Bribe & Fork: Cheap PCN Bribing Attacks via Forking Threat

Zeta Avarikioti
Department of Informatics, TU Wien, Austria

Paweł Kędzior
University of Warsaw, Poland

Tomasz Lizurej
NASK, Warsaw, Poland
University of Warsaw, Poland

Tomasz Michalak
IDEAS NCBR, Warsaw, Poland
University of Warsaw, Poland

Abstract

In this work, we reexamine the vulnerability of Payment Channel Networks (PCNs) to bribing attacks, where an adversary incentivizes blockchain miners to deliberately ignore a specific transaction to undermine the punishment mechanism of PCNs. While previous studies have posited a prohibitive cost for such attacks, we show that this cost can be dramatically reduced (to approximately $125), thereby increasing the likelihood of these attacks. To this end, we introduce *Bribe & Fork*, a modified bribing attack that leverages the threat of a so-called feather fork which we analyze with a novel formal model for the mining game with forking. We empirically analyze historical data of some real-world blockchain implementations to evaluate the scale of this cost reduction. Our findings shed more light on the potential vulnerability of PCNs and highlight the need for robust solutions.

2012 ACM Subject Classification Security and privacy → Systems security

Keywords and phrases Blockchain, Payment Channels Networks, Timelock Bribing, Feather Forking

Digital Object Identifier 10.4230/LIPIcs.AFT.2024.11

Related Version *Full Version*: `https://arxiv.org/abs/2402.01363`

Funding This work was supported by the Austrian Science Fund (FWF) through the SFB SpyCode project F8512-N, the project CoRaF (grant agreement ESP 68-N), and by the WWTF through the project 10.47379/ICT22045.
This result is part of a project that received funding from the European Research Council (ERC) under the European Union's Horizon 2020 and Horizon Europe research and innovation programs (grant PROCONTRA-885666). This work was also partly supported by the National Science Centre, Poland, under research project No. 46339.

Acknowledgements We thank Paul Harrenstein for his help in defining the model presented in this work.

1 Introduction

The financial world was transformed by blockchains such as Bitcoin [24] and Ethereum [32]. While blockchains offer a number of benefits, their scalability remains a significant challenge when compared to traditional centralized payment systems [10]. One promising solution to this issue is the so-called *payment channel networks (PCNs)* that move most of the transaction workload off-chain [15].

Several PCN proposals [1, 2, 3, 4, 5, 11, 12, 13, 14, 16, 25, 28] have been laid forward so far, each design offering some unique combination of features. Nonetheless, the core idea behind payment channels remains the same, that is to facilitate off-chain transactions among parties connected, either directly or indirectly via a path, on a network operating on top of the blockchain layer, the PCN.

To participate in a PCN, two parties can lock funds in a joint account on-chain, thereby opening a payment channel. Subsequently, the parties can transact off-chain by simply updating (and signing) the distribution of their funds. When either party wants to settle the account, or in other words close the payment channel with its counterparty, they can publish the last agreed distribution of the channel's funds. However, each update constitutes a valid closure of the channel from the perspective of the blockchain miners. As a result, a malicious party may publish an outdated update to close the channel holding more than it currently owns. To secure the funds against such attacks, payment channels enforce a *dispute period*. During this period, the funds remain locked to allow the counterparty to punish any malicious behavior, and if so, claim all the funds locked in the channel.

Hence, the security of PCNs, like the most widely deployed Bitcoin Lighting Network [25], crucially relies on financial incentives. Specifically, during the dispute period, the punishment mechanism should enforce that a malicious party is always penalized and an honest party should never lose its funds. Unfortunately, this is not always the case, as argued by Khabbazian et al. [23] and Avarikioti et al. [6]. For instance, Lighting channels are susceptible to the so-called *timelock bribing attacks*. In such an attack, a malicious party posts an old update transaction on-chain, attempting to close its channel with more funds than it presently possesses. Concurrently, the party bribes the miners to ignore the punishment transaction of the counterparty for the duration of the dispute period. This bribe is typically offered to the miners in the form of high transaction fees.

Naturally, the success of this attack depends on the value of the bribe. Avarikioti et al. [6] showed that a bribing attack will be successful if the bribe is no smaller than: $\frac{f_1 - f}{\lambda_{min}}$, where λ_{min} is the fraction of the mining power controlled by the least significant miner in the underlying proof-of-work blockchain, f_1 is the sum of the fees of a block containing the punishment transaction, and f is the sum of the fees of a block containing average transactions. We observe, however, that as λ_{min} can be arbitrarily small, the bribing amounts required can significantly exceed the funds typically locked in PCNs, rendering the bribing attack impractical. Moreover, even in blockchains with rather concentrated mining power, like in Bitcoin [18], the cost of a bribing attack is very high. For example, conservatively assuming that the smallest miner has 10^{-4} of the total mining power[1], the cost of the attack as analyzed in [6] would be at least 1 BTC, for $f_1 - f \approx 10^{-4}$ BTC. As a result, it would be irrational to perform a bribing attack of this sort, as the average closing price for 1 BTC between, for instance, 2019 and 2022 was $23,530.92$ USD[2], which is more than 10-fold the current total value locked on average in a Lighting channel[3]. This naturally leads to questioning *whether there is potential to amplify such attacks to the extent they pose a genuine threat to the security of PCNs.*

[1] Details can be found in the full version of the paper. We experimentally show, that the mining power of the weakest miner in the system can be fairly assumed to be of magnitude 10^{-12}

[2] `statmuse.com`

[3] `https://1ml.com/statistics`

Our Contribution

In this work, we show that *bribing costs can be significantly reduced*, thereby making timelock bribing attacks a realistic threat. We do so by extending the bribing attack to leverage not only the structure of transaction fees but also a threat to fork the blockchain, known as a *feather fork attack* [19, 27]. In our context, a given miner executes a feather fork attack by announcing a self-penalty transaction tx_p. Whenever the self-penalty transaction appears on the blockchain, the miner is incentivized to fork the punishment transaction tx_1 on the blockchain, i.e., the miner will try to extend the blockchain based on the predecessor of the block tx_1 including some other block. Specifically, a feather-forking miner is bribed to commit collateral, betting that their fork will win the race. As a consequence, their threat of forking becomes considerably credible, incentivizing other miners to follow their fork. The collateral is of a similar magnitude as the bribe in [6], however, the miners only lock it *temporarily*. We call our attack *Bribe & Fork*. With the feather fork at hand, the bribing cost may now be reduced from $\frac{f_1-f}{\lambda_{min}}$ to approximately:

$$\frac{2\overline{f}+2(f_1-f)}{\lambda_s},$$

where $\overline{f}$ is the average fee of a single transaction, and λ_s is the mining fraction of the most significant miner. Recall that f_1 and f denote the sum of fees of a block containing the punishment transaction and only average transactions respectively. To demonstrate the cost reduction of *Bribe & Fork*, we reexamine the previous example for Bitcoin, with $\overline{f} \approx 10^{-4}$ BTC, $f_1 - f \approx \overline{f}$, and $\lambda_s \approx 20\%$. Now, λ_s replaces in the denominator the previously presented $\lambda_{min} << 10^{-4}$, thus *yielding a bribe at least 1000 times smaller than the one in [6].*

To derive this result, we present a formal model of mining games with forking, extending the conditionally timelocked games introduced in [6]. In the game with forks, miners may now choose, in each round, (a) which transactions to mine, and (b) whether they want to continue one of the existing chains or they intend to fork one of the chains. All miners know the choices of the winner of each block, as a feather-forking miner locks collateral on-chain.

To empirically estimate the cost reduction of *Bribe & Fork*, we analyze the historical data of real-world blockchain implementations. Among others, we analyze the average block rewards and fees, as well as the hash power present in the system and available to a single miner, primarily for Bitcoin in 2022. Given the officially available data, we observe that *the cost of our attack can be as cheap as $125* (for 1 BTC $\approx$ $25.000). In general, the cost of our attack can be up to 10^{10} times cheaper than the bribe required in [6] according to our findings. Hence, even considering a collateral of around $\$30,000$, *Bribe & Fork* is substantially more cost-efficient, and, by extension, more probable to occur.

2 Background

In this section, we first describe the necessary context required to understand *Bribe & Fork*.

2.1 Timelocked Bribing Attack

Whenever miners decide to create a new block, they select some set of transactions from all transactions posted on the mempool, which is a database of all publicly visible transactions. Mining pools and individual miners usually choose the transactions with the highest fees first, as they are part of their reward for a successfully mined block. Miners are aware that some

transactions may be dependent on each other. For instance, two transactions that spend the same Unspent Transaction Output (or UTXO) in Bitcoin, cannot be both published on-chain; the transaction that is validated first, i.e., is included on a block of the longest chain, immediately deems invalid the other transaction. If from two interdependent transactions, only one can be published directly, while the other is timelocked and thus can only be published after some time elapses, we refer to this pair of transactions as a *conditionally timelocked output*. This conditionally timelocked output is the target of timelock bribing attacks: the owner of the coins of the transaction that is valid only after the timelock expires attempts to bribe the miners to ignore the currently valid competing transaction. Thus, for the miners that observe transactions on the mempool, sometimes it may be profitable to censor one transaction in order to mine another one that provides a greater gain in the future.

2.2 Timelock Bribing in the Bitcoin Lightning Network

Next, we describe the timelock bribing attack in the context of the most widely deployed payment channel network, namely the Bitcoin Lightning Network (LN). In LN, a single on-chain transaction called the funding transaction, opens a channel between parties $\mathcal{P}_1, \mathcal{P}_2$. Next, parties exchange with each other signed messages off-chain which update the state of their accounts. If the parties are honest and responsive, they may close the channel in collaboration. To do so, the parties post a mutually signed transaction that spends the output of the funding transaction and awards each party their fair share of funds. However, if a dishonest party $\mathcal{P}_2$ attempts to publish on-chain an old state that she profits from comparably to the latest agreed state, her funds will remain locked for the so-called dispute period. During this period, the other (rational) party $\mathcal{P}_1$ will try to revoke the old state, by sending a transaction tx_1 to the mempool called the revocation transaction (or breach remedy). Transaction tx_1 awards all the channel funds to the cheated party $\mathcal{P}_1$. We denote by f_1 the miner's fee to include a block with tx_1.

