Casper the Friendly Finality Gadget

摘要

我们介绍了 Casper，一种基于股权证明（PoS）的覆盖已有的工作量证明（PoW）区块链的终结系统。Casper 是结合了 PoS 算法研究和拜占庭容错共识理论的一个部分共识机制。我们介绍我们的系统，证明一些令人满意的特性，并且展示对抗长期修正和灾难性崩溃的防御措施。这种 Casper 覆盖提供了几乎任何具有对抗区块逆转的额外保护措施的 PoW 链。

1. 介绍

过去几年有大量的工作研究基于 PoS 的区块链共识算法。在一个 PoS 系统中，一个区块链通过一个任何在系统中持有币的人都可参与的过程来附加并同意新的区块，并且一个代理人具有的影响力与其持有的币（或“股份 stake”）是成比例的。这是 PoW“挖矿”的一个极其富有效率的替代方案，并且可以使得区块链不需要高昂的硬件和电力消耗代价来进行操作。

对于 PoS 的设计思想主要有两种流派。第一种，基于链的股权证明[1, 2]，模仿 PoW 的机制，以一个区块的链条为特点，并且通过伪随机地给股权持有人分配权限以创建新的区块来模拟挖矿的工作。这包括 Peercoin[3]，Blackcoin[4]以及 Iddo Bentov 的工作研究[5]。

另一种流派，基于拜占庭容错（BFT）的股权证明，它是以一个对于比如 PBFT[6]这样的 BFT 共识算法的三十年研究为基础的。BFT 算法通常证明了数学的属性；举个例子，一个算法通常可以数学地证明，只要超过 2/3 的协议参与者诚实的遵守协议，那么，忽略网络延迟，这个算法就不会决定为有冲突的区块。重定义 BFT 算法的目的为 PoS 是有 Tendermint[7]最先提出来的，并且具有现代的灵感比如[8]。尽管进行了一些修改，但是 Casper 仍然遵守着 BFT 的传统。

1.1. 我们的工作

Casper the Friendly Finality Gadget 是一个在提议机制上（proposal mechanism）的包装，一个提议区块的机制[1]。Casper 负责决定这些区块，本质上就是选择一个独一无二的可以代表权威的交易账本的链。Casper 提供安全性，但是活性要取决于选择的提议机制。也就是说，如果攻击者完全控制了提议机制，Casper 防止决定出两个矛盾的检查点，但是攻击者可以防止 Casper 决定任何未来的检查点。

Casper 介绍了几种新的特征，这些都是 BFT 不必须支持的：

- 责任制（Accountability） 如果一个验证者违反了规则，我们可以检测到它并且知道是哪个验证者违反了规则。责任制允许我们惩罚违法的验证者，解决使基于链的 PoS 承受麻烦的“nothing at stake”问题。对于违规的惩罚是验证者的所有押金。这种最大化的惩罚是对违反协议的防御手段。因为 PoS 安全性基于惩罚的大小，这可以被设置为大大超过挖矿奖励的所得，所以 PoS 提供了严格的比 PoW 更强的安全性刺激。
- 动态验证者（Dynamic validators） 我们介绍了一个安全的方法来随时间改变验证者集合（第 3 部分）。
- 防御（Defenses） 我们介绍了对抗长期修正攻击和超过 1/3 的验证者离线的攻击的防御办法，它以一个非常弱的同步性假设权衡为代价（第 4 部分）。
- 模块化覆盖（Modular overlay） Casper 作为一个覆盖的设计使其更简单地实现成为一个已有 PoW 链的升级。

我们按步骤介绍 Casper，由一个简单的版本开始（第 2 部分），然后渐进地加入验证者

[1] 这个功能扮演一个类似通常用在传统 BFT 算法中“领导选举”的抽象角色，但是它适用于 Casper 作为已有区块链终结包装的结构。

集合变更（第 3 部分），并且最终防御攻击（第 4 部分）。

2. The Casper 协议

在以太坊中，提议机制最初会是已有的 PoW 链，使第一个版 Casper 是一个 PoW/PoS 的混合物。在之后的版本中，PoW 提议机制会被一些更有效的所替代。比如，我们可以想象把区块提议转变成一种 PoS 区块循环签署的方案。

在这个简单版本的 Casper 中，我们假设有一个固定的验证者集合和一个产生已有区块的子区块的提议机制（比如，熟悉的 PoW 提议机制），形成一个一直增长的区块树。从[9]可知，这棵树的根通常称为“创世区块”。

在正常情况下，我们期望提议机制通常会在一个链表里一个接一个的提出区块（即，每个“父”区块只有一个“子”区块）。但是，在网络延迟和蓄意攻击情况下，这个提议机制有时将会不可避免的产生同一个父区块的多个子区块。Casper 的工作就是从每个父区块只选择一个子区块，从而从区块树中选择出一个权威的链。

