He-HTLC: Revisiting Incentives in HTLC

Sarisht Wadhwa[§]
Duke University
sarisht.wadhwa@duke.edu

Jannis Stöter[§]
Duke University
jannis.stoeter@alumni.duke.edu

Fan Zhang
Duke University
fan.zhang@duke.edu

Kartik Nayak
Duke University
kartik@cs.duke.edu

***Abstract*—Hashed Time-Locked Contracts (HTLCs) are a widely used primitive in blockchain systems. Unfortunately, HTLC is incentive-incompatible and is vulnerable to bribery attacks. *MAD-HTLC* (Oakland'21) is an elegant solution aiming to address the incentive incompatibility of HTLC.**

In this paper, we show that *MAD-HTLC* is also incentive-incompatible. The crux of the issue is that *MAD-HTLC* only considers passively rational miners. We argue that such a model fails to capture *active rational* behaviors. We demonstrate the importance of taking actively rational behaviors into consideration by showing three novel *reverse-bribery* attacks against *MAD-HTLC* that can be implemented using Trusted Execution Environments (TEEs) or zero-knowledge proofs (ZKPs). We further show that reverse bribery can be combined with original delaying attacks to render *MAD-HTLC* insecure regardless of the relationship between v^{col} and v^{dep}. Based on the learnings from our attacks, we devise a new smart contract specification, He-HTLC,[1] that meets the HTLC specification even in the presence of actively rational miners.

1. Introduction

Blockchain-based cryptocurrencies like Bitcoin [1] and Ethereum [2] enable secure transfer of *tokens* without a central authority and allow users to set elaborate and programmable *smart contracts* to govern token transfers. Hashed Time-Locked Contract (HTLC) [3] is a widely used smart contract implementable both on Ethereum and Bitcoin. HTLC is prominently used in the Lightning network [4], [5] to securely route payments through multiple payment channels, but HTLC is also essential to contingent payments [6], atomic swaps [7], etc. At a high level, an HTLC is parameterized with a timelock T and a hash lock H (hence the name), enforcing a conditional transfer of v tokens from a payer (Bob) to a payee (Alice): Before timeout T, Alice can spend the v tokens by sending a transaction embedding a pre-image of H to the blockchain; after the time T, Bob can spend the tokens.

Unfortunately, as shown by previous works [8], [9] HTLC is vulnerable to *incentive manipulation*. In order for Alice to be able to spend the tokens, HTLC assumes that *miners* will include Alice's transaction to the blockchain in a timely fashion before the timeout T. However, as rational agents seeking to maximize profit, miners may not adhere to the desired behavior when properly incentivized by a malicious Bob. For example, Winzer et al. [8] showed that Bob can set up a smart contract to reward (or *bribe*) miners if they ignore Alice's transaction until after the timeout, causing Alice to lose the tokens. Indeed, not only a problem for HTLC, generic bribery attacks where the adversary corrupts participants with money (potentially in a creditable fashion with the help of a smart contract) is a real concern in consensus security [10], [11], blockchain oracles [12], [13], on-chain voting [14], etc.

As a solution to incentive manipulation in HTLC, Tsabary et al. [9] proposed an elegant solution called *MAD-HTLC*. The key idea is to require the payer to deposit collateral, and any misbehavior by the payer will lead to the collateral becoming spendable (thus can be confiscated) by miners. *MAD-HTLC* ensures that rational miners will always penalize cheating payer, thus deterring the payer from misbehaving.

While *MAD-HTLC* explicitly takes miners' rationality into account—a step forward compared to the original HTLC scheme—its analysis only considers a narrow set of possible strategies by the miners. Specifically, the only allowed action is *transaction placement optimization* where miners choose from the mempool the best transactions to include in a block. In other words, *MAD-HTLC* modeled miners as *passively rational* in that miners will pick the best opportunities made available by the environment, but will not actively create better opportunities by themselves. In reality, however, there are no such constraints on the miners' action space. Sophisticated miners might actively engage in external protocols (in addition to mining) to further increase their gains. As an example, a significant faction of Ethereum miners now offer for-profit "private relay" services [15]–[17] as an additional source of revenue [18].

We refer to miners' strategies beyond simple transaction placement optimization as *actively rational*. Motivated by the rise of actively rational strategies in real-world blockchains (e.g., [15]–[17]), we aim to understand their security implications for HTLC. Unfortunately, we show that *MAD-HTLC*—the state-of-the-art bribery-resistant realization of HTLC—is insecure with the presence of actively rational miners. On the other hand, we also show a modified

[§] Equal contribution

[1] Our specification is lightweight and inert to incentive manipulation attacks. Hence, we call it He-HTLC where He stands for Helium.

contract He-HTLC that satisfies the HTLC specification even in the presence of actively rational miners.