In this case, the malicious party $\mathcal{P}_2$ can launch a timelocked bribing attack, attempting to bribe the miners to ignore tx_1 for the dispute period T such that $\mathcal{P}_2$ gains the additional funds. Specifically, $\mathcal{P}_2$ may send in the mempool a block with fee f_2 that includes transaction tx_2, with $f_2 > f_1$, that is only valid if no miner includes in their winning block containing tx_1 within time T. Consequently, if the revocation transaction tx_1 is not published on-chain within T, $\mathcal{P}_2$ can spend the funds of the old state and the next winning miner will be awarded f_2. The pair of transactions tx_1 and tx_2 is now a conditionally timelocked output.

Assuming that for an average block of transactions, the users get in total f fees, the following holds [6]: if $f_2 - f > \frac{f_1 - f}{\lambda_{min}}$ then all rational miners will choose to wait for T rounds and publish a block containing tx_2.

2.3 Feather Fork Attack

A feather fork, as introduced in [20], is an attack on Bitcoin wherein a miner threatens to fork the chain if selected transactions are included. This intimidation mechanism aims to subtly alter the miners' block acceptance policy: the threatened miners may exclude the selected transactions in order to mitigate the risk of losing their mining reward [31]. As feather forking relies on economic incentives, the attacker may increase their probability of success by bribing other miners to follow their short-lived fork, e.g., committing to pay them the block rewards they may lose by censoring the selected transactions.

Unlike a "hard" fork, where miners exclusively mine their own chain version regardless of its length compared to other versions, a feather fork entails mining on the longest chain that excludes selected transactions and does not fall significantly behind its alternatives. Thus, a feather fork is less disruptive and more likely to be adopted by the network, making it a potentially powerful tool in the hands of a malicious actor. In this work, we employ this tool to enhance the likelihood of a successful timelocked bribing attack.

3 *Bribe & Fork* Attack

We now introduce our novel attack, termed *Bribe & Fork*, that combines the timelock bribing attack in LN with the feather fork attack. We assume the existence of a payment channel between parties $\mathcal{P}_1$ and $\mathcal{P}_2$. Similarly to timelock bribing, we consider $\mathcal{P}_2$ to be malicious and attempt to close the channel with $\mathcal{P}_1$ in an old state using the transaction tx_2. Consequently, $\mathcal{P}_1$ is expected to attempt to revoke the old state. To prevent the inclusion of the revocation transaction, $\mathcal{P}_2$ bribes the miner N_s with the highest mining power λ_s to threaten others with a fork if they add the unwanted transaction txs_1 on-chain. This action is implemented through a self-penalty mechanism where the bribed miner temporarily locks collateral which can only be reclaimed in case a block txs_2 containing tx_2 appears on-chain (and thus any block txs_1 containing tx_1 is not included on-chain). In essence, the bribed miner "bets" that the revocation transaction will be censored, thus rendering the threat credible for the rest of the miners.

To realize *Bribe & Fork*, there are two mechanisms that should be implementable on-chain: a) the bribe transaction that should only be spendable if txs_2 is included on-chain, and b) the self-penalty mechanism that enables the bribed miner N_s to lock collateral P (with transaction tx_{p_1}) and then reclaim it only if txs_2 is included on-chain (with transaction tx_{p_2}). We implement the bribing and self-penalty mechanisms in Bitcoin script, using the conditioning enabled by the UTXO (Unspent Transaction Output) structure. We note that in Ethereum, preparing a smart contract that has access to the state of the closing channel suffices to implement the bribing and self-penalty mechanisms.

In detail, a single bribing transaction tx_b and two special transactions tx_{p_1}, tx_{p_2} are introduced. Let us assume that the cheating party $\mathcal{P}_2$ is bribing the miners to launch the attack conditioned on the inclusion of its transaction tx_2. To do so, $\mathcal{P}_2$ creates a bribe transaction tx_b with input the party's money from tx_2 and outputs three UTXOs, one given to the miner that mines this transaction, one dummy output owned by player N_s (i.e., the miner bribed to perform feather forking), and one that returns the rest of the money to $\mathcal{P}_2$. Now, N_s creates a transaction tx_{p_1}, locking a deposit P, that is spendable via a multisignature of m-out-of-n parties (e.g., $m = n/2$), one of which is N_s's signature. Then, tx_{p_2} is created with two inputs: the output of tx_{p_1} and the dummy output of tx_b. Consequently, tx_{p_2} is spendable only if it is signed by at least m parties of the predefined set n and tx_2 is validated on-chain. Upon receiving tx_{p_2} signed by at least $m-1$ parties, N_s signs and posts it on-chain. Assuming that no subset of size $m-1$ of the rest $n-1$ parties will collude with the miner to spend the deposit, the deposit can be claimed by the miner only if transaction tx_2 is included on-chain. The security of this scheme depends on the selection of the $n-1$ parties, which can be in principle conditioned on the honest majority assumption of the blockchain via subsampling.[4]

[4] One could also consider using an instantiation of a Trusted Execution Environment (TEE) that outputs tx_{p_2} only when txs_2 appears on the blockchain. Additionally, note that on Ethereum, preparing a smart contract that has access to the state of the closing channel is sufficient to implement the penalty mechanism.

Now, the malicious party $\mathcal{P}_2$ together with a selected miner N_s can launch the *Bribe & Fork* attack as follows: They can create special transactions tx_b (as a companion transaction for tx_2) and the self-penalty transactions tx_{p_1}, tx_{p_2} to publicly announce a credible threat that the transaction tx_1 will be forked once it appears on the blockchain. They publish these special transactions in the mempool as transaction sets txs_2 (that contains tx_2 and tx_b) and txs_{p_1}, txs_{p_2} (that contain tx_{p_1}, tx_{p_2}, respectively).

We show later that this attack significantly reduces the cost of the required bribe from $\frac{f_1 - f}{\lambda_{min}}$ to approximately $\frac{2\overline{f}+2(f_1-f)}{\lambda_s}$. The required collateral that is eventually reclaimed by the bribed miner is expected to be $\lambda_s \cdot B$ (where B is a constant in Bitcoin block reward independent of the user fees), which is comparable to the original bribe needed in [6]. Figure 1 illustrates the comparison of *Bribe & Fork* with [6], while Figure 2 below depicts the details of *Bribe & Fork*.

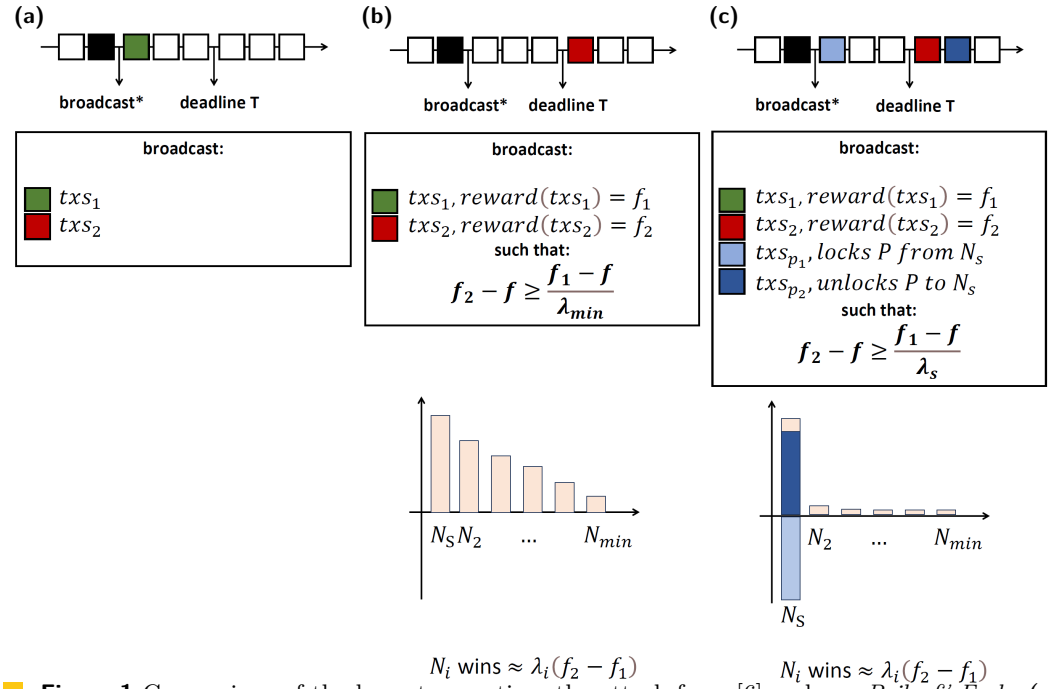

Figure 1 Comparison of the honest execution, the attack from [6] and our *Bribe & Fork*. **(a) Honest execution:** once an old state appears on-chain (black rectangle), $\mathcal{P}_1$ gets an option to revoke this state with a transaction tx_1 (included in the block txs_1 which is published in the first round). **(b) Attack in [6]:** the bribing party publishes tx_2 and tx_b included in a single block txs_2, with a large miner fee (reversely proportional to the fraction of the mining power λ_{min} of the least significant miner). The miners skip mining txs_1 in the first round, and mine txs_2 in the last round. **(c) *Bribe & Fork*:** the bribing party publishes txs_2 with a fee sufficient to bribe only the strongest miner (with $f_2 - f$ reversely proportional to λ_s). The strongest miner publishes the self-penalty transactions tx_{p_1}, tx_{p_2} that can be mined in transaction sets txs_{p_1}, txs_{p_2}. In the first round, the miner N_s locks $P \approx \lambda_s \cdot B$ to the deposit transaction tx_{p_1}, thus threatening other miners that they will be forked once txs_1 is mined before the deadline. After the deadline the transaction set txs_2 is published and the miner N_s may collect back the deposit using txs_{p_2}.