无需处理整个区块树，出于高效的目的[2]，Casper 只考虑检查点的子树，形成检查点树（图 1a）。创世区块是一个检查点，每个在区块树（或区块数）中的高度是恰好 100 的倍数的区块也是一个检查点。区块高度为 100*k 的区块的“检查点高度”就是 k；同样地，一个检查点 c 的高度 h(c)是在检查点链中从 c 一直沿着父辈链路向回拉伸到根的元素的数量（图 1b）[3]。

每个验证者有一个押金；当一个验证者加入时，他的押金是已存入的币的数量。加入之后每个验证者的押金会随着奖励和惩罚增多和变少。PoS 的安全性源于押金的大小，而不是验证者的数量，所以对于本文余下内容，当我们说“2/3 的验证者”的时候，我们指的是押金权重部分；也就是一组押金和大小等于整个验证者集合的总押金大小的 2/3 验证者。

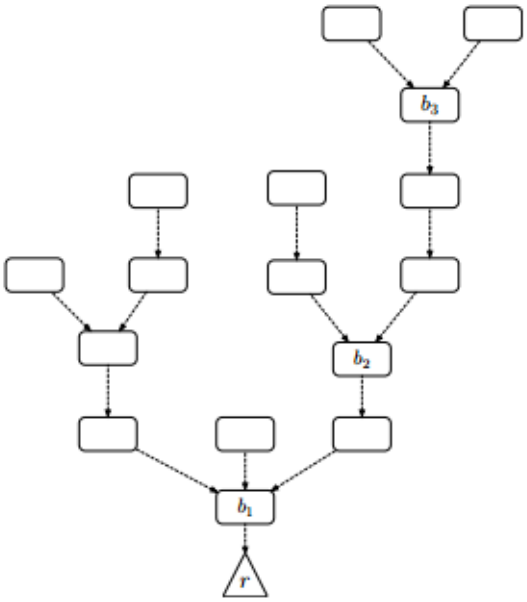
验证者可以广播一个包含四条信息投票（vote）消息（表 1）：两个检查点 s 和 t 以及它们的高度 h(s)和 h(t)。我们需要 s 是 t 在检查点树中的一个祖先，否则这个投票会被认为是无效的。如果验证者 v 的公钥不在验证者集合中，投票会被认为是无效的。与验证者的签名一起，我们把这些投票写成<v, s, t, h(s), h(t)>的格式。

我们定义以下述语：

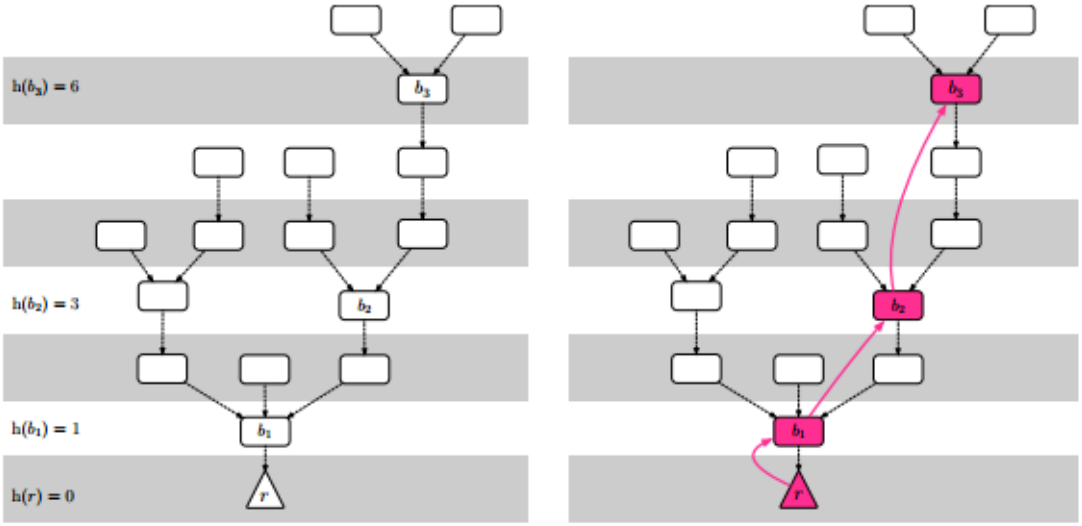
- 一个绝大多数链路（supermajority link）是一对有序的检查点(a, b)，也可以写作 a→b，以使得至少 2/3 的验证者（按押金）有发布的源为 a 目标为 b 的投票。
- 两个检查点 a 和 b 当且仅当它们是不同分支的节点的时候，被称作是冲突的（conflicting），也就是，互相不是祖先或后代。
- 一个坚持点被认为是被证实的（justified），如果(1)它是根，或者(2)存在一个绝大多数链路 c′→c，其中 c′是被证实的。图 1c 显示了一条有四个被证实的区块的链。
- 一个检查点被认为是终结的（finalized），如果它是被证实的，并且有一个绝大多数链路 c→c′，其中 c′是 c 的一个直接子检查点。同样地，检查点 c 是终结的，当且仅当：检查点 c 是被证实的，而且存在一个绝大多数链路 c→c′，然后检查点 c 和 c′不是冲突的，并且 h(c′) = h(c) + 1。

