Suborn Channels: Incentives Against Timelock Bribes

Zeta Avarikioti[1] and Orfeas Stefanos Thyfronitis Litos[2⋆]

[1] IST Austria zetavar@ethz.ch
[2] TU Darmstadt orfeas.thyfronitis@tu-darmstadt.de

Abstract. As the Bitcoin mining landscape becomes more competitive, analyzing potential attacks under the assumption of rational miners becomes increasingly relevant. In the rational setting, blockchain users can bribe miners to reap an unfair benefit. Established protocols such as Duplex Micropayment Channels and Lightning Channels are susceptible to bribery, which upends their financial guarantees. Indeed, we prove that in a two-party contract in which the honest party can spend an output right away, whereas the malicious can only spend the same output after a timelock, the latter party can promise a high fee to the miners, who then intentionally ignore the transaction of the honest party in anticipation of the higher fee. This effectively prevents a valid transaction from ever entering the blockchain, resulting in potentially severe financial losses for the honest and considerable gains for the malicious party.
We expand previous results on timelock bribes to more realistic blockchains, proving that a general class of contracts are susceptible. We then apply our results to Duplex Micropayment Channels and Lightning Channels, providing exact bounds on their safe operating region. Furthermore, we enhance the Bitcoin Script of Duplex Micropayment Channels so that the coins of a party that attempts to bribe are given to the miners as fees, therefore effectively disincentivizing bribes. Our solution, named Suborn channels, is implemented as a proof-of-concept. We also propose a small change to Lightning Channels that achieves a similar effect. Moreover, we formally express the exact circumstances under which our two proposals ensure alignment of miner incentives with the prescribed protocol outcome.

Keywords: Bitcoin · Security · Layer 2 · Payment channels · Lightning network · Incentives · Bribing

1 Introduction

Blockchains like Bitcoin [23] and Ethereum [28] reformed the financial landscape. Nevertheless, blockchains scale poorly in comparison to conventional centralized payment systems [9]. One of the major approaches to alleviate the scalability issue of blockchains is *payment channel networks* (PCNs).

⋆ Work done while the author was at the University of Edinburgh.

Payment channels allow two parties to lock funds on the blockchain and thereafter securely transact off-chain. A number of PCN proposals exist [2–8, 10–15, 17, 21, 24, 25], each improving on previous designs, exploiting features of different blockchains or balancing various trade-offs differently. Two of the earliest PCNs are the *Lightning Network* (LN) [24] and the *Duplex Micropayment Channels* (DMC) [11], both applicable on Bitcoin.

As Bitcoin implements a (crypto-)currency, financial incentives are critical to the security of the protocol. These financial incentives naturally transfer to the off-chain network operating on top of Bitcoin, e.g., DMC or LN, since the off-chain network also involves locked cryptocurrency funds. As a result, several bribing attacks have been proposed on PCNs [22, 26, 27]. In this work, we focus on a specific type of bribing attacks, the so-called *timelocked bribes*: a briber pays the miners to include the briber's transaction which will only be valid in the future, and *not* include a conflicting but currently valid transaction from an honest party. This attack affects directly the security of most PCNs.

The success of a timelock bribing attack is conditional on several variables. Determining those variables and therefore the parameter regions in which parties can transact securely against a briber is a challenging task. Furthermore, we ask whether expanding these safe regions is possible, as this would imply a wider functioning area for payment channels. In this work, we take up these challenges.

Our Contributions. We first formally describe the dynamics governing miners' choices on whether to mine a future transaction with a high fee or a currently spendable but conflicting transaction with a smaller fee. To this end, we perform a game-theoretic analysis in Section 3. We then formulate and prove in Theorem 1 under which circumstances it is a *strictly dominant strategy* for miners to ignore the currently spendable transaction in favor of the future one. This theorem generalizes the incentive analysis performed in [26] to blockchains with more than 1 transaction per block and to a more generic smart contract than HTLC [11]. At a high level, miners prefer the future transaction if it offers a very high fee (a.k.a. bribe) compared to the currently spendable transaction. The exact bound depends also on the fees paid by ordinary transactions and the mining power of the weakest miner but, somewhat surprisingly, is independent of the length of the timelock for large enough bribes.

In Section 4 we apply our theorem to DMC, providing exact bounds on the cases in which a timelock bribe is possible. Subsequently, we modify the DMC protocol and propose a new scheme which we term Suborn channels in order to greatly expand those bounds. The core idea is that Suborn channels allow miners to claim the coins the briber owns in the channel when the honest party proves the briber cheated. The exact script for Suborn channels is provided as well, along with its proof of concept implementation[3].

Lastly, we apply our theorem to LN, characterizing exactly when a timelock bribe is beneficial. We then propose a straightforward change to the protocol that completely nullifies timelock bribes; we simply increase the transaction fees

[3] `https://gitlab.com/fc22-submission-69/suborn`

to include the coins owned by the briber. We further analyze the circumstances under which our proposal would not cost money to the honest party and recommend how the honest party can avoid cost-inducing situations entirely. We note that no change in LN Script is necessary for implementing our proposal.

2 Background and Notation

2.1 Bitcoin

Bitcoin users publish *transactions*, which are temporarily stored by miners. Each miner composes a *block* that consists of valid transactions. Miners compete with each other in a lottery which periodically selects a winner with probability proportional to their *mining power* – a quantity that we assume is constant and common knowledge among participants. *Mining* is a resource-intensive process that each miner performs locally. The winning miner gets their block included in the blockchain and gains a (constant) *block reward* plus the sum of the fees of all included transactions; thus rational miners attempt to maximize their received fees. Then miners verify that all transactions in the new block are valid, compose a new block compatible with all past ones (including the new valid block they just received) and attempt to win the next lottery. In case a miner receives two or more conflicting blocks of the same *height* (a.k.a. when they encounter a *fork*), they can mine on top of any one. With high probability one of the forks will eventually overtake the others by accumulating more blocks, so all miners will switch to the longest chain, dropping the other forks and resolving the conflict.

Smart contracts. Blockchains like Bitcoin [23] and Ethereum [28] enable *smart contracts*, i.e., programmable scripts that attach a wide variety of rules which must be satisfied in order to spend coins. In Bitcoin, coins are attached to *transaction outputs*, which in turn are locked with a specific script. Bitcoin smart contracts commonly employ the use of multisignatures, timelocks, and hashlocks.

The most commonly used smart contract requires a single *signature* from a specific public key: such a contract ensures that the coins of interest are exclusively owned by whoever knows the associated private key. An m-of-n *multisignature* is a contract that demands at least m signatures which correspond to any m of the n predefined public keys.

Hashlocks are another type of contract, available also in Bitcoin Script. If a transaction output is hashlocked, it requires the pre-image of the specific hash to become valid and thus spendable. For instance, suppose $h(s)$ denotes the hash of a secret s. If an output is hashlocked with $h(s)$, it is valid only if the secret s is revealed.

Timelocked outputs can only be spent after a specified time in the future. One of the simplest practical smart contracts that uses timelocks is the *conditional timelock*, which allows the associated output to be spent either with the signature of party P_1 right away, or with the signature of party P_2 after a timelock – possibly additional requirements encumber one or both spending methods, e.g. hashlocks or multisigs.

2.2 Payment channels

The core idea behind payment channels is the same across different constructions: two parties may lock coins on an escrow on the blockchain, or a so-called channel, and then perform arbitrarily many transactions off-chain. Each off-chain transaction is a signed message that depicts the current balance of coins between the parties. Any party can close the channel at any time, either in collaboration with the counterparty, or unilaterally by publishing the last message signed by both parties. Therefore, the blockchain is only used to open and close the channel, and to resolve potential conflicts between parties. The conflict resolution mechanism differs significantly among channel constructions. Please see [29] for a survey of PCNs.

In this work, after establishing a general result for conditionally timelocked contracts, we apply our results to two specific PCNs: DMC [11] and LN [24]. We now describe these constructions, excluding their HTLCs.

DMC overview. At a high level, a DMC between parties P_1 and P_2 works as follows: At first the parties agree on a *setup* transaction, which spends their initial coins and moves them to an output locked with a 2-*of*-2 *multisig*. They then establish a series of *opt-in* transactions. These transactions form a chain of a pre-agreed length and are all timelocked until a common pre-agreed future time $T_{\max}$. The first transaction consumes the setup transaction output and provides a similar 2-of-2 multisig output. Each subsequent transaction consumes the output of the previous opt-in transaction and provides a similar 2-of-2 multisig output, with the exception of the last one. This opt-in transaction has *two* 2-of-2 multisig outputs instead, each carrying coins equal to one party's initial coins.