1.1. Reverse bribery attacks against *MAD-HTLC*

We introduce a family of novel attacks in the presence of actively rational miners. Specifically, we propose *reverse bribery* where the miner bribes the payer to divulge certain information in a way that both the miner and the payer are better off, at the expense of the payee. Reverse bribery is an example of an *actively rational action* because a miner initiates the attack. Since the role of parties sending and receiving bribes has reversed compared to the original attack by Winzer et al. [8] and *MAD-HTLC* [9], we call this a *reverse bribery attack*.

Key intuition. To understand the intuition behind our attacks, we briefly review the design of *MAD-HTLC*. It has the same functionality as HTLC with two key changes. First, in a transaction where Bob pays Alice v^{dep} tokens, Bob also deposits collateral of v^{col} tokens. Second, upon any bribery attempt of Bob, miners will confiscate both v^{dep} and v^{col}. In other words, in honest executions, Bob gets back v^{col}; if Bob attempts to bribe, Bob gets 0 (Alice too), and miners earn v^{dep} and v^{col}. The design goal is to disincentivize Bob from bribery, which this accomplishes. However, we observe that it leads to another attacking opportunity because *miners and Bob together can earn more than they would in honest executions*; by "dividing up the loot", both may be better off attacking.

Specifically, in a reverse bribery attack, the miners will agree with Bob on the following deal: Bob will divulge certain information to the miners (thus forgoing the collateral v^{col}), but the miners will compensate Bob with $bribe = v^{col} + \epsilon$ in a separate payment. Both, the miners and Bob, will earn more than they would in honest executions: miners will earn $v^{dep} - \epsilon$ more,[2] and Bob will earn ϵ more.

Challenges. While we establish the feasibility of an attack with the above intuition, there are two key challenges that we need to overcome to make these attacks work.

First, we need to establish whether there exists a feasible range of values such that both Bob and the miner are better off. Observe that if there was only one miner, then any bribe value $br \in (v^{col}, v^{col} + v^{dep})$ would allow Bob and the miner to be better off. However, when we have multiple miners competing, each with different mining power, each of them being either passively or actively rational, it is unclear if any of them or all of them should bribe Bob and what amount should be bribed.

Second, of course, the miners and Bob do not trust each other. The way they would "divide up the loot" in a mutually distrusting fashion poses a technical challenge. The problem resembles that of a fair exchange where the miners offer a bribe in exchange for certain information from Bob, but existing blockchain-based fair exchange protocols (such as [19]–[21]) do not apply to our setting because they focus on fair exchanges between two non-mining parties, and rely on miners as trusted parties to facilitate the exchange. In our setting, however, the exchange takes place between Bob and the miners where there is an imbalance of power; miners can choose which transactions to (and not to) include on-chain while Bob cannot, so miners can easily censor undesirable transactions (e.g., payment to Bob).

At a high level, we address the first challenge by proposing three different attacks and analyzing their properties through rigorous game-theoretic analyses. To address the second challenge, we construct *fair exchange* protocols between Bob and a miner.

1.2. Fixing HTLC with He-HTLC

Our attacks exploit two key aspects of *MAD-HTLC*. First, *MAD-HTLC* overly compensates miners with $v^{dep} + v^{col}$ tokens when pre_b is available. Second, since all of these tokens can be redeemed in a single transaction, it enables a reverse bribery attack through a fair exchange between Bob and the miners even if they do not trust each other.

To design our incentive-compatible protocol He-HTLC, we first appropriately reduce the amount of tokens that miners can confiscate as enforcers of the contract. Then, we enforce that, in honest executions, Bob can redeem $v^{dep} + v^{col}$ in two separate transactions that are some blocks apart. By introducing this separation and revealing the secret pre_b in the first of the two transactions, we utilize the distrust between Bob and the miners to force both parties to follow the protocol. There are several subtleties in determining the amount of tokens that miners can confiscate as well as ensuring Bob is unable to bribe the intermediate miners within these blocks.

1.3. Summary of Contributions

In summary, this paper revisits the incentive attacks against HTLC and makes the following contributions:

- To the best of our knowledge, we propose the first model that captures miners' actively rational actions. While observed in practice [15]–[17], miners' strategic actions besides mining have not been captured by existing models, leaving a gap in game theoretic analyses that makes our attacks possible.
- We introduce novel *reverse bribery attacks* which allow actively rational miners to profitably deviate from *MAD-HTLC*. We analyze the profitability of our attacks in a rigorous game theory model and outline their implementation using trusted execution environments or zero-knowledge proofs. We further show that reverse bribery can be combined with ordinary delay attacks (e.g., those in [8]) to form a hybrid attack that renders *MAD-HTLC* insecure regardless of the relationship between v^{col} and v^{dep} with constant probability.
- We present He-HTLC, the first HTLC scheme that is secure in the presence of actively rational miners.

[2] We currently ignore transaction fees, but consider them in later sections.

