适用于采用最长链规则的区块链基于交易费机制的手续费再分配智能合约

Rastislav Budinsk´y布尔诺理工大学信息技术学院
xbudin05@fit.vutbr.cz

Ivan Homoliak布尔诺理工大学信息技术学院
ihomoliak@fit.vutbr.cz

Ivana Stanˇc´ıkov´a布尔诺理工大学信息技术学院
istancikova@fit.vutbr.cz

arXiv:2302.04910v1 [cs.CR] 9 Feb 2023

摘要——**本文回顾了采用最长链分叉选择规则的工作量证明（PoW）区块链在基于交易费的机制下面临的削价攻击。接下来，我们重点讨论挖矿收入波动和挖矿缺口问题——即交易费带来的即时奖励无法覆盖矿工支出的情况。**

为缓解这些问题，我们提出一种解决方案，将已挖出的区块所产生的交易费分成两部分——（1）给予区块矿工的即时奖励；（2）存入一个或多个作为共识协议一部分的手续费再分配智能合约（$\mathcal{FRSC}$s）。与此同时，这些再分配智能合约会在预先设定的时间内，使用入账手续费累积资金中的一定比例奖励区块矿工。这种机制使我们能够实现若干有益于协议激励稳定性和安全性的特性。

通过我们提出的解决方案，严格不执行削价攻击的默认合规矿工比例从当前最先进结果的 66% 降至 30%。

I. 引言

加密货币通过提供区块奖励来激励矿工生成新区块。交易费也构成矿工收入的一部分，促使他们将交易纳入区块。矿工通过优先处理手续费最高的交易来最大化收益。在比特币及其众多克隆链中 [1]，，区块奖励大约每四年减半一次（即每经过 210k 个区块），最终将形成纯粹的基于交易费的机制。

2016年之前，人们普遍认为，矿工收入的主要来源不会影响区块链的安全性。然而，Carlsten 等人 [2] 指出，在基于交易手续费的协议中，呈指数分布的区块到达时间会导致矿工每个区块的收入方差很高，并由此产生影响。作者表明，对于恶意矿工来说，对一个奖励丰厚的区块发动削价攻击（即分叉）是一种有利可图的策略。尽管如此，Daian 等人 [3]表明，即使在包含传统区块奖励的区块链中，这种攻击仍然可行，因为套利机器人之间存在抢先交易竞争，它们愿意大幅提高交易手续费，以获取最大可提取价值带来的利润。

本文聚焦于缓解工作量证明区块链基于交易费的机制中的削价攻击。我们还讨论基于交易费的机制中存在的其他相关问题。具体而言，我们着重于尽量缩小挖矿缺口 [2], [4]，（即交易费带来的即时奖励不足以覆盖矿工支出的情况），并平抑矿工收入的大幅波动。

为缓解这些问题，我们提出一种方案，将已挖出的区块所产生的交易费分为两部分：（1）给予矿工的即时奖励；（2）存入一个或多个手续费再分配智能合约（$\mathcal{FRSC}$s）的资金。同时，这些 $\mathcal{FRSC}$s会在固定时段（即固定数量的区块）内，将累积资金中的一定比例奖励给区块的矿工。这种设置使我们能够实现若干有助于提升协议稳定性和安全性的特性。

贡献：具体而言，我们的贡献如下：1）我们提出了一种方法，通过一个或多个$\mathcal{FRSC}$s对部分交易费进行移动平均，从而规范来自交易费的挖矿奖励。2）我们使用区块中的不同交易费比例（在矿工与 $\mathcal{FRSC}$s之间分配）评估了该方法。我们试验了不同数量和长度的 $\mathcal{FRSC}$s，并证明使用多个不同长度的 $\mathcal{FRSC}$s在缓解我们所要解决的问题方面最具优势；不过，即使只使用一个 $\mathcal{FRSC}$ 也有益处。3）我们证明，采用该方法可以最大限度地缩小挖矿缺口，因为在挖矿轮次开始时的矿工能够从 $\mathcal{FRSC}$s获得奖励，从而稳定其收入。4）我们通过实证证明，采用该方法后，严格不实施削价攻击的默认合规矿工比例门槛可从 66%（原始研究报告的数值 [2]）降至 30%。

© 2023 IEEE。允许将本材料用于个人用途。对于任何其他用途，包括在当前或未来的任何媒体中重印或再版本材料用于广告或宣传目的、创作新的合集、转售或重新分发至服务器或列表，或在其他作品中重复使用本作品的任何受版权保护的组成部分，均必须获得 IEEE 的许可。

II. 问题定义

基于交易手续费的机制方案中出现了一些问题，即使在如今的比特币协议中也能观察到这些问题 [2]。我们选取了三个主要问题，并旨在降低它们对仅依赖交易费的协议造成的影响。具体而言，我们关注以下问题：

1) 削价攻击。 在这种攻击中，恶意矿工试图通过重新挖掘最长链上的顶部区块来获取交易手续费，从而促使其他矿工在其区块之上挖矿 [2]。具体而言，考虑这样一种情况：一名诚实矿工挖出一个包含手续费极高的交易的区块。恶意矿工可以对该区块进行分叉，同时将“慷慨的”交易中的一部分留在内存池中不予挖掘。这些交易会促使其他矿工在攻击者的链上继续挖矿，从而削价攻击原始区块。这种恶意行为可能导致更高的孤块率、系统不可靠以及双重支付。

2) 挖矿缺口。如 [2], 所讨论的，一旦内存池中没有足够的交易手续费来激励矿工挖矿，就会出现挖矿缺口问题。假设某名矿工在前一个区块创建后不久成功挖出一个新区块；由于工作量证明区块链中区块创建时间服从众所周知的指数分布，这种情况是可能发生的。因此，该矿工可能无法获得足够的奖励来覆盖其支出，因为内存池中的大多数交易都已被纳入前一个区块，而新交易可能尚未到达，或者手续费较低。于是，矿工会被激励推迟挖矿，直到内存池中合理地积累了足够多的交易（及其手续费）。Tsabary 和 Eyal 的研究还通过仿真方法分析了挖矿缺口 [4],，并进一步证明，这种挖矿缺口会激励规模更大的挖矿联盟，从而影响去中心化。**3) 随时间变化的交易手续费**。 在基于交易手续费的机制中，交易手续费的任何波动都会直接影响矿工收入。某些时间段内交易手续费的大幅波动，例如在一天或一周的时间跨度内 [5],，可能导致矿工奖励缺乏不理想的可预测性，并间接影响底层协议的安全性。