Each of the two last outputs constitutes the *setup output* for a *simple micropayment channel* (not to be confused with the setup transaction of the DMC itself). A simple channel can only facilitate payments in *one direction*, so there is one channel for each direction. We here explain briefly how the channel in which P_2 pays P_1 functions; the other one is symmetric. The channel starts off with P_1 and P_2 agreeing on a *refund* transaction that is timelocked until $T_{\max}$, spends the setup output and provides one output that carries P_2's initial coins that are spendable by P_2 alone. Once both parties know the relevant opt-in and refund transactions along with the necessary signatures by their counterparty, they only need to put the DMC setup transaction on-chain to open the DMC. When P_2 holds c_2 and wants to pay δ coins to P_1, who holds c_1 coins, P_2 signs and sends to P_1 an *update* transaction which has no timelock, spends the setup output and has one output per party; P_1's output carries $c_1 + \delta$ and P_2's carries $c_2 - \delta$ coins. The balances c_1, c_2 are as in the last update transaction if any, otherwise as in the refund. Note that P_1 can put on-chain any update transaction if needed, so he prefers the latest update transaction in which he has most coins – this mechanism is called *replace by incentive*.

Due to their unidirectional nature, one of the two simple channels may eventually get depleted. In such a case, the parties *invalidate* them along with the last opt-in transaction by creating a new competing opt-in transaction with a

lower timelock. This opt-in transaction provides two new simple micropayment channels, each initially containing the sum of the payer's coins in the just invalidated simple channels. In case the timelock of the new opt-in transaction is smaller than a pre-agreed value $T_{\min}$, the two parties replace the last two opt-in transactions instead. Both new opt-in transactions use the same timelock, which is lower than the timelock of the second-last opt-in transaction in the old chain. The same replacement logic, called *replace by timelock*, can be extended backwards to the entire length of the opt-in transactions' chain. This way, an *invalidation tree* is created that consists of opt-in transactions as non-leaf nodes and pairs of simple micropayment channels as leaves.

When an invalidation tree is itself depleted, cooperative parties can *refresh* their DMC and obtain a new invalidation tree with a single on-chain transaction. Similarly, cooperative parties can close their channel with a single on-chain *teardown* transaction.

The above construction depicted in Figure 1, ensures that an honest party can always retrieve its coins unilaterally by publishing the opt-in transactions of the latest branch when their timelock expires, even if the counterparty stops cooperating in arbitrary ways. This security guarantee holds only if a transaction with a lower timelock is always included on-chain when competing only with transactions with a higher timelock. As we see in this work however, this assumption does not always hold.

LN overview. LN bases its functionality on an entirely different construction. A central premise is that, in contrast to DMC, not all of the transactions stored locally by the two parties are the same: some have differing scripts.

The two parties first negotiate the *funding* output, which carries all of the channel's coins in a 2-of-2 multisig. They then build a pair of *commitment* transactions, one for each party, each of which can spend the funding output. P_1's commitment transaction is signed by P_2 and has two outputs. One carries P_2's initial coins and can be spent with a simple signature by P_2. The other carries P_1's initial coins and can be spent in one of two ways: either with a signature by P_1 after a pre-agreed timelock (the *honest* spending method), or with a signature by a special *revocation* key that is generated by the two parties cooperatively (the *punishment* spending method). The latter private key has the unusual property that it can remain unknown to both parties while allowing the corresponding public key to be computed cooperatively: Each party has a *secret share*, from which it can generate a *public share*. The two secret shares combine to the private key, whereas the two public ones combine to the public key, thus the two parties can cooperatively derive the public key without disclosing their secret shares. This construction is formalized and proved secure in [18]. P_2's commitment transaction is symmetric. Once each party holds its first commitment transaction, they can put the funding output on-chain to open the channel.

Conceptually, an off-chain payment is performed in two steps. First, the two parties generate and sign a new pair of commitment transactions of which the outputs pay out the newly agreed coins to each party. New revocation keys

are used. Second, each party sends to its counterparty the secret share of the revocation key used in the previous commitment transaction, revoking the latter. This way, if a party publishes an old commitment transaction, the counterparty can take all coins in the channel as punishment as long as it uses the punishment spending method before the timelock of the honest spending method expires. Note that the actual update procedure is slightly more complicated than in this simplified, but morally correct, description.

Lastly, the two parties can cooperatively close the channel by building a single *closing* transaction that spends the funding output and gives each party its coins without a timelock.

The LN construction ensures that an honest party which checks the blockchain periodically can always unilaterally retrieve its coins or more, either by publishing its latest commitment transaction and waiting for its timelock to expire or by punishing its counterparty in case the latter published a revoked commitment transaction. This guarantee though holds only under the assumption that a non-timelocked transaction which competes only with a timelocked one can always go on-chain. As we mentioned however, this assumption is violated under certain circumstances. The LN construction is illustrated in Figure 2.

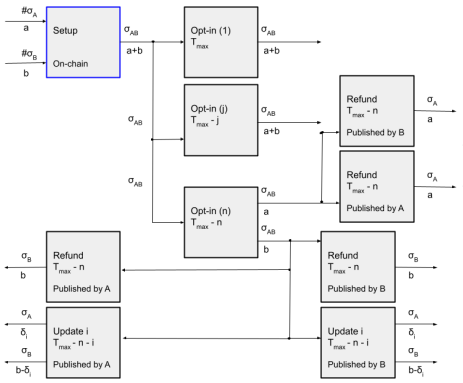

Fig. 1: Duplex Micropayment Channels

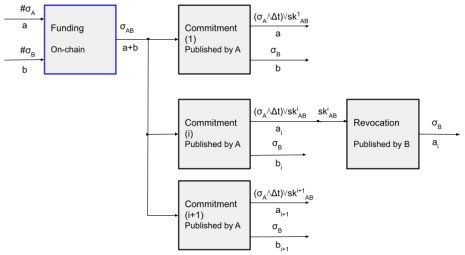

Fig. 2: Lightning Channels

3 Incentive Analysis

3.1 Model

As in [26], we assume that all $n \geq 2$ miners are rational and each has a proportion $0 < \lambda_i < 1$ of the total mining power, constant throughout the execution. Block rewards are ignored to simplify the analysis, but would not change our results as long as they remain constant throughout the time frame of interest. Let $\lambda_{\min} = \min_{i \in [n]} \lambda_i$. We assume that each block is comprised of a fixed number of transactions N (as opposed to, e.g., a fixed block size like Bitcoin or a

fixed gas limit like Ethereum). The game evolves in rounds. At the beginning of each round, each miner decides on a set of transactions to include in her block. Subsequently a single miner is chosen at random according to the mining power distribution, her block is appended to the blockchain, the included transaction consumes its designated UTXO(s) and potentially provides one or more new unspent outputs, the winning miner obtains the fee of the transaction and all miners learn who won. This completes the round and miners attempt to mine a new block.

The utility of each miner u_i is equal to the sum of fees she obtains over all game rounds – we restrict our attention to games with a finite number of rounds, say T. All transactions that may be included in a block are publicly known and carry a constant fee f, unless stated otherwise. The mining power distribution along with the rest of the model discussed above is common knowledge.

In each round $k \in [T]$, the i-th miner employs a *strategy* σ_i^k that takes values in the set Σ_i^k, which consists of the transactions that the miner chooses for its block at round k. A *strategy profile* for round k is the tuple of the strategies of all miners for round k and is denoted with $\sigma^k = (\sigma_1^k, \ldots, \sigma_n^k) \in \Sigma^k$. A strategy profile for rounds from k_1 to k_2 is the concatenation of the strategy profiles of rounds k_1 to k_2 and is denoted with $\sigma^{k_1 \ldots k_2} = \sigma^{k_1} \ldots \sigma^{k_2} \in \Sigma^{k_1 \ldots k_2}$. The Nash equilibrium strategy profile for rounds from k_1 to k_2 is denoted with $\overline{\sigma}^{k_1 \ldots k_2}$. Note that the latter constitutes a slight abuse of notation since the strategy profiles in rounds after k_2 may in principle influence what is the Nash equilibrium of rounds up to k_2, but in our games of interest every future round has exactly one Nash equilibrium that is also the unique strictly dominant strategy profile (ignoring inclusion of a different set of transactions unrelated to the conditionally timelocked output O of interest, c.f. Definition 1, as such differences do not change the utility), thus no problem arises. A strategy profile for all rounds is denoted with $\sigma = \sigma^{1 \ldots T} \in \Sigma$. The Nash equilibrium strategy profile for all rounds is denoted with $\overline{\sigma}$. We denote the tuple of all miners' strategies apart from that of the i-th miner with $\sigma_{-i} \in \Sigma_{-i}$, and we may add a superscript to denote one or more rounds as above. Note that our notation cannot represent games in which there are multiple Nash equilibria. This is not a concern, as we will not come across such games.

3.2 Conditionally timelocked game and analysis

In this section, we define a game that captures the race between two transactions $\texttt{tx}_1$ and $\texttt{tx}_2$ that spend the same unspent output but under different conditions. On the one hand, $\texttt{tx}_1$ allows the output to be spent immediately, while $\texttt{tx}_2$ bounds the output to a timelock. On the other hand, $\texttt{tx}_1$ pays the miner a fee f_1, while $\texttt{tx}_2$ pays a fee f_2. Both $f_1, f_2 > f$, f the fee of ordinary transactions.

Naturally, if $f_1 > f_2$ any rational miner will immediately include $\texttt{tx}_1$. We are therefore interested in the case where $f_2 > f_1$. For this case, we determine the exact conditions under which all rational miners will wait out the timelock and include $\texttt{tx}_2$ (irrespective of the timelock). We observe that these conditions

depend solely on the two fees f_1, f_2, the minimum mining power λ_{min}, and the number of necessary bribing transactions m (e.g., 2 for DMC and 1 for LN).