2. Overview of Results

2.1. HTLC, Delay Attacks, and Proposed Fixes

To aid understanding, before describing our results, we start by reviewing Hash Time-Locked Contracts (HTLC), concerns with the existing protocol, and attempts to fix it.

HTLC. As introduced earlier, an HTLC where Bob pays Alice v^{dep} tokens is specified by $(\mathsf{pk}_A, \mathsf{pk}_B, v^{dep}, pre_a, T)$, where pk_A and pk_B are public keys of Alice and Bob, pre_a is a secret value picked by Bob whose hash is on-chain, and T is the agreed-upon timeout. The above HTLC stipulates that Bob's deposit of v^{dep} tokens can be spent (or redeemed) in two ways: (i) Alice can spend if she obtains pre_a from Bob and broadcasts a signed transaction tx_A^{dep} including pre_a, or (ii) Bob can spend by broadcasting a signed transaction tx_B^{dep} after time T (i.e., Bob gets his money back if Alice is inactive for T time).

In practice, there are several variants of HTLC. The above description abstracts away implementation- and application-specific details and allows us to focus on the core issues (similar to HTLC-Spec in *MAD-HTLC* [9]). We refer readers to [9] for a survey of applications of HTLC.

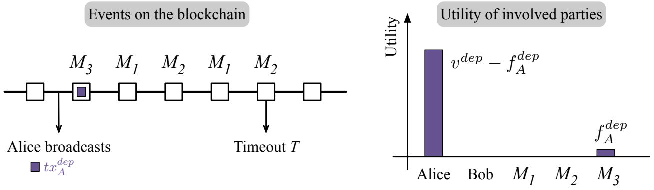


Figure 1. An honest execution of HTLC where miner M_3 includes Alice's transaction before timeout. In terms of utility, Alice gets the funds minus the fees paid to the miner.

We refer to executions of HTLC where Alice redeems v^{dep} before T as *honest executions*. Figure 1 depicts an honest execution, showing the relevant events on the blockchain and the utility of involved parties in the end. We assume there are three miners $\{M_i\}_{i=1}^{3}$, and the blocks are labeled by the miner mining it. We assume Alice obtains pre_a from Bob through off-chain channels and Alice releases tx_A^{dep} (paying a transaction fee f_A^{dep}) at some time before T. A miner, say M_3, includes the transaction before T and earns f_A^{dep}; Alice earns the rest $v^{dep} - f_A^{dep}$.

Bribery attacks on HTLC. Winzer et al. [8] and Tsabary et al. [9] showed attacks where Bob bribes miners to censor tx_A^{dep} so that Bob can redeem v^{dep} for himself at time T. Figure 2 depicts the attack, showing the relevant events on the blockchain and the utility of involved parties in the end. We assume there are three miners $\{M_i\}_{i=1}^{3}$, and the blocks are labeled by the miner of it. We assume Alice obtains pre_a from Bob through off-chain channels and Alice releases tx_A^{dep} (paying a transaction fee f_A^{dep}) at some time before T. As shown in Fig. 2, Bob can bribe b tokens (represented as circles) to each of the miners before timeout T to exclude tx_A^{dep}. The bribe can be paid offline or enforced through a separate contract. Then, Bob can create tx_B^{dep} at time T, which is included by M_2. As depicted in Fig. 2, Alice does not receive her expected gain whereas Bob gains $v^{dep} - f_B^{dep} - 4 \times b$. (The figure shows a party's earning above the x axis and the cost below it. The net gain is the difference between the two.)

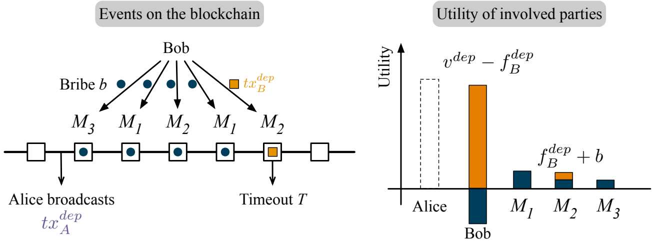


Figure 2. A delaying attack by Bob on HTLC. Bob bribes miners to censor tx_A^{dep} until the timeout, and then broadcasts tx_B^{dep}. In terms of utility, miners earn fees and bribes, and Bob earns v^{dep} at the expense of Alice (dashed rectangle denotes Alice's expected gain in honest executions).

MAD-HTLC. To make HTLC bribery-resistant, [9] proposed a modified HTLC protocol called *MAD-HTLC* (where MAD stands for mutually assured destruction).

As introduced above, *MAD-HTLC* implements the same functionality as HTLC, but with two key changes. First, it introduces a second hash lock (whose pre-image is denoted pre_b) and an additional redemption path where miners can redeem v^{dep} if they have access to both pre_a and pre_b. Second, *MAD-HTLC* introduces an additional collateral contract initiated by Bob which contains some collateral tokens v^{col} that can also be confiscated by miners if they know (pre_a, pre_b). These modifications aim to disincentivize Bob from revealing pre_b unless he needs to be truly refunded, in situations where pre_a is not released.