III. 提出的方法

本节将更详细地介绍我们所提出的解决方案。该方案会将已挖出的区块中以原生加密货币支付的交易费的一定比例，汇集到一个或多个手续费再分配智能合约（即 $\mathcal{FRSC}$s）中。必须向这些合约缴款的区块矿工，同时也会从中获得奖励；所获奖励近似于固定区块滑动窗口内流入交易费的移动平均值。已挖出的区块所产生的交易费比例（即 $\mathbb{C}$）会发送至 $\mathcal{FRSC}$ ，其余部分

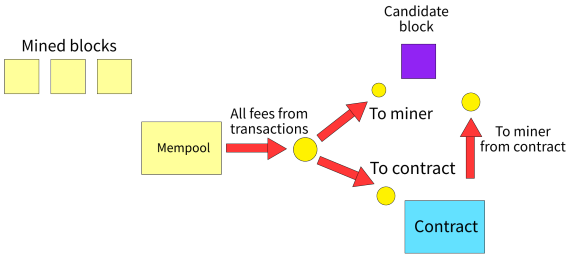


图 1：我们解决方案的概览。

交易费（即 $\mathbb{M}$）则直接分配给矿工，因此 $\mathbb{C}+\mathbb{M}=1$.

$\mathbb{M}$ 的作用是激励矿工优先处理手续费较高的交易，而 $\mathbb{C}$的作用是缓解削价攻击和挖矿缺口问题。

概览： 我们在图 1 中展示了所提方法的概览，该方法包含以下步骤：

1）使用 $\mathcal{FRSC}$，矿工计算下一个区块 B 的奖励（即 $nextClaim(\mathcal{FRSC})$——见公式 4），该奖励将由 $\mathcal{FRSC}$ 支付给该区块的矿工。2）矿工从其内存池中选择手续费最高的一组交易来挖掘区块 B 。3）已挖出的区块 B 直接将一定比例的交易费（即 $B.fees * \mathbb{M}$）奖励给矿工，并将剩余部分（即 $B.fees * \mathbb{C}$）奖励给 $\mathcal{FRSC}$。4）矿工从 $\mathcal{FRSC}$获得 $nextClaim$ 。

我们的方法嵌入在共识协议中，因此共识节点必须遵守该方法，以确保其区块有效。该方法可以通过区块链平台的标准智能合约实现，也可以在共识协议的原生代码中实现。

A. 高交易费交易的优先处理

在交易费恒定的环境中，无论是否采用我们的方法，矿工都会获得相同的金额。然而，在公有区块链中（尤其是在基于交易费的制度下），存在一种机制，用于确保在处理交易时优先处理交易费较高的交易，这可能导致矿工奖励的波动。在我们的方法中，我们保留了交易优先级，因为我们直接将部分交易费归属于矿工（即， $\mathbb{M}$）。

B. 手续费再分配智能合约

我们将手续费再分配智能合约定义为

$$\mathcal{FRSC}=(\nu,\lambda,\rho), \tag{1}$$

其中， ν 是合约中累积的代币数量，λ 表示 $\mathcal{FRSC}$的滑动窗口大小，即对 ν有贡献的前序区块数量，而 ρ 是用于定义流入

交易费在多个合约之间重新分配比例的参数，而所有 $\mathcal{FRSC}$ 中的 ρ之和必须等于 1：

$$\sum_{x \in \mathcal{FRSC}s} x.\rho = 1. \quad (2)$$

与单个 $\mathcal{FRSC}$ 相比，多个 $\mathcal{FRSC}$s 能在交易费波动较大或环境不可预测时，更好地调整对矿工的补偿（我们将在第 V-C 节中展示这一点）。

我们将区块链高度为H 时 $\mathcal{FRSC}$s 的状态记为 $\mathcal{FRSC}_{[H]}$。然后，我们按如下方式确定高度为 $H+1$ 的下一个区块的矿工从$\mathcal{FRSC}_{[H]} \in \mathcal{FRSC}s_{[H]}$ 获得的奖励：

$$\partial Claim_{[H+1]}^{\mathcal{FRSC}_{[H]}} = \frac{\mathcal{FRSC}_{[H]}.\nu}{\mathcal{FRSC}_{[H]}.\lambda}, \quad (3)$$

而从所有 $\mathcal{FRSC}$s 获得的奖励为

$$nextClaim_{[H+1]} = \sum_{\mathcal{X}_{[H]} \in \mathcal{FRSC}s_{[H]}} \partial Claim_{[H+1]}^{\mathcal{X}_{[H]}}. \quad (4)$$

然后，挖出区块$B_{[H+1]}$ 且该区块包含所有交易费 $B_{[H+1]}.fees$ 的矿工，其总奖励为

$$rewardT_{[H+1]} = nextClaim_{[H+1]} + \mathbb{M} * B_{[H+1]}.fees. \quad (5)$$

高度为 $H+1$ 时合约的新状态为

$$\mathcal{FRSC}s_{[H+1]} = \{\mathcal{X}_{[H+1]}(\nu, \lambda, \rho) \mid \quad (6)$$

$$\lambda = \mathcal{X}_{[H]}.\lambda, \quad (7)$$
$$\rho = \mathcal{X}_{[H]}.\rho, \quad (8)$$
$$\nu = \mathcal{X}_{[H]}.\nu - \partial Claim_{[H+1]} + deposit * \rho, \quad (9)$$
$$deposit = B_{[H+1]}.fees * \mathbb{C}\}, \quad (10)$$

其中，$deposit$ 表示区块 $B_{[H+1]}$ 中所有交易费中存入所有 $\mathcal{FRSC}$s 的比例 $\mathbb{C}$，各合约的存入比例遵循公式 2。

C. 示例

我们以比特币 [6] 为例来说明我们的方法。假设当前区块链高度为 H，并且我们仅使用一个 $\mathcal{FRSC}$，其参数如下：

$$\mathcal{FRSC}_{[H]} = (2016, 2016, 1).$$

我们设定 $\mathbb{M} = 0.4$ 和 $\mathbb{C} = 0.6$，这意味着矿工直接获得 $B_{[H+1]}.fees$ 的 40%，而 $\mathcal{FRSC}$ 获得剩余的 60%。