Definition 1 (Conditionally timelocked output). *A* conditionally timelocked output *is an on-chain transaction output with spending condition* $\mathtt{cond}_1 \vee \mathtt{cond}_2$ *such that* $\mathtt{cond}_1$ *is not encumbered with any timelock and* $\mathtt{cond}_2$ *is encumbered with a timelock that expires* T *blocks after block with height* T_0*.*

Definition 2 (Conditionally timelocked game). *A* conditionally timelocked game *is a game that consists of* T *rounds, starting from a blockchain of height* T_0 *which includes an unspent conditionally timelocked output* $O = (\cdot, \mathtt{cond}_1 \vee \mathtt{cond}_2)$*. From the onset of the game, miners are aware of a set of transactions* $\mathtt{txs}_1$ *that fits in a single block.* $\mathtt{txs}_1$ *contains a transaction* $\mathtt{tx}_1$ *which spends* O *by satisfying* $\mathtt{cond}_1$*. All other transactions in* $\mathtt{txs}_1$ *spend at least one output of* $\mathtt{tx}_1$ *or of another transaction in* $\mathtt{txs}_1$*. Miners are also aware of another set of transactions* $\mathtt{txs}_2$ *that fits in a single block as well.* $\mathtt{txs}_2$ *contains a transaction* $\mathtt{tx}_2$ *that spends* O *by satisfying* $\mathtt{cond}_2$*. All other transactions in* $\mathtt{txs}_2$ *spend at least one output of* $\mathtt{tx}_2$ *or of another transaction in* $\mathtt{txs}_2$*. Furthermore, there are at every round enough valid, "unrelated" transactions to fill a block that do not spend* O *or any output spent or produced by any transaction in* $\mathtt{txs}_1 \cup \mathtt{txs}_2$*, and each offers fee* f*. We denote any set of unrelated transactions with* $\mathtt{txs}_u$*.*

Let $m = \max\{|\mathtt{txs}_1|, |\mathtt{txs}_2|\}$*. For* $i \in \{1, 2\}$*, we denote by* f_i *the maximum value a miner can extract (as fees or outputs that can be spent by anyone) by including in her block an* m*-sized set of transactions that includes* $\mathtt{tx}_i$ *and by* $\mathtt{txs}_i^*$ *any such set of transactions.*

Additionally, for $k \in [T]$ *we denote with* Γ_k *the* subgame *of* Γ *at the beginning of the* k*-th round with* O *still unspent. Likewise we denote with* Γ_k^* *the subgame of* Γ *at the beginning of the* k*-th round with* O *having already been spent.*

We note the following in the context of a conditionally timelocked game:

- $\mathtt{tx}_1$ is an ancestor of all other transactions in $\mathtt{txs}_1$,
- $\mathtt{tx}_2$ is an ancestor of all other transactions in $\mathtt{txs}_2$,
- $\mathtt{tx}_1$ and $\mathtt{tx}_2$ are mutually exclusive, therefore no pair of transactions from $\mathtt{txs}_1$ and $\mathtt{txs}_2$ respectively can coexist in the blockchain.
- For $i \in \{1, 2\}$, a set of transactions $\mathtt{txs}_i^*$ that extracts value f_i for the miner may contain anywhere from 1 to m transactions from $\mathtt{txs}_i$. The remaining transactions in $\mathtt{txs}_i^*$, as well as the rest of the transactions in the block, are unrelated transactions.
- If O is unspent at a round before T, a miner cannot mine $\mathtt{tx}_2$.
- If O is unspent at round T, a miner may mine either of $\mathtt{tx}_1$, $\mathtt{tx}_2$.
- If O is spent, a miner cannot mine either of $\mathtt{tx}_1$, $\mathtt{tx}_2$.
- We ignore games in which O is initially spent, as they provide no opportunity to bribe.
- Since $\Gamma_1 = \Gamma$ and O is initially unspent, there is no Γ_1^* game.
- The notation $\mathtt{txs}_u$ does not clarify the exact size of the set, but it will always be clear from context, keeping in mind that each block must contain exactly N transactions.

Intuitively, $\texttt{txs}_1$ represent honest and $\texttt{txs}_2$ represent bribing sets of transactions. Looking forward, the reason we consider sets of transactions (as opposed to just a single transaction) is because in Lightning a briber cannot offer the bribe just with $\texttt{tx}_2$ (i.e. "HTLC-Timeout" [1]), since the fee of this transaction is agreed upon by both protocol parties; the briber has to publish one more transaction instead, which would spend HTLC-Timeout and offer the bribing fee. This observation renders the analysis of [26] technically inapplicable to Lightning. Our model generalizes that of [26] to cover such situations. Furthermore, our approach applies to bribing scenarios in protocols that do not include hashlocks, such as DMC.

Lemma 1. *Consider a conditionally timelocked game Γ. If $mf > f_1$, then attempting to mine $\texttt{tx}_1$ at any round is a strictly dominated strategy for all miners.*

Proofs to all lemmas and theorems can be found in Appendix B.

Lemma 2. *Consider a conditionally timelocked subgame Γ_k^* in which O has been spent. $\forall \sigma \in \Sigma, \forall i \in [n]$, it is $u_i(\sigma, \Gamma_k^*) = \lambda_i(T-k+1)Nf$.*

Theorem 1. *Consider a conditionally timelocked game Γ. If $f_2 > \frac{f_1 - mf}{\lambda_{\min}} + mf > f_1$, then the unique Nash Equilibrium is for every miner to attempt to mine only $\texttt{txs}_u$ at each round before T and attempt to mine $\texttt{txs}_2^* \cup \texttt{txs}_u$ at round T, in other words that $\overline{\sigma} = (\underbrace{\texttt{txs}_u, \ldots, \texttt{txs}_u}_{n})^{T-1}(\underbrace{\texttt{txs}_2^* \cup \texttt{txs}_u, \ldots, \texttt{txs}_2^* \cup \texttt{txs}_u}_{n})$.*

Intuitively, Theorem 1 asserts that for a big enough bribe, every miner is incentivized to ignore the honest transactions, wait instead for the timelock to expire and then mine the bribing transactions, thus ensuring the success of the bribing attempt. The minimum required size of the bribe is proportional to the fees of the honest transactions, inversely proportional to the minimum mining power and independent of the timelock length.

4 Timelock bribe analysis

In this section, we leverage the analysis of Section 3 to examine the race between a briber that publishes an old transaction alongside with a bribe, and an honest party that follows the protocol specification; meaning that the honest party attempts to include on-chain the last update transaction or the revocation transaction in DMC and Lightning channels respectively. As explained in Section 2, the old transaction is timelocked but typically offers a bribe, while the honest transaction can be spent immediately but typically pays the miner less coins.

We first determine the parameter region under which the DMC channels are susceptible to such bribing attacks. Then, we modify the DMC channels, and propose a novel scheme, which we term SUBORN channels, to limit the bribing region. The core idea is that, if a party tries to bribe, its coins in the last agreed transaction are awarded to the miners by-design, in addition to the transaction

fee. We note that a rational briber will at most bribe the miners with its gain between the two competing transactions. For instance, suppose the cheating transaction awards 7 coins to the briber and 3 coins to the honest party, while the last agreed transaction awards 4 and 6 respectively. Then, the 4 coins (of the briber in the last state) can be claimed by the miner that mines the honest party's transaction, while the briber can only profitably bribe for less than $7 - 4 = 3$ coins, clearly losing the race. Our construction thus limits the parameter region in which timelock bribes are effective.

Thereafter, we identify the parameter region in which bribes are effective in LN. Finally, we propose the use of an increased fee in the revocation transaction, depending on the value of each transaction, to expand the aforementioned parameter region with similar effects to SUBORN channels.

4.1 Timelock bribe

Now, let P_1 be an honest party and P_2 a rational party which tries to maximize its coins like the miners. We assume that both parties have no mining power.

Definition 3 (Timelock Bribe). *Consider parties P_1, P_2 and a publicly known transaction $\mathtt{tx}$ with one output O that can be either spent by P_1 with a transaction $\mathtt{tx}_1$, possibly after a timelock, such that $\mathtt{tx}_1$ offers miners a value f_1, or by P_2 with a transaction $\mathtt{tx}_2$ which has a timelock that is strictly greater than that of $\mathtt{tx}_1$ (if the latter has any). Consider a set of transactions $\mathtt{txs}_2$, $|\mathtt{txs}_2| = m$, that contains $\mathtt{tx}_2$, offers total value f_2 to miners and all transactions in $\mathtt{txs}_2$ apart from $\mathtt{tx}_2$ spend at least one output of $\mathtt{tx}_2$ or another transaction in $\mathtt{txs}_2$. We say that P_2 offers a* timelock bribe *if P_2 publishes all $\mathtt{txs}_2$ before the timelock of $\mathtt{tx}_2$ has expired and $f_2 > \frac{f_1 - mf}{\lambda_{\min}} + mf$.*

Theorem 1 implies that the excessive fee paid by P_2 intends to discourage miners from including P_1's transaction before P_2's timelock expires and eventually include P_2's transaction instead. We now prove that the briber prefers to use the fewer (bribing) transactions possible (denoted by m).

