Casper the Friendly Finality Gadget

1. **Introduction**

在过去几年中，对基于“股权证明”（PoS）的区块链一致性算法进行了大量研究。在PoS系统中，区块链通过一个过程附加并同意新的块，在该过程中，任何持有系统内币的人都可以参与，并且代理人的影响与持有币（或“赌注”）的数量成比例。这是一种非常有效的替代工作证明（PoW）“采矿”的替代方案，并使区块链能够在没有采矿的高硬件和电力成本的情况下运行。

PoS设计有两个主要的思想流派。第一个基于链的股权证明，模仿工作量证明机制，并以一系列区块为特征，并通过伪随机向利益相关者分配创建新区块的权利来模拟挖掘。

另一流派，基于拜占庭容错（BFT）的股权证明，是基于对BFT共识算法（如PBFT）的一项已有30年历史的研究。 BFT算法通常具有经过验证的数学特性;例如，通常可以在数学上证明只要大于2/3的协议参与者诚实地遵循协议，那么，无论网络延迟如何，算法都无法最终确定冲突块。 Tendermint首次引入了重新利用BFT算法进行股权证明，并具有现代启示。 Casper遵循这种BFT传统，但进行了一些修改。

**1.1我们的工作**

Casper the Friendly Finality Gadget是一个提议机制的顶层 - 一个提出块的机制。 Casper负责最终确定这些块，主要是选择一个代表分类帐规范交易的唯一链。 Casper提供安全性，但是活性取决于所选择的提议机制。 也就是说，如果攻击者完全控制提议机制，Casper可以防止最终确定两个冲突的检查点，但是攻击者可以阻止Casper完成任何未来的检查点。

Casper引进了一些BFT算法不支持的特点：

* 问责。 如果验证者违反规则，我们可以检测到违规并知道哪个验证者违反了规则。 问责制允许我们惩罚恶意验证者，解决困扰基于链的PoS的“nothing at stake”问题。 违反规则的处罚是验证人的全部保证金。 这种最大的惩罚是违反协议的辩护（This maximal penalty is the defense against violating the protocol.）。因为POS的安全性是基于惩罚的多少，而这可以设定为大大超过采矿奖励的收益，所以股权证明提供了比工作证明更严格的安全激励。
* 动态验证人。 我们使用了一种安全的方法让验证人按时间推移进行改变。（We introduce a safe way for the validator set to change over time）
* 防御。我们引入了对抗长程攻击和超过三分之一验证人下线情况下的攻击的防御，以非常弱的权衡同步性假设为代价。（the cost of a very weak tradeoff synchronicity assumption ）
* 覆盖。 Casper的设计作为上层使其更容易实现，作为现有工作量证明机制的升级。

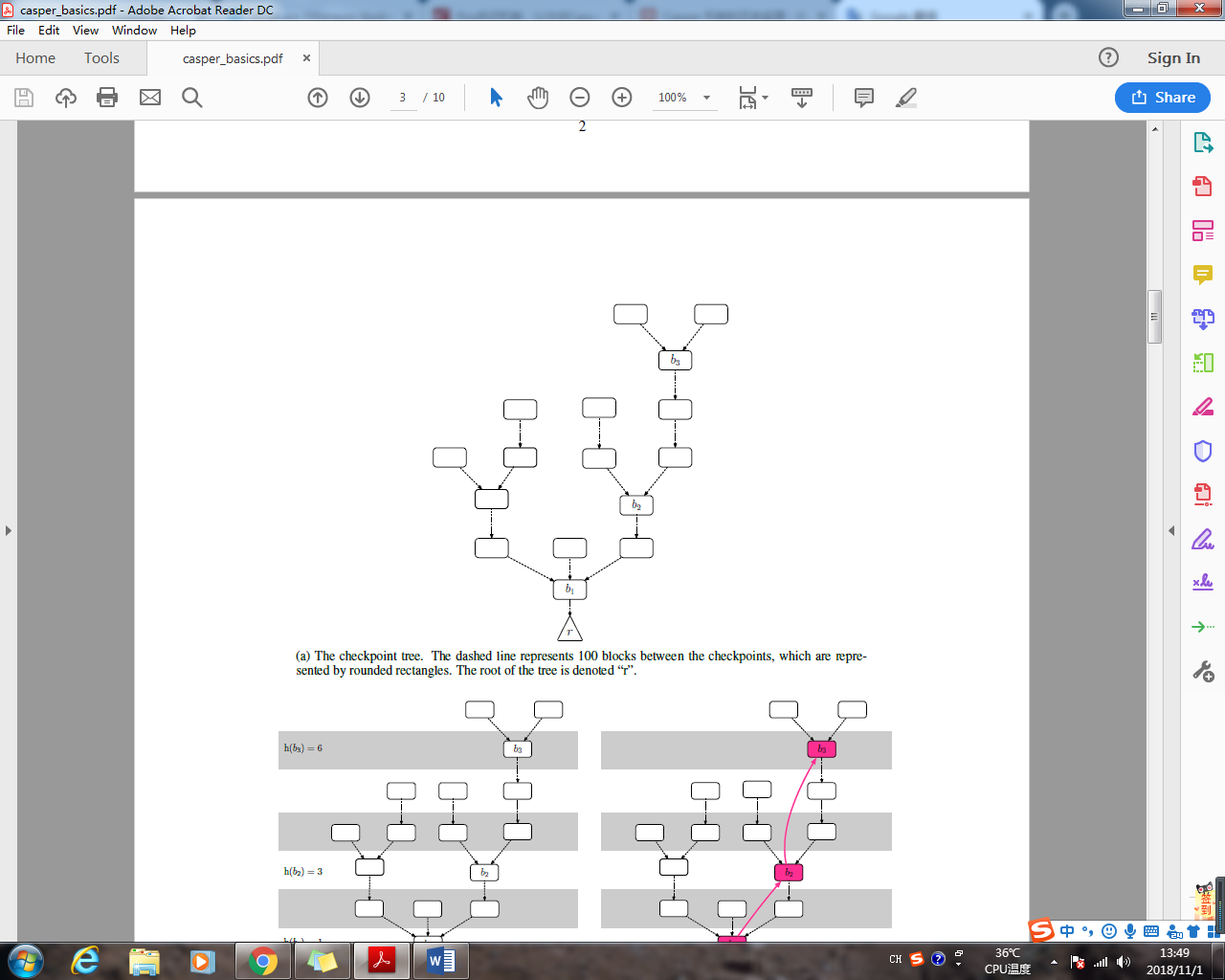
1. **Casper协议**

在以太坊中，提案机制最初将是现有的工作链证明，使Casper的第一个版本成为混合PoW / PoS系统。 在未来版本中，Pow提议机制将被更有效的机制替代。 例如，我们可以想象将块提议转换为某种PoS循环块签名方案。