接下来，我们计算高度为 $H+1$ 的区块的矿工从 $\mathcal{FRSC}$ 获得的奖励：

$$\partial Claim_{[H+1]} = \frac{\mathcal{FRSC}_{[H]}.\nu}{\mathcal{FRSC}_{[H]}.\lambda} = \frac{2016}{2016} = 1 \text{ BTC},$$

从而导致

$$nextClaim_{[H+1]} = \partial Claim_{[H+1]} = 1 \text{ BTC}.$$

此外，我们假设高度为 $H+1$ 的区块中交易产生的总奖励为 $B_{[H+1]}.fees = 2$ 比特币。因此，该区块 $B_{[H+1]}$ 的矿工获得的总奖励为

$$\begin{aligned} rewardT_{[H+1]} &= nextClaim_{[H+1]} + \mathbb{M} * B_{[H+1]}.fees \\ &= 1 + 0.4 * 2 \\ &= 1.8 \text{ BTC}, \end{aligned}$$

以及从 $B_{[H+1]}$ 到$\mathcal{FRSC}$ 的交易费贡献：

$$deposit = B_{[H+1]}.fees * \mathbb{C} = 1.2 \text{ BTC}.$$

因此，$\mathcal{FRSC}$ 中 ν 的值在高度 H + 1处更新如下：

$$\begin{aligned} v_{[H+1]} &= \mathcal{FRSC}_{[H]}.\nu - nextClaim_{[H+1]} + deposit \\ &= 2016 - 1 + 1.2 \text{ BTC} \\ &= 2016.2 \text{ BTC}. \end{aligned}$$

交易费制度下的传统方式： 在传统的交易费制度中，$rewardT_{[H+1]}$ 将等于所有交易手续费之和 $B_{[H+1]}.fees$（即 2 BTC）；因此，使用 $\mathbb{M} = 1$。在我们的方法中，只有在以下条件成立时，$rewardT_{[H+1]}$才等于区块 $B_{[H+1]}$中所有交易手续费之和：

$$B_{[H+1]}.fees = \frac{nextClaim_{[H+1]}}{\mathbb{C}}. \quad (11)$$

在我们的示例中，如果交易包含 1.66 BTC 的手续费，矿工可以挖出区块 $B_{[H+1]}$，同时获得与该区块中所有交易手续费之和相同的总奖励：

$$B_{[H+1]}.fees = \frac{1}{0.6} = 1.66 \text{ BTC}.$$

D. $\mathcal{FRSC}$合约的初始设置

为确保公平的起点，我们提出通过创世值来初始化我们方法中的 $\mathcal{FRSC}$。以下公式计算每个 $\mathcal{FRSC}$ 的创世值，并初始化 $\mathcal{FRSC}s_{[0]}$的初始状态：

$$\{\mathcal{FRSC}_{[0]}^{x}(\nu, \lambda, \rho) \mid \nu = \overline{fees} * \mathbb{C} * \rho * \lambda\}, \quad (12)$$

其中，$\overline{fees}$ 是预期的平均入账手续费。

IV. 实现

我们的实现基于比特币挖矿模拟器 [7]，该模拟器于 [2], 推出，之后根据我们的需求进行了调整。

1) 模拟器中的更改： 我们创建了一个配置文件，用于模拟传入交易的自定义场景，以替代原始设计中的 [2]手续费。我们新增了一个选项，可将模拟切换到完整内存池模式，从而限定一个区块内可获得的总手续费（以及相应的交易总数）——这主要涉及挖矿耗时超过平均区块挖矿时间的区块。[1]

[1]请注意，原始模拟器 [2] 假设区块中的交易数量（因此总手续费）仅受挖出区块所需时间的限制，这一点也在 [8]中受到批评。

接下来，我们将多个参数移至模拟器的参数中，旨在消除频繁重新编译程序的需要，从而简化使用模拟器运行各种实验的过程。最后，我们将基于 $\mathcal{FRCS}$的解决方案集成到了模拟器中。 $\mathcal{FRSC}$由相应的配置文件启动。我们修改后的模拟器源代码可通过匿名链接获取：https://www.dropbox.com/s/gwlbedq47csthqp/sources.zip?dl=0。

V. 评估

我们在一个长期场景中评估了我们的$\mathcal{FRCS}$概念验证实现。该场景由我们设计，旨在展示内存池中的交易费总额随时间演变时发生的显著变化。大多数实验的结果图都描绘了这一场景，其中以“内存池中的手续费”曲线表示——参见第 V-A 节和第 V-B 节。

我们通过实验采用了不同的参数，并研究这些参数如何影响矿工从$\mathcal{FRSC}$获得的总奖励，与不采用我们方案的基线进行比较。主要考察了 $\mathbb{C}$ 的设置，以及$\mathcal{FRSC}$的不同长度 λ 。请注意，我们采用的每区块交易费为 50 比特币，与最初提出削价攻击的论文 [2]中相同。此外，除最后一项实验（即第 V-D 节）外，我们在所有实验中均启用了完整内存池选项，以确保条件更加贴近现实。

A. 实验I

1) 方法：本实验旨在研究采用我们的方法时矿工获得的奖励数量，并与基线进行比较（即，全部奖励均基于所有交易费计算）。在本实验中，我们研究了 $\mathbb{C}$ 如何影响矿工的总奖励，以及滑动窗口的 λ 如何对奖励取平均值。具体而言，我们创建了两个具有不同 λ 的独立 $\mathcal{FRSC}$：一个设为 2016 年（即 $\mathcal{FRCS}^1$），另一个设为 5600（即$\mathcal{FRCS}^2$）。我们使用 $\mathbb{C} \in \{0.5,\ 0.7,\ 0.9\}$的三个不同取值对这两个 $\mathcal{FRCS}$进行了模拟。

2) 结果：本次实验的结果如图 2 所示。在所有实验运行中，我们都可以观察到， $\mathcal{FRSC}^2$ 的适应速度比 $\mathcal{FRSC}^1$ 慢，因此支付给矿工的总奖励平均化程度更高。尽管这是我们所希望的，但我们也能看到， $\mathcal{FRSC}^1$能更快地适应不断变化的环境，这正是矿工对手续费上涨的预期。