[2] 检查点之间的长距离减少了算法的开销，但是也增加了达成共识的时间。我们选择检查点之间的 100 个区块的空间作为妥协。

[3] 明确地，一个检查点的高度与在检查点树中沿着绝大多数链路回到根的祖先区块的数量不同（在下一节中定义）。



(a) The checkpoint tree. The dashed line represents 100 blocks between the checkpoints, which are represented by rounded rectangles. The root of the tree is denoted "r".



(b) The height function

(c) The justified chain $r \rightarrow b_1 \rightarrow b_2 \rightarrow b_3$

Figure 1: Illustrating a checkpoint tree, the height function, and a justified chain within the checkpoint tree.

Notation	Description
s	the hash of any justified checkpoint (the "source")
t	any checkpoint hash that is a descendent of s (the "target")
$\mathrm{h}(s)$	the height of checkpoint s in the checkpoint tree
$\mathrm{h}(t)$	the height of checkpoint t in the checkpoint tree
$\mathcal{S}$	signature of $\langle s, t, \mathrm{h}(s), \mathrm{h}(t)\rangle$ from the validator ν's private key

Table 1: The schematic of a single **VOTE** message denoted $\langle \nu, s, t, \mathrm{h}(s), \mathrm{h}(t)\rangle$.

Casper 最值得注意的一个属性是，在没有大于等于 1/3 的验证者违反 Casper Commandments/slashing 的两个[4]条件之一的情况下，来决定两个冲突的检查点是不可能的（图 2）。

如果一个验证者违反了其中一个消减条件（slashing condition），违反的证据可以被作为一个交易而包含进区块链中，在这个时候，这个验证者的全部押金会被拿走，其中一小部分当作“发现者费用”奖励给证据交易的提交者。在当前的以太坊中，停止一个消减条件的强制实施需要一次成功的对以太坊的 PoW 区块提议者的 51%攻击。

AN INDIVIDUAL VALIDATOR ν MUST NOT PUBLISH TWO DISTINCT VOTES,

$$\langle \nu, s_1, t_1, \mathrm{h}(s_1), \mathrm{h}(t_1) \rangle \quad \text{AND} \quad \langle \nu, s_2, t_2, \mathrm{h}(s_2), \mathrm{h}(t_2) \rangle \ ,$$

SUCH THAT EITHER:

I. $\mathrm{h}(t_1) = \mathrm{h}(t_2)$.
Equivalently, a validator must not publish two distinct votes for the same target height.

OR

II. $\mathrm{h}(s_1) < \mathrm{h}(s_2) < \mathrm{h}(t_2) < \mathrm{h}(t_1)$.
Equivalently, a validator must not vote within the span of its other votes.

Figure 2: The two Casper Commandments. Any validator who violates either of these commandments gets their deposit slashed.

2.1. 证明安全性和似合理的活性

我们证明 Casper 的两个根本属性：责任安全性（accountable safety）和似合理的活性（plausible liveness）。责任安全性是指，两个冲突的检查点不可能都是终结的，除非大于等于 1/3 的验证者违反了一个消减条件（意味着至少三分之一的总押金丢失）。似合理的活性是指，忽略任何之前的事件（例如，消减事件，延迟区块，审查攻击等等），如果大于等于 2/3 的验证者遵守协议，那么没有任何验证者违反消减条件而决定一个新的检查点总是有可能的。

在假设按权重 2/3 的验证者没有违反消减条件的情况下，我们得到以下属性：

(i) 如果 $s_1 \rightarrow t_1$ 和 $s_2 \rightarrow t_2$ 是不同的绝大多数链路，那么 $h(t_1) \neq h(t_2)$。

(ii) 如果 $s_1 \rightarrow t_1$ 和 $s_2 \rightarrow t_2$ 是不同的绝大多数链路，那么不等式 h(s1) < h(s2) < h(t2) < h(t1) 则无法保持。