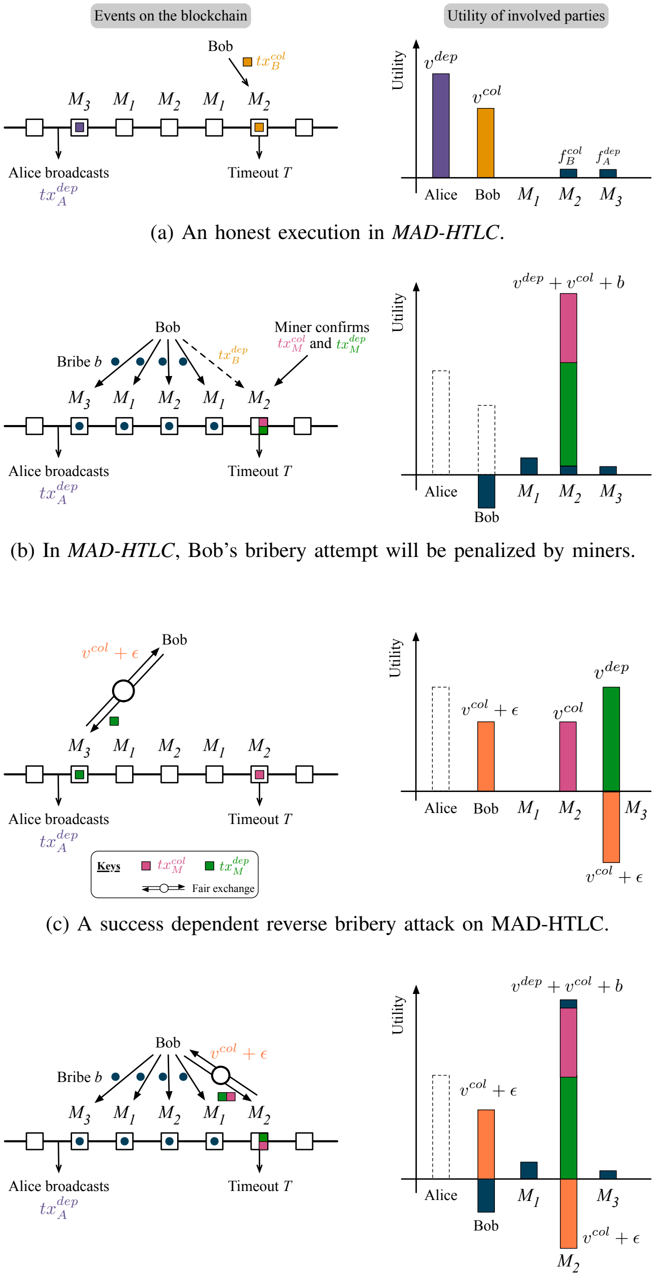
To recall notations in [9], Bob's payment of v^{dep} tokens is deposited in a contract called *MH-Dep*, which stipulates that v^{dep} can be redeemed in one of the following three paths (we use "paths" and "transactions" interchangeably, and to redeem through *dep-A* is the same as broadcasting tx_A^{dep}; the notation $t \geq T$ indicates that this transaction is invalid until T):

$$\begin{aligned}
tx_A^{dep} &= (pre_a, sig_a) && \text{// } \textit{dep-A}\text{: Alice can spend with } pre_a \\
tx_B^{dep} &= (pre_b, sig_b, t \geq T) && \text{// } \textit{dep-B}\text{: Bob can spend after } T \\
tx_M^{dep} &= (pre_a, pre_b) && \text{// } \textit{dep-M}\text{: Anyone can spend with both pre-images}
\end{aligned} \tag{1}$$

Bob's collateral v^{col} in contract *MH-Col* [9] can be redeemed in two ways:

$$\begin{aligned}
tx_B^{col} &= (sig_b, t \geq T) && \text{// } \textit{col-B}\text{: Bob can spend after } T \\
tx_M^{col} &= (pre_a, pre_b, t \geq T) && \text{// } \textit{col-M}\text{:Anyone can spend with both pre-images}
\end{aligned} \tag{2}$$

Figures 3a and 3b present two example scenarios in *MAD-HTLC*. In Fig. 3a, Bob is honest and he broadcasts tx_B^{col} after the timeout to get back the collateral. Figure 3b illustrates how *MAD-HTLC* prevents Bob's bribery attempts. At time T, rational miners will not let Bob spend v^{dep} via tx_B^{dep}, but instead, they will confiscate both v^{dep} and v^{col}. Thus, Bob loses not only v^{col} but also all bribes. Miners earn Bob's bribe b, and the miner who confiscates also earns v^{dep} and v^{col}. *MAD-HTLC* strongly disincentivizes Bob from pulling off the delaying attack described earlier and incentivizes miners to act as the enforcers.