B. 实验 II

1) 方法：在本实验中，我们研究了多个 $\mathcal{FRSC}$ 如何处理与之前相同的情景，即改变 $\mathbb{C}$。具体而言，我们研究了各个 $\mathcal{FRSC}$ 如何通过各自的 $\partial Claim^{\mathcal{FRSC}_{[H]}}_{[H+1]}$ 为 $nextClaim_{[H+1]}$ 做出贡献。这次我们只改变参数 $\mathbb{C} \in \{0.5,\ 0.7,\ 0.9\}$，并考虑了四种 $\mathcal{FRSC}$：

$$\mathcal{FRSC}s = \{$$
$$\mathcal{FRSC}^1(_, 1008, 0.07), \mathcal{FRSC}^2(_, 2016, 0.14),$$
$$\mathcal{FRSC}^3(_, 4032, 0.28), \mathcal{FRSC}^4(_, 8064, 0.51)\},$$

其中，它们的长度 λ 设为连续的二次幂（以观察在更长区间内进行更密集平均化所产生的差异），而它们的再分配比率 ρ 则设为最大限度地发挥较长 $\mathcal{FRSC}$ 进行平均化的潜力（详见下文第 V-B3 节）。

2) 结果：本次实验的结果如图 3 所示。我们可以观察到，较短的 $\mathcal{FRSC}$迅速适应了新的变化，而较长的 $\mathcal{FRSC}$ 则为矿工带来了更稳定的收入。从这个意义上说，我们可以看到，$\partial Claim^4$ 在整个场景中都保持稳定，而例如$\partial Claim^1$ 则波动更大。

由于在我们所有实验中（第 V-C 节除外），内存池中的手续费变化场景都相同，因此我们可以将 $\mathcal{FRSC}$ 与第 V-A 节中的 λ = 5600 进行比较，并与当前涉及四个 $\mathcal{FRSC}$的设置进行对比——两者存在一些相似之处。这让我们有了用单个$\mathcal{FRSC}$替代多个$\mathcal{FRSC}$的思路，我们在第 V-C 节中对此进行了进一步研究。

3) 不同 ρs：在图 4 中，我们研究了同一组四个合约中 ρ 的不同取值及其对 $\partial Claim$s 的影响。结果表明，ρ 的取值应与多个 $\mathcal{FRSC}$s 的 λ 相关联，以最大限度地发挥较长 $\mathcal{FRSC}$s 进行平均化的潜力。

C. 实验 III

*1)*方法：在本实验中，我们研究了是否可以使用单一的 $\mathcal{FRSC}$ 设置替代多个 $\mathcal{FRSC}$s，同时保持对 $nextClaim$ 的相同影响。为了量化这两种情况之间的差异，我们引入了一种新的 $\mathcal{FRSC}s$ 度量指标，称为有效 $\bar{\lambda}$，其计算方法如下：

$$\text{effective_}\bar{\lambda}(\mathcal{FRSC}s) = \sum_{x \in \mathcal{FRSC}s} x.\rho * x.\lambda. \tag{13}$$

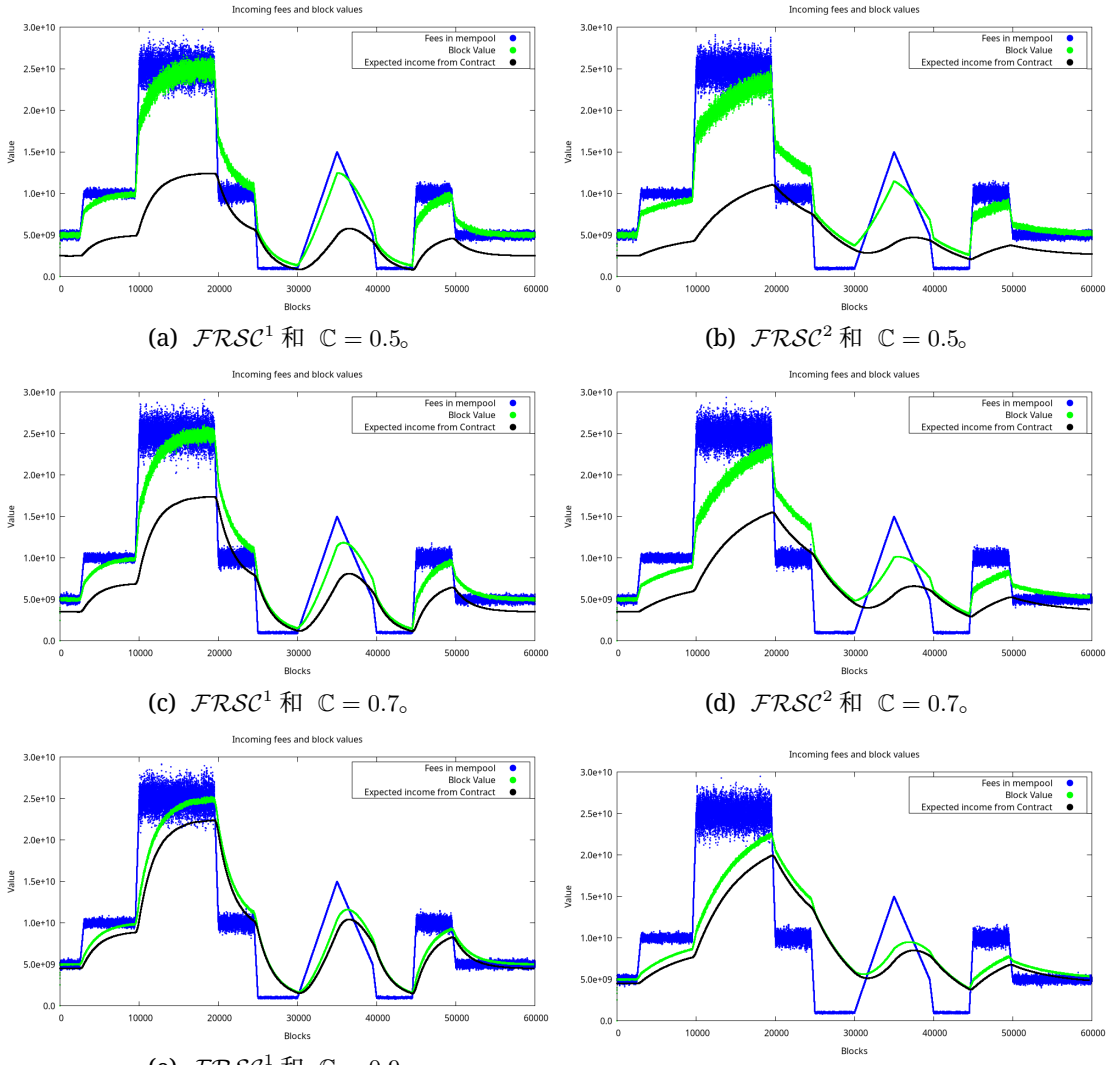
例如，我们希望比较单个 $\mathcal{FRSC}$ 与 4 $\mathcal{FRSC}$s，这两种配置具有相同的有效 $\bar{\lambda}$。这两种情况的配置如下：(1) $\mathcal{FRSC}$(_, 5292, 1) 和 (2)