从这两个属性我们可以立即得出，对于任何高度 n：

(iii) 存在最多一个绝大多数链路 s→t，其中 h(t) = n。

(iv) 存在最多一个被证实的高度为 n 的检查点。

通过目前的这四个属性，我们转到主要理论。

理论 1（责任安全性） 两个冲突的检查点 a_m 和 b_n 不可能都是终结的。

证明：让 a_m（及被证实的直接子检查点 a_{m+1}）和 b_n（及被证实的直接子检查点 b_{n+1}）为不同的终结检查点，如图 3 所示。现在假设 a_m 和 b_n 冲突，并且没有普遍性 $h(a_m) < h(b_n)$损失（如果 $h(a_m) = h(b_n)$，那么很清楚有 1/3 的验证者违反了条件 I）。令 $r \rightarrow b_1 \rightarrow b_2 \rightarrow \bullet\bullet\bullet \rightarrow b_n$ 是一条检查点的链，这样就存在一个绝大多数链路 $r \rightarrow b_1, \ldots, b_i \rightarrow b_{i+1}, \ldots, b_n \rightarrow b_{n+1}$。我们知道没有 $h(b_i)$等于 $h(a_m)$或 $h(b_n)$，因为那样违反属性(iv)。令 j 为最小整数使得 $h(b_j) > h(a_{m+1})$；然后 $h(b_{j-1}) < h(a_m)$。然而，这意味一条从一个时期号小于 $h(a_m)$的检查点到一个时期号大于 $h(a_{m+1})$的检查点的绝大多数链路的存在，这是与从 a_m 到 a_{m+1} 的绝大多数链路不相容的。

[4] Casper 的早期版本中有两个消息类型和四个消减条件[10]，但是我们已经简化到一个消息类型和两个严格条件。我们移除了条件：(i)已提交的哈希必须已经是被证实的，(ii)准备消息必须指向一个已经被证实的祖先。这是一个设计选择。我们做出这样的选择使得严格条件的违反独立于链的状态。

理论 2（似合理的活性） 绝大多数链路总是可以被添加以产生新的终结检查点，倘若存在子辈继承终结链。

证明：令 a 为有最大高度的被证实检查点，令 b 为任何验证者已经为其投票的有最大高度的目标检查点。任何是具有高度 h(a') = h(b) + 1 的检查点 a 的后代检查点 a'可以被证实而不违反规则 I 或 II，然后 a'可以通过加一条从 a'到 a'的一个直接子检查点的绝大多数链路而被决定，同样也不违反规则 I 或 II。