Lemma 3.

$$\forall m \in [N-1], \frac{f_1 - mf}{\lambda_{\min}} + mf < \frac{(f_1 + f) - (m+1)f}{\lambda_{\min}} + (m+1)f$$

Note that the $(f_1 + f)$ in the numerator of the right-hand side of the inequality stems from the fact that an additional unrelated transaction has to be added to $\mathtt{txs}_1^*$ if the number of briber's transactions $\mathtt{txs}_2$ are increased by 1 while it is already $|\mathtt{txs}_2| \geq |\mathtt{txs}_1|$. In other words, Lemma 3 states the following: *Given that briber's transactions are more than the honest party's transactions, timelock bribes involving fewer transactions are cheaper for the briber.* This holds because a lower number of bribe transactions means that the briber has to surpass a lower minimum bribe in order to incentivize miners in her favor.

4.2 Timelock bribe in DMC

Let parties P_1 and P_2 that have a DMC channel and consider one of the two latest transactions, i.e., the only two non-invalidated, refund transactions gives $c_{r,1}$ to P_1, $c_{r,2}$ to P_2 and offers fee f_r, whereas the latest corresponding update transaction gives $c_{u,1}$ to P_1, $c_{u,2}$ to P_2 and offers fee f_u. Note that the update transaction is not timelocked, whereas the refund transaction is, and that the two transactions are mutually exclusive. Assume $c_{r,2} > c_{u,2}$. Then in this simple micropayment channel payments flow from P_2 to P_1, thus it is in the benefit of P_2 if the refund transaction is put on-chain instead of any of the replacement update transactions. Furthermore assume that all the timelocks of the opt-in transactions of the branch of interest have expired and that P_1 has published them along with the latest update transaction of the simple micropayment channel under discussion, but no child transaction that spends its $c_{u,1}$ coins – this is honest behavior according to the DMC protocol. Note that in the other simple micropayment channel of the current branch, payments flow from P_1 to P_2, therefore P_2 prefers the latest update transaction to the refund transaction in that channel and would not attempt to timelock bribe there.

The result intended by the DMC construction is for the update transaction, and not the refund transaction, to be included on-chain. Unfortunately, under the assumption of rational miners, there are cases in which this expectation is violated. In particular, P_2 can offer a timelock bribe and turn the inclusion of the refund transaction into a strictly dominant strategy profile for the miners. We identify the parameter region for which this is possible.

Theorem 2. *A DMC bribe is possible if* $c_{r,2} - c_{u,2} > \frac{f_u - 2f}{\lambda_{\min}} + 2f - f_r$*, where* $c_{r,2}, c_{u,2}$ *are* P_2*'s coins in the refund and update transactions respectively, and* f_r, f_u *are the fees of the refund and updated transactions.*

P_1 should therefore take care to avoid such a situation by invalidating the current refund transaction before such a state is reached. Note that due to Lemma 3 it does not make sense for P_2 to attempt to bribe using more transactions than just the refund transaction and $\mathtt{tx}_b$, lest she wants to pay a higher bribe. Also note that it is essentially risk-free for P_2 to attempt a timelock bribe, since if it fails the latest update transaction will be mined and P_2 will receive her fair share without any punishment. Due to symmetry between the two parties, the analysis above holds with the roles of P_1 and P_2 reversed.

Observe that in practice parties have the ability to locally re-estimate the value of $\lambda_{\min}$ on the fly and act accordingly: if a change to apparent mining power distribution makes one of the two parties decide that the current balance is reaching risky values, it can ask its counterparty to invalidate the current leaf and refuse to do any further payments until this is done.

4.3 Improving DMC incentives: simple Suborn channels

Simple Suborn channel design. Our goal is to drastically reduce the effectiveness of timelock bribing in DMC. To that end, we propose the following

changes. Remember that the only valid state of a DMC channel is essentially two unidirectional channels. We denote with $(1 \to 2)$ the channel in which P_1 pays P_2, and with $(2 \to 1)$ the reverse; this notation is also used as a superscript.

Each party locally stores a different refund transaction (instead of having identical ones). In channel $(2 \to 1)$, P_2's refund transaction has two outputs: (a) an output with P_1's coins, spendable just with P_1's signature, (b) an output with P_2's coins, spendable with P_2's signature and the preimage of a specified hash. P_1's refund transaction in $(2 \to 1)$ is as in DMC (only signatures required).

The update transaction of channel $(2 \to 1)$ (held by P_1) is changed as follows. (a) P_1's output can be spent with P_1's signature, whereas (b) P_2's output has two spending methods: either with P_2's signature, *or* with the preimage of the aforementioned hash (same as the refund transaction) *without any signature*. Channel $(1 \to 2)$ is symmetric. The changes are depicted in Figure 3.

To establish a channel, each party generates a secret preimage and sends to the counterparty its hash. Upon receiving the hash, the party sends to the counterparty its signature on the refund transaction. To perform a payment, the payer signs and sends the new update transaction to the counterparty. The closing of a simple Suborn channel is similar to DMC (collaboratively, or unilaterally with a refund or update transaction).

When P_2 attempts to spend her $c_{r,2}^{2\to1}$ coins in her own refund transaction, she has to reveal the preimage. This secret can be used by a miner to claim P_2's coins from P_1's update transaction. This effectively increases the fee of P_1's update transaction using P_2's coins. The miner only knows the preimage if P_2 attempts to timelock bribe (disclosing the secret in the process) while neither the refund nor the update transaction is on-chain.

Note that this change does not jeopardize P_2's ability to use her refund transaction honestly. In case the timelock of P_2's refund transaction expires, she can publish it, wait for it to be confirmed deep enough in the blockchain, and only then publish a transaction that spends her $c_{r,2}$ coins. At that moment it is safe for P_2 to reveal the preimage, since the update transaction cannot be included on-chain anymore.

Analysis. In order to determine the exact bounds within which our technique prevents timelock bribes, we perform a similar analysis as for the original DMC.

Theorem 3. *A bribe in the simple* Suborn *channels is possible if* $c_{r,2} - c_{u,2}(1 + \frac{1}{\lambda_{\min}}) > \frac{f_u - 2f}{\lambda_{\min}} + 2f - f_r$*, where* $c_{r,2}, c_{u,2}$ *are* P_2*'s coins in the refund and update transactions respectively, and* f_r, f_u *are the fees of the refund and updated transactions.*

We see that the bounds of balances within which timelock bribes may take place is much smaller than in the DMC construction. For example, if $\lambda_{\min} = 0.02$ (as estimated in [22]), then $c_{u,2}$ may become 51 times smaller than in plain DMC before a bribing opportunity arises. Unfortunately, these bounds are still tighter than the ones originally recommended in DMC, which allowed simple micropayment channels to be completely depleted before moving on to a new

branch – as we showed, this would risk a timelock bribe opportunity. Once again, Lemma 3 precludes a case where it is in the benefit of P_2 to bribe using more transactions. Note that this change exposes P_2 to some risk if she attempts a timelock bribe even if the Nash equilibrium for the miners is to ignore P_1's update transaction. In case any winning miner is irrational and chooses to mine P_1's update and take P_2's coins, P_2 is *punished* and takes no coins. The analysis for P_1 is symmetric.

Overhead over DMC. When opening the channel, each party has to generate a single preimage (32 bytes), send its hash (32 bytes) to the counterparty, receive and store the counterparty's hash. This has to be done only once for the entire lifetime of the channel. No additional communication rounds are needed, as the hashes can be appended to existing messages. When closing the channel, if a refund transaction is used, then the on-chain overhead is a hash (33 bytes, the extra byte specifies the length of the hash), its preimage (33 bytes as well) and the corresponding opcodes for its verification (`OP_SHA256` & `OP_EQUAL`, 1 byte each, c.f. Appendix A), adding 68 bytes. If one party publishes its update transaction and its counterparty is honest, then the on-chain overhead is the branch with the hashlock in the counterparty's output, which adds a hash and 4 opcodes (`OP_IF` & `OP_ELSE` & `OP_ENDIF`, c.f. Appendix A), adding 38 bytes. The overhead of the update transaction can be eliminated if the taproot[4] optimization is used.

4.4 Incentivizing DMC across branches: Suborn channels

Suborn channel design. The previous technique can be extended to discourage cross-branch bribes. Suppose the briber attempts to incentivize miners to ignore the valid branch of the invalidation tree altogether in favor of an invalidated branch, i.e., one which is encumbered with a longer timelock than the valid one. Now the briber may instead use an old update transaction to cheat. To address this issue, we require update transactions to include a hashlock as well. More specifically, the output of P_2, both in the refund and in the update transactions of P_2, should require the preimage of the hash along with P_2's signature. The changes are mirrored for the other party. This way all avenues for bribing are encumbered with preimage revelation. The two hashes (one per party) must remain the same across branches in all update and refund transactions. This way bribing in an old branch can be punished in the last branch. See Appendix A for the exact Script, and Figure 3 for an illustration of SUBORN channels.