$$\mathcal{FRSC}s = \{$$
$$\mathcal{FRSC}^1(_, 1008, 0.07), \mathcal{FRSC}^2(_, 2016, 0.19),$$
$$\mathcal{FRSC}^3(_, 4032, 0.28), \mathcal{FRSC}^4(_, 8064, 0.46)\}.$$

我们可以轻松验证， 4 $\mathcal{FRSC}$s 的有效 $\bar{\lambda}$ 与单个 $\mathcal{FRSC}$ 的情况相同，具体可用公式 13 计算： $0.07 * 1008 + 0.19 * 2016 + 0.28 * 4032 + 0.46 * 8064 = 5292$。

我们使用一种自定义费用演化场景进行了这项实验，该场景主要涉及内存池中线性递增/递减费用（见图 5a），并将两种配置的 $\mathbb{C}$ 均设为 0.7。选择这种新的内存池费用演化场景，是为了纳入费用的极端变化，从而凸显可能存在的差异。

2) 结果：图 5b 展示了设置为4 $\mathcal{FRSC}$s 和设置为 1 $\mathcal{FRSC}$时， $nextClaim$ 奖励的相对百分比差异。显然，与单个 $\mathcal{FRSC}$ 相比，设置为 4$\mathcal{FRSC}$s 能带来更高的奖励



(a) $\mathcal{FRSC}^1$ 和 $\mathbb{C} = 0.5$。 (b) $\mathcal{FRSC}^2$ 和 $\mathbb{C} = 0.5$。

(c) $\mathcal{FRSC}^1$ 和 $\mathbb{C} = 0.7$。 (d) $\mathcal{FRSC}^2$ 和 $\mathbb{C} = 0.7$。

(e) $\mathcal{FRSC}^1$ 和 $\mathbb{C} = 0.9$。 (f) $\mathcal{FRSC}^2$ 和 $\mathbb{C} = 0.9$。

图2：实验I，研究单个 $\mathcal{FRSC}$ 的不同 $\mathbb{C}s$ 和 λs，其中 $\mathcal{FRSC}^1.\lambda = 2016$ 和 $\mathcal{FRSC}^2.\lambda = 5600$。内存池中的手续费 表示已挖出的区块中手续费的总价值（即代表基线）。区块价值 是矿工在区块 B 中获得的奖励，即他直接获得的手续费总额（即 $\mathbb{M} * B.fees$）与他从 $\mathcal{FRSC}$ 获得的奖励（即 $nextClaim_{[H]}$）之和。合约的预期收入 表示矿工从 $\mathcal{FRSC}$ 获得的奖励（即 $nextClaim_{[H]}$）。

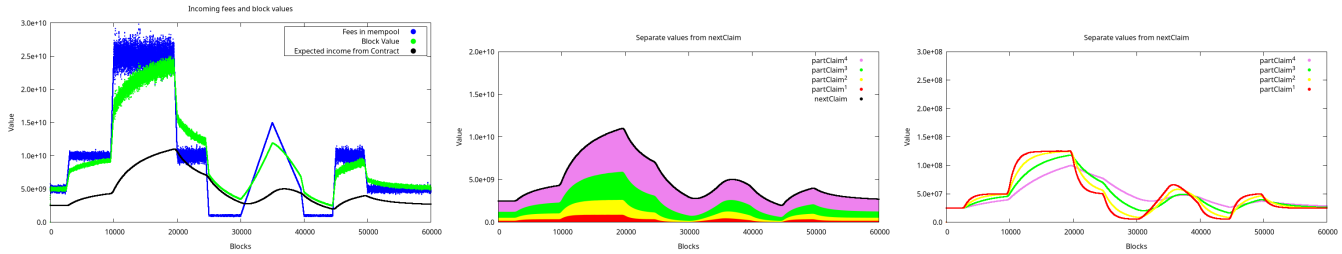
在内存池手续费极低时提供补偿，但在内存池手续费较高时提供的奖励较少。因此，我们得出结论，无法用单个 $\mathcal{FRSC}$ 替代由多个 $\mathcal{FRSC}$ 组成的配置。

D. 实验*IV*

我们重点复现了 [2] 第 5.5 节中的实验。我们旨在寻找一个最小的默认合规矿工占比，使得削价攻击不再有利可图。默认合规矿工是指遵循共识协议规则的诚实矿工，例如在最长链之上继续构建区块。

我们执行了多次模拟，每次模拟均包含多场博弈（即 [2] 中的 30 万场），并设置了不同比例的默认合规矿工。我们将其余矿工平均创建了学习型矿工，他们根据先前的博弈运行结果进行学习，并以一定概率切换到以下挖矿策略中最优的一种。