(a) An honest execution in *MAD-HTLC*.

(b) In *MAD-HTLC*, Bob's bribery attempt will be penalized by miners.

(c) A success dependent reverse bribery attack on MAD-HTLC.

(d) A hybrid bribery attack on MAD-HTLC.

Figure 3. Example execution scenarios in *MAD-HTLC* and our reverse bribery attacks against *MAD-HTLC*.

2.2. Reverse Bribery: Revisiting Incentives in MAD-HTLC

Our key observation in this work is that *MAD-HTLC* is secure only assuming *passively rational* miners or miners who would maximize their utility in terms of the number of tokens based on transactions available in the mempool. However, if (some) miners are *actively rational*, i.e., they can actively seek out other users in the system and bribe them to obtain a higher utility, then this additional available action enables a new vector of attacks.

To intuitively see why this is possible, compare the two scenarios in Fig. 3a and Fig. 3b, and observe that in *MAD-HTLC*, Bob only redeems his collateral v^{col} at the end of an honest execution whereas miners do not earn any tokens other than transaction fees. However, the contract allows the miners and Bob to together redeem $v^{col} + v^{dep}$ if they can amicably find a way to retrieve it without necessarily trusting another party (but only relying on rationality assumptions). In a successful attack, this total gain can be split such that Bob and miners are individually better off (at least in expectation) than following the protocol. We present our attack in three parts, each of which is qualitatively different from the others.

***SIRBA*: Success-independent reverse bribery attack.** We first present (in Section 3) a warm-up attack that essentially shows that when $v^{col} - f_B^{col} < \frac{\lambda_i}{1-\lambda_i}(v^{dep} - f_A^{dep})$ following *MAD-HTLC* is not an incentive compatible strategy for a single actively rational miner M_i with mining power λ_i.

***SDRBA*: Success-dependent reverse bribery attack.** Our warmup attack only establishes that following *MAD-HTLC* is not incentive compatible for a given miner when all other miners are passively rational. As is, the result is not shown to extend under all possible actions and distributions of actively and passively rational miners (though likely there exists an equilibrium). Moreover, in this attack, while an actively rational miner is better off in expectation, it may end up losing tokens if it is not elected a miner in specific rounds of interest.

To reduce these uncertainties, we devise an attack called *success-dependent* reverse bribery attack (in Section 4). As the name suggests, in this attack, miners only pay a bribe to Bob if it has the opportunity to redeem v^{dep} through *dep*-M. This eliminates some risk for the miner w.r.t. bribing and not having the opportunity to redeem v^{dep}. However, since pre_b is revealed when v^{dep} is redeemed for a miner, all miners may engage in a competition to redeem v^{col} at time T. As shown in Fig. 3c, the winning miner M_3 exchanges some bribe $v^{col} + \epsilon$ where $\epsilon > 0$ for a gain of $v^{dep} - (v^{col} + \epsilon)$. However, in this execution, miner M_2 redeems v^{col}. (The double arrows with a circle between M_3 and Bob indicate a fair exchange of the bribe for redeeming v^{dep}.)

There are two key challenges to realizing this attack. The first is the game-theoretic formulation to show that there exists a set of bribe values under which all parties (except Alice) are better off in performing this attack under any given distribution of actively and passively rational miners and the action spaces available to them. The second challenge is to show that there exists a mechanism to perform a fair exchange between a bribing M_i and Bob. We show two realizations of this attack: the first relies on using trusted execution environments (TEEs) [22], [23] and the second uses zero-knowledge proofs [24].

***HyDRA*: Hybrid delay-reverse bribery attack.** Observe that *SDRBA* reduces risk w.r.t. mining v^{dep} but not w.r.t. v^{col}. In the example described earlier, the bribing miner earns $v^{dep} - (v^{col} + \epsilon)$. Thus, if $v^{dep} \leq v^{col}$, then the attack is not always beneficial for miner M_3. This concern can be eliminated if the same miner redeems both v^{dep} and v^{col} together (possibly in the same transaction). However, since *MAD-HTLC* requires v^{col} to be redeemed at time $\geq T$, it is necessary to delay redeeming v^{dep} until then.

This brings us to our third attack (Section 5) where we use a combination of delay attacks similar to Winzer et. al. [8] and *SDRBA*. Thus, we call this attack a *hybrid bribery attack*. As shown in Fig. 3d, in this attack, miners are first bribed to exclude transaction tx_A^{dep} until time T. Subsequently, miner M_2 at time T engages in *SDRBA* where it exchanges $v^{col} + \epsilon$ for redeeming $v^{dep} + v^{col}$. This attack works for any value of v^{dep} and v^{col} so far as the original delay attack (e.g., those in [8]) is feasible.