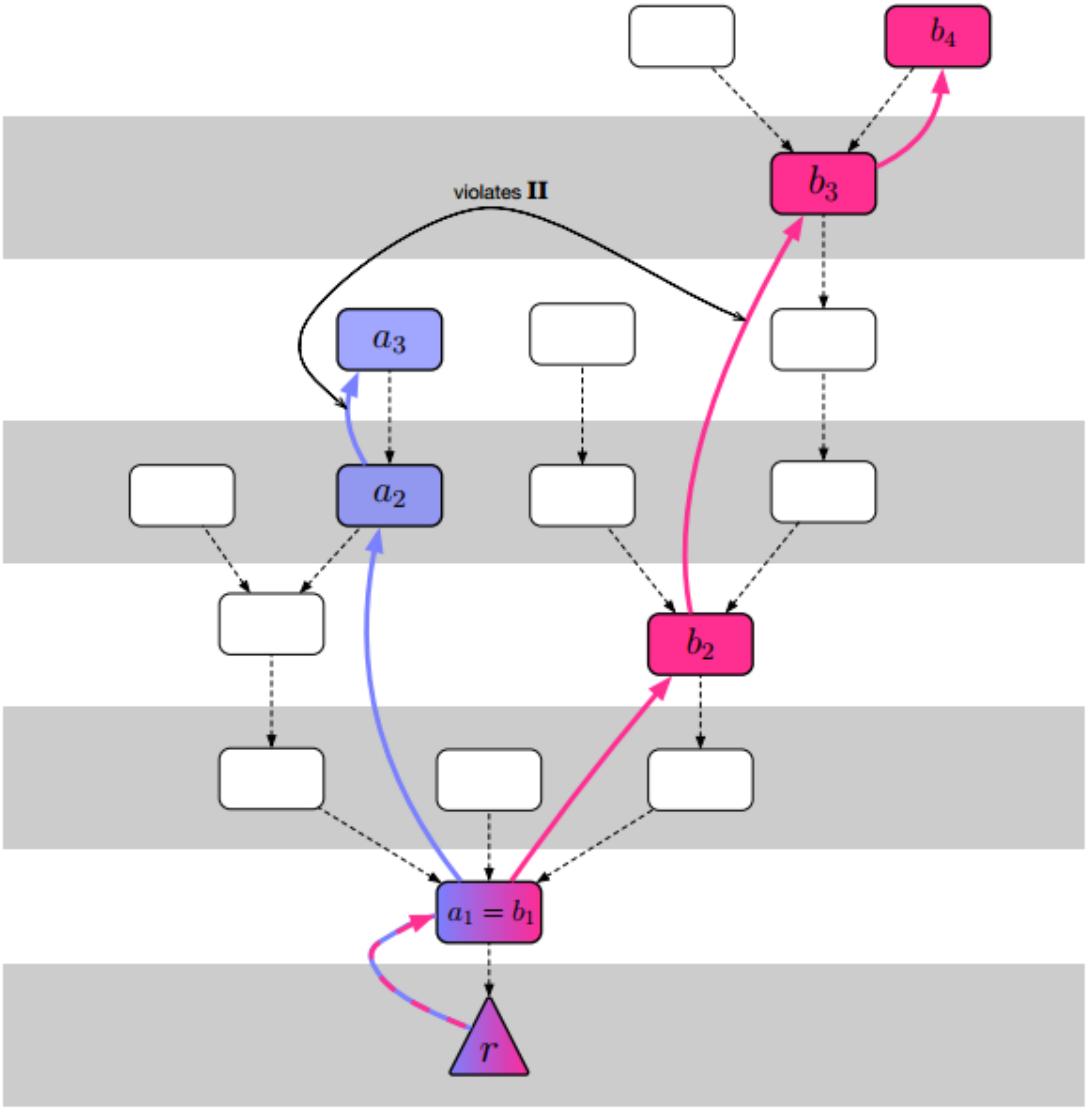


Figure 3: Figure for Theorem 1 (Accountable Safety).

2.2. Casper 分支选择规则

Casper 比标准 PoW 设计要复杂的多。例如，分支选择必须是调节好的。我们修改过的分支选择规则应该被所有用户、验证者、甚至潜在的区块提议机制所遵守。如果用户，验证者或者区块提议者不遵守标准 PoW 分支选择的“总是在最长的链之上建立”的规则的话，就会产生病态的情况，Casper 被“卡住”，任何建立在最长链之上的区块不能在没有验证者无私牺牲自己押金的情况下被决定（或者甚至是被证实）。为了避免这一情况，我们介绍一种新的，构建校正的，分支选择规则：遵循包含最大高度被证实的检查点的链。这个分支选择规则是由构建校正的，因为它遵循似合理的活性证明（理论 2），这精确地陈述了在最大

高度上的被证实的检查点之上决定一个新的检查点是可行的。这个分支选择规则将会在第 3、4 部分中进行调整。

3. 启动动态验证者集合

验证者集合需要能够修改。新的验证者必须能够加入进来，已有的验证者必须能够离开。为了实现这点，我们定义一个区块的王朝（dynasty）。区块 b 的 dynasty 就是从根到区块 b 的父区块的链中终结检查点。当一个可能的验证者的押金消息被包含在一个 dynasty 为 d 的区块中时，那么验证者 v 将会具有 d + 2 的 dynasty 并在第一个区块加入验证者集合。我们将 d + 2 称为这个验证者的 start dynasty，DS(v)。

一个验证者为了离开验证者集合，它必须发送一个“取款 withdraw”消息。如果验证者 v 的取款消息包含在一个 dynasty 为 d 的区块中，它类似地在第一个 dynasty 为 d + 2 的区块处离开验证者集合；我们把 d + 2 称作这个验证者的 end dynasty，DE(v)。如果一个取款消息还没有被包含，那么 DE(v) = ∞。一旦验证者 v 离开了验证者集合，则该验证者的公钥就会被永久禁止再次加入到验证者集合。这就移除了为同一检验人处理多次 start/end dynasty 的需要。

在 end dynasty 的开始，验证者的押金会在被取走之前的很长一段时间内上锁，这被称为取款延迟（withdrawal delay）（考虑“区块的四个月价值 four months' worth of blocks”）。如果在取款延迟期间，验证者违反了任何命令，押金会被大幅度扣除。

我们定义两个函数，可以为任何给定的 dynasty d 生成两个验证者子集，前验集（forward validator set）和后验集（rear validator set）。它们被如下定义：

$$\mathcal{V}_{\mathrm{f}}(d) \equiv \{\mathrm{v} : \mathrm{DS}(\mathrm{v}) \leq d < \mathrm{DE}(\mathrm{v})\}$$
$$\mathcal{V}_{\mathrm{r}}(d) \equiv \{\mathrm{v} : \mathrm{DS}(\mathrm{v}) < d \leq \mathrm{DE}(\mathrm{v})\}$$

注意，这意味着 dynasty d 的前验集就是 dynasty d+1 的后验集。

注意，为了让链能够“知道”其自己当前的 dynasty，我们需要稍微约束一下我们对“终结（finalization）”的定义：之前的定义是，一个检查点 c 如果是被证实的并且有一个从 c 到检查点树中任何它的直接子检查点的绝大多数链路，那么该检查点就被称为是终结的。现在，终结有一个附加条件——只有为绝大多数链路 c→c′的投票和所有的绝大多数链路递归地证明，在 c′的子检查点之前被包含到区块链中，c 才是终结的，也就是说，在区块号 h(c′) * 100 之前。为了支持动态验证者集合，我们重定义“绝大多数链路”和“终结”如下：