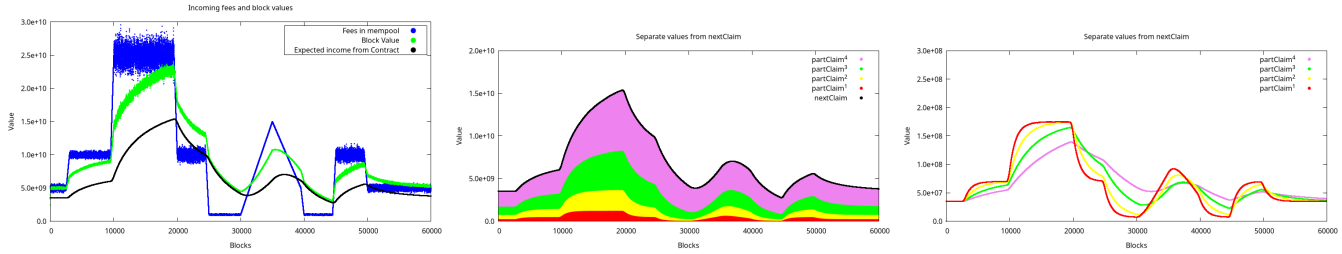
- 小幅合规：该矿工的行为与默认合规矿工相同，只有一点不同：当看到两条链时，他不会在最早的区块上挖矿，而是选择利润最高的区块。因此，该矿工并非（直接）发动攻击的矿工。
- LAZYFORK：该矿工会比较两个选项的利润：(1) 在最长链区块上挖矿，或 (2) 对该区块发动削价攻击。无论选择哪种方式，他都会将一半的内存池费用留给后续矿工，从而阻止另一名 LAZYFORK 矿工对他发动削价攻击。
- FUNCTION-FORK() 矿工的行为可以用函数 f(.) 参数化，该函数表示其削价攻击的程度。输出数值越高，程度越低



(a) 包含 4 $\mathcal{FRSC}$s 的场景，$\mathbb{C} = 0.5$。

(b) $\partial Claim$s 和 $nextClaim$，$\mathbb{C} = 0.5$。

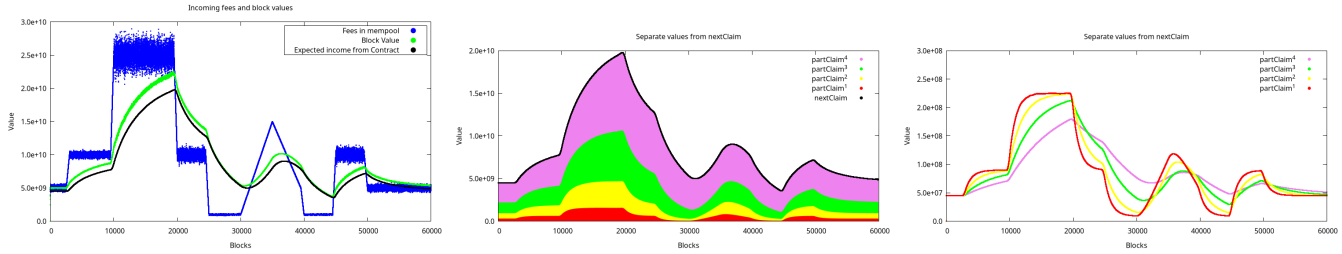
(c) 经 ρ 归一化的 $\partial Claim$s，$\mathbb{C} = 0.5$。



（d）包含 4 $\mathcal{FRSC}$s 的情形，$\mathbb{C} = 0.7$。

（e） $\partial Claim$s 和 $nextClaim$，$\mathbb{C} = 0.7$。

（f）以 ρ 归一化的 $\partial Claim$s，$\mathbb{C} = 0.7$。

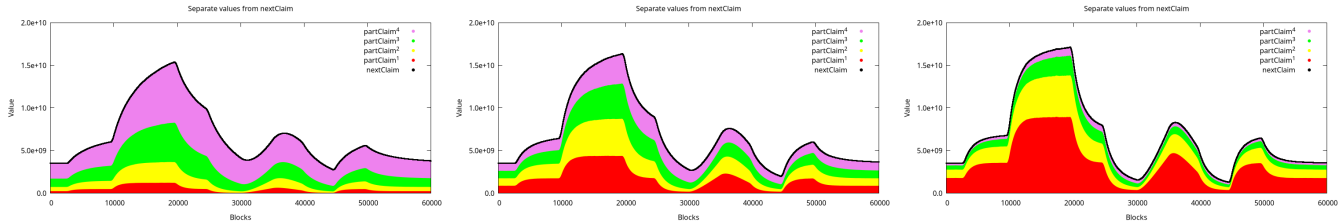


（g） 4 $\mathcal{FRSC}$s 情景，$\mathbb{C} = 0.9$。

（h） $\partial Claim$s 和 $nextClaim$，$\mathbb{C} = 0.9$。

（i）按 ρ 归一化的 $\partial Claim$s，$\mathbb{C} = 0.9$。

图 3：实验 II：在包含多个 $\mathcal{FRSC}$ 的场景中研究各种 $\mathbb{C}$s，以及它们对应的 λ ={1008, 2016, 4032, 8064} 和 ρ = {0.07, 0.14, 0.28, 0.51}。$\partial Claim$s 表示各个 $\mathcal{FRSC}$s 对矿工总奖励的贡献（即其 $nextClaim$ 组成部分）。

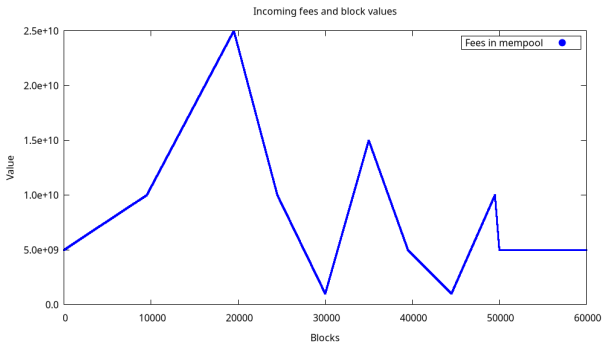


(a) 与 λ相关的 ρ 。

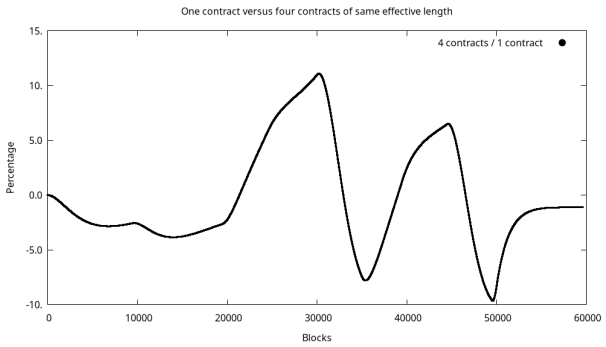
(b) 对每个 $\mathcal{FRSC}$均相等的 ρ 。

(c) 与 λ负相关的 ρ 。

图 4：实验 II – μ 个 $\mathcal{FRSC}$s 使用 ρ 的各种分布及其对 $\partial Claim$ 的影响，其中 $\mathbb{C}$ = 0。7 .