Implementation Details of *Bribe & Fork*

Figure 2 contains a diagram depicting the details of our attack. At first, a Lightning Channel is opened with a single funding transaction and allows parties $\mathcal{P}_1, \mathcal{P}_2$ to make an arbitrary number of off-chain state transitions of their funds. Once one of the parties ($\mathcal{P}_2$ in our

example) decides to publish an old state ($comm_i$ in our example) at a chain of length T_0, the opportunity to manipulate the behaviour of miners' is opened. Before the chain reaches length $T_0 + 1$, the transactions tx_1 and tx_2 are published. Transaction tx_1 allows $\mathcal{P}_1$ to revoke a dishonestly committed state. Transaction tx_2 allows $\mathcal{P}_2$ to collect and manage dishonest funds after T rounds. Along with tx_1 and tx_2, the bribing transaction tx_b and self-penalty transactions tx_{p_1}, tx_{p_2} are published. On the chain of length T_0, the miners may decide to mine the transaction tx_{p_1} that would lock some amount of coins of one of the miners (say N_s). If so, all miners are threatened that they will be forked once tx_1 appears on the blockchain until the chain reaches length $T_0 + T$. On the chain of length $T_0 + T$, the transaction tx_2 along with bribing transaction tx_b may be published and the selected miner N_s may collect back the deposit with the transaction tx_{p_2}.

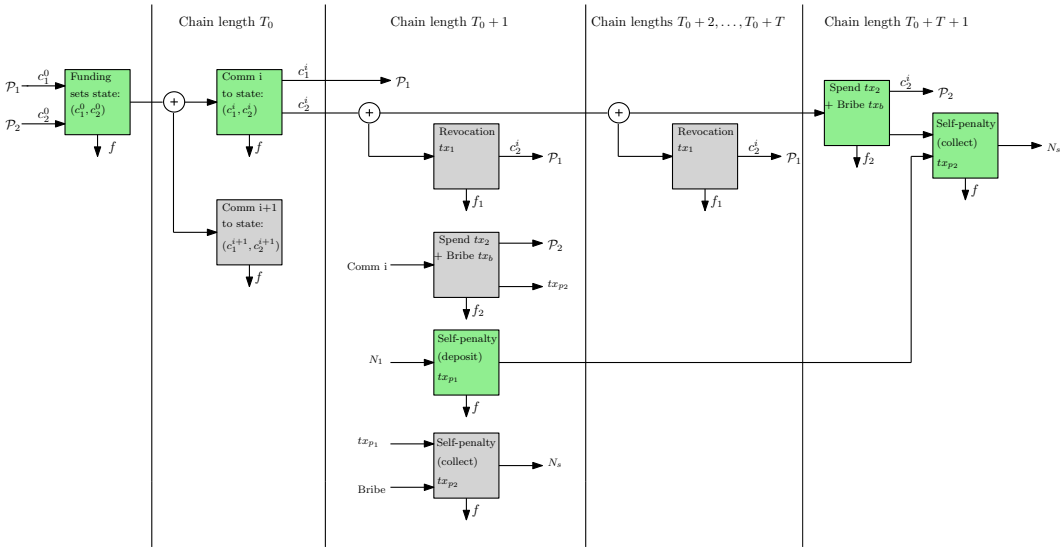

Figure 2 The *Bribe & Fork* attack. The green boxes indicate the transactions that should be put on-chain to run a successful *Bribe & Fork* attack. The grey boxes indicate the transactions that should be published on the mempool before the chain reaches a specific length. For instance, Spend transaction tx_2 has to be published on the mempool before the chain reaches length $T_0 + 1$, even though it can not be published on the blockchain until the chain reaches length $T_0 + T$. The arrows going into the boxes indicate the spending conditions of the transactions and the arrows going out of the boxes indicate how the funds of the boxes can be spent.

4 Our Model

In this section, we gradually define our game that models the process of mining that takes into account the forks. A summary of our notation can be found in the full version of this paper.

4.1 Preliminaries

Let us begin by recalling the conditionally-timelocked output definition from [6].

▶ **Definition 1** (Conditionally timelocked output [6]). *A conditionally timelocked transaction output $txo(T_0, T, cond_1, cond_2)$ is a transaction output of a transaction tx with spending condition $cond_1 \vee cond_2$. Condition $cond_1$ is not encumbered with any timelock and condition $cond_2$ is encumbered with a timelock that expires T blocks after the block with height T_0, where tx was published.*

The game with forks is defined for a fixed set of players (miners) $N = \{N_1, \ldots, N_n\}$ with a tuple of mining powers $\boldsymbol{\lambda} = (\lambda_1, \ldots, \lambda_n)$ and will last R rounds. Notice that we focus on proof-of-work blockchains that employ the so-called Nakamoto consensus, such as Bitcoin [24]. We assume that in such environments miners (players) tend to form mining pools to (a) bypass the task of verifying transactions – the pool's manager dispatches a list of valid transactions for inclusion – and more importantly, (b) to guarantee a more stable income as individual mining carries substantial deviation. Empirical evidence supporting this assumption, drawn from Bitcoin, is detailed in the full version of the paper.

For the rest of this work, each miner is assumed to either mine independently or stick to a selected mining pool throughout the execution of the game, i.e., we treat the mining pools as single players in the game. As already mentioned, we assume that the game lasts for a fixed number of rounds. Alternatively, we could consider a scenario where the game lasts until the main chain reaches a predefined length. The first assumption is more suitable for the time periods when the block rate is constant. On the other hand, the second modeling approach is better for longer periods where the mining difficulty of blockchains is adjusted to achieve a given number of blocks within a given time unit.

4.1.1 Global State Object

We introduce a global state object $\mathcal{S} = \{S_1, \ldots, S_{|S|}\}$ that describes a set of currently mined chains on the blockchain. Each S_i consists of a *list* (chain) of pairs $S_i = [\ (\mathsf{block}_1, W_1), \ldots, (\mathsf{block}_{|S_i|}, W_{|S_i|})]$ describing successfully mined blocks. In each pair $(\mathsf{block}_j, W_j) \in S_i$, block_j describes a set of transactions included in the block, and $W_j \in N$ indicates a player that successfully mined the block.

4.1.2 Allowed Actions

We define the classes of possible actions in our game:

- All chains in a state can be *continued*. When the operation *continue* is successful, a new pair (block, W) is appended to the continued chain in the global state object.
- Chains of length at least 1 in the state can be *forked*. Whenever one of the players successfully *forks*, the new (duplicate) chain is created in the global state object in the following manner:
  - the source fork is duplicated; and
  - a new block replaces the latest block in the duplicate.

  For instance, let $\mathcal{S} = \{\ [(\mathsf{block}_1, W_1), (\mathsf{block}_2, W_2)]\ \}$ be a current state with a single chain S_1. Then, after a successful fork of S_1 with (B_3, W_3), one gets $\mathcal{S} = \{\ [(\mathsf{block}_1, W_1), (\mathsf{block}_2, W_2)], [(\mathsf{block}_1, W_1), (\mathsf{block}_3, W_3)]\ \}$.

Notice that on existing blockchains, miners can fork a chain or mine on top of an arbitrary block in one of the existing chains. However, forking that starts at old blocks is less likely to outrun the longest chain. For that reason, we exclude this possibility from the game (following [20, 31, 26]). In other words, miners in our model can fork only the last transaction on one of the chains, and then either the original chain or the fork becomes stable whenever it reaches a length equal to the length of the original chain plus one. The forks can be modeled differently, e.g., assuming a longer fork length or using a finite automaton definition. We expect our results to hold in the alternative modeling as well, but with different parameters of our solution would change.

4.1.3 The Abandon Rule

Let us define the abandon rule $\mathsf{abandon} : \mathbb{S} \to \mathbb{S}$ that is later used to abandon old chains no longer useful in the game. As we allow forking only the newest block in a chain, our $\mathsf{abandon}$ rule will make each chain that outruns the length of other chains the only chain in the game. That is, for any $S_i \in \mathcal{S}$: $\exists_{S_j \in \mathcal{S}}\ len(S_j) \geq len(S_i) + 1$ the abandon rule removes S_i from the state $\mathcal{S}$.

4.1.4 Types of Transaction Sets

Each block mined in the game (denoted as block) includes only one of the transaction sets listed below:

- An unlimited amount of unrelated transaction sets txs_u that contain average transactions tx_u unrelated to any special transactions listed below. These transaction sets can be mined at any point in the game.
- A transaction set txs_1 that contains the transaction tx_1 that spends money of txo under $cond_1$. As long as txo is not spent, this transaction set can be mined on a chain of any length. The rest of the transactions in this transaction set are unrelated transactions tx_u.
- A (bribing) transaction set txs_2 that contains the transaction tx_2 that spends money of txo under $cond_2$ and a bribing transaction tx_b. As long as txo is not spent, this transaction set can be mined on a chain of length $\geq T$. The rest of the transactions in this transaction set are unrelated transactions tx_u.
- A special transaction set txs_{p_1}. In the first round of the game, one of the players (say N_1) might decide to create a transaction set txs_{p_1} with a self-punishment transaction tx_{p_1} (see the description of the penalty mechanism in the Section 3). The player chooses the amount P, which he deposits to the transaction. The rest of the transactions in this transaction set are unrelated transactions tx_u.
- A special transaction set txs_{p_2} with transaction tx_{p_2}. The transaction tx_{p_1} assures that the player N_1 that created the transaction tx_{p_1} may collect back the deposit P by publishing the transaction tx_{p_2}, but only after the transaction set txs_2 is published on the blockchain (see the Section 3). The rest of the transactions in this transaction set are unrelated transactions tx_u.

4.1.5 Rewards

We assume that a miner, after successfully mining a transaction set txs_i on the main chain, gets a reward $\mathsf{reward}(txs_i)$ equal to $B + f_i + P$, where B is a constant block reward and f_i is a sum of user fees input by users posting transactions in the transaction set txs_i. Whenever txs_i contains a transaction that locks C coins from the miner's account, we set P to be equal to $-C$. Analogously, when a miner collects C as one of the transactions from txs_i, we set the parameter P to C. The reward for mining a block depends on the number of transactions within the block and their fees. The fee for a more complex transaction is typically higher as it occupies more space in a block. In this respect, we make the following assumptions that correspond to the current Bitcoin values (more details can be found in the full version of the paper):

- Each unrelated transaction set txs_u has on average m transactions, its reward

$$\mathsf{reward}(txs_u) = B + f = B + m \cdot \overline{f},$$

  where $\overline{f}$ is an average user transaction fee. We also assume that $\overline{f} < 10^{-4}B$.

- For other transaction sets with an uncommon functionality, e.g., txs_j, we assume it contains in total $m - c_j$ unrelated (average) transactions, a special transaction tx_j that takes space of c_j average transactions, where $c_j < m$. In total,

  $$\mathsf{reward}(txs_j) = B + (m - c_j)\overline{f} + c_j\overline{f_j} + P = B + f_j + P,$$

  where $f_j = (m - c_j)\overline{f} + c_j\overline{f_j}$. The interpretation of the parameter c_j is that it describes the number of transactions needed in a block to implement the uncommon functionality, each of them with fee $\overline{f_j}$.