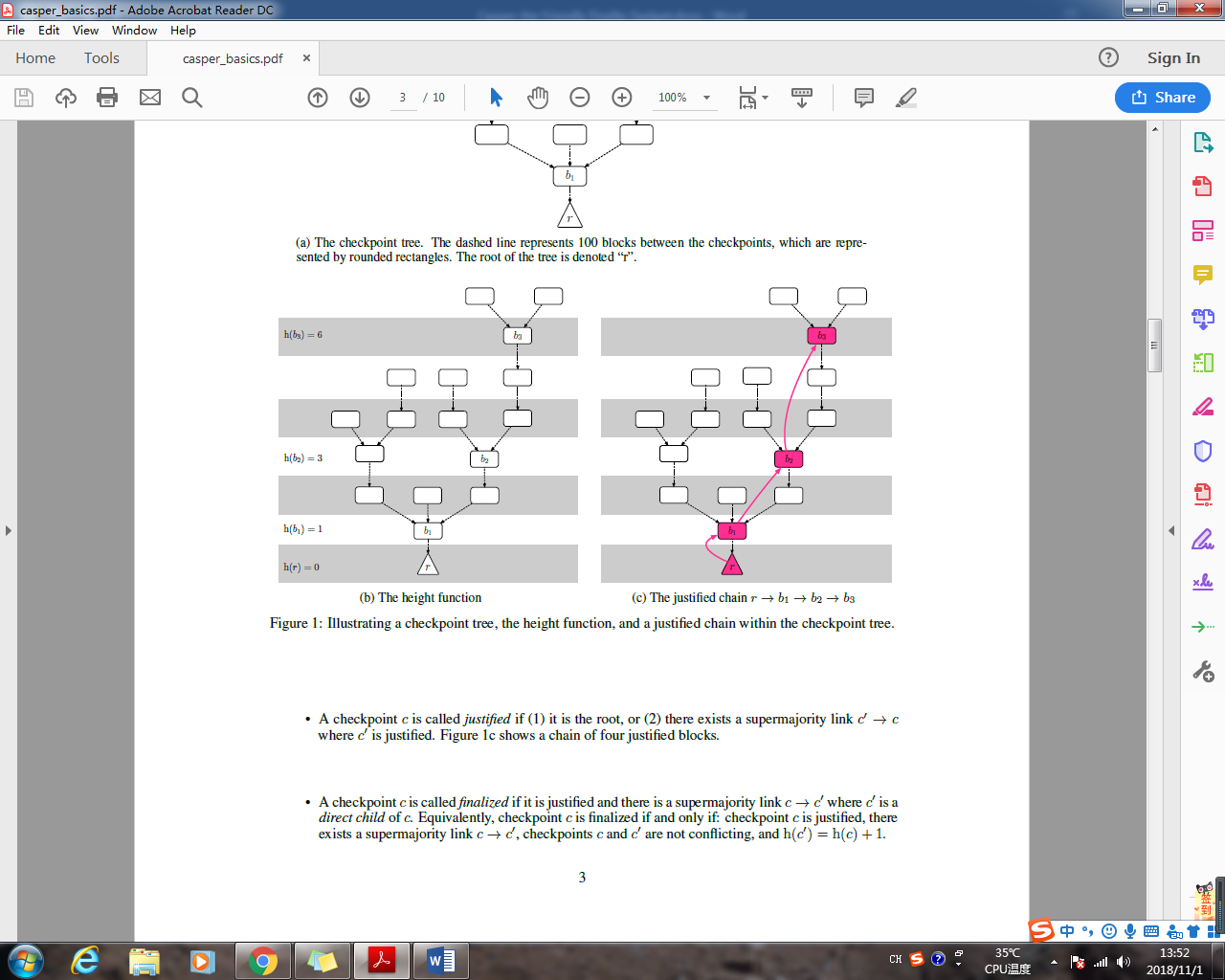
在简单版本的Casper中，我们假设有一组固定的验证人和一个提议机制（例如，熟悉的工作量证明机制），它产生现有块的子块，形成一个不断增长的块树。树的根通常被称为“创世块”。

在这种普通情况下，我们期望提议机制通常会在链表中一个接一个地提出块（即每个“父”块恰好有一个“子”块）。 但是在网络延迟或故意攻击的情况下，提议机制将不可避免地偶尔产生同一父母的多个孩子。 Casper的工作是从每个父母中选择一个孩子，从而从块树中选择一个规范链。

为了提高效率，Casper只考虑形成检查点树的检查点子树，而不是处理完整的块树（图1a）。 创世块是一个检查点，块树（或块号）中高度为100的倍数的每个块也是一个检查点。 块高度为100\*k的块的“检查点高度”是k; 同样地，检查点c的高度是检查点链中的元素数量，从c沿着父链一直延伸到根的高度（图1b）。

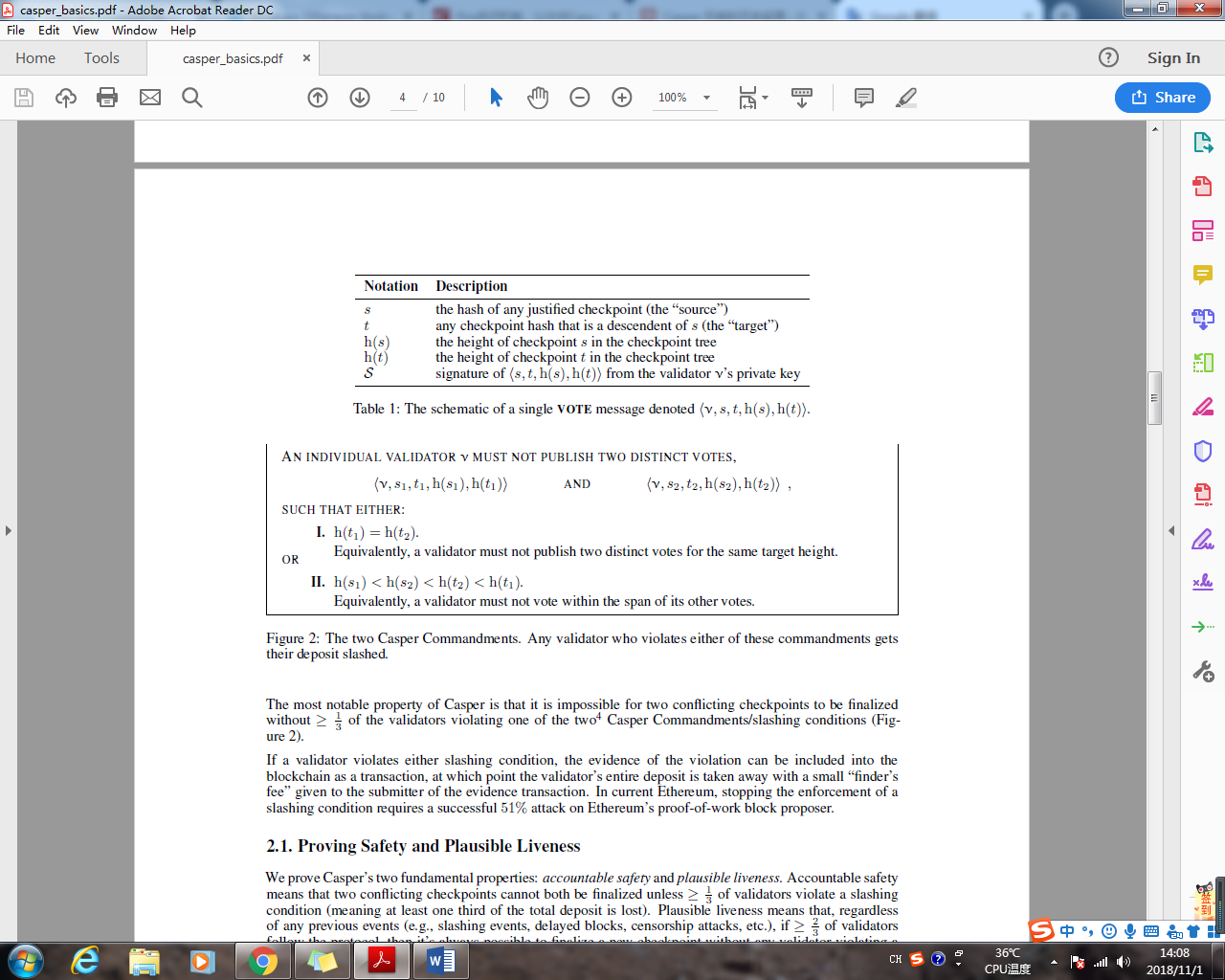


（a）检查点树。 虚线表示检查点之间的100个块，其由圆角矩形表示。 树的根用“r”表示。



每个验证人都有保证金; 当验证者加入时，其保证金是存入的币的数量。 加入后，每个验证人的保证金都会随着奖励和罚款而上升和下降。Pos的安全性来自保证金的大小，而不是验证人的数量，因此对于本文的其余部分，当我们说三分之二的验证者时，我们指的是保证金加权分数; 也就是说，一组验证人总存款大小等于整个验证器组的总存款大小的三分之二。

