概率是1015·10-51 = 10-36。因此，攻击者最多可以有两对，其中i + 3j> 3850，在这种情况下，我们都有i + 3j <3950。这使得我们可以分析以下两种情况（在上述攻击策略中）：

•两个目标循环/步骤对，其中3850 <i + 3j≤3950。 可以表明（对于情况i +3j≤4100，类似的计算）对于一个目标循环/步骤对，成功对手成功的概率小于10-9，因此小于2·10-9 的总。

•所有循环/步骤对都有i + 3j≤3850。同样可以看出对手的成功概率小于10-20。 由于每轮最多有102步，并且由相同的选择种子设置R = 103轮，所以攻击者成功的概率以102·103·10-20 = 10-15为界。

总之，策略2比策略1提供了更低的攻击者成功率，其中攻击者的成功率以R = 103的每个序列的10-7为界。

**C.2活泼与同步性强-Liveness with strong synchrony**

在本节中，我们表明，在强烈的同步假设下，Algorand保持活力。

**定理4**.在强同步下，所有的用户最终都会在一个块上达成一致。如果最高优先级的提议者是诚实的，那么所有诚实的用户在该块上达成最终共识。

**证明**。上面的活性定理的证明来自于下面我们证明的强同步下的BA⋆的几个性质。首先，在引理1中，我们展示了在弱同步下，CountVotes过程（算法5）不会在同一回合和步骤中为两个用户返回两个不同的非超时值。其次，我们证明定理2，它证明了CountVotes的这个性质保证了如果一个最后的块存在于一个Algorand回合中，那么在这一回合中就没有其他的块被约定了。第三，考虑到这个性质，我们在引理2中表明，如果一个诚实的用户执行的BA⋆返回某个特定回合的某个价值达成共识，那么对于所有其他诚实的用户，BA⋆表示同一价值上的共识。第四，在引理3中，我们证明一个诚实的用户执行的BA⋆总是最终返回一个值。最后，在引理4中，我们表明，如果所有诚实的用户都以相同的价值轮回称为BA⋆，那么他们都同意这一轮的价值，而且共识是最终的。

**引理1**CountVotes被两个诚实的用户调用相同的循环和步长的概率返回一个值v1≠一个诚实的用户超时，并且为另一个诚实的用户返回一个值v2∈{timeout，v1}是可以忽略的。

**证明。** 在强烈的同步下，§7.5（§B.2分析）中有关委员会的两个条件拥有压倒性的概率。 由于CountVotes返回v1，对于一个诚实的用户超时，那么该用户计算超过Tstep·τstep的委员会成员在该特定轮次和步骤中对v1的投票。 为了跨越步骤门槛，必须得到半数以上诚实委员的选票。 我们选择一个足够大的委员会，使得安全条件1/2＃good +＃bad≤Tstep·τstep具有绝对的可能性。 由于诚实委员只投一个价值，所以在安全条件成立的情况下，没有其他的价值v2可以得到足够的选票。

**引理2**：在强同步的情况下，如果一个诚实的用户完成对r值的BA⋆，并且对v值达成一致，那么所有诚实的用户都可以完成具有相同值的r的BA⋆。

**证明**。 在强同步的情况下，我们假设所有的用户在第r-1轮的块上达成共识（如果由于较早的非高度同步期而导致fork，则§8中的fork分辨协议已经解决了这个问题）。 在这种情况下，减少过程（算法7）返回每个诚实用户两个值之一：或者是空块的散列，它由前一个块完全确定，因此对于所有用户都是相同的，或者Reduce过程返回相同的block\_hash（应用引理1，我们观察到只有一个值可以在减少的第一步中超过CountVotes的阈值，因此只有一个不同于空块的散列的值是可能的）。..

由于BA⋆对BinaryBA返回的结果是一致的，所以只要BinaryBA返回值为v的诚实用户就足够了，那么对于所有其他诚实的用户来说，这是足够的。

不失一般性，让我们假设用户用block\_hash完成了BinaryBA⋆。因此，该用户使用round r和step 1（mod 3）调用的CountVotes过程返回了block\_hash。通过引理1，其他用户也可以用block\_hash结束这个步骤，或者超时。因此，没有完成该步骤中的协议的所有诚实用户将接着投票给block\_hash（因为步骤索引是1 mod 3）。完成协议的用户将传播所有他们收到的关于block\_hash的选票。

在下面的步骤（索引2 mod 3）中，由于所有在上一步超时（因此没有完成）的诚实用户将投票block\_hash，任何诚实用户调用的CountVotes过程将返回block\_hash。此外，在这一步完成的诚实的委员会成员也发了票。因此，由于我们选择了一个足够大的委员会，CountVotes将会以很高的可能性#good> Tstep·τstep返回block\_hash。这个共识将继续下去，直到下一步索引1 mod 3，其中所有用户都会同意block\_hash。