（a）实验 III 的自定义费用场景。



（b） $nextClaim$ between 4 $\mathcal{FRSC}$s 与单个 $\mathcal{FRSC}$之间的相对差异。

图 5：实验 III 比较 4 $\mathcal{FRSC}$s 和 1 $\mathcal{FRSC}$，两种配置具有相同的有效 λ。 –

他获得的奖励，以及他为激励其他矿工在其区块之上继续挖矿而留下的更多奖励。该矿工每次分叉区块链时都会压低奖励。

1) 方法： 由于缺少难度重新调整功能（我们的工作以及 [2]中同样存在这一问题），孤块率会更高，这可能会直接影响我们基于 $\mathcal{FRSC}$ 的方法。如果孤块率约为 40%，大致对应于 [2],，我们的区块平均需要多花 40% 的时间才能创建，从而增加区块创建时间（即挖出一个区块所需的时间）。这不会影响原始模拟器，因为不存在会改变找到该区块的矿工总奖励的 $\mathcal{FRSC}s$ 。

然而，对于基于 $\mathcal{FRSC}$ 的模拟，情况并非如此，因为 $\mathcal{FRSC}s$ 的初始设置是按 $fees = 50$ 比特币计算的（与原始模拟一致）。但是，随着区块创建时间延长，且交易费根据该时间计算， $fees$ 的数量也会发生变化。如果不作任何调整， $\mathcal{FRSC}s$ 在达到饱和之前，最初支付给矿工的奖励会较少。为缓解这一问题，我们在每次运行之前，按照上一轮博弈的孤块率，增加各个 $\mathcal{FRSC}$ 的初始值。这样得到的条件非常相似，可通过比较我们的模拟与原始模拟中最长链的最终值加以验证。

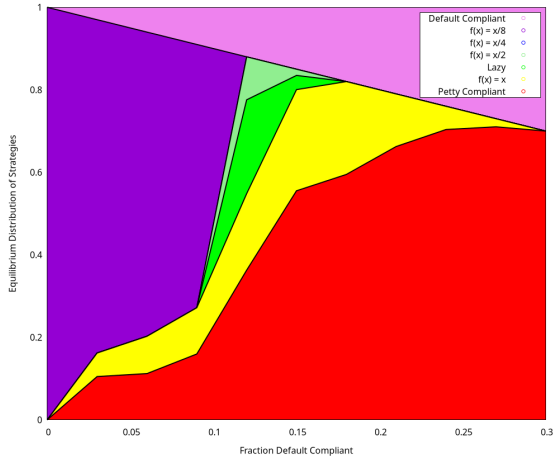


图6：在我们的方法中，默认合规矿工的数量为 ∼30%（相比之下， [2]为 ∼ 66%）。

结果与原始模拟的最终值非常接近。我们决定采用这种方法，以尽可能贴近原始实验。当完整内存池的参数等于 $false$ 时（见第 IV 节），这一点尤为重要，因为这意味着流入内存池的交易费是根据区块创建时间计算的。在我们的模拟中，使用了以下参数：100 名矿工、每轮博弈 10 000 个区块、300 000 轮博弈（每次模拟运行）、EXP3 学习模型，以及 $\mathbb{C} = 0.7$。费用建模采用了与原论文 [2] 相同的参数：禁用完整内存池参数，并且每 600 秒恒定流入 5 000 000 000 聪（即 50 比特币）。有关学习策略和其他参数的详细信息，请参阅 [2]。

设置 $\mathcal{FRSC}s$：由于手续费持续流入内存池，我们不需要对矿工的收入取平均值。因此，我们仅使用了单个 $\mathcal{FRSC}$ ，其定义为 $\mathcal{FRSC}$(7 056 000 000 000, 2016年, 1)，其中 $\mathcal{FRSC}.\nu$ 的初始值根据公式 12 进行了调整，并假设$\overline{fees} = 50$ 比特币。在之后的任何一轮博弈中，$\mathcal{FRSC}.\nu$都会按照前几轮中提到的孤块率进行增加。

*2)*结果： 该实验的结果如图6所示，结果表明，在我们使用 $\mathcal{FRSC}s$ 的方法下，我们将默认合规矿工的数量（即紫色与红色相交部分）从原来的 66% 降低到了 30%。这意味着，只要有至少 30% 的默认合规矿工，就可以避免矿工通过压价获利，从而表明结果更加稳健。

VI. 安全性分析与讨论

A. 合约干涸攻击

这是我们方案中可能出现的一种新型攻击；不过，这是一种旨在破坏我们方案功能的滥用型攻击，而非以最大化对手的利润为目的。在这种攻击中，对手只打算从 $\mathcal{FRSC}s$中获取奖励，而不在区块中纳入交易（或只纳入其中一部分）。

这可能导致 $\mathcal{FRSC}$s中的资金缓慢耗尽，从而使未来诚实矿工获得的奖励减少。此外，攻击者可以在$\mathcal{FRSC}$s饱和度较高的有利时机进行挖矿，并在一段时间后决定停止挖矿。这可能会降低诚实矿工的盈利能力，进而削弱针对削价攻击的安全性。

攻击者可以不断重复这种行为，这很可能会导致内存池中的交易费增加，因为与诚实矿工相比，攻击者会在内存池中保留更多尚未挖出的交易。因此，如果诚实矿工挖出一个区块，他就能获得更高的奖励，同时将更多交易费存入 $\mathcal{FRSC}$s，这表明我们的方法具有一定的自我调节能力，能够缓解这类攻击。

此外，我们可以考虑通过一种方式降低此类攻击的影响：只有当区块包含足够的交易费（例如，达到网络规定的最低阈值）时，才向矿工发放全部 $nextClaim_{[H+1]}$ by $\mathcal{FRSC}$s。不过，这种做法假设内存池中始终有合理数量的手续费，但实际情况未必如此；如果可供挖掘的交易不足，矿工可能会暂时停止挖矿。尽管如此，我们仍需开展更多研究，以探讨这一方案或能缓解此类潜在滥用攻击的更优方案；这将留待未来工作。