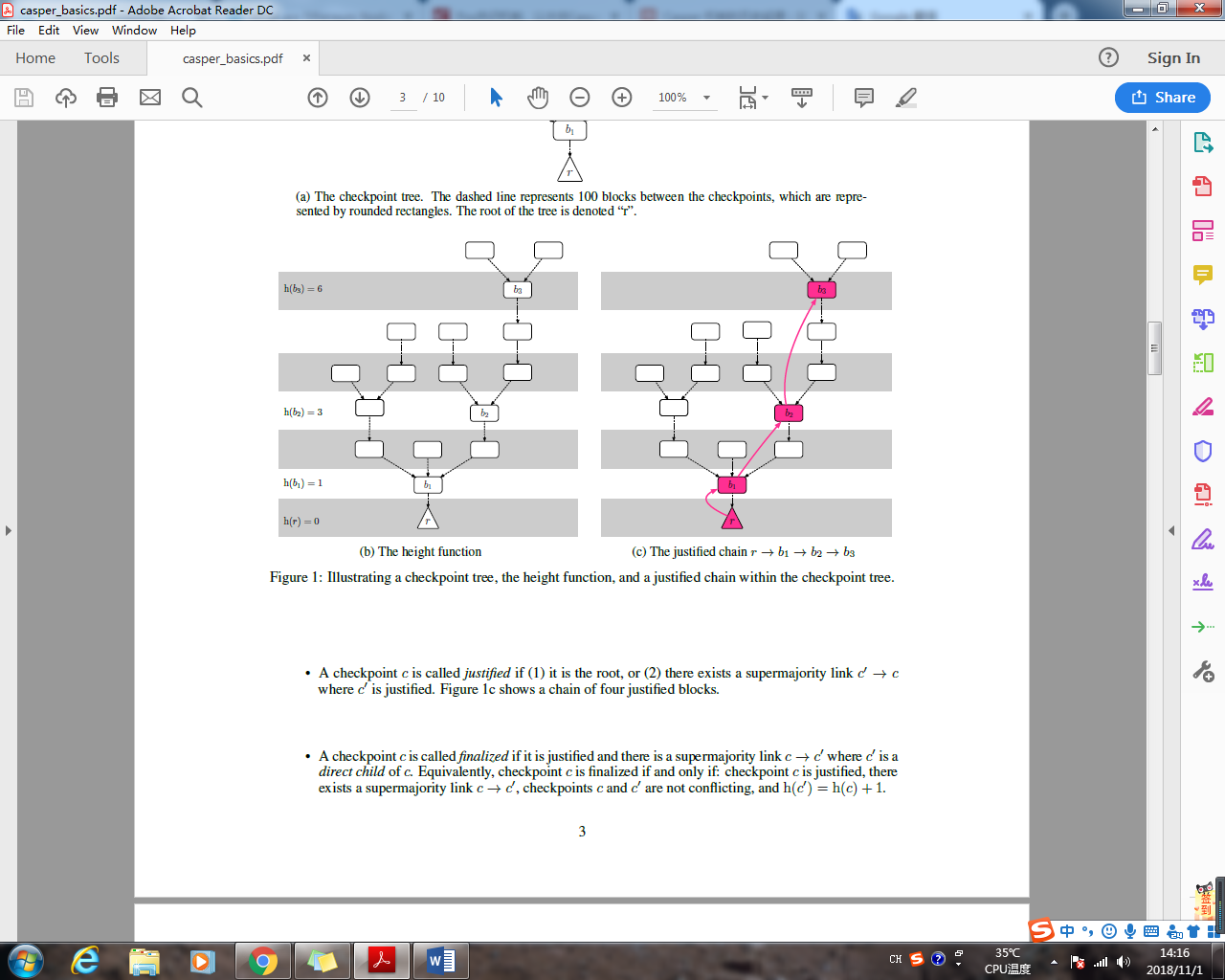
验证人会广播一个投票消息，它包含了四个信息（表1）：1..两个检查点s和t，以及他们的高度h（s）和h（t）。我们要求s是t在检查点树中的祖先，否则这个投票被视为无效。如果验证人的公钥v不在验证人集中，这个投票被视为无效。连同验证人的签名，我们把这个投票写成<v,s,t,h(s),h(t)>.



单个VOTE消息的示意图

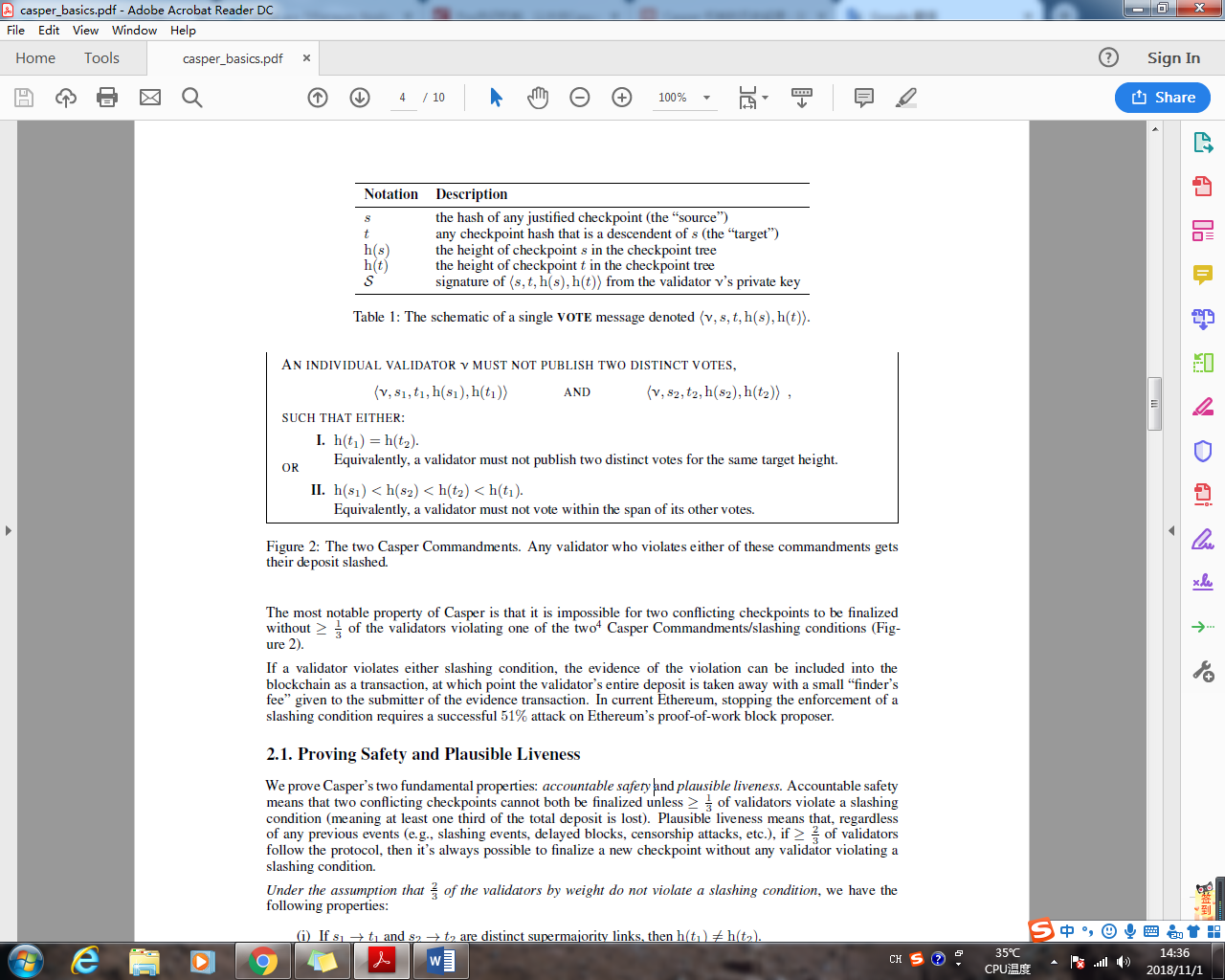
我们定义如下的术语:

* 一个**绝对多数链接（supermajority link）**是一组有序对（a,b），也写作a→b，至少三分之二的验证人发布了从源a到目标b的投票。Supermajority link可以跳过检查点，也就是对h（b）>h（a）+1是可以的。图1c展示了三个supermajority link。



* 两个检查点a和b当且仅当在它们不在同一分支上时才叫做**冲突（conflicting）**，也就是，它们彼此之间不是祖孙关系。
* 一个检查点c只有当1..它是根，2..存在supermajority link c‘→c（c‘是justified）时才被叫做**justified**，图1c展示了有4个justified块的链。’
* 如果一个检查点c是justified并且有一个supermajority link c→c‘（c‘是c的直接孩子）时，c被叫做**finalized**。同样的，检查点c被finalized当且仅当：检查点c是justified，存在一个supermajority link c→c‘，c和c‘不冲突，并且h（c’）=h（c）+1.

Casper最值得注意的特性是：如果没有超过三分之一的验证人违反两个Casper Commandments/slashing 条件（图2）之一，两个冲突的节点被finalized是不可能的。



一个单独的验证人不能发布两个不同的投票，

1..h（t1）=h（t2）。同样的，验证人不能在同一目标高度不能发布不同的投票。

2..h(s1)<h(s2)<h(t2)<h(t1)。同样的，验证人不能在其他投票范围内进行投票。

如果验证者违反了任何削减条件，则违规的证据可以作为交易包含在区块链中，此时验证者的全部存款将被提取给证据交易提交者的小“发现者费”。 在目前的以太坊中，停止执行削减条件需要对以太坊的工作量证明块提议者进行成功的51％攻击。

**2.1证明安全性和合理（可能）的活性**

我们证明了Casper的两个基本属性：accountable（问责？可靠？）安全性和合理的活性。安全意味着两个相互冲突的检查点不能都被finalized，除非超过三分之一的验证人违反了削减条件（意味着至少有三分之一的保证金丢失）。 合理的活性意味着，无论以前发生过什么事件（例如，削减事件，延迟阻止，审查攻击等），如果超过三分之二的验证人遵守协议，那么总是可以在没有任何验证人违反削减条件的情况下finalized新的检查点。

在有三分之二的验证人不会违反削减条件的假设下，我们有如下属性：

1. 如果s1→t1和s2→t2是不同的supermajority link，那么h（t1）！=h（t2）。
2. 如果s1→t1和s2→t2是不同的supermajority link，那么不等式h(s1)<h(s2)<h(t2)<h(t1)不成立

从这两个属性中我们可以立即看到，对于任一高度n：

1. 存在最多一个supermajority link s→t，且h（t）=n。
2. 在高度n上存在最多一个justified检查点.

有了这四个属性，我们转向主要定理。

|  |
| --- |
| 定理1（accountable安全）。 两个相互冲突的检查点a和b不能同时finalized。  证明：    其中j是满足h（bj）>b（am+1）的最小整数  定理2（合理的活性）。Supermajority link总会被添加来产生新的finalized的检查点，使得存在孩子加入到最终确定的链上。  证明： |

**2.2 Casper的分叉选择规则**

Casper比标准PoW设计更复杂。因此，必须调整分叉选择。我们修改后的分叉选择规则应该被所有用户，验证人，甚至底层块提议机制使用。如果用户，验证人或块提议者改为遵循“始终在最长链上建立”的标准PoW分叉选择规则，则存在Casper被“卡住”的病态场景，并且无法finalized（或是justified）在最长链上构建的任何块，除非一些验证者利他主义地牺牲他们的保证金。为了避免这种情况，我们引入一种新颖的，正确的构造，分叉选择规则：遵循包含最高高度的justified的检查点的链。这个分叉选择规则是正确的，因为它遵循合理的活性证明（定理2），它确切地说，总是可以在具有最大高度的justified的检查点之上完成新的检查点。此分叉选择规则将在第3节和第4节中进行调整。