2.3. Fixing HTLC (Once Again) with He-HTLC

To design any incentive-compatible HTLC protocol, we need to satisfy the following constraints:

(i) when pre_a is *not* available to the miners within round T, Bob must be able to redeem $v^{dep} + v^{col}$.

(ii) when pre_a is available to the miners within round T, Alice must receive v^{dep} and Bob must receive v^{col}, i.e., Bob and miners together must not be incentivized to form a coalition and cheat Alice.

(iii) Alice must not be able to collude with miners and earn more than v^{dep}.

Observe that since the availability of pre_a is not recorded on the chain, miners may be willing to ignore pre_a even if it is available if they stand to gain more in the process. This creates a tension between achieving both constraints (i) and (ii). Indeed, this is exactly what our attacks exploited over *MAD-HTLC* in the previous sections. Two key ingredients of the attacks are: (a) Bob and miners together can get high earnings $v^{dep} + v^{col}$ through *dep-M* + *col-M* or *dep-B* + *col-B*, and (b) these redemptions could happen atomically in a single transaction.

At a high level, our solution breaks the atomicity and for some paths, burns some tokens. To address miners' high earnings through *dep-M* + *col-M*, we ensure that miners can never confiscate the total amount $v^{dep} + v^{col}$. We partially burn some of these tokens so that they earn only an "appropriate amount" of tokens. To address Bob's high earnings through *dep-B* + *col-B* (when pre_a is available), we break the atomicity of redemption by Bob by requiring Bob to redeem v^{dep} and v^{col} in separate steps: in the first step, Bob reveals pre_b to miners to redeem v^{dep}. In a second step, at least ℓ blocks later (where ℓ is a parameter), Bob can redeem v^{col} using *col-B*. This separation provides two guarantees. First, if pre_a is not available, miners would include the transaction redeeming v^{col} and Bob receives v^{col}. Second, if pre_a is known, then miners can be (and are) presented with an incentive-compatible path where they can confiscate an "appropriate amount" of tokens for themselves using $(pre_a, \ pre_b)$.

We now address the value of the "appropriate amount", namely how much miners should confiscate (and the rest will be burnt). Observe that if this amount is too high, then reverse bribery attacks become feasible (as is the case in *MAD-HTLC*). If the amount is too low, miners might be easily bribed to not confiscate. We set this amount to $v^{col} - v^{dep}$. This ensures that neither reverse nor ordinary bribery is feasible. First, observe that in all redemption paths, the combined earnings of a miner and Bob cannot exceed v^{col}. Thus the miner cannot afford to reverse bribe more than v^{col}. Since Bob would have earned v^{col} if he participated in the protocol honestly in the first place, miners will never have enough earnings to reverse bribe Bob. Second, to prevent Bob from bribing miners out of confiscation, we introduced the ℓ block separation between the two steps of redemption. The idea is to force Bob to have to bribe multiple miners that mine blocks between the two steps of redemption. Since every distinct miner needs to be paid more than $v^{col} - v^{dep}$ (the amount they earn from following the protocol honestly), Bob cannot afford to bribe all of them if there are sufficiently many distinct miners in between. For instance, if ℓ is large enough so that there are κ distinct miners in ℓ blocks on average, then for $v^{col} = \frac{\kappa}{\kappa-1} v^{dep}$, it would not be incentive compatible for Bob to pay more than $v^{col} - v^{dep}$ to all the κ miners, since the total bribe $\kappa \times (v^{col} - v^{dep})$ he has to pay will exceed his earning v^{col}.

Finally, to achieve constraint (iii), i.e., Alice and miners combined cannot cheat Bob, we ensure that when $v^{col} \leq 2v^{dep}$, through any combination of paths, miners and Alice together earn $\leq v^{dep}$. Thus, redeeming v^{dep} through *dep-A* is the ideal path for Alice.

3. System Model and a Success-Independent Reverse Bribery Attack

In this section, we present the system model and the game-theoretical framework in which we analyse our attacks. As a warm-up, we present *success-independent reverse bribery attack* (*SIRBA*) and rigorous game-theoretical analysis to showcase our proof techniques.

3.1. System Model

We will use a model similar to the one in *MAD-HTLC* [8], [9]. We assume the existence of a blockchain-based cryptocurrency that facilitates transactions of *tokens* among a set of participants. In our model, the participants are *Alice*, *Bob*, other *users*, and a fixed set of n *miners*, denoted by $\mathbf{M} = \{M_1, \ldots, M_n\}$. Below, we detail the modeling assumptions about blockchains and participants.