In the full version of the paper, based on empirical data, we show how block rewards fluctuate in the real world. However, following [6], we assume that standard transactions have a constant (average) reward and that all blocks have a constant number of transactions. In the list above, we refer to each standard transaction as an average transaction.

4.1.6 Mining Power Distribution

We assume the following mining power allocation $\lambda = (\lambda_1, \ldots, \lambda_n)$ among the players (see discussion in the full version of this paper).

- There exists a single "strong" player (say player s) with mining power $\lambda_s \geq 20\%$. All other players have mining power smaller than λ_s.
- There exists a "relatively" strong player (say player i) with mining power $1\% < \lambda_i < 2\%$.
- We assume that all players with mining power less than 1% have collective power at most 5%.
- The smallest mining power is of any miner in the game is $\lambda_{min} > 10^{-100}$.

4.1.7 Players' reluctance to believe a threat

In the mining process, players can threaten other players that they will fork their blocks, once these blocks appear on the blockchain, as in the feather forking attacks. However, without any additional assumptions, there exist multiple solutions for such a setting [17]. To derive a single solution in our game, we make a conservative assumption that the players do not conduct the forking action if it *can* result in financial losses to them. In other words, we accept only threats from a player who strictly profits from forking a selected transaction, i.e., the forking action is a dominating one for the player in this particular state.

4.1.8 No Shallow Forks Conjecture

The Conjecture 1 below is a second assumption (together with the assumption that players are reluctant to believe a threat) that allows us to achieve a unique solution in the game with forks. In the conjecture, we assume that players have the option to fork a transaction only when they see an explicit opportunity of mining any other transaction with a *higher* miner's fee[5], initially blocked by the currently forked transaction. That is, we forbid shallow forks in the model.

[5] We denote that, alternatively to Conjecture 1, one could assume that the size of the mining fees in the game is limited, as excluding transactions with outstandingly high fees can also discourage the players from forking these transactions.

▶ **Conjecture 1** (No shallow forks). *At any point in the game Γ_F, the players will not start a fork of a chain ending with a transaction set txs_a, unless they see an explicit opportunity to mine a fork containing at some point txs_b, initially blocked by txs_a (e.g., by the conditionally timelocked output transaction mechanism, or the self-penalty mechanism). What is more, the players must be aware that* $\mathsf{reward}(txs_b) > \mathsf{reward}(txs_a)$.

Note that given the feasible transaction sets in the Section 4.1.4 and the reward structure defined in the Section 4.1.5, whenever $\mathsf{reward}(txs_2) > \mathsf{reward}(txs_1)$ and $\mathsf{reward}(txs_1) > \mathsf{reward}(txs_{p_1})$, according to the Conjecture 1, the players in our game can attempt to fork only txs_1 to get txs_2 or txs_{p_1} to mine txs_1. In other words, whenever txs_u, txs_2, or txs_{p_2} appear on one of the chains, they will not be forked.

4.2 The Game

Finally, we describe a game that models the process of PoW blockchain mining taking into account the option to conduct a forking of a block. The game proceeds in rounds; in each round, the miners can choose whether they want to continue mining one of the chains or they want to fork one of the chains.

▶ **Definition 2** (Conditionally timelocked game with forks). *A* conditionally timelocked game with forks $\Gamma_F(N, R)$ *is a game with a finite set of players (miners) $N = \{N_1, \ldots, N_n\}$, where $n = |N|$, that lasts R rounds. We define a tuple of mining powers $\boldsymbol{\lambda} = (\lambda_1, \ldots, \lambda_n)$ associated with the players, such that $\sum_{\lambda_i \in \lambda} = 1$. In the following, we will write Γ_F, instead of $\Gamma_F(N, R)$, when N, R are obvious from the context.*

Given the global state object, the set of possible actions, the rewards structure and the mining power distribution defined above, the game is played as follows:

1. *The game starts with the state $\mathcal{S} = \{[\,]\}$ which is updated exactly R times. All players are aware that this state is built upon a blockchain of height T_0 which includes an unspent conditionally timelocked transaction output $txo(T_0, T, cond_1, cond_2)$, where $T < R$.*
2. *At each round $1 \leq r \leq R$, players $N_i \in N$ choose which of the subchains $\mathcal{S}_k \in \mathcal{S}$ they build upon, whether they will* continue *or* fork *this chain and which of the feasible blocks (built upon one of the transaction sets) they want to add in case they are declared as the winner. Let $\Omega(S, r)$ denote the set of all feasible actions for the state $(\mathcal{S}, r)$ described as triplets $(S, \mathsf{decision}, \mathsf{transaction\_set})$, where $S \in \mathcal{S}$, and $\mathsf{decision} \in \{continue, fork\}$. Based on $\boldsymbol{\lambda} = (\lambda_1, \ldots, \lambda_n)$, one player is declared as the winner in the round r, and the state object is modified accordingly.*
3. *After each round, the abandon rule* abandon *is run on the current state.*

When the final round R of the game is over, it finishes in some state $\mathcal{S}$, and rewards are given to the players. By $\mathcal{S}^*$, let us denote the longest chain in the state $\mathcal{S}$. Whenever the state has multiple longest chains, $\mathcal{S}^*$ denotes the oldest of the longest chains of $\mathcal{S}$. After the final round R, in the state $\mathcal{S}$, the reward given to a player N_i is: $\mathsf{reward}_i(\mathcal{S}) = \sum_{(\mathsf{block}, W) \in \mathcal{S}^* : W = i} \mathsf{reward}(\mathsf{block})$.

4.2.1 Strategies

Notice that given the set of players N, the actions continue and mine defined above, and the set of transaction sets possible to mine, one can determine the set $\mathbb{S}$ of all states that may happen in the game.

A strategy σ_i for a player N_i is given by a function mapping each pair $(\mathcal{S}, r) \in \mathbb{S} \times [R]$ to a triplet feasible for this pair $(S, \mathsf{decision}, \mathsf{transaction\_set})$.

Let σ denote a strategy profile of all players - a tuple of strategies of all players - i.e., $\sigma = (\sigma_1, \ldots, \sigma_n)$. Given a fixed index i, with σ_{-1}, we will denote a strategy profile of all players, but the selected N_i.

The distribution of mining power among the players $\boldsymbol{\lambda} = (\lambda_1, \ldots, \lambda_n)$, the strategy profile σ, current state $\mathcal{S} \in \mathbb{S}$ and current round $r \in [R]$ define a probability distribution function $\boldsymbol{p}_{\boldsymbol{\lambda},\sigma,\mathcal{S},r,r'} : \mathbb{S} \to [0,1]$ that assigns a probability that a certain state $\mathcal{S}' \in \mathbb{S}$ is activated after round $r' \in [R]$, where $r' > r$. Given $\boldsymbol{p}_{\boldsymbol{\lambda},\sigma,\mathcal{S},r,r'}$ and the reward function, we can define the utility (the expected reward) $\mathbb{E}_i(\sigma)$ of each player i, when strategy σ is played. We say $\sigma^* = (\sigma_1^*, \ldots, \sigma_n^*)$ is a Nash Equilibrium if for all players $N_i \in N$ it holds that

$$\mathbb{E}_i(\sigma_i^*, \sigma_{-i}^*) \geq \mathbb{E}_i(\sigma_i, \sigma_{-i}^*),$$

for all alternative strategies σ_i for the player i.

We denote by $\Gamma_F^{\mathcal{S},r}$ the subgame of the game Γ_F in a round r, at a state $\mathcal{S}$. We denote by $\mathbb{E}_i(\sigma, \mathcal{S}, r)$ the utility of a player N_i in this subgame, which is the expected reward for this player once the game is over.

5 Analysis of *Bribe & Fork*

In this section, we formally analyze *Bribe & Fork* where a bribe transaction tx_2 is published, large enough to bribe a chosen miner with the highest mining power, yet significantly smaller than the value required to directly persuade all miners to skip mining the transaction txs_1. The selected miner is then asked to threaten others with a fork if they add the unwanted transaction txs_1 to the blockchain. To make these threats credible, we implemented the self-penalty mechanism (see Section 3).

5.1 About the proofs

In the proofs, we aim to find a *dominating strategy* for a player N_i in a given state $\mathcal{S}$ and a round r, i.e., a strategy that outweighs other strategies of a selected player in the given state and round. As we will move from the very last round of the game till the first round of the game, we will be able to conclude our reasoning with a single NE of the full game. Whenever needed, we use the mathematical induction technique to show that some choice of strategy is optimal for a sequence of rounds. Usually, the base case is the last round of the game and the induction step proves that if a given strategy is dominating in a round $k+1$, then it is also a dominating strategy in a round k.

When we compare how the player N_i benefits from taking two distinct actions A, B in a given state $\mathcal{S}$ and a round r, we often say that there exists a constant C common for these strategies. To this end, we assume that action A refers to some strategy σ_a of the player N_i, and action B refers to some strategy σ_b of the player N_i, such that σ_a differs only in its definition from σ_b on the selected state $\mathcal{S}$ and the selected round r. The utility of the player N_i is the same for both strategies whenever in the state $\mathcal{S}$ and r someone else than N_i is selected as the winner of the round. With C, we denote the utility of player N_i multiplied by the probability of this event when player N_i is not the winner of the round. This reasoning gives us an easy-to-use method to compare utility between the strategies σ_a, σ_b. We can thus compare the utilities of the player N_i in the state $\mathcal{S}$ and round r when the two distinct strategies σ_a, σ_b are selected as:

$$\mathbb{E}_i(\sigma_a, \mathcal{S}, r) = \lambda_i(\text{utility of the player } N_i \text{ when action A was taken}) + C,$$
$$\mathbb{E}_i(\sigma_b, \mathcal{S}, r) = \lambda_i(\text{utility of the player } N_i \text{ when action B was taken}) + C.$$

5.2 Transaction Order in a Single Chain

Although, due to the definition of the player's utility function, the bribing transaction (txs_2) may encourage the players not to skip mining transactions with high rewards during the dispute period (e.g. txs_1), we first show that once timelock is over and txs_1 was not mined, the players will follow the default strategy to mine transactions with highest rewards first.