In our scheme, to decide whether our balance is within safe bounds, we must consider all past update and refund transactions. This must be taken into account in the parameter region analysis. Note that in every simple channel the payer only stores the refund transaction, whereas the payee stores both the refund and the update transactions. The payee always prefers the update to the refund transaction, as simple channels are unidirectional.

[4] `https://github.com/bitcoin/bips/blob/master/bip-0341.mediawiki`

Let $k_l \in \mathbb{N}$ be the total number of leaves – the only valid leaf is the k_l-th. Let $k \in [k_l - 1]$. Furthermore, assume that branches k_l and k have j distinct opt-in transactions. The coins held in the outputs of the two last update transactions of the k-th branch that belong to P_2 are denoted with $c^{1\to2}_{k,u,2}$ and $c^{2\to1}_{k,u,2}$. Analogous notation is used for the refund transactions and P_1's coins. Let f_b be the necessary bribe given by $\texttt{tx}_b$ to incentivize miners to ignore the latest leaf – P_2 may only have $\texttt{tx}_b$ consume her $c^{2\to1}_{k,r,2}$ and optionally in addition her $c^{1\to2}_{k,u,2}$ coins, as all opt-in transactions only include multisig outputs that can only be spent according to the protocol. All opt-in, update and refund transactions have a fee f_o, f_u and f_r respectively.

Theorem 4. *A bribe in* Suborn *channels is possible if* $\forall k \in [k_l - 1], c^{1\to2}_{k,u,2} + c^{2\to1}_{k,r,2} - (c^{1\to2}_{k_l,u,2} + c^{2\to1}_{k_l,u,2})(1 + \frac{1}{\lambda_{\min}}) < \frac{1}{\lambda_{\min}}(jf_o + 2f_u - (j+3)f) + (j+3)f - jf_o - f_r - f_u$.

Before every payment P_1 must ensure that Theorem 4 will still hold to prevent a timelock bribe from P_2. Otherwise P_1 should refuse to facilitate the payment and propose creating a new branch instead. Lemma 3 ensures that it is in the benefit of P_2 to bribe with only one additional transaction $\texttt{tx}_b$. Similarly to the previous subsection, P_2 is exposed to some risk if it attempts a timelock bribe even when the Nash equilibrium is in her favor, since there may be a winning irrational miner that chooses to mine P_1's transaction and punish P_2. The analysis for P_1 is symmetric.

Overhead over Simple Suborn channels. The only additional overhead compared to Subsection 4.3 is a hash, a preimage and 2 opcodes, for a total of 68 bytes, when the party that publishes an update transaction spends its own output.

4.5 Timelock bribe in LN

LN is also susceptible to timelock bribes. P_2 can timelock bribe by publishing an old, revoked commitment transaction together with a bribing transaction $\texttt{tx}_b$. $\texttt{tx}_b$ spends P_2's output of the commitment transaction, offers fee f_b and pays the rest to P_2. For big enough f_b, this incentivizes miners to ignore P_1's revocation transaction, which carries a fee f_r; the revocation transaction gives all P_2's coins to P_1 (without a timelock) as a punishment for publishing an old commitment transaction. Let c_{old} and c_{new} be P_2's coins in the old and new commitment transactions respectively.

Theorem 5. *A bribe in LN channels is possible if* $c_{\text{old}} - c_{\text{new}} > \frac{f_r - f}{\lambda_{\min}} + 2f$.

Therefore, in order to avoid a timelock bribe, P_1 must not allow the channel balance to reach the condition of Theorem 5 for any old channel state. Lemma 3 ensures that it is not in P_2's benefit to attempt to bribe with more than one transaction. The analysis for P_1 is symmetric.

4.6 Fixing LN incentives

In order to shrink the bounds in which a timelock bribe is possible in LN, we propose the following change: Instead of having revocation transactions only offer fee f_r, they would instead offer a higher fee f'_r, such that bribes are not possible (reversing the inequality of Theorem 5). Note that we only consider countermeasures where the honest party does not lose coins. Figure 4 demonstrates the proposed modification to the lightning channel construction.

Theorem 6. *A bribe in modified LN channels is not possible if* $f'_r \geq f + \lambda_{\min}(c_{\text{old}} - c_{\text{new}} - 2f)$ *and* $\lambda_{\min} \leq \frac{c_{\text{new}} - f}{c_{\text{old}} - c_{\text{new}} - 2f}$*, where* c_{old} *are the maximum coins* P_2 *owned in any old channel state and* c_{new} *the coins* P_2 *currently owns.*

Before each payment, P_1 should ensure that Theorem 6 holds for the new balance. To do so, P_1 substitutes the values $\lambda_{\min}$, f, c_{old} and c_{new} – the latter is P_2's coins after the prospective update.

The maximum possible $\lambda_{\min}$ occurs when there are exactly two equal miners, meaning that $\lambda_{\min} = 0.5$. Then, $\frac{c_{\text{new}} - f}{c_{\text{old}} - c_{\text{new}} - 2f} \geq 0.5 \Leftrightarrow 2(c_{\text{new}} - f) \geq c_{\text{old}} - c_{\text{new}} - 2f \Leftrightarrow 3c_{\text{new}} \geq c_{\text{old}}$, meaning that P_1 can always nullify P_2's bribes if $3c_{\text{new}} \geq c_{\text{old}}$. For any lower $\lambda_{\min}$, the safe region is even larger.

Conveniently for P_1, the fee f'_r does not have to be determined in advance; it can be directly calculated and applied only if P_2 attempts to timelock bribe. Indeed, the revocation transaction can be built unilaterally by P_1 when it is needed and with the desired fee, as the punishment path of a commitment transaction is locked with a single key (namely "`revocationpubkey`" [1]) and P_1 knows the corresponding private key (namely "`revocationprivkey`" [1]). Moreover, f'_r does not need to be applied at all in case P_2 publishes an old commitment transaction without bribing. *No change in Script is necessary, just a suitable increase in the fee of revocation transactions*, as discussed above. The analysis for P_1 is symmetric.

Overhead over LN. Since our solution does not change any of the data exchanged neither adds outputs nor complicates scripts, its only overhead is the calculation of the new fee f'_r, a local computation that in practice is negligible.

5 Related Work

Bribing attacks on blockchains with Nakamoto-style consensus have been identified in the past, initially ones that incentivize miners to double-spend transactions [19]. Accepting such bribes carries an associated risk for the miner. Specifically, while the honest miners extend the longest chain, the bribed miner would have to ignore the last block and instead mine enough new blocks on top of an older block to catch up with the last block – a high risk/high returns strategy. Furthermore, attempts to fork the blockchain are often publicly visible and attributable, and could lead to damaged miner reputation.

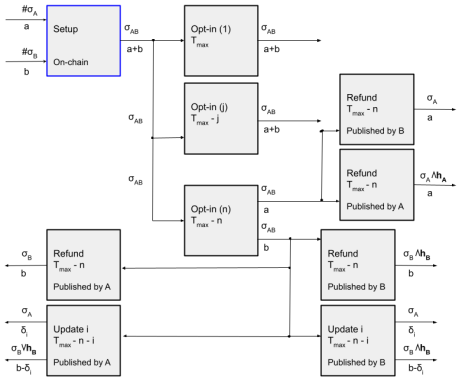

Fig. 3: SUBORN Channels

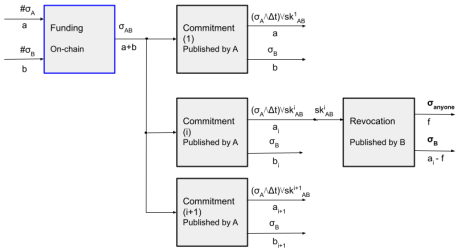

Fig. 4: Modified Lightning Channels

On the other hand, accepting a timelock bribe is risk-free for the miner, as it does not involve creating a fork; it simply has the miner ignore a particular transaction when forming a block and then mining on top of the longest chain. Previous analyses [22, 26, 27] of timelock bribes cover slightly different scenarios with distinct approaches and thus arrive to varying conclusions.

In particular, [22] focuses on the resilience of the HTLC smart contract under a timelock bribe, which is analyzed in the context of LN and *Cross-Chain Atomic Swaps* [16] under the assumption of rational miners and for all possible ranges of bribe values. A simple utility function is used, as it only considers the miners' payouts of the competing bribing and honest transactions, not taking into account fees offered by candidate unrelated transactions that could be included instead in a block. Modulo these differences, the subset of their analysis that treats the same bribe ranges as ours is indeed compatible with the results of the current work. For smaller bribes, the authors conclude that no opportunity for bribery exists given that the timelock is long enough – its exact length depends on the honest transaction fee, the bribe and the mining power distribution. Given the results of their analysis, the authors provide recommendations on safe parameters for LN and Atomic Swaps.