**引理3.**在强烈的同步下，Algorand最终在每一轮中确认一个区块。

证明。当BA⋆达成共识时，Algorand确认一个区块。这足以说明BinaryBA⋆最终达成了共识，这意味着BA⋆也达成共识，因为BA⋆其余的步骤数量有限。

由引理2可以看出，只有一个诚实的用户确认一个块（因为其余的将在最多3个步骤之后完成）。让我们考虑三个序列中BA⋆的步骤，就像算法8中的一个循环迭代一样。所以如果一个诚实的用户在前两个步骤中完成BinaryBA⋆，我们就完成了。

让我们假设情况并非如此，因此，所有诚实的用户都达到BinaryBA⋆的第三个“共同的硬币”步骤。如果在这个步骤中，所有诚实的用户都会得到超过价值v（两种可能的选项之一）的Tstep·τstep投票，那么他们都会在接下来的步骤中为这个价值投票，并在接下来的两个步骤中达成一致。

现在让我们假设一个互补的例子，至少有一些诚实的用户根据普通的硬币改变他们的投票。进一步假定用户设置硬币（根据算法9）是诚实的;因此，在投币步骤中没有足够多的投票值v的所有诚实用户看到相同的随机投币并且改变他们对该投币的投票（因为网络是强烈同步的，所有诚实的用户在有限的延迟内接收消息）。因此，只有三种情况需要考虑：

•**一些诚实的用户为block\_hash计算了足够的选票，其余的则观察硬币。** 在这种情况下，在硬币步骤结束时，如果普通硬币为0（即概率为1/2），所有诚实用户将投票给block\_hash。

•**一些诚实的用户计算了足够的空位投票，其余的观察硬币。**在这种情况下，在硬币步骤结束时，如果普通硬币是1（即，概率为1/2），所有诚实的用户将投票给empty\_hash。

•**所有诚实的用户观察硬币。** 在这种情况下，所有诚实的用户都将为普通硬币投票。 协议将继续进行，直到BinaryBA⋆在接下来的两个步骤中完成。

重要的是，一些诚实的用户对empty\_block计数足够的投票的情况下，其他人对block\_hash计数足够的投票的概率可以忽略不计。 见引理1。

因此，如果用户设置硬币是诚实的，我们达到一致的概率至少为1/2。 由于设置硬币的用户是从委员会中随机选出的，在委员会中至少有2/3的成员是诚实的，那么达成共识的可能性至少是1/2・2/3 = 1/3。由于达成共识的可能性在3个步骤的每个序列中大于0，协议最终达成一致。

**引理4**，在强同步的情况下，如果所有诚实的用户在同一回合中开始具有相同价值的BA⋆，那么他们就该价值达成最终共识。

**证明**。 让我们按照BA⋆的协议，并表明这是正确的。 所有诚实的用户都会使用同一个块的哈希来调用它，称之为block\_hash。 因此，在减少的第一步中，所有诚实的用户都会为block\_hash计算超过阈值的选票，并且还会在减少的第二步中对它投票，在那里他们将再次计算超过block\_hash的选票阈值，并且 因此，所有诚实的用户都会使用block\_hash调用BinaryBA⋆，并在完成一步后返回，因此也将block\_hash投票作为最终的共识。

**C.3效率**

**定理5**.在强同步的情况下，如果最高优先级的提议者是诚实的，那么BA⋆将以4个步骤达成最终共识，否则有望在13个步骤中达成共识。即使在最坏的情况下，BinaryBA⋆在MaxSteps = 150步内也没有完成，可以忽略不计（<3·10-9）。

**证明**。定理的第一部分（当最高优先级块提议者是诚实的时候达到快速和最终共识）在引理4中被显示。我们集中在第二部分，BA⋆在最坏的情况下预计在13个步骤内达成共识。

让我们分析一下共同的硬币导致共识的可能性。 BA⋆的二元一致阶段由三个重复的步骤组成。一个恶意的最高优先级的提议者可以使Binary BA⋆的前两个步骤没有达成共识，但是在每个“共同硬币”步骤之后，诚实的用户将在以下两个步骤中以概率1/3达成共识（参见引理3）。因此，在前两个步骤之后，三个步骤的每个序列以概率1/3达到一致。因此，在最坏的情况下，二进制协议总共需要2 + 3 \* 3 = 11个步骤的预期数量，并且BA⋆具有两个额外的初始步骤以减少二进制协议，即总共预期13个步骤。

由于在二进制BA⋆的3个步骤（除了第一次迭代）的每个序列中，BA⋆以概率1/3达到一致，所以在MaxSteps = 150步之后是2/3 150/3 -1<3·10-9。