▶ **Lemma 3.** *Let $\Gamma_F(\mathcal{S}, T+1)$ be a subgame in a state $\mathcal{S} = \{S\}$, where the state $\mathcal{S}$ contains a single chain S and the transaction set txs_{p_1} was mined in the first round. In the next $T-1$ rounds, miners mined unrelated transactions sets txs_u. Furthermore, it holds that*

$$\mathsf{reward}(txs_2) > \mathsf{reward}(txs_1) > \mathsf{reward}(txs_u) \text{ and } \mathsf{reward}(txs_{p_2}) > \mathsf{reward}(txs_u).$$

Then the dominating strategy for all players in the subgame $\Gamma_F(\mathcal{S}, T+1)$ is to mine transaction sets in the following order: txs_2, txs_{p_2}, and for the rest of rounds txs_u.

Proof. As txs_{p_1} was mined in the first round, then in any round after the round $T+1$ there are available to mine the following transaction sets:

- mutually exclusive txs_1 and txs_2 with expired timelock,
- one txs_{p_2} that can be mined only after txs_2 appears on blockchain,
- and an unlimited amount of unrelated transaction sets txs_u.

Since only txs_1 and txs_2 are mutually exclusive and $\mathsf{reward}(txs_1) < \mathsf{reward}(txs_2)$, then by Conjecture 1, whenever txs_2, txs_{p_2} or txs_u appear on the blockchain, they will not be forked. Thus only txs_1 may be forked in the subgame.

Once both txs_2, txs_{p_2} are on the chain, miners can not mine any special transaction sets, and all of the miners will mine txs_u till the end of the game.

Next we show that for any round $r \in \{T+2, \ldots, R\}$, in a state $\mathcal{S}'$ created by extending the chain in $\mathcal{S}$ with txs_u and one txs_2 at any point in this chain, the dominating strategy for all players is to mine txs_{p_2} first if it was not mined until this point. We will prove it by induction. The statement trivially holds in the last round R, because $txs_{p_2} > txs_u$. Now, assuming that it holds in round $R-k$, we prove that it also holds in round $R-k-1$. Any player N_i will be chosen with probability λ_i as the winner of the round. The utility of the player N_i following a strategy σ_{p_2} that first mines txs_{p_2} in the state $\mathcal{S}'$ is:

$$\mathbb{E}_i(\sigma_{p_2}, \mathcal{S}', R-k-1) = \lambda_i(\mathsf{reward}(\mathcal{S}') + f_{p_2} + B + \lambda_i k(f+B)) + C,$$

for some constant C that describes the expected reward of N_i in case he/she is not chosen as the winner of this round.

The utility of the player N_i following a strategy σ_u that first mines txs_u in the state $\mathcal{S}'$ is:

$$\mathbb{E}_i(\sigma_u, \mathcal{S}', R-k-1) = \begin{cases} \lambda_i(\mathsf{reward}(\mathcal{S}') + f_u + B) + C & \text{when } k = 0 \\ \lambda_i(\mathsf{reward}(\mathcal{S}') + f_u + B + \lambda_i(f_{p_2} + B) + \lambda_i(k-1)(f+B)) + C & \text{when } k \geq 1 \end{cases}$$

From the above it follows that

$$\mathbb{E}_i(\sigma_{p_2}, \mathcal{S}', R-k-1) > \mathbb{E}_i(\sigma_u, \mathcal{S}', R-k-1).$$

Next we show that for any $r \in \{R, \ldots, T+1\}$, in a state $\mathcal{S}''$ created by extending the chain in $\mathcal{S}$ with txs_u, the dominating strategy for all players is to mine txs_2 first if it was not mined until this point. Observe that in the state $\mathcal{S}''$ the miners can only mine txs_u, txs_1 or txs_2. The statement trivially holds in the last round. Now, assuming that it holds in

round $R-k$, we prove that it also holds in round $R-k+1$. Any player N_i will be chosen with probability λ_i as the winner of the round, and with probability $1-\lambda_i$ someone else will be selected as the winner of the round. Then for some constant C, the utility of the player N_i in a strategy σ_2 that first mines txs_2 is

$$\mathbb{E}_i(\sigma_2, \mathcal{S}', R-k+1) = \begin{cases} \lambda_i(\mathsf{reward}_i(\mathcal{S}'') + f_2 + B) + C & \text{when } k=0 \\ \lambda_i(\mathsf{reward}_i(\mathcal{S}'') + f_2 + B + \lambda_i(f_{p_2} + B)) + C & \text{when } k=1 \\ \lambda_i(\mathsf{reward}_i(\mathcal{S}'') + f_2 + B + \lambda_i(f_{p_2} + B) + (k-2)\lambda_i(f+B)) + C & \text{when } k \geq 2 \end{cases}$$

Whereas the utility of the player N_i in a strategy σ_1 that first mines txs_1 is

$$\mathbb{E}_i(\sigma_1, \mathcal{S}'', R-K+1) \leq \begin{cases} \lambda_i(\mathsf{reward}_i(\mathcal{S}'') + f_1 + B) + C & \text{when } k=0 \\ \max\{\lambda_i(\mathsf{reward}_i(\mathcal{S}'') + f_1 + B + \lambda_i(f+B)) + C, & \text{when } k=1 \\ \quad \lambda_i(\mathsf{reward}_i(\mathcal{S}'') + f_2 + B) + C\} & \\ \max\{\lambda_i(\mathsf{reward}_i(\mathcal{S}'') + f_1 + B + 2\lambda_i(f+B)) + C, & \text{when } k=2 \\ \quad \lambda_i(\mathsf{reward}_i(\mathcal{S}'') + f_2 + B + \lambda_i(f_{p_2} + B)) + C\} & \\ \max\{\lambda_i(\mathsf{reward}_i(\mathcal{S}'') + f_1 + B + k\lambda_i(f+B)) + C, & \text{when } k \geq 3 \\ \quad \lambda_i(\mathsf{reward}_i(\mathcal{S}'') + f_2 + B + \lambda_i(f_{p_2} + B)) + (k-3)\lambda_i(f+B))C\} & \end{cases}$$

It is again easy to see that $\mathbb{E}_i(\sigma_2, \mathcal{S}'', R-k+1) > \mathbb{E}_i(\sigma_1, \mathcal{S}'', R-K+1)$. ◀

The details of the proof of the above Lemma imply the following result.

▶ **Lemma 4.** *Let $\Gamma_F(\mathcal{S}, T+1)$ be a subgame in a state $\mathcal{S} = \{S\}$, where the state contains a single chain S where in T rounds miners mined unrelated transaction sets txs_u. Furthermore, for all miners*

$$\mathsf{reward}(txs_2) > \mathsf{reward}(txs_1) > \mathsf{reward}(txs_u).$$

Then the dominating strategy for all players in the subgame $\Gamma_F(\mathcal{S}, T+1)$ will result in the following transactions order: txs_2, and for the rest of rounds txs_u.

5.3 Decisions of an Individual Miner are Consistent

In this section, we show that without a high-cost reward f_2, once someone is successful with mining txs_1, the miner will continue mining this chain, as it might be too costly for the miner to lose the block reward that he already mined. As txs_1, txs_2 is the only pair of conflicting transactions in the game whenever txs_{p_1} was not created, it follows from Conjecture 1 that the forks may occur only when one miner successfully mines txs_1, and the other player wants to profit from mining txs_2. Thus, in the following, we study the behavior of the players whenever one of the players decides to mine txs_1.

▶ **Theorem 1.** *Assuming subgame $\Gamma_F(\mathcal{S}, r)$ in a state $\mathcal{S}$ with a single chain of length $r \leq T$, formed until round r where player N_j mined txs_1 in the last round, and txs_{p_1} is not on the chain, then the player N_j will* continue *to mine this chain unless $f_2 - f \geq f_1 + B$, even when other miners decide to fork the chain with txs_1 and* continue *mining the new subchain created during the fork.*

Proof. At every point of the game, each player N_i can choose a strategy for the remaining M rounds to collect at least $M\lambda_i(f+B)$ if he simply always chooses to mine txs_u from this point.
Thus, whenever txs_1 was just mined by N_j and $R-r$ rounds are left till the end of the game, then for some C:

- N_j chooses a strategy σ_1 where he continues the current chain of the state $\mathcal{S}$, thus:

$$\mathbb{E}_j(\sigma_1, \mathcal{S}, r+1) \geq \lambda_j(\mathsf{reward}_j(\mathcal{S}) + f + B + (R-r-1)\lambda_j(f+B)) + C.$$

- when the N_j "forks" himself, then at least one of the blocks txs_1 or txs_u will be canceled out in the final chain, therefore for any strategy σ_2 that involves forking txs_1:

$$\mathbb{E}_j(\sigma_2, \mathcal{S}, r+1) \leq \lambda_j(\mathsf{reward}_j(\mathcal{S}) - (f_1+B) + f_2 + B + (R-r-1)\lambda_j(f+B)) + C.$$

In conclusion $\mathbb{E}_j(\sigma_1, \mathcal{S}, r+1) > \mathbb{E}_j(\sigma_2, \mathcal{S}, r+1)$, unless $f_2 - f \geq f_1 + B$.

Now, since N_j that already mined txs_1 will not fork himself in the first round, it is easy to see that the same follows in the next round. ◀

5.4 Only a High-Cost Reward May Encourage Miners to Fork

The next result shows that it is not possible to credibly threaten with forks without high fees. In particular, we show that for any miner with mining power λ_j that *considers* mining the block txs_1, any forking threat in the game where txs_{p_1} was not created, will not be credible unless $f_2 - f \geq \lambda_j(f+B)$, as the miner that mined the transaction will continue to mine his transaction.

▶ **Theorem 2.** *Let $\Gamma_F(\mathcal{S}, r+1)$ be a subgame in a state $\mathcal{S}$ that contains only a single chain of length r consisting of $r-1$ unrelated transaction sets txs_u and one (just mined) txs_1 (mutually exclusive with txs_2 with $\mathsf{reward}(txs_2) > \mathsf{reward}(txs_1) > \mathsf{reward}(txs_u)$) mined by some miner N_j. The txs_{p_1} was not created and $r \leq T$. Other miners will not fork txs_1, unless $f_2 - f \geq \lambda_j(f+B)$, where λ_j is the mining power of the miner N_j.*

Proof. For brevity, the proof of this statement was moved to the full version of this paper. ◀

5.5 Without a High-Cost Reward, All Players Mine txs_1

As we already observed, once txs_1 is mined, it will not be forked unless the bribing fee is sufficient. We will show that for a sufficiently large number of rounds T, all of the players will mine transaction set txs_1 in the first round. A similar result was introduced in [23], but we prove that this result still holds in the game with forks.