In [27] three different bribery mechanisms are presented and analyzed for a general setting of transaction censorship attacks. Fees from unrelated transactions are taken into account in the miners' utility function. The first attack involves paying out a separate bribe to each miner if it succeeds, not just to the winner of the last round. The second attack pays out bribes throughout the execution to each winning miner as long as the honest transactions are ignored. This attack can be cheaper than the first, but cannot, to the best of our knowledge, be implemented in Bitcoin Script without explicit cooperation by the honest party. The last attack is inspired by *feather forks* [20]: It bribes a miner with enough mining power to commit to ignoring blocks with undesired transactions, effectively threatening other miners to orphan their blocks if they act honestly. If the committed miner defects, she loses a deposit. This attack is cheaper than

the other two, but needs a miner that is willing to publicly commit to a malicious strategy. Due to incompatible assumptions on the setting, particularly on the payout schedule of the honest transactions, the results of [27] and the current work are not directly comparable.

Lastly in [26] bribing attacks against standard HTLC contracts are analyzed, bribing ranges beyond which the attack succeeds are provided, an extension to the Bitcoin Core code that allows miners to specify arbitrary strategies is implemented and a collateral-based modification of HTLC that provably withstands bribing attacks is built. Our incentive analysis constitutes a generalization of the approach of [26]. We further apply our analysis to both DMC and LN. We also provide an alternative method to discourage bribes, which does not employ collateral.

6 Conclusion and future work

6.1 Future work

In the context of a conditionally timelocked output, another direction of interest is to formally analyze miner incentives when $\frac{f_h - mf}{\lambda_{\min}} + mf > f_b > f_h$, where f_h is the fee of the timelock-free transaction and f_b is the bribe. Such a study would highlight opportunities for cheaper bribing, formulate the effects of the transition of the bribe value from one regimen to the other in a unified framework, and examine the effectiveness of our proposals against such lower bribe values. Ideally it would also unify the settings of [22, 26, 27] and the current work.

HTLCs, which are used both in DMC and LN for multi-hop atomic payments, leverage timelocks for their functionality. The methodology used in this work can be extended to techniques for mitigating timelock bribing for HTLCs as well.

6.2 Conclusion

In this work, we analyzed the circumstances under which a general form of timelock bribes may be carried out by a rational participant of a two-party protocol, assuming rational miners. We further applied our findings to provide bounds on the applicability of timelock bribes in DMC [11] and LN [24]. Subsequently, using specially tailored novel techniques that allow the honest party to use the rational party's funds to counter-bribe the miners, we reduced the opportunities for timelock bribes compared to the original constructions and effectively expanded their safe operating region.

References

1. *Lightning Network Specification, BOLT #3: Bitcoin Transaction and Script Formats.* `https://github.com/lightning/bolts/blob/master/03-transactions.md`.

2. L. Aumayr, O. Ersoy, A. Erwig, S. Faust, K. Hostakova, M. Maffei, P. Moreno-Sanchez, and S. Riahi. Generalized bitcoin-compatible channels. Cryptology ePrint Archive, Report 2020/476, 2020. https://eprint.iacr.org/2020/476.
3. L. Aumayr, O. Ersoy, A. Erwig, S. Faust, K. Hostáková, M. Maffei, P. Moreno-Sanchez, and S. Riahi. Bitcoin-compatible virtual channels. In *IEEE Symposium on Security and Privacy, Oakland, USA; 2021-05-23 - 2021-05-27*, 2021. https://eprint.iacr.org/2020/554.pdf.
4. L. Aumayr, P. Moreno-Sanchez, A. Kate, and M. Maffei. Donner: Utxo-based virtual channels across multiple hops. Cryptology ePrint Archive, Report 2021/855, 2021. https://eprint.iacr.org/2021/855.
5. Z. Avarikioti, E. K. Kogias, R. Wattenhofer, and D. Zindros. Brick: Asynchronous incentive-compatible payment channels. In *International Conference on Financial Cryptography and Data Security*, 2021.
6. Z. Avarikioti, O. S. T. Litos, and R. Wattenhofer. Cerberus channels: Incentivizing watchtowers for bitcoin. In *International Conference on Financial Cryptography and Data Security*, pages 346–366. Springer, 2020.
7. C. Burchert, C. Decker, and R. Wattenhofer. Scalable funding of bitcoin micropayment channel networks. In *The Royal Society*. 2018.
8. M. M. T. Chakravarty, S. Coretti, M. Fitzi, P. Gazi, P. Kant, A. Kiayias, and A. Russell. Hydra: Fast isomorphic state channels. Cryptology ePrint Archive, Report 2020/299, 2020. https://eprint.iacr.org/2020/299.
9. K. Croman, C. Decker, I. Eyal, A. E. Gencer, A. Juels, A. Kosba, A. Miller, P. Saxena, E. Shi, E. G. Sirer, D. Song, and R. Wattenhofer. On scaling decentralized blockchains. In *International Conference on Financial Cryptography and Data Security*, pages 106–125. Springer, 2016.
10. C. Decker, R. Russell, and O. Osuntokun. eltoo: A simple layer2 protocol for bitcoin. https://blockstream.com/eltoo.pdf.
11. C. Decker and R. 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.
12. M. Dong, Q. Liang, X. Li, and J. Liu. Celer network: Bring internet scale to every blockchain, 2018.
13. S. Dziembowski, L. Eckey, S. Faust, and D. Malinowski. Perun: Virtual payment hubs over cryptocurrencies. In *2019 2019 IEEE Symposium on Security and Privacy (SP)*, pages 344–361, Los Alamitos, CA, USA, May 2019. IEEE Computer Society.
14. S. Dziembowski, S. Faust, and K. Hostáková. General state channel networks. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, CCS 2018, Toronto, ON, Canada, October 15-19, 2018*, pages 949–966, 2018.
15. C. Egger, P. Moreno-Sanchez, and M. Maffei. Atomic multi-channel updates with constant collateral in bitcoin-compatible payment-channel networks. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, CCS '19, pages 801–815, New York, NY, USA, 2019. Association for Computing Machinery.
16. M. Herlihy. Atomic cross-chain swaps. In *Proceedings of the 2018 ACM Symposium on Principles of Distributed Computing, PODC 2018, Egham, United Kingdom, July 23-27, 2018*, pages 245–254, 2018.
17. M. Jourenko, M. Larangeira, and K. Tanaka. Lightweight virtual payment channels. In S. Krenn, H. Shulman, and S. Vaudenay, editors, *Cryptology and Network Security*, pages 365–384, Cham, 2020. Springer International Publishing.

18. A. Kiayias and O. S. T. Litos. A composable security treatment of the lightning network. In *33rd IEEE Computer Security Foundations Symposium*, pages 334–349. IEEE, 2020.
19. K. Liao and J. Katz. Incentivizing blockchain forks via whale transactions. In *Financial Cryptography and Data Security - FC 2017 International Workshops, WAHC, BITCOIN, VOTING, WTSC, and TA, Sliema, Malta, April 7, 2017, Revised Selected Papers*, pages 264–279, 2017.
20. A. Miller. Feather-forks: enforcing a blacklist with sub-50% hash power. `https://bitcointalk.org/index.php?topic=312668.0`. Accessed: 2020-11-22.
21. A. Miller, I. Bentov, R. Kumaresan, C. Cordi, and P. McCorry. Sprites and state channels: Payment networks that go faster than lightning, 2017. arXiv preprint arXiv:1702.05812.
22. T. Nadahalli, M. Khabbazian, and R. Wattenhofer. Timelocked bribing. In *Financial Cryptography and Data Security*, Lecture Notes in Computer Science, 2021.
23. S. Nakamoto. Bitcoin: A peer-to-peer electronic cash system, 2008.
24. J. Poon and T. Dryja. The bitcoin lightning network: Scalable off-chain instant payments. https://lightning.network/lightning-network-paper.pdf, January 2016.
25. J. Spilman. Anti dos for tx replacement. `https://lists.linuxfoundation.org/pipermail/bitcoin-dev/2013-April/002433.html`. Accessed: 2020-11-22.
26. I. Tsabary, M. Yechieli, and I. Eyal. MAD-HTLC: because HTLC is crazy-cheap to attack. *IEEE S&P*, 2021.
27. F. Winzer, B. Herd, and S. Faust. Temporary censorship attacks in the presence of rational miners. In *2019 IEEE European Symposium on Security and Privacy Workshops (EuroS &PW)*, pages 357–366. IEEE, 2019.
28. G. Wood. Ethereum: A secure decentralised generalised transaction ledger. *Ethereum Project Yellow Paper*, 2014.
29. L. Zhao, H. Shuang, S. Xu, W. Huang, R. Cui, P. Bettadpur, and D. Lie. Sok: Hardware security support for trustworthy execution, 2019.