Blockchain. We model a blockchain as an append-only ledger consisting of an ordered sequence of blocks containing *transactions*. *Miners* create blocks whereas other participants create transactions to be submitted. We denote the j-th block as b_j. We consider block-creation as a discrete-time, memory-less stochastic process. In each round, only one miner creates a block. For simplicity, we assume that the blockchain does not fork (discussed in Section 8). The probability with which a miner M_i creates a block in a round is given by its *mining power* λ_i. We assume $\lambda_i < 0.5$ for each M_i and $\sum_{i=1}^{n} \lambda_i = 1$. We assume λ_i's are fixed and known to everyone.

We consider a transaction as confirmed once it has been included in a block. Miners receive a fee for including transactions in their block. For simplicity, we assume that there are always unrelated transactions in the mempool and that all transactions, unless specified otherwise, pay the same transaction fee f.

Rationality: active and passive. We consider all participants rational, risk-neutral, and non-myopic, and will break tie randomly (i.e., they act to maximize their expected utility at the end of the game). We assume no discount factor in the utility of a rational player, i.e., payment of x today has the same utility as payment of x after a long time. These assumptions are consistent with *MAD-HTLC*.

The key difference from *MAD-HTLC* is that we consider an extended action space for miners. *MAD-HTLC* assumes that miners will maximize their utility (in terms of the number of tokens earned) only based on transactions and information available in the mempool. We refer to this type of miners as *passively rational miners* because they passively act upon information provided by other players. Our model, however, permits miners to reach out to other players and engage in external protocols with them (hence we call them *actively rational miners*). For instance, miners can engage in bribing Alice or Bob if doing so increases their utility. Formally, we divide the set of miners $\mathbf{M} = \{M_1, M_2, \ldots, M_n\}$ into two fixed-size mutually disjoint subsets $\mathbf{M_P}$ and $\mathbf{M_A}$, where $\mathbf{M_P}$ refers to the set of passively rational miners and $\mathbf{M_A}$ refers to the set of actively rational miners. We refer to the total mining power of all miners in $\mathbf{M_P}$ as λ_{pa} and miners in $\mathbf{M_A}$ as λ_{ac}.

3.2. Success-Independent Reverse Bribery Attack

As a warmup, we now present success-independent reverse bribery attack (*SIRBA*), the first of the three reverse bribery attacks. Intuitively, the *SIRBA* attack is straightforward. Recall that in *MAD-HTLC*, Bob deposits v^{dep} tokens to contract *MH-Dep* (Eq. (1)), along with v^{col} tokens of collateral to contract *MH-Col* (Eq. (2)). The attack proceeds as follows: once Alice tries to redeem *MH-Dep* and reveals pre_a, miners pay a bribe to Bob in exchange for pre_b and try to redeem *MH-Dep* and *MH-Col* for themselves via paths *dep-M* and *col-M*.

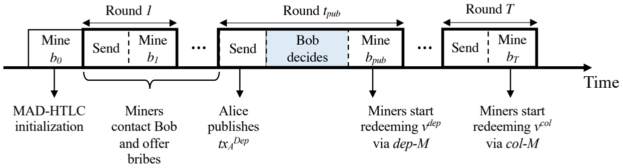

Round 1
Round t_{pub}
Round T
Mine b_0
Send
Mine b_1
...
Send
Bob decides
Mine b_{pub}
...
Send
Mine b_T
Time
MAD-HTLC initialization
Miners contact Bob and offer bribes
Alice publishes tx_A^{Dep}
Miners start redeeming v^{dep} via *dep-M*
Miners start redeeming v^{col} via *col-M*

Figure 4. Timeline of the *SIRBA* game. The game starts after *MAD-HTLC* are initialized and proceeds in T rounds. Each round consists of a *send* step and a *mine* step, except for the round t_{pub} which in addition has an intermediate step for Bob to decide if he accepts miners' offers.

To establish its feasibility, we need to answer two questions: (i) how does *SIRBA* change the incentives for parties? Specifically, is following *MAD-HTLC* an equilibrium? and (ii) how can we enable a fair exchange between two mutually distrusting parties?

We answer the first question through a rigorous game-theoretical analysis. At a high level, we will prove that everyone following the *MAD-HTLC* protocol is not an equilibrium. In particular, we will show in the following subsections that there exists a scenario where the existence of even a single actively rational miner can render *SIRBA* a preferred strategy over honest execution. To answer the second question, we present a practical implementation leveraging trusted execution environments (TEEs) or zero-knowledge proofs (Section 3.5).

3.3. Game Setup

To analyze *SIRBA*, we construct a game G^{ind} between Alice, Bob, and miners $M_1, M_2, \ldots, M_n$ as follows. Note that *ind* refers to success-independent.

3.3.1. Timeline of the *SIRBA* game.

Like in [9], without loss of generality, we say the game begins when *MH-Dep* and *MH-Col* contracts are initiated in some block b_0. It spans T *rounds*, corresponding to the creation of blocks b_1 through b_T. We denote the special round in which Alice publishes pre_a as t_{pub}.

Figure 4 visualizes the timeline of the *SIRBA* game. Every round except t_{pub} consists of two *steps*.