▶ **Theorem 3.** *Let $\Gamma_F(\mathcal{S}, 1)$ be a subgame where none of the miners decides to create txs_{p_1} before the first round, and the bribing fee is not too high, i.e. $f_2 - f < 10^{-2}(f+B)$. What is more $f_1 > f$, and if we define $Y = \sum_{j=i:\lambda_j>0.01, f_2-f<\frac{f_1-f}{\lambda_j}}^{|N|} \lambda_j$, then T, Y are big enough, such that $(1 - 1.01(1-Y)^T) > 0$. Every miner with $\lambda_j > 0.01$ will decide to mine f_1 in the first round.*

Proof. In the game where none of the miners decides to create the transaction set txs_{p_1}, miners may choose to mine txs_u and txs_1 in all rounds, or txs_2 only after round T. Now, since the game contains only one pair of mutually exclusive transactions txs_1, txs_2 with $\mathsf{reward}(txs_2) > \mathsf{reward}(txs_1)$, then by Conjecture 1 players can start to fork only when txs_1 appears on the blockchain. What is more, since $f_2 - f < 10^{-2}(f+B)$, by Theorem 2, whenever some player with $\lambda_j > 10^{-2}$ successfully mines txs_1 in a chain of length $\leq T$, none of the players will decide to fork his block.

We prove that in the above game miners with collective mining power at least Y will decide to mine txs_1 in rounds $\{T, T-1, \ldots, 1\}$ if not mined up to this point. Let's take any miner with $\lambda_i > 10^{-2}$ that makes a decision in round $T-k$, for $k \in \{0, \ldots, T-1\}$.

As already mentioned, once he successfully mines the block txs_1, it will not be forked. In round T, whenever the block txs_1 was not mined, then the miners had only mined txs_u so far ending up in a state $\mathcal{S}_T$. Then for some constant C, the utility of the player N_i in all strategies σ_1 that choose to mine txs_1 in $\mathcal{S}_T$, and all strategies that choose to mine txs_u in $\mathcal{S}_T$:

$$\mathbb{E}_i(\sigma_1, \mathcal{S}_T, T) \geq \lambda_i(\mathsf{reward}_i(\mathcal{S}_T) + f_1 + B + \lambda_i(R - T - 1)(f + B)) + C,$$

$$\mathbb{E}_i(\sigma_2, \mathcal{S}_T, T) \geq \lambda_i(\mathsf{reward}_i(\mathcal{S}_T) + f + B + \lambda_i(f_2 + B) + (R - T - 2)\lambda_i(f + B)) + C.$$

Now, $\mathbb{E}_i(\sigma_2, \mathcal{S}_T, T) < \mathbb{E}_i(\sigma_1, \mathcal{S}_T, T)$ only if $(*) f_2 - f \geq \frac{f_1 - f}{\lambda_i}$. This implies that miners with collective mining power at least Y will prefer to mine txs_1 in this round.

In round $T - k$, where $k > 0$, whenever the block txs_1 was not mined, then the miners had only mined txs_u so far, ending up in a state $\mathcal{S}_{T-k}$. Then for some constant C, the utility of the player i in all strategies σ_1 that choose to mine txs_1 in $\mathcal{S}_T$, and all strategies that choose to mine txs_u in $\mathcal{S}_T$:

$$\mathbb{E}_i(\sigma_1, \mathcal{S}_{T-k}, T - k) = \lambda_i(\mathsf{reward}_i(\mathcal{S}_{T-k}) + f_1 + B + \lambda_i(k - 1)(f + B) + \lambda_i(R - T + k - 1)(f + B)) + C,$$

$$\mathbb{E}_i(\sigma_2, \mathcal{S}_{T-k}, T - k) \leq \lambda_i(\mathsf{reward}_i(\mathcal{S}_{T-k}) + f + B + \lambda_i(k - 1)(f + B) + \lambda_i(f_2 + B) + (R - T + k - 2)\lambda_i(f + B)) + C.$$

Similiary, the above equation implies that at least Y miners will prefer to mine txs_1 in this round.

Now, after the first round there are T rounds till the moment of mining txs_2, player's N_i benefit from mining txs_1 (with $\lambda_i > 0.01$) and not waiting for txs_2 is at least $benefit = \lambda_i(f_1 + B) - \lambda_i(1 - Y)^T(f_2 + B)$, and since $f_2 - f < 10^{-2}(f + B)$, then $benefit \geq \lambda_i(f_1 + B - (1 - Y)^T)(1.01f + 1.01B)$. Now, assuming that $f_1 > f$ we have $benefit \geq \lambda_i(f(1 - 1.01(1 - Y)^T)) + B(1 - 1.01(1 - Y)^T)$. This implies that whenever $(1 - 1.01(1 - Y)^T) > 0$, then all miners with $\lambda_i > 0.01$ will mine txs_1 in the first round. ◀

A similar results holds in any state where sufficiently large number of $T - r + 1$ rounds are left till the round T, txs_{p_1} was not created in the game (or txs_u was mined in the first round), and txs_1 was not mined yet. We leave it as a lemma without a proof.

▶ **Lemma 5.** *Given a game with forks $\Gamma_F(\mathcal{S}, r)$ with $r < T$, where none of the miners decides to create txs_{p_1} before the first round (or txs_u is mined in the first round), and the bribing fee is not too high, i.e. $f_2 - f < 10^{-2}(f + B)$ and given $Y = \sum_{j=i:\lambda_j > 0.01, f_2 - f < \frac{f_1 - f}{\lambda_j}}^{|N|} \lambda_j$; $T - r + 1, Y$ are big enough, such that $(1 - 1.01(1 - Y)^{T-r+1}) > 0$, every miner with $\lambda_j > 0.01$ will decide to mine f_1 in this round.*

5.6 Discouraging Miners to Mine txs_1

In the previous sections, we have shown that it is rather expensive to force the players not to mine txs_1 in the first round, even when the players can fork this transaction. In this section, we leverage the self-penalty mechanism introduced in Section 3. The proof is inductive, and its base case starts in round T. For each round, we first show that the miner N_s with the highest mining power λ_s will not mine txs_1, as we assume that $f_2 - f > \frac{f_1 - f}{\lambda_s}$. Next, given a

sufficiently large penalty $P > \lambda_s(f + B)$, we show that the selected player N_s will fork the transaction txs_1, once it appears on the blockchain, even though it poses a risk of losing the block reward. Finally, we show that in this round all players other than the player N_s are afraid to mine txs_1, when the self-penalty transaction is on the chain.

▶ **Theorem 4.** *Let $\Gamma_F(\mathcal{S}, 2)$ be a subgame where txs_{p_1} defined by a player N_s with mining power λ_s was mined in the first round with $P > \lambda_s(f + B)$. What is more $f_2 - f > \frac{f_1 - f}{\lambda_s}$, and $\frac{f+B}{f_1+B} > 1 - \lambda_s^2$. None of the miners will decide to mine txs_1 in rounds $2, \ldots, T$.*

Proof. For brevity, the proof of the theorem was moved to the full version of this paper. ◀

5.7 Encouraging the Strongest Miner to Use the Penalty Mechanism

Finally, we observe the benefit that comes from using the penalty mechanism. First for the miner with the strongest mining power λ_s, we observe that using the self-penalty mechanism and threatening others to mine the transaction txs_1, once it appears on the blockchain is beneficial for him whenever $f_2 - f > \frac{2\overline{f}+2(f_1-f)}{\lambda_s} + \overline{f}$. Next, for any miner with a smaller mining power, we show that merely the fact that he is threatened to mine txs_1 can force them to skip mining this transaction.

▶ **Theorem 5.** *In the game with forks Γ_F that starts with an empty state $\mathcal{S}$, whenever $f_2 - f > \frac{2\overline{f}+2(f_1-f)}{\lambda_s} + \overline{f}$, $\frac{f+B}{f_1+B} > 1 - \lambda_s^2$, $f_1 > f$, $f_{p_2} > f$, $\lambda_{min} > 0.05^{T/2}$, $f_2 - f < 10^{-2}(f + B)$ and given $Y = \sum_{j=i:\lambda_j > 0.01, f_2 - f < \frac{f_1 - f}{\lambda_j}}^{|N|} \lambda_j$, it holds that $(1 - 1.01(1 - (1 - Y)^{T/2}) > 0$, the dominating strategy for all players in the game Γ_F is is to mine txs_{p_1} with $P > \lambda_s(f + B)$ created in the first round by the strongest player N_s with the mining power λ_s, then mine txs_u until round T, then txs_2, txs_{p_2}, and txs_u until the end.*

Proof. By Theorems 3 and 4, utility of the player N_s that chooses to create txs_{p_1}, txs_{p_2} and mine txs_{p_1} in the first round[6] (strategy σ_p) is at least:

$$\mathbb{E}_s(\sigma_p, \mathcal{S}, 1) \geq -\lambda_{p_1} c_{p_1} \overline{f_{p_1}} + \lambda_s((m - c_{p_1})\overline{f} + B) + (\lambda_{p_1} + \lambda_s)F_2' + (1 - \lambda_s - \lambda_{p_1})F_1',$$

where $F_1' = (\lambda_s(T - 1)(f + B) + \lambda_s(R - T)(f + B))$,

$$F_2' = (\lambda_s((T - 1)(f + B)) + \lambda_s(f_2 + B) - \lambda_{p_2} c_{p_2} \overline{f}_{p_2} + \lambda_s((m - c_{p_2})\overline{f} + B) + \lambda_i(R - T - 2)(f + B)).$$