A Suborn Transactions Script for Incentivized DMC

```
OP_IF
    <remote_pubkey>
    OP_CHECKSIG
OP_ELSE
    OP_SHA256
    <remote_hash>
    OP_EQUAL
OP_ENDIF
```

Fig. 5: script for P_{3-i}'s output of P_i's update transactions, $i \in \{1, 2\}$

```
OP_SHA256
<local_hash>
OP_EQUALVERIFY
<local_pubkey>
OP_CHECKSIG
```

Fig. 6: script for P_i's output in P_i's refund and update transactions, $i \in \{1, 2\}$

```
1
<remote_sig>
```

Fig. 7: witness script spending honest ("IF") branch of Fig. 5 script

```
<preimage>
<local_sig>
```

Fig. 8: witness script spending Fig. 6 script

```
0
<preimage>
```

Fig. 9: witness script spending punishment ("ELSE") branch of Fig. 5 script

B Omitted proofs

Proof of Lemma 1. For round $k \in [T]$, the game is either Γ_k or Γ_k^*. If a miner attempts to mine $\mathtt{tx}_1$ in round k, the maximum value she can extract is if she chooses to mine $\mathtt{txs}_1^*$ and fill the remaining $N - m$ slots with unrelated transactions. There is no benefit to be gained in this or later rounds if a different way of including $\mathtt{tx}_1$ is chosen, so we ignore such other options. The expected fee she gains from this round is $\lambda_i(f_1 + (N-m)f)$ in the first case and 0 in the second (as her block would be invalid). If instead she attempts to mine only unrelated transactions, her expected gains from this round are $\lambda_i N f$. It is $mf > f_1 \Leftrightarrow Nf > f_1 + (N-m)f \Leftrightarrow \lambda_i Nf > \lambda_i(f_1 + (N-m)f)$ and $\lambda_i Nf > 0$, so attempting to mine only unrelated transactions offers higher value in both cases. Since the expected utility is the sum of the expected gains of all rounds, attempting to mine $\mathtt{txs}_1^*$ in any round is strictly dominated by attempting to mine $\mathtt{txs}_u$ in their place. □

Proof of Lemma 2. Since O is spent, all remaining valid transactions offer fee f. Therefore the i-th miner has a probability λ_i to obtain fee Nf for each of the remaining $T - k + 1$ rounds, for a total expected utility $u_i(\sigma, \Gamma) = \lambda_i(T - k + 1)Nf$. □

Proof of Theorem 1. We will prove the theorem using induction and iterated elimination of strictly dominated strategies.

First of all, we note that

$$f_2 > f_1 > mf \ . \tag{1}$$

The first inequality stems directly from the theorem precondition, whereas the second arises when we solve $\frac{f_1 - mf}{\lambda_{\min}} + mf > f_1$ for f_1 while keeping in mind that $0 < \lambda_{\min} < 1$.

Consider now the i-th miner, $i \in [n]$ when she decides which transaction to include for the last round, T. If O is unspent, then

$$\begin{gathered}
\forall \sigma_{-i}^T \in \Sigma_{-i}^T \text{ it is} \\
u_i(\sigma_{-i}^T; \sigma_i^T = \mathtt{txs}_u, \Gamma_T) = \lambda_i Nf \ , \\
u_i(\sigma_{-i}^T; \sigma_i^T = (\mathtt{txs}_1^* \cup \mathtt{txs}_u), \Gamma_T) = \lambda_i(f_1 + (N-m)f) \ , \\
u_i(\sigma_{-i}^T; \sigma_i^T = (\mathtt{txs}_2^* \cup \mathtt{txs}_u), \Gamma_T) = \lambda_i(f_2 + (N-m)f) \ .
\end{gathered}$$

From inequalities (1) we deduce that $\sigma_i^T = \mathtt{txs}_2^* \cup \mathtt{txs}_u$ is a strictly dominant strategy for any $i \in [n]$, so $\overline{\sigma}^T = (\underbrace{(\mathtt{txs}_2^* \cup \mathtt{txs}_u), \ldots, (\mathtt{txs}_2^* \cup \mathtt{txs}_u)}_{n})$ in subgame Γ_T with $u_i(\overline{\sigma}^T, \Gamma_T) = \lambda_i(f_2 + (N-m)f)$.

We will now prove via induction that $\overline{\sigma}^{1\ldots T-1} = (\underbrace{\mathtt{txs}_u, \ldots, \mathtt{txs}_u}_{n})^{T-1}$ for subgame Γ_k, in other words that the Nash equilibrium in all rounds prior to

the last one in which O is unspent is for all players to attempt to mine only unrelated transactions.

The base of the induction is $k = T-1$. For $i \in [n]$, it is either $\sigma_i^{T-1} = \mathtt{txs}_1^* \cup \mathtt{txs}_u$ or $\sigma_i^{T-1} = \mathtt{txs}_u$ (as in the proof of Lemma 1, we ignore all configurations that include $\mathtt{tx}_1$ except for $\mathtt{txs}_1^*$). Let $\sigma_{-i}^{T-1} \in \Sigma_{-i}^{T-1}$ and λ_u the sum of mining power of miners who try to mine only unrelated transactions in round $T-1$, excluding the i-th miner. If $\mathtt{tx}_1$ is mined, then the last round is Γ_T^* and by Lemma 2 the utility obtained by the i-th miner at the last round is $\lambda_i N f$. It is

$$
\begin{aligned}
&u_i((\sigma_{-i}^{T-1}; \sigma_i^{T-1} = \mathtt{txs}_u)\overline{\sigma}^T, \Gamma_{T-1}) = \\
&\qquad \lambda_i(Nf + u_i(\overline{\sigma}^T, \Gamma_T)) + \lambda_u u_i(\overline{\sigma}^T, \Gamma_T) + (1-\lambda_u-\lambda_i)\lambda_i N f = \\
&\lambda_i(Nf + \lambda_i(f_2 + (N-m)f)) + \lambda_u\lambda_i(f_2+(N-m)f) + (1-\lambda_u-\lambda_i)\lambda_i N f \ , \\
&u_i((\sigma_{-i}^{T-1}; \sigma_i^{T-1} = (\mathtt{txs}_1^* \cup \mathtt{txs}_u))\overline{\sigma}^T, \Gamma_{T-1}) = \\
&\qquad \lambda_i((f_1 + (N-m)f) + \lambda_i N f) + \lambda_u u_i(\overline{\sigma}^T, \Gamma_T) + (1-\lambda_u-\lambda_i)\lambda_i N f = \\
&\lambda_i((f_1 + (N-m)f) + \lambda_i N f) + \lambda_u\lambda_i(f_2+(N-m)f) + (1-\lambda_u-\lambda_i)\lambda_i N f \ .
\end{aligned}
$$

It is

$$
\begin{aligned}
u_i((\sigma_{-i}^{T-1}; \sigma_i^{T-1} = \mathtt{txs}_u)\overline{\sigma}^T, \Gamma_{T-1}) &> u_i((\sigma_{-i}^{T-1}; \sigma_i^{T-1} = \mathtt{txs}_1^* \cup \mathtt{txs}_u)\overline{\sigma}^T, \Gamma_{T-1}) \\
\Leftrightarrow \lambda_i(Nf + \lambda_i(f_2 + (N-m)f)) &> \lambda_i((f_1 + (N-m)f) + \lambda_i N f) \\
\Leftrightarrow f_2 &> \frac{f_1 - mf}{\lambda_i} + mf \ .
\end{aligned}
$$

It is $\frac{f_1-mf}{\lambda_i} + mf \leq \frac{f_1-mf}{\lambda_{\min}} + mf$ so the above is true. Therefore $\overline{\sigma}^{T-1} = (\underbrace{\mathtt{txs}_u, \ldots, \mathtt{txs}_u}_{n})$, thus $\lambda_u = 1-\lambda_i$ and $u_i(\overline{\sigma}^{T-1\ldots T}, \Gamma_{T-1}) = \lambda_i(Nf + \lambda_i(f_2 + (N-m)f) + (1-\lambda_i)\lambda_i(f_2+(N-m)f) = \lambda_i((2N-m)f + f_2)$.

Let $k \in [T-2]$. The inductive assumption for $k+1$ is firstly that $\overline{\sigma}^{k+1} = (\underbrace{\mathtt{txs}_u, \ldots, \mathtt{txs}_u}_{n})$ and secondly $u_i(\overline{\sigma}^{k+1\ldots T}, \Gamma_{k+1}) = \lambda_i((T-k)Nf + f_2 - mf)$.

For the inductive step, let once again $i \in [n]$. It is either $\sigma_i^k = \mathtt{txs}_1^* \cup \mathtt{txs}_u$ or $\sigma_i^k = \mathtt{txs}_u$ (again ignoring suboptimal transaction sets that include $\mathtt{tx}_1$ but are not $\mathtt{txs}_1^*$). Let $\sigma_{-i}^k \in \Sigma_{-i}^k$ and λ_u the sum of mining power of miners who try to mine only unrelated transactions in round k, excluding the i-th miner. If $\mathtt{tx}_1$ is mined, then the next round is Γ_{k+1}^* and by Lemma 2 the utility obtained

by the i-th miner from all rounds after k is $\lambda_i(T-k)Nf$. It is

$$\begin{aligned}
&u_i((\sigma^k_{-i};\sigma^k_i=\texttt{txs}_u)\overline{\sigma}^{k+1\ldots T},\Gamma_k)=\\
&\lambda_i(Nf+u_i(\overline{\sigma}^{k+1\ldots T},\Gamma_{k+1}))+\lambda_u u_i(\overline{\sigma}^{k+1\ldots T},\Gamma_{k+1})+(1-\lambda_u-\lambda_i)\lambda_i(T-k)Nf=\\
&\lambda_i(Nf+\lambda_i((T-k)Nf+f_2-mf))+\\
&\lambda_u\lambda_i((T-k)Nf+f_2-mf)+(1-\lambda_u-\lambda_i)\lambda_i(T-k)Nf,\\
&u_i((\sigma^k_{-i};\sigma^k_i=\texttt{txs}^*_1\cup\texttt{txs}_u)\overline{\sigma}^{k+1\ldots T},\Gamma_k)=\\
&\lambda_i(f_1+(N-m)f+\lambda_i(T-k)Nf)+\\
&\lambda_u u_i(\overline{\sigma}^{k+1\ldots T},\Gamma_{k+1})+(1-\lambda_u-\lambda_i)\lambda_i(T-k)Nf=\\
&\lambda_i(f_1+(N-m)f+\lambda_i(T-k)Nf)+\\
&\lambda_u\lambda_i((T-k)Nf+f_2-mf)+(1-\lambda_u-\lambda_i)\lambda_i(T-k)Nf.
\end{aligned}$$