- 如果至少 2/3 的 dynasty 为 d 的前验集已经发布投票 s→t，并且至少 2/3 的 dynasty 为 d 的后验集也已经发布了投票 s→t，那么一组有序检查点对(s, t)（其中 t 在 dynasty d 中）具有一个绝大多数链路。
- 如果 c 是被证实的，并且存在一条绝大多数链路 c→c′，其中 c′是 c 的子检查点，那么检查点 c 就被称为是当前终结的。我们加入条件，仅当绝大多数链路 c→c′的投票以及该绝大多数链路证明 c，在 c′的子检查点之前被包含在 c′的区块链时，c 是终结的——即，在区块号 h(c′) * 100 之前。

前验集和后验集通常有很大的重叠；但是如果这两个验证者集合本质上不同，那么这种“缝合”机制防止安全故障以免一个终结检查点的孙辈由于证据包含在一个链而非另一个链中而导致存在不同的 dynasty 的情况出现。图 4 中可以看到一个这样的例子。

4. 停止攻击

有两种著名的针对 PoS 系统的攻击：长期修正（long range revisions）和灾难性崩溃（catastrophic crashs）。我们轮流介绍。

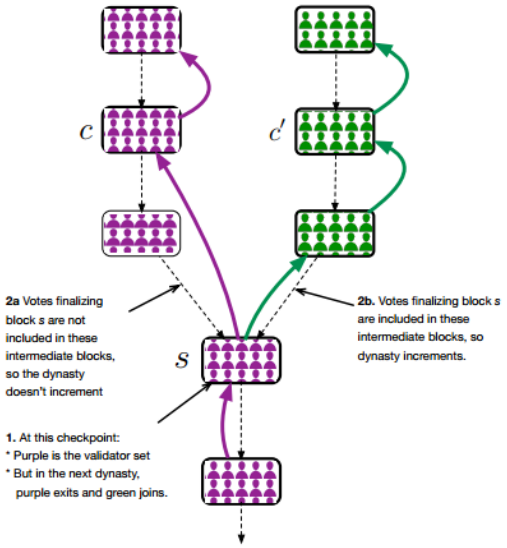


Figure 4: **Attack from dynamic validator sets**. Without the validator set stitching mechanism, it's possible for two conflicting checkpoints c and c' to both be finalized without any validator getting slashed. In this case c and c' are the same height thus violating commandment **I**, but because the validator sets finalizing c and c' are disjoint, no one gets slashed.

4.1. Long Range Revisions

一个验证者的 end dynasty 之后的取款延迟介绍了一种验证者和客户端之间的同步性假设。一旦某些验证者已经取出他们的押金，如果这些押金超过很久以来的押金的 2/3，那么他们可以使用他们的历史绝大多数链路来决定冲突检查点而无需担心押金被扣（因为他们已经取出他们的钱）。这就被称为长期修正攻击，详见图 5。

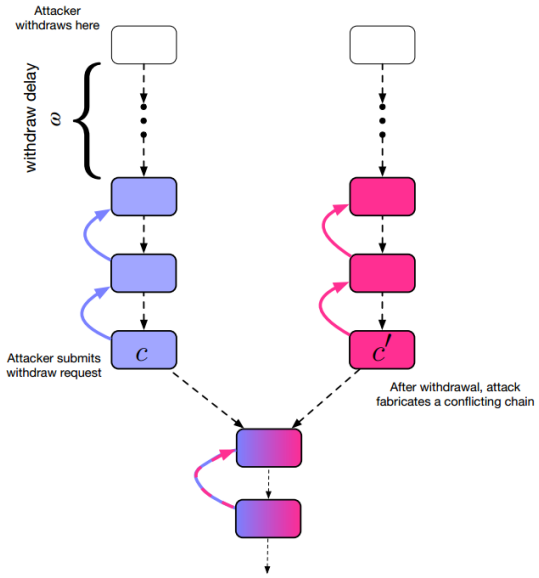


Figure 5: **The long range attack**. As long as a client gains complete knowledge of the justified chain at a regular interval, it will not be susceptible to a long range attack.

简单说，可以由一个使终结区块永远不会被恢复的分支选择规则，以及一个期望，即每个客户端都会“登录”并得到一个以某个正常频率（比如每 1-2 个月一次）完全更新的链的视图，来防御 long-range 攻击。一个决定比其更老区块的“long range revision”分支会简单地被忽略掉，因为所有客户端已经在那个高度看见过一个终结区块并且会拒绝恢复它。

如下，我们对该机制做一个非正式的证明。假设：

• 在两个客户端之间有一个最大通信延迟 δ，所以如果一个客户端在时间 t 收到某个消息，所有其他客户端保证在时间 t + δ 之前也会收到该消息。这意味着我们可以讨论一个在一个区块被网络接收的期间的“时间窗口 time window”[t_{min}, t_{max}]，其宽度 $t_{max} - t_{min}$ 最大为 δ。