Recall that we assume that all players with mining power less than 1% have collective power at most 5%. As the players with mining power more than 0.01 will prefer to mine txs_1 in the first place when txs_{p_1} is not created, the utility of the player N_s that does not decide to create txs_{p_1} (strategy σ_1) is at most (by Lemma 5 and the Theorem 4):

$$\mathbb{E}_s(\sigma_1, \mathcal{S}, 1) \leq ((1 - 0.05^{T/2})F_1 + 0.05^{T/2}F_2),$$

where $F_1 = F_1' + \lambda_s(f_1 + B)$, $F_2 = F_2' + \lambda_s(f + B)$. Further, if $\lambda_{p_1} c_{p_1} \overline{f}_{p_1} + \lambda_s(f + B) - \lambda_s c_{p_1} \overline{f} + F_1' + (\lambda_{p_1} + \lambda_s)(F_2' - F_1') > F_1 + 0.05^{T/2}(F_2 - F_1)$, then $\mathbb{E}_s(\sigma_p, \mathcal{S}, 1) > \mathbb{E}_s(\sigma_1, \mathcal{S}, 1)$. This condition holds whenever:

$$(\lambda_{p_1} + \lambda_s)[\lambda_s f_2 - \lambda_s f_1] > 0.05^{T/2}[\lambda_s f_2 - \lambda_s f_1] + \lambda_{p_1} c_{p_1} \overline{f}_{p_1} + \lambda_s c_{p_1} \overline{f} + (\lambda_{p_1} + \lambda_s)(\lambda_{p_2} c_{p_2} \overline{f}_{p_2} + \lambda_s c_{p_2} \overline{f}) + \lambda_s(f_1 - f).$$

[6] In the analysis we omit the strategy where the player N_s creates txs_{p_1}, txs_{p_2} and does not decide to mine txs_{p_1}

What concludes that the following bribe is enough to encourage the strong miner to wait for txs_2:

$$f_2 - f > \frac{c_{p_1}\overline{f}_{p_1} + c_{p_1}\overline{f} + c_{p_2}\overline{f}_{p_2} + f_1 - f}{\lambda_s - 0.05^{T/2}} + c_{p_2}\overline{f}.$$

Now, if the txs_{p_1}, txs_{p_2} are created then every player N_i other than the player N_s may mine txs_{p_1}, once it is published (strategy $\sigma_{p_1^*}$). When txs_{p_1} is successfully mined in the first round, then all miners will be encouraged to wait until txs_2 may be mined after the T'th round. $1 - \lambda_{p_1} - \lambda_i$ miners may decide to mine txs_u (or txs_1) in the first round. In this case, when the txs_u is mined, all other players will be able to mine at least $f + B$ for the rest of the rounds $\mathbb{E}_i(\sigma_{p_1^*}, \mathcal{S}, 1) \geq \lambda_i(f_{p_1} + B) + (\lambda_{p_1} + \lambda_i)F_2 + (1 - \lambda_{p_1} - \lambda_i)F$, where $F_2 = (T-1)\lambda_i(f+B) + \lambda_i(f_2 + B) + \lambda_i(f_{p_2} + B) + (R - T - 2)\lambda_i(f + B)$ and $F = (R-1)\lambda_i(f+B)$.

On other hand the players may first decide to mine either txs_u or txs_1 in the first round (strategy σ_{1*}). In the worst case scenario the block with txs_1 is not forked. What is more, whenever $1 - \lambda_{p_1}$ miners decide to mine txs_u in the first round, then all miners with mining power more than 0.01 will make an attempt to mine txs_1. In conclusion, by Lemma 5 and the Theorem 4:

$$\mathbb{E}_i(\sigma_{1*}, \mathcal{S}, 1) \leq \lambda_i(f_1 + B) + \lambda_{p_1}F_2 + (1 - \lambda_{p_1} - \lambda_1)((1 - 0.05^{T/2})F + 0.05^{T/2}F_2),$$

where F and F_2 are defined as previously. $\mathbb{E}_i(\sigma_{p_1*}, \mathcal{S}, 1) > \mathbb{E}_i(\sigma_{1*}, \mathcal{S}, 1)$ holds whenever:

$$\lambda_i(f_{p_1} + B) + \lambda_i(F_2 - F) > \lambda_i(f_1 + B) + (1 - \lambda_{p_1})0.05^{T/2}(F_2 - F)$$

Which holds for any $f_{p_1} \geq f_1$ and $\lambda_i > 0.05^{T/2}$.
Now, by setting $c_{p_1} = 1, c_{p_2} = 1$, $\overline{f_{p_1}} = f_1 - f$, $\overline{f_{p_2}} = \overline{f}$, we get a condition $f_2 - f > \frac{2\overline{f} + 2(f_1 - f)}{\lambda_s - 0.05^{T/2}} + \overline{f}$, what for $\lambda_s \approx 20\%$ and sufficiently large $T/2$ concludes $f_2 - f \gtrsim \frac{2\overline{f} + 2(f_1 - f)}{\lambda_s} + \overline{f}$. ◀

6 Example Evaluation

Using the real-world data analysis of Bitcoin fees and hashpower distribution in major PoW blockchains (see discussion in the full version of this paper), we visualize the improvement our bound $f_2 - f > \frac{2\overline{f} + 2(f_1 - f)}{\lambda_s} + \overline{f}$ brings compared to the previous result from [6], namely $f_2 - f \geq \frac{f_1 - f}{\lambda_{min}}$. Additionally, the Theorem 5 requires that $f_2 - f < 10^{-2}(f + B)$ and there exists a player N_j with mining power $\lambda_j > 0.01$ for which $f_2 - f < \frac{f_1 - f}{\lambda_j}$.

For example, let us assume that $f_1 - f \approx \overline{f}$, and set $T > 110$. Now, since λ_{min} can be fairly estimated to be $\lambda_{min} < 10^{-12}$, we can see that the attack without forking threats could cost in practice around $10^{12}\overline{f}$. On the other hand, the new bound requires only $f_2 - f > \frac{2\overline{f} + 2(f_1 - f)}{\lambda_s} + \overline{f}$, for $\lambda_s \approx 0.2$, this costs around $f_2 - f > 21\overline{f}$. The only condition left is that for some miner with $\lambda_j > 0.01$, the following condition must hold $f_2 - f < \frac{f_1 - f}{\lambda_j}$, but the data shows that miners that control approximately $1.5\% - 2\%$ of the total mining power usually exist, thus for a miner with mining power $1\% < \lambda_j < 2\%$ it holds that $f_2 - f < 50 \cdot \overline{f}$. In summary, if we take any f_2 that is larger than f by 21 up to 50 times, then the default strategy for all miners is to wait for the bribing transaction.

7 Related Work and Countermeasures

In the landscape of constructing financially stable systems on blockchain [21, 7], our work falls into the class of incentive manipulation attacks which have been widely applied to undermine blockchain's security assumptions [22]. To the best of our knowledge, we are the first to combine feather forking attacks [8] with timelock bribing attacks on payment channel networks and to achieve a bribing cost that is approximately only constant times larger than the cost of an average transaction fee.

Incentive manipulation attacks on timelocked puzzles were introduced with the so-called timelock bribing attack [23]. Later, Avarikioti et al. [6] applied timelock bribing attacks in payment channel networks, such as the Lightning Network and Duplex Micropayment Channels, and proposed countermeasures. Our work extends [6], modifying the timelock bribing attack for payment channels to facilitate a miner bribing strategy that incorporates feather forking. As a result, our work reduces the cost of bribing attacks significantly in comparison to [6], i.e., the cost is now inversely proportional to the mining power of the largest miner instead of the smallest miner which results in at least 1000 times smaller bribes. Our model is similar to the one in [17] that introduced forks, but we were able to craft reasonable assumptions for the PoW blockchains which secured a *unique* NE solution. In particular, we restrict the strategy of the miner by forbidding him to conduct shallow forks and allowing him to fork only in a case when the player strictly profits from conducting the fork action (compare Sections 4.1.7, 4.1.8).

The bribing strategies for the payment channels are similar in their nature to the bribing strategies for the HTLC mechanism. Perhaps the closest to our work is [31], where the authors introduced a way to bribe HTLCs, leveraging the power of smart contracts and feather forking. The cost of the attack in [31] is, however, proportional to the sum of the fees ($\gtrsim \sum_{i=1}^{T} f \cdot \lambda_{max}$) of all blocks before the deadline T. In contrast, we achieve a cost proportional to the cost of fees of a single block ($\gtrsim \frac{f_1 - f}{\lambda_S}$).

Furthermore, MAD-HTLC [29] underlined the vulnerability of HTLCs to bribing attacks, achieving the same attack cost as [6], specifically $\approx \frac{f_1 - f}{\lambda_{min}}$. MAD-HTLC presupposes that the minimum fraction of mining power controlled by a single user, λ_{min}, is at least 0.01, to achieve low bribing costs. This is, however, an impractical assumption, as the analysis of the actual data (see discussion in the full version of this paper) shows that λ_{min} can be reasonably estimated to be less than 10^{-12}, making the bribing attack exceedingly expensive. The reduction of the bribing costs *Bribe & Fork* achieves in comparison to MAD-HTLC is similar to that of [6] analyzed above.

MAD-HTLC additionally proposed a countermeasure for bribing attacks where miners are allowed to claim the locked coins in the HLTC in case a party misbehaves, similar to [6]. Later, He-HTLC [30] pointed out that MAD-HTLC is susceptible to counter-bribing attacks. In particular, one party may (proactively) collude with the miners to cooperatively steal the coins of the counterparty in the MAD-HTLC construction. He-HTLC also proposed a modification on MAD-HTLC to mitigate the counter-bribing attack: now the coins are partially burned in case of fraud instead of being fully awarded to the miners. Recently, Rapidash [9] revisited the counter-bribing attack and proposed yet another improvement on He-HTLC. These works are orthogonal to ours as the proposed attacks apply only to the specific MAD-HTLC construction and not to Lighting Channels that are the focus of this work. Furthermore, our focus is not on designing countermeasures against timelocked bribing attacks. Instead, we demonstrate how employing feather forking can make timelocked bribing attacks very cheap for the attacker, therefore highlighting the need for robust mitigating strategies.

Nonetheless, it is crucial to acknowledge that the previously mentioned countermeasures can be used to defend against *Bribe & Fork*– inheriting their respective vulnerabilities. For example, one can employ the mitigation technique for timelocked bribing on the Lighting Network proposed by Avarikioti et al. [6]. In our model, this countermeasure ensures that announcing txs_2 also involves revealing a secret that anyone can use to claim the money before time T. This implies that if txs_2 is announced in the mining pool before time T, all the money to be collected only after time T can be immediately claimed by another party. We assert, without proof, that the same countermeasure mechanism remains effective even in a model that considers forks. Intuitively, the "strong" miner in our analysis does not benefit from waiting for the bribing transaction if it is not announced, thus preventing the creation of the self-penalty transaction. Conversely, if the transaction is announced and the secret is revealed, any (winning) miner could claim the reward.