It is

$$\begin{aligned}
&u_i((\sigma^k_{-i};\sigma^k_i=\texttt{txs}_u)\overline{\sigma}^{k+1\ldots T},\Gamma_k)>u_i((\sigma^k_{-i};\sigma^k_i=\texttt{txs}^*_1\cup\texttt{txs}_u)\overline{\sigma}^{k+1\ldots T},\Gamma_k)\\
\Leftrightarrow\ &\lambda_i(Nf+\lambda_i((T-k)Nf+f_2-mf))>\lambda_i(f_1+(N-m)f+\lambda_i(T-k)Nf)\\
\Leftrightarrow\ &f_2>\frac{f_1-mf}{\lambda_i}+mf\ .
\end{aligned}$$

Like in the induction base, it is $\frac{f_1-mf}{\lambda_i}+mf\leq\frac{f_1-mf}{\lambda_{\min}}+mf$ so the above is true. Therefore $\overline{\sigma}^k=(\underbrace{\texttt{txs}_u,\ldots,\texttt{txs}_u}_{n})$, thus $\lambda_u=1-\lambda_i$ and

$$\begin{aligned}
&u_i(\overline{\sigma}^{k\ldots T},\Gamma_k)=\\
&\lambda_i(Nf+\lambda_i((T-k)Nf+f_2-mf))+(1-\lambda_i)\lambda_i((T-k)Nf+f_2-mf)=\\
&\lambda_i((T-k+1)Nf+f_2-mf)\ .
\end{aligned}$$

We have proven that $\forall k\in[T-1]$ it is $\overline{\sigma}^k=(\underbrace{\texttt{txs}_u,\ldots,\texttt{txs}_u}_{n})$ thus we deduce that $\overline{\sigma}=(\underbrace{\texttt{txs}_u,\ldots,\texttt{txs}_u}_{n})^{T-1}(\underbrace{\texttt{txs}^*_2\cup\texttt{txs}_u,\ldots,\texttt{txs}^*_2\cup\texttt{txs}_u}_{n})$. □

Proof of Lemma 3. Let $m\in[N-1]$.

$$\begin{aligned}
\frac{f_1-mf}{\lambda_{\min}}+mf<\frac{(f_1+f)-(m+1)f}{\lambda_{\min}}+(m+1)f\Leftrightarrow\\
\frac{f_1-mf}{\lambda_{\min}}<\frac{f_1-mf}{\lambda_{\min}}+f\Leftrightarrow\\
0<f
\end{aligned}$$

The latter is true, thus the proof is complete. □

Proof of Theorem 2. P_2 publishes the refund transaction, along with a transaction $\mathtt{tx}_b$ that spends her $c_{r,2}$ coins, transferring some of them to a new address that belongs to P_2 and offering the rest as fee f_b, such that $f_r + f_b > \frac{f_u - 2f}{\lambda_{\min}} + 2f$. Due to Theorem 1, miners will ignore the update transaction, wait for the timelock of the refund transaction to expire and mine it along with $\mathtt{tx}_b$. In order for this timelock bribe to be beneficial to P_2, it must hold that $c_{r,2} - f_b > c_{u,2} \Leftrightarrow c_{r,2} - c_{u,2} > f_b$. Therefore, a suitable f_b exists if $c_{r,2} - c_{u,2} > \frac{f_u - 2f}{\lambda_{\min}} + 2f - f_r$. □

Proof of Theorem 3. More specifically, consider P_2 evaluating whether to timelock bribe. Publishing the refund transaction and $\mathtt{tx}_b$ offers to miners a total fee $f_r + f_b$, of which f_b is taken from $c_{r,2}$, therefore bribing makes sense only if $c_{r,2} - f_b > c_{u,2} \Leftrightarrow c_{r,2} - c_{u,2} > f_b$. In that case the published update transaction offers an effective fee of $f_u + c_{u,2}$. Leveraging Theorem 1, we deduce that miners will accept the bribe if $f_r + f_b > \frac{f_u + c_{u,2} - 2f}{\lambda_{\min}} + 2f \Leftrightarrow f_b > \frac{f_u + c_{u,2} - 2f}{\lambda_{\min}} + 2f - f_r$. Therefore, a suitable f_b exists if and only if $c_{r,2} - c_{u,2} > \frac{f_u + c_{u,2} - 2f}{\lambda_{\min}} + 2f - f_r \Leftrightarrow c_{r,2} - c_{u,2}(1 + \frac{1}{\lambda_{\min}}) > \frac{f_u - 2f}{\lambda_{\min}} + 2f - f_r$. □

Proof of Theorem 4. For each $k \in [k_l - 1]$, P_2 prefers the update transaction of $(1 \to 2)$ and the refund transaction of $(2 \to 1)$ k-th leaf to the update transactions of the currently valid leaf if $c^{1\to2}_{k,u,2} + c^{2\to1}_{k,r,2} - f_b > c^{1\to2}_{k_l,u,2} + c^{2\to1}_{k_l,u,2} \Leftrightarrow c^{1\to2}_{k,u,2} + c^{2\to1}_{k,r,2} - (c^{1\to2}_{k_l,u,2} + c^{2\to1}_{k_l,u,2}) > f_b$. Since branches k and k_l have j distinct opt-in transactions, then $j + 3$ transactions are implicated in the bribe. Thus, according to Theorem 1 miners will choose the bribe if $jf_o + f_r + f_u + f_b > \frac{1}{\lambda_{\min}}(jf_o + 2f_u + c^{2\to1}_{k_l,u,2} + c^{1\to2}_{k_l,u,2} - (j+3)f) + (j+3)f \Leftrightarrow f_b > \frac{1}{\lambda_{\min}}(jf_o + 2f_u + c^{2\to1}_{k_l,u,2} + c^{1\to2}_{k_l,u,2} - (j+3)f) + (j+3)f - jf_o - f_r - f_u$. Therefore, a compatible fee f_b exists if $c^{1\to2}_{k,u,2} + c^{2\to1}_{k,r,2} - (c^{1\to2}_{k_l,u,2} + c^{2\to1}_{k_l,u,2}) > \frac{1}{\lambda_{\min}}(jf_o + 2f_u + c^{2\to1}_{k_l,u,2} + c^{1\to2}_{k_l,u,2} - (j+3)f) + (j+3)f - jf_o - f_r - f_u$. □

Proof of Theorem 5. For the bribe to be profitable for P_2, it must be $c_{\text{old}} - f_b > c_{\text{new}} - f \Leftrightarrow c_{\text{old}} - c_{\text{new}} - f > f_b$ – the fee f is included because this is the minimum fee P_2 would have to pay anyway in order to use its c_{new} coins. By applying Theorem 1, we deduce that miners will accept the bribe if $f_b > \frac{f_r - f}{\lambda_{\min}} + f$, therefore a suitable f_b exists if and only if $c_{\text{old}} - c_{\text{new}} - f > \frac{f_r - f}{\lambda_{\min}} + f \Leftrightarrow c_{\text{old}} - c_{\text{new}} > \frac{f_r - f}{\lambda_{\min}} + 2f$. □

Proof of Theorem 6. To discourage bribes, from Theorem 5, the fee of the honest party should satisfy the following: $c_{\text{old}} - c_{\text{new}} \leq \frac{f'_r - f}{\lambda_{\min}} + 2f$. This means that $f'_r \geq f + \lambda_{\min}(c_{\text{old}} - c_{\text{new}} - 2f)$. We will now ensure that this f'_r does not lead to loss of coins for P_1. Let c be the total channel value, which stays constant throughout the channel lifetime. P_1 has to own enough coins in the old state, so that their sum with the counterparty's coins minus the fee f'_r exceeds or matches P_1's coins in the latest state. Formally, $c - c_{\text{old}} + c_{\text{old}} - f'_r \geq c - c_{\text{new}} \Leftrightarrow c_{\text{new}} \geq f'_r$. Combining the above, it has to be $c_{\text{new}} \geq f + \lambda_{\min}(c_{\text{old}} - c_{\text{new}} - 2f) \Leftrightarrow \lambda_{\min} \leq \frac{c_{\text{new}} - f}{c_{\text{old}} - c_{\text{new}} - 2f}$. The last step is valid since $c_{\text{old}} - c_{\text{new}} - 2f > 0$. This is true since, as we saw

above, P_2 only attempts to bribe if $c_{\text{old}} - c_{\text{new}} - f > f_b$ and we know that $f_b \geq f$. □