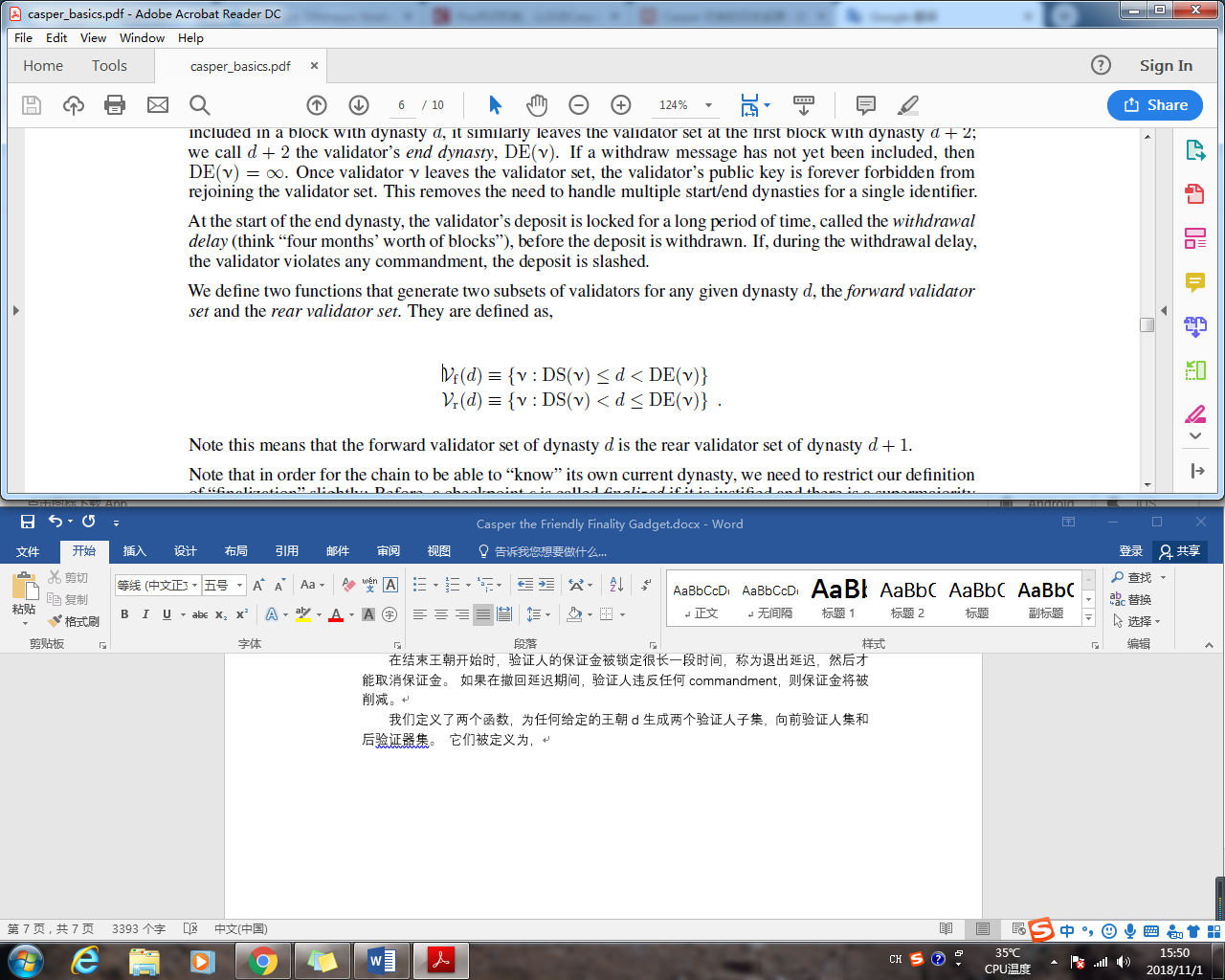
1. **启用动态验证人**

验证人集需要能够更改。 新的验证人必须能够加入，现有的验证人必须能够离开。 为了实现这一点，我们定义了一个块的王朝（dynasty）。 块b的王朝是从根到块b的父节点链中的最终确定的检查点的数量。 当一个可能成为验证人的保证金消息包含在具有王朝d的块中时，验证人v将加入有d+2王朝的第一个块的验证人集.我们称d + 2这个验证人的起始王朝，记为DS（v）。

要离开验证人集，验证器必须发送“撤销”消息。 如果验证者v的撤销消息包含在带有王朝d的块中，它同样会将验证人集留在第一个有王朝d+2的块中; 我们称d + 2为验证人的末日王朝，记为DE（v）。 如果撤销消息还没有被包含（到块中），那么DE（v）= 无穷.一旦验证人v离开验证人集，验证者的公钥永远不会重新加入验证人集。 这消除了处理单个标识符的多个开始/结束王朝的需要。

在结束王朝开始时，验证人的保证金被锁定很长一段时间，称为退出延迟，然后才能取消保证金。 如果在撤回延迟期间，验证人违反任何commandment，则保证金将被削减。

我们定义了两个函数，为任何给定的王朝d生成两个验证人子集，向前验证人集和后验证器集。 它们被定义为，



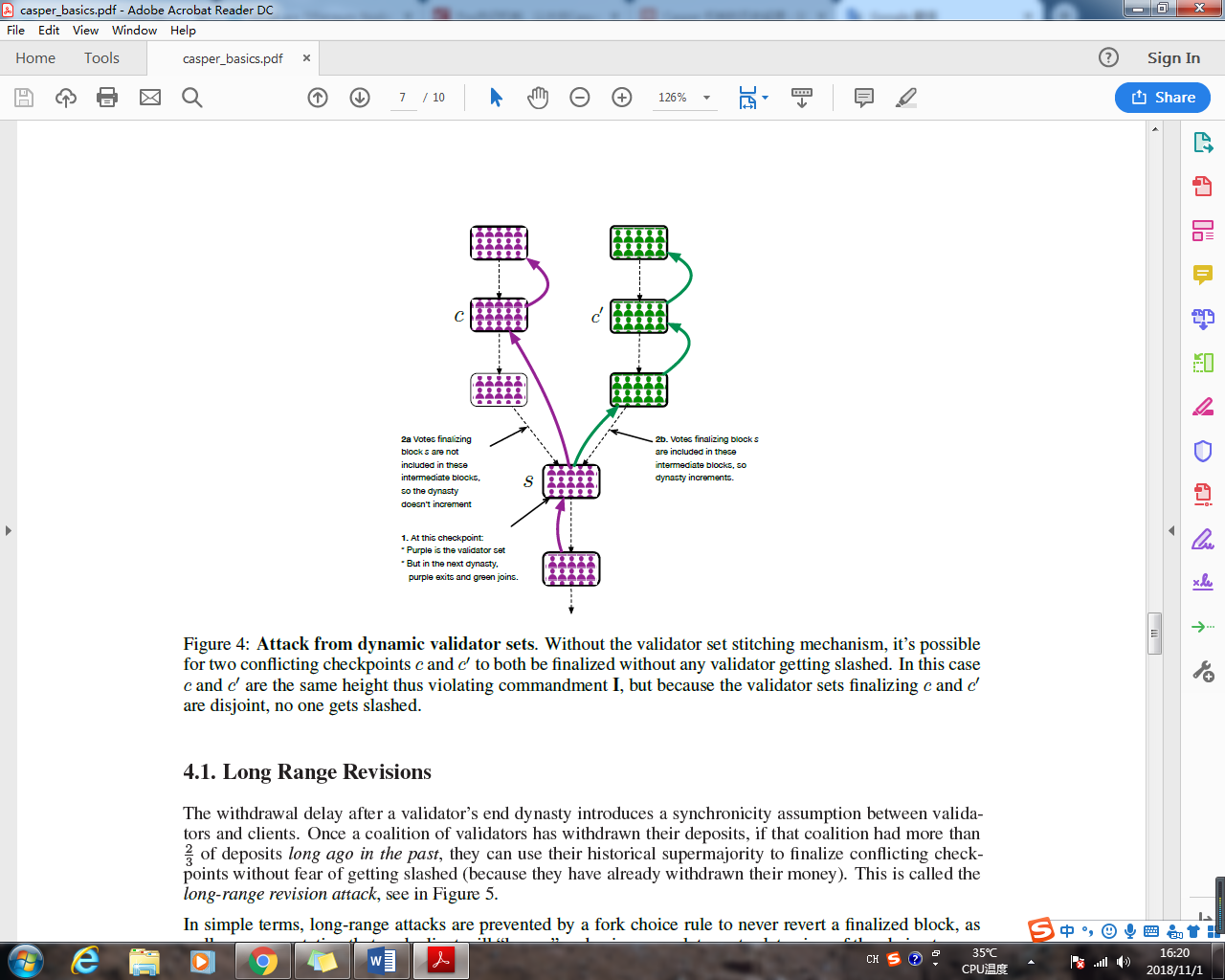
\*Vr表示达到王朝d之前的验证人集，Vf表示王朝d之后的验证人集

注意：这就意味着d王朝的向前验证人集是d+1王朝的后验证人集。

注意：为了使链能够“知道”它自己当前的王朝，我们需要稍微限制我们对“finalization”的定义：之前，如果检查点c是justified的并且存在supermajority link 由c到检查点树中它的任一直接孩子，则称之为finalized。 现在，finalized还有一个附加条件，只有在supermajority link c→c‘，（也就是所有递归justifing了c的surmajority link）都被包含在c‘的孩子之前的区块链。为了支持动态验证人集，我们重定义了supermajority link和finalization，如下：