8 Conclusions and Future Work

In conclusion, our work reexamines the vulnerability of PCNs to bribing attacks and introduces a modified attack leveraging the threat of forking. We introduce a formal model of a mining game with forking extending the conditionally timelocked games introduced by Avarikioti et al. [6]. In particular, in our extended model, miners not only choose which transactions to mine in each round but also decide whether to continue existing chains or initiate forks. In this model, we demonstrate that the cost of the bribing attack can be significantly reduced compared to the previous analysis. In more detail, it can be reduced from $\frac{f_1 - f}{\lambda_{min}}$ to approximately $\frac{2\overline{f} + 2(f_1 - f)}{\lambda_s}$, where $\overline{f}$ represents the cost of an average fee for a single transaction and λ_s denotes the reduction factor compared to significantly smaller λ_{min} calculated in previous work [6]. To validate our findings, we empirically analyze the historical data of real-world blockchain implementations. This analysis confirms that staging a bribing attack on a PCN is significantly less costly (approximately 125$) than considered previously.

The results of our study have implications for the design and implementation of PCNs, as well as for the broader applications of timelocked contracts, e.g., atomic swaps. Our findings underscore the need for proactive measures to mitigate the risk of bribing attacks.

Possible avenues for future research include exploring whether our penalty mechanism implementation can be implemented without the honest majority assumption or whether our results still hold in the presence of more general abandon rules. Another interesting question is whether our results extend in a Proof-of-Stake setting.

References

1 Lukas Aumayr, Ozgur Ersoy, Andreas Erwig, Sebastian Faust, Kristina Hostakova, Matteo Maffei, Pedro Moreno-Sanchez, and Sabrina Riahi. Generalized bitcoin-compatible channels. *Cryptology ePrint Archive*, 2020:476, 2020. URL: `https://eprint.iacr.org/2020/476`.

2 Lukas Aumayr, Ozgur Ersoy, Andreas Erwig, Sebastian Faust, Kristina Hostakova, Matteo Maffei, Pedro Moreno-Sanchez, and Sabrina Riahi. Bitcoin-compatible virtual channels. In *IEEE Symposium on Security and Privacy*, 2021. URL: `https://eprint.iacr.org/2020/554.pdf`.

3 Lukas Aumayr, Sri AravindaKrishnan Thyagarajan, Giulio Malavolta, Pedro Moreno-Sanchez, and Matteo Maffei. Sleepy channels: Bi-directional payment channels without watchtowers. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, pages 179–192, 2022.

4 Zeta Avarikioti, Eleftherios Kokoris Kogias, Roger Wattenhofer, and Dionysis Zindros. Brick: Asynchronous incentive-compatible payment channels. In *International Conference on Financial Cryptography and Data Security*, 2021. URL: `https://fc21.ifca.ai/preproceedings/50.pdf`.

5 Zeta Avarikioti, Orestis S. T. Litos, and Roger Wattenhofer. Cerberus channels: Incentivizing watchtowers for bitcoin. In *International Conference on Financial Cryptography and Data Security*, pages 346–366. Springer, 2020. URL: `https://link.springer.com/chapter/10.1007/978-3-030-60276-7_18`.

6 Zeta Avarikioti and Orfeas Stefanos Thyfronitis Litos. Suborn channels: Incentives against timelock bribes. In *Financial Cryptography and Data Security - 26th International Conference, FC 2022, Grenada, May 2-6, 2022, Revised Selected Papers*, volume 13411 of *Lecture Notes in Computer Science*, pages 488–511. Springer, 2022. `doi:10.1007/978-3-031-18283-9_24`.

7 Iddo Bentov and Ranjit Kumaresan. How to use bitcoin to design fair protocols. In *Advances in Cryptology - CRYPTO 2014 - 34th Annual Cryptology Conference*, volume 8617 of *Lecture Notes in Computer Science*, pages 421–439. Springer, 2014. `doi:10.1007/978-3-662-44381-1_24`.

8 Joseph Bonneau, Andrew Miller, Jeremy Clark, Arvind Narayanan, Joshua A. Kroll, and Edward W. Felten. Sok: Research perspectives and challenges for bitcoin and cryptocurrencies. In *2015 IEEE Symposium on Security and Privacy*, pages 104–121, 2015. `doi:10.1109/SP.2015.14`.

9 Hao Chung, Elisaweta Masserova, Elaine Shi, and Sri AravindaKrishnan Thyagarajan. Rapidash: Foundations of side-contract-resilient fair exchange. Cryptology ePrint Archive, Paper 2022/1063, 2022. URL: `https://eprint.iacr.org/2022/1063`.

10 Kyle Croman, Christian Decker, Ittay Eyal, Adem Efe Gencer, Ari Juels, Ahmed Kosba, Andrew Miller, Prateek Saxena, Elaine Shi, Emin Gün Sirer, Dawn Song, and Roger Wattenhofer. On scaling decentralized blockchains. In *International Conference on Financial Cryptography and Data Security*, pages 106–125. Springer, 2016.

11 Christian Decker, Rusty Russell, and Olaoluwa Osuntokun. eltoo: A simple layer2 protocol for bitcoin. `https://blockstream.com/eltoo.pdf`, 2019.

12 Christian Decker and Roger Wattenhofer. A fast and scalable payment network with bitcoin duplex micropayment channels. In *Stabilization, Safety, and Security of Distributed Systems*, pages 3–18. Springer, 2015.

13 Stefan Dziembowski, Lisa Eckey, Sebastian Faust, and Daniel Malinowski. Perun: Virtual payment hubs over cryptocurrencies. In *2019 IEEE Symposium on Security and Privacy (SP)*, pages 344–361. IEEE, 2019.

14 Stefan Dziembowski, Sebastian Faust, and Kristína Hostáková. General state channel networks. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pages 949–966. ACM, 2018.

15 Lewis Gudgeon, Pedro Moreno-Sanchez, Stefanie Roos, Patrick McCorry, and Arthur Gervais. Sok: Layer-two blockchain protocols. In Joseph Bonneau and Nadia Heninger, editors, *Financial Cryptography and Data Security - 24th International Conference, FC 2020, Kota Kinabalu, Malaysia, February 10-14, 2020 Revised Selected Papers*, volume 12059 of *Lecture Notes in Computer Science*, pages 201–226. Springer, 2020. `doi:10.1007/978-3-030-51280-4_12`.

16 Michael Jourenko, Nicolas Larangeira, and Koji Tanaka. Lightweight virtual payment channels. In *Cryptology and Network Security*, pages 365–384. Springer International Publishing, 2020.

17 Dimitris Karakostas, Aggelos Kiayias, and Thomas Zacharias. Blockchain bribing attacks and the efficacy of counterincentives, 2024. `arXiv:2402.06352`.

18 Sishan Long, Soumya Basu, and Emin Gün Sirer. Measuring miner decentralization in proof-of-work blockchains. *arXiv preprint arXiv:2203.16058*, 2022.

19 Antonio Magnani, Luca Calderoni, and Paolo Palmieri. Feather forking as a positive force: incentivising green energy production in a blockchain-based smart grid. In *Proceedings of the 1st Workshop on Cryptocurrencies and Blockchains for Distributed Systems*, pages 99–104, 2018.

20 Andrew Miller. Feather-forks: enforcing a blacklist with sub-50% hash power. URL: https://bitcointalk.org/index.php?topic=312668.0.

21 Andrew Miller and Iddo Bentov. Zero-collateral lotteries in bitcoin and ethereum, 2017. arXiv:1612.05390.

22 Michael Mirkin, Yan Ji, Jonathan Pang, Ariah Klages-Mundt, Ittay Eyal, and Ari Juels. Bdos: Blockchain denial of service, 2020. arXiv:1912.07497.

23 Tejaswi Nadahalli, Majid Khabbazian, and Roger Wattenhofer. Timelocked bribing. In *Financial Cryptography and Data Security - 25th International Conference, FC*, volume 12674 of *Lecture Notes in Computer Science*, pages 53–72. Springer, 2021. doi:10.1007/978-3-662-64322-8_3.

24 Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system, 2008. URL: http://bitcoin.org/bitcoin.pdf.

25 Joseph Poon and Thaddeus Dryja. The bitcoin lightning network: Scalable off-chain instant payments. *https://lightning.network/lightning-network-paper.pdf*, January 2016.

26 Yahya Shahsavari, Kaiwen Zhang, and Chamseddine Talhi. A theoretical model for fork analysis in the bitcoin network. In *IEEE International Conference on Blockchain, Blockchain 2019, Atlanta, GA, USA, July 14-17, 2019*, July 2019. doi:10.1109/Blockchain.2019.00038.

27 Santhi Shalini and H Santhi. A survey on various attacks in bitcoin and cryptocurrency. In *2019 International Conference on Communication and Signal Processing (ICCSP)*, pages 0220–0224. IEEE, 2019.

28 Joseph Spilman. Anti dos for tx replacement. https://lists.linuxfoundation.org/pipermail/bitcoin-dev/2013-April/002433.html, 2013. Accessed: 2020-11-22.

29 Itay Tsabary, Matan Yechieli, Alex Manuskin, and Ittay Eyal. MAD-HTLC: because HTLC is crazy-cheap to attack. In *42nd IEEE Symposium on Security and Privacy, SP*, pages 1230–1248. IEEE, 2021. doi:10.1109/SP40001.2021.00080.

30 Sarisht Wadhwa, Jannis Stoeter, Fan Zhang, and Kartik Nayak. He-htlc: Revisiting incentives in HTLC. In *30th Annual Network and Distributed System Security Symposium, NDSS*. The Internet Society, 2023. URL: https://www.ndss-symposium.org/ndss-paper/he-htlc-revisiting-incentives-in-htlc/.

31 Fredrik Winzer, Benjamin Herd, and Sebastian Faust. Temporary censorship attacks in the presence of rational miners. In *2019 IEEE European Symposium on Security and Privacy Workshops, EuroS&P Workshops*, pages 357–366. IEEE, 2019. doi:10.1109/EuroSPW.2019.00046.

32 Gavin Wood. Ethereum: A secure decentralised generalised transaction ledger. *Ethereum Project Yellow Paper*, 2014.