• 我们假设所有客户端的本地时钟是完美同步的（任何矛盾都可以被当做通信延迟 δ 的部分来对待）。

• 区块需要有时间戳。如果一个客户端有本地时间 T_L，那么他会拒绝时间戳 $T_B > T_L$ 的区块（即未来的区块），并且他们会拒绝接受 $T_B < T_L - \delta$ 的区块（即很久以前的区块）作为终结区块（但可能仍然会接受作为链的一部分）。

• 如果一个验证者在时间 t（即他们收到的两个投票中更晚的那一个的时间）看见一个消减条件的违反，那么他们会拒绝具有是还未包含该消减证据的链的一部分的大于 t + 2δ 的时间戳的区块。

假设，大量对消减条件的违反会导致两个冲突的终结检查点，c_1 和 c_2。如果两个时间窗口不相交，那么所有的验证者都会与先到的检查点达成一致，并且每个人都会遵守规则不去恢复终结检查点，然后就不会有问题了。

如果两个时间窗口确实相交了，那么我们可以用如下的方法处理这种情况。令 c_1 的时间窗口为[0, δ]，c_2 的时间窗口为[δ - ϵ, 2δ - ϵ]。那么它们的时间戳都至少为 0。到时间 2δ 的时候，保证所有的客户端都看到了消减条件的违反，所以它们拒绝其链还没有包含证据交易的具有时间戳大于 4δ 的区块。因此，只要 w > 4δ，就保证违规的验证者将会在任何客户端接受的所有链上失去他们的押金，其中 w 是“取款延迟”（图 5），即 end-epoch 和验证者实际收到他们的押金的时候之间的延迟。

由于网络延迟，有可能客户端不同意一块给定的消减证据是否被准时提交到指定的链或者被视作接受它太晚。然而，这只是一个活性故障，而不是安全性故障，并且这种可能性不会削弱我们的安全性声明，因为已经知道一个损坏的提议机制（可能被要求用来阻止证据包含（evidence inclusion））可以阻止终结（finality）。

我们也可以通过非正式地声明攻击会是短期的来回避证据包含超时的问题，因为验证者会将一个没有包含消减证据的长时间运行的链视为攻击并转换到另一个由少数不是攻击的一部分的诚实验证者支持的分支（见第 4.2 节），从而停止这次攻击并消减攻击者。

4.2. Castastrophic Crashes

假设大于 1/3 的验证者同时出现崩溃故障——即，由于一个网络分区、计算机故障或者验证者本身就是恶意的等原因，这些验证者不在链接到网络上。直观地，从这些点开始，不会再有绝大多数链路产生，并且因此不会有未来检查点成为终结检查点。

我们可以通过创立一个缓慢耗尽任何没有为检查点投票的验证者的“静止泄漏（inactivity leak）”来恢复，直到最终其押金大小降到足够低以至于正在投票的验证者成为一个绝大多数（supermajority）。最简单的公式就像“在每个 epoch 一个具有押金大小 D 的验证者没能投票成功，他就会损失 D * p（0 < p < 1）”，尽管为了更快地解决灾难性崩溃故障，一个在非终结区块长期增加泄漏率的公式可能就是一个最佳的选择。

这个耗尽的以太（ether）可以在 w 天之后被销毁或者返还给验证者。泄漏的资产应该销毁还是返还以及静止泄漏的精准公式不在这篇文章讨论范围内因为这些是经济激励的问题，而不是拜占庭容错问题。

静止泄漏介绍了终结两个冲突检查点而不需要有验证者被消减的可能性（图 6），验证者只会在两个检查点其中的一个上损失钱。假设验证者被分成两个子集，其中子集 V_A 投票链 A，子集 V_B 投票链 B。在链 A 上，V_B 的押金会泄漏，反之亦然，导致每个子集都要分别在各

自的链上有一个绝大多数（supermajority），允许两个冲突的检查点可以在没有验证者被明确地消减情况下被终结（但是每个子集由于泄漏都会损失两个链中其一上的一大部分押金）。如果发生这种情况，那么每个验证者应该赞成它第一个看到的终结检查点。

从这些不同攻击恢复的精确算法仍然是一个开放的问题。到目前为止，我们假设验证者可以检测到明显的违规行为（比如，不包含证据）并且手动创建“少数软分支”。这种少数分支可以被看作一个以其自己的权限在市场中与大多数链竞争的区块链，并且如果大多数链真地被串通的恶意攻击者操控的话，那么我们可以假设市场可以支持少数分支。