* Supermajority link：一个有序的检查点对（s，t）（t在王朝d中），有supermajority link，如果至少三分之二的d王朝的向前验证人集发布了投票s→t，而且至少三分之二的d王朝的后验证人集发布了投票s→t。
* Finalization：如果检查点c是justified并且有一个c→c’（c’是c的孩子）的supermajority link我们就叫c是justified。我们增加了一个条件：只有投票supermajority link c→c’（也就是justifing了c的supermajority link）被包含在c’的区块链中而且要在c’的孩子之前，才说c是justified。

前后验证人集通常会大大重叠; 但是如果两个验证人集相当不同，这种“缝合”机制可防止在finalized的检查点的两个孙子具有不同朝代的情况下的安全失败，因为证据包含在一个链中而不包含在另一个链中。 有关此示例，请参见图4。

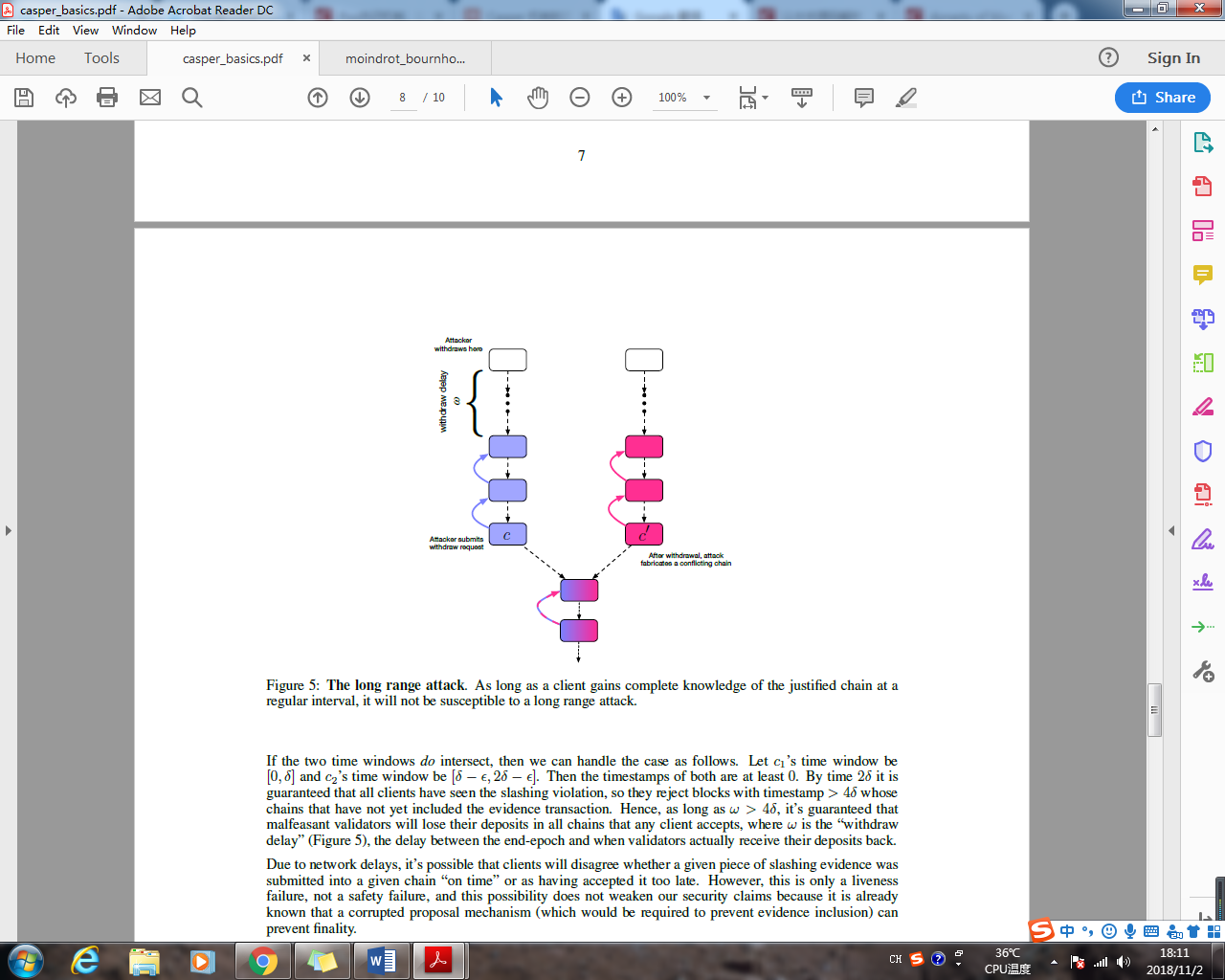


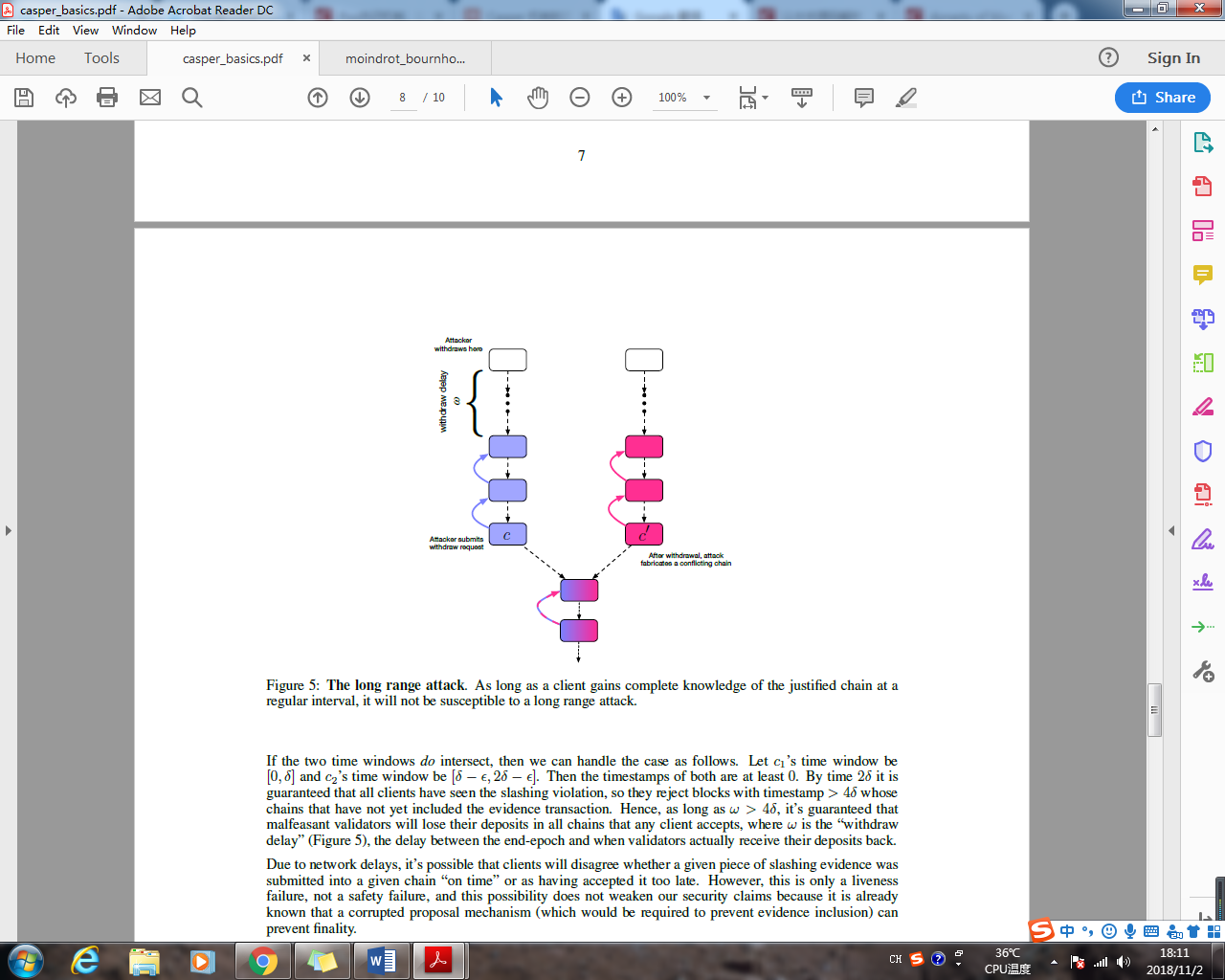
1. **停止攻击**

针对pos有两个著名的攻击：长程攻击和大规模崩溃。我们会依次讨论。

**4.1长程攻击**

验证者结束王朝之后的退出延迟引入了验证者与客户之间的同步性假设。 一旦验证者联盟撤回了他们的保证金，如果该联盟过去很久以前有超过三分之二的保证金，他们可以利用他们的历史绝对多数来finalized相互冲突的检查点而不必担心被削减（因为他们已经撤回了他们的钱）。 这称为长程攻击，见图5。





只要客户定期对合理链有一个完整认识，它就不会遭受长程攻击。

简单来说，分叉选择规则阻止远程攻击永远不会恢复finalized的块，并期望每个客户端将按常规频率（例如，每1-2个月一次）登陆来获得链的完整最新视图。 “长程攻击”产生的分支finalized比这更“older”的块将被忽略，因为所有客户端都已经看到在该高度的最终块，并拒绝恢复它。

如下我们做了这个机制的非正式证明。假设：

* 在两个客户之间存在最大通信延迟δ，所以如果一个客户端在时间t听到一些消息，那么所有其他客户端都保证在时间t +δ时听到它。 这意味着当一个块被网络接收的时候我们可以谈论“时间窗口”（time window）[tmin，tmax]，最大宽度tmax-tmin为δ。