- *send*: Users, including Alice and Bob, send transactions to the mempool.
- *mine*: A miner M_i is chosen at random according to its mining power λ_i and creates a block with transactions of its choice, including ones created by itself.

In round t_{pub}, there is an intermediate step in between: as soon as Alice reveals pre_a in a "send" step, an actively rational miner can engage in reverse bribery with Bob to obtain pre_b, followed by a "mine" step as above. We will elaborate on reverse bribery when specifying Bob's and miners' action spaces.

States. In a given round k, the game can be in one of three states: 1) *red*: *MH-Dep* is still redeemable; 2) *irred-nrev*: *MH-Dep* has already been redeemed, but pre_b is not known to the miners, and 3) *irred-rev*: *MH-Dep* has already been redeemed, and pre_b is known to some miners. We define *States* as $\{\mathit{red}, \mathit{irred\text{-}nrev}, \mathit{irred\text{-}rev}\}$.

Subgames. We define a subgame for each round $k \in [1, T]$. We denote the subgame starting at the beginning of round k as $G^{\mathit{ind}}(k, s)$, where $s \in \mathit{States}$. $T - k$ more blocks are to be created after this subgame. We use $\cdot$ as wildcard when denoting subsets of games, e.g., $G^{\mathit{ind}}(\cdot, \mathit{red})$ refers to the set of all subgames in which *MH-Dep* is still redeemable.

3.3.2. Action spaces.

Now we specify the action space of Alice, Bob, and two types of rational miners.

Alice and Bob. Alice follows the *MAD-HTLC* protocol. Specifically, Alice can choose a round $t_{\mathit{pub}} \in [1, T)$ to publish $\mathit{tx}_A^{\mathit{dep}}$ (c.f., Eq. (1)) with a fee f_A^{dep} of her choice. Note that $f < f_A^{\mathit{dep}} < v^{\mathit{dep}}$ is necessary for Alice's transaction to outbid all other transactions in the mempool.

After $\mathit{tx}_A^{\mathit{dep}}$ (and pre_a) has been revealed by Alice, Bob's action is limited to those that do not reveal pre_b on-chain since otherwise his collateral will be confiscated by miners. One possible action is to get back v^{col} in round T, by publishing $\mathit{tx}_B^{\mathit{col}}$ offering a fee $f_B^{\mathit{col}} > f$ of his choice. For ease of exposition, we assume Bob always publishes $\mathit{tx}_B^{\mathit{col}}$ in round T even if he accepts bribes (see below). This does not change the game because $\mathit{tx}_B^{\mathit{col}}$ does not reveal any information about pre_b.

Another possible action of Bob is to engage in reverse bribery with actively rational miners, as we will elaborate on shortly.

Passively rational miners. The standard rationality assumptions (as in [9]) state that miners will strategically choose transactions to mine to maximize utility. We refer to this type of miner as *passively rational*. The transactions available to miners depend on the round k as well as miners' knowledge of pre_a and pre_b.

In any subgame $G^{\mathit{ind}}(\cdot, \cdot)$, M_i can include unrelated transactions from the mempool for fee f. In subgames $G^{\mathit{ind}}(k, \mathit{red})$, where $k \geq t_{\mathit{pub}}$, M_i can include $\mathit{tx}_A^{\mathit{dep}}$ for fee f_A^{dep}. In subgames $G^{\mathit{ind}}(k, \cdot)$, where $k \geq T$, M_i can include $\mathit{tx}_B^{\mathit{col}}$ for fee f_B^{col} if $\mathit{tx}_B^{\mathit{col}}$ has not already been included.

In addition, if M_i has knowledge of pre_a and pre_b, then in subgames $G^{\mathit{ind}}(k, \mathit{red})$, where $k \geq t_{\mathit{pub}}$, M_i can create and include a transaction $\mathit{tx}_M^{\mathit{dep}}$ that redeems *MH-Dep* for itself via path *dep-M*. In any subgame $G^{\mathit{ind}}(T, \cdot)$, M_i can create and include a transaction $\mathit{tx}_M^{\mathit{col}}$ that redeems *MH-Col* for itself via path *col-M*.

Actively rational miners and reverse bribery. A key difference from the model in [9] is that we permit miners to actively reach out to other players. We hence refer to such miners as *actively rational*. Specifically, we allow miners to perform reverse bribery. As soon as Alice reveals pre_a, actively rational miners can decide to pay a pre-agreed br_i to Bob in exchange for pre_b. For ease of exposition, we assume that Bob and a bribing miner have reached an agreement about the value of br_i before t_{pub}.

In *SIRBA*, Bob can independently decide whether to accept or reject each bribe br_i (the corresponding action is denoted accept_i and reject_i). In case Bob accepts, he will share pre_b with the briber before the creation of block $b_{t_{\mathit{pub}}}$. Since Bob does not trust