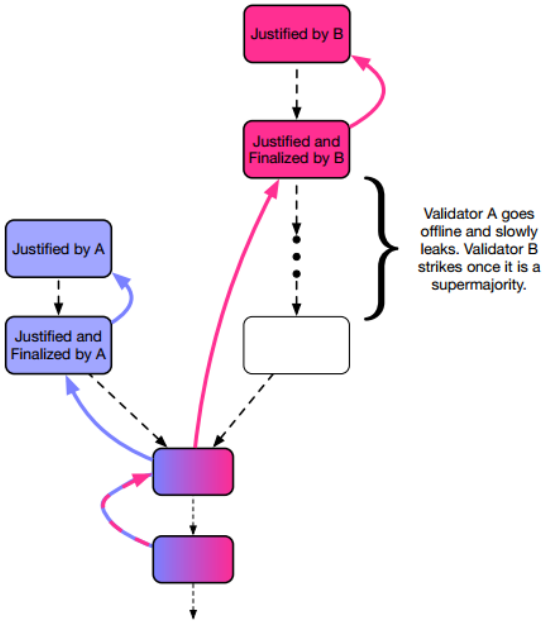


Figure 6: **Inactivity leak**. The checkpoint on the left can be finalized immediately. The checkpoint on the right can be finalized after some time, once offline validator deposits sufficiently deplete.

5. 结论

我们介绍了 Casper，一个新颖的源自拜占庭容错文献的 PoS 系统。Casper 包括：两个消减条件，一个由[11]启发的 correct-by-construction 分支选择规则，以及一个动态的验证者集合。最终我们介绍了 Casper 的扩展（不恢复终结检查点和静止泄漏）来防御两种常见的攻击。

Casper 仍然存在不完善的地方。比如，一个完全妥协的区块提议机制将会组织 Casper 决定新的区块。Casper 是一个基于 PoS 的对于几乎任何 PoW 链的严格安全性的提升。Casper 没有完全解决的问题，尤其是与 51%攻击相关的问题，通过使用用户激活（user-activated）的软分支，仍然可以是正确的。未来的开发将毫无疑问地提升 Casper 的安全性并减少对于用户激活的软分支的需求。

未来工作

目前的 Casper 系统是在一个 PoW 区块提议机制之上建立的。我们希望把区块提议机制转变为 PoS。我们希望即使在验证者集合的权重随着奖励和惩罚变更时，也可以证明责任安全性和似合理的活性。另外一个留给未来工作的问题是一个考虑到 PoS 上的常见攻击的分支选择规则的正式说明。未来的论文将会解释并分析 Casper 及其后续的金融激励。一个特别的与这种对于区块攻击者的自动化策略相关的经济问题正在证明不同客户端之间的不一致程度和由攻击者引发的消耗之间的比例的上限。

致谢

我们感谢与 Jon Choi，Karl Floersch，Ozymandias Haynes，以及 Vlad Zamfir 频繁的讨论。

参考文献

[1] Pass, R. & Shi, E. Fruitchains: A fair blockchain. In *Proceedings of the ACM Symposium on Principles of Distributed Computing*, PODC 2017, 315–324 (ACM, New York, NY, USA, 2017). URL http://doi.acm.org/10.1145/3087801.3087809.

[2] Introducing dfinity crypto techniques (2017). URL https://dfinity.org/pdf-viewer/pdfs/viewer.html?file=../library/intro-dfinity-crypto.pdf.

[3] King, S. & Nadal, S. Ppcoin: Peer-to-peer crypto-currency with proof-of-stake **19** (2012). URL https://decred.org/research/king2012.pdf.

[4] Vasin, P. Blackcoin's proof-of-stake protocol v2 (2014). URL http://blackcoin.co/blackcoin-pos-protocol-v2-whitepaper.pdf.

[5] Bentov, I., Gabizon, A. & Mizrahi, A. Cryptocurrencies without proof of work. In Sion, R. (ed.) *International Conference on Financial Cryptography and Data Security*, 142–157 (Springer, 2016).

[6] Castro, M., Liskov, B. & et. al. Practical byzantine fault tolerance. In Leach, P. J. & Seltzer, M. (eds.) *Proceedings of the Third Symposium on Operating Systems Design and Implementation*, vol. 99, 173–186 (1999).

[7] Kwon, J. Tendermint: Consensus without mining (2014). URL https://tendermint.com/static/docs/tendermint.pdf.

[8] Chen, J. & Micali, S. ALGORAND: the efficient and democratic ledger. *CoRR* **abs/1607.01341** (2016). URL http://arxiv.org/abs/1607.01341.

[9] Nakamoto, S. Bitcoin: A peer-to-peer electronic cash systems (2008). URL https://bitcoin.org/bitcoin.pdf.

[10] Buterin, V. Minimal slashing conditions (2017). URL https://medium.com/@VitalikButerin/minimal-slashing-conditions-20f0b500fc6c.

[11] Sompolinsky, Y. & Zohar, A. Accelerating bitcoin's transaction processing. fast money grows on trees, not chains. **2013** (2013)