* 我们假设所有客户端都有完全同步的本地时钟（任何差异都可以被视为通信延迟δ的一部分）。
* 块需要有时间戳。 如果客户端具有本地时间TL，那么它将拒绝其时间戳TB满足TB> TL（即将来）的块，并且他们将拒绝接受finalized（但仍可接受作为链的一部分）时间戳TB <TL-δ （即过去太远了）的块。
* 如果验证者在时间t看到一次slashing违规（那是他们听到两个选票中较后一个的时间），那么他们会拒绝时间戳> t + 2δ的块，这些块是尚未包含此slashing证据的链的一部分。

假设一大堆slashing违规导致finalized了两个冲突的检查点c1和c2。 如果两个时间窗口不相交，那么所有验证人都会同意首先出现的那个检查点，并且每个人都遵循该规则，不会恢复最终检查点，那么就没有问题。

如果两个时间窗口相交，那么我们可以按如下方式处理这种情况。 设c1的时间窗为[0，δ]，c2的时间窗口是[δ-ε，2δ-ε]。 然后两者的时间戳至少为0。到时间2δ，保证所有客户端都看到了slashing违规，因此他们拒绝时间戳> 4δ的块，其链尚未包含证据交易。 因此，只要ω> 4δ，保证恶意验证人将在任何客户接受的所有链中丢失保证金，（ω是“撤销延迟”（图5），即结束epoch与验证人实际收到保证金之间的延迟）。

由于网络延迟，可能会发生客户不同意是否将某一特定的slashing证据“按时”提交到给定的链中，或者是否已经接受了它。 然而，这只是一个活性的失败，而不是安全性失败，这种可能性并没有削弱我们的安全要求，因为已经知道一个损坏的提案机制可以防止终结。

我们还可以通过非正式地认为攻击将是短暂的来回避证据包含超时的问题，因为验证者将会认为一个不包含slashing证据的长期运行的链条是攻击，并会切换到由诚实的少数验证者支持的另一个分支。这条链不是攻击的一部分（见4.2节），因此可以阻止攻击并slashing攻击者。

**4.2大规模奔溃**

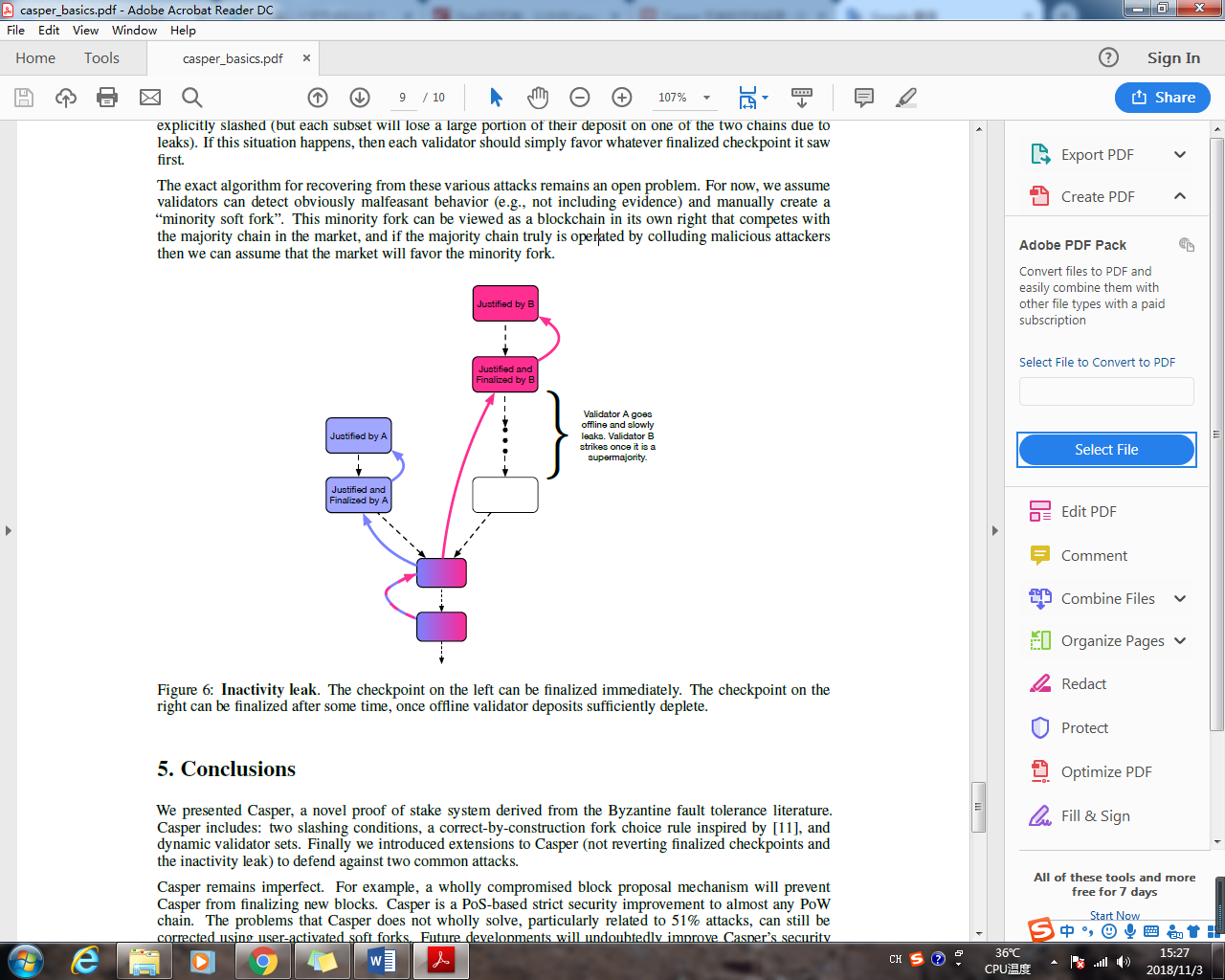
假设大于三分之一的验证人同时崩溃，即由于网络分区，计算机故障或验证人本身是恶意的，它们不再连接到网络。直观地说，从这一点来看，不能创建supermajority link，因此无法finalized未来的检查点。