B. 可能的改进

$\mathcal{FRSC}$ 可以包含一个参数，用于设定 $\mathcal{FRSC}^x$支付的奖励（即 $nextClaim^x_{[H+1]}$）相对于其数值中位数（在 λ上计算）的可能变化区间。如果 $nextClaim^x_{[H+1]}$ 因进入内存池的手续费显著增加而大幅高于其中位数值，那么 $\mathcal{FRSC}^x_{[H]}$ 将从区间 $\langle average, nextClaim^x_{[H+1]}\rangle$ 中选取某个值（由该参数指定）奖励矿工，而不是支付全部 $nextClaim^x_{[H+1]}$。这对于 λ 参数较小的$\mathcal{FRSC}$尤其有用。用于抽样的参数可以包含一个随机函数（例如指数函数），使得接近中位数的值具有更高的出现概率。不过，我们将这项技术的评估留待未来工作。

C. 挖矿难度的调整

如果采用最长链分叉选择规则的工作量证明区块链使用基于交易费的机制，矿工的盈利能力可能会更加不稳定，进而降低其抵御削价攻击的安全性。虽然我们提出的解决方案借助 $\mathcal{FRSC}$s 有助于缓解这一问题，但我们还提出了另一项功能：根据一个时期内收取的交易费总额调整挖矿难度。具体而言，一个时期内从交易中收取的费用越高，难度就可以相应提高，反之亦然。还需要进一步研究来评估这一方案。

VII. 相关工作

Carlsten 等人 [2]的研究启发了我们的论文。该研究首次描述了由区块创建时间服从指数分布以及交易费存在显著差异所引发的削价攻击。作者对采用基于交易费机制的比特币进行了模拟，并发现 DEFAULT- CO MPLIANT 矿工的最低门槛为 66%。

Gong 等人 [8] 指出，无论区块大小限制如何，使用内存池中累积的全部手续费在实践中都不可行，而且会夸大最初在 [2]中描述的削价攻击的盈利能力。此外，Houy [9]证明，对区块大小限制（进而也限制交易数量）的约束具有经济意义，并能避免交易费降至零。因此，Gong 等人 [8] 在给定区块大小限制的情况下，对削价攻击的盈利能力进行建模，该限制限定了每个挖矿轮次中可领取的费用。作者提出了一种对策：有选择地将交易打包进新区块，同时少领取一些费用，以避免削价攻击。我们认为，与我们的方法不同，这种方案无法通过共识协议强制实施，因此仍可能发生削价攻击。

Zhou 等人 [10]研究了挖矿缺口问题；当区块链的吞吐量较高时，这一问题尤为突出。因此，作者提出了一种自适应算法，每 1000 个区块调整一次区块大小，从而确保区块有足够的空间打包新交易。

尽管 Bitcoin-NG [11]提出了一种新的共识机制，但其中也包含了将交易费在两个实体之间分配的理念——当前领导者和区块的矿工，这应能激励矿工纳入由领导者创建的区块。然而，Bitcoin-NG 在某种意义上是中心化的，因此削价攻击并非其研究对象。

VIII. 结论

在这项工作中，我们聚焦于与采用最长链分叉选择规则的基于交易费的区块链机制相关的三个问题：(1) 挖矿缺口；(2) 发生削价攻击的可能性；以及 (3) 挖矿奖励的不稳定性。为缓解这些问题，我们提出了一种基于费用再分配智能合约、对移动平均值进行近似的方法。这些合约会积累一定比例的交易费，同时利用其储备金向矿工发放奖励。通过这种方式，即使在交易费非常低的时期，例如挖矿轮次开始时、新矿工加入挖矿协议时以及出现市场偏差时，矿工也能获得足够的奖励。

此外，我们的方法对削价攻击的容忍度更高，并将完全遵循协议且严格不实施削价攻击的矿工所需的最低门槛从现有最先进研究报告的 66% 降至 30%。

参考文献

[1] Q. Hum、W. J. Tan、S. Y. Tey、L. Lenus、I. Homoliak、Y. Lin 和 J. Sun，《Coinwatch：一种用于检测加密货币漏洞的基于克隆的方法》，载于*2020*年*IEEE*国际区块链会议（*Blockchain*）。IEEE，2020年，第17–25页。[2] M. Carlsten、H. Kalodner、S. M. Weinberg 和 A. Narayanan，《论比特币在没有区块奖励时的不稳定性》，载于*2016*年*ACM SIGSAC*计算机与通信安全会议论文集，2016年，第154–167页。[3] P. Daian、S. Goldfeder、T. Kell、Y. Li、X. Zhao、I. Bentov、L. Breidenbach 和 A. Juels，《闪电男孩2.0：去中心化交易所中的抢先交易、矿工可提取价值与共识不稳定性》，载于*2020*年*IEEE*安全与隐私研讨会（*SP*）。IEEE，2020年，第910–927页。[4] I. Tsabary 和 I. Eyal，《间隙博弈》，载于*2018*年*ACM SIGSAC*计算机与通信安全会议论文集，2018年，第713–728页。[5] B. Wiki，《矿工费》，2022年。 [在线]。网址：https://en.bitcoin.it/ wiki/Miner fees

[6] S. Nakamoto，《比特币：一种点对点电子现金系统》，去中心化商业评论，第21260页，2008年。[7] H. Kalodner，《比特币挖矿模拟器》，在线资源，2015年。 [在线]。网址：https://github.com/citp/mining simulator [8] T. Gong、M. Minaei、W. Sun 和 A. Kate，《迈向解决压价问题》，载于国际金融密码学与数据安全会议。施普林格，2022年，第444–463页。[9] N. Houy，《比特币交易费的经济学》，*GATE*工作论文，第1407卷，2014年。[10] D. Zhou、N. Ruan 和 W. Jia，《一种无需区块奖励的比特币网络稳健吞吐量方案》，载于*2019*年*IEEE*第*21*届高性能计算与通信国际会议；*IEEE*第*17*届智慧城市国际会议；*IEEE*第*5*届数据科学与系统国际会议（*HPCC/SmartCity/DSS*）。IEEE，2019年，第706–713页。[11] I. Eyal、A. E. Gencer、E. G. Sirer 和 R. Van Renesse，《{比特币NG}：一种可扩展区块链协议》，载于第*13*届*USENIX*网络系统设计与实现研讨会（*NSDI 16*），2016年，第45–59页。