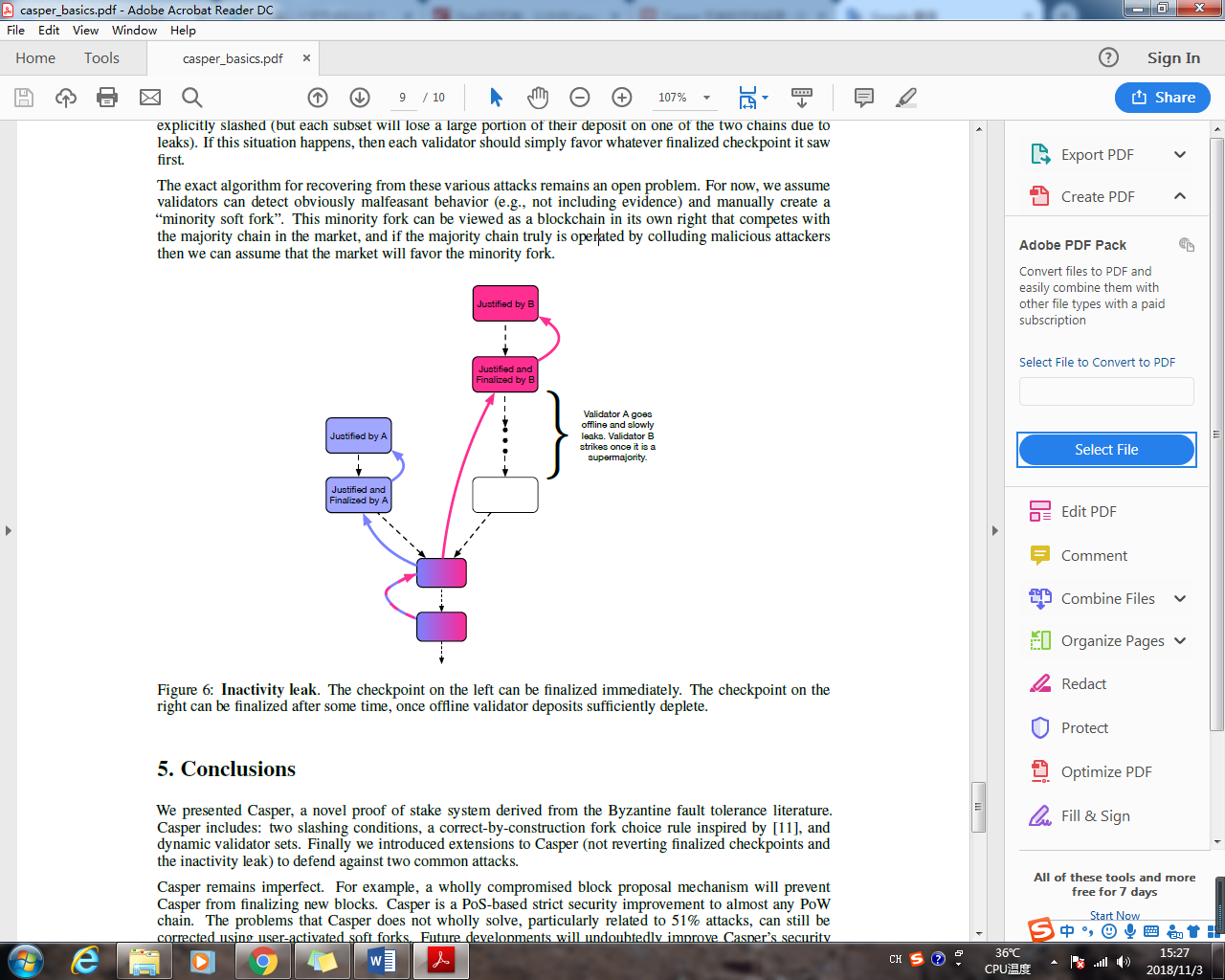
我们可以通过建立“不活动泄漏”（inactivity leak）来缓解这种情况，这种泄漏会慢慢消耗任何不投票给检查点的验证人的保证金，直到最终其保证金规模减少到足以使投票的验证人为绝对多数。 最简单的公式就像“在每个epoch，保证金大小为D的验证人未能投票，它失去D \* p（0 <p <1）”，虽然为了更快地解决灾难性崩溃，在非最终块的long streak的情况下增加泄漏率的公式可能是最佳的。

这种消耗了的以太可在ω天后燃烧或返回验证器。 泄漏资产是否应该被烧毁或返还以及不活动泄漏的确切公式超出了本文的范围，因为这些是经济激励的问题，而不是拜占庭容错。

不活动泄漏引入了两个冲突的检查点被finalized而没有任何验证人被削减的可能性（如图6所示），验证人只在两个检查点中的一个上损失保证金。 假设验证人被分成两个子集，在链A上投票的子集VA和在链B上投票的子集VB。在链A上，VB的存款将泄漏，反之亦然，导致每个子集在其各自的链上具有绝对多数，使得在没有明确削减任何验证人的情况下最终确定两个冲突的检查点（但由于泄漏，每个子集将在两个链中的一个链上丢失大部分保证金）。 如果发生这种情况，那么每个验证人应该只是支持它首先看到的最终检查点。

从各种攻击中恢复的确切算法仍是一个悬而未决的问题。目前，我们假设验证者可以检测到明显的恶意行为（例如，不包括证据）并手动创建“少数软分叉”。这种少数分叉可以被视为区块链，它用自己的权利同市场上多数链竞争，如果多数链真正通过串联恶意攻击者来操作，那么我们可以假设市场将偏向少数派。





1. **结论**

我们介绍了Casper，这是一种来自拜占庭容错的新的pos。 Casper包括：两个削减条件，一个正确构造的分叉选择规则，以及动态验证人集。 最后，我们介绍了Casper的扩展（不还原最终检查点和不活动泄漏）来防御两种常见的攻击。

卡斯珀仍然不完美。 例如，完全受损的块提议机制将阻止Casper finaling新块。 Casper是基于PoS的几乎所有PoW链的严格安全性改进。 Casper没有完全解决的问题，特别是与51％攻击相关的问题，仍然可以使用用户激活的软分叉进行纠正。 未来的发展无疑将提高Casper的安全性并减少对用户激活的软分叉的需求。

**未来的工作。**当前的Casper系统建立在pow块提议机制上。 我们希望将块提案机制转换为pos。 即使验证器组的权重随奖励和处罚而变化，我们也希望证明其可靠的安全性和合理的活性。 未来工作的另一个问题是对叉选择规则的正式规范，考虑到对股权证明的常见攻击。 未来的工作报告将解释和分析casper的财务激励及其后果。 与阻止攻击者的这种自动化策略相关的特定经济问题证明了不同客户端之间的不一致程度与攻击者所产生的成本之间的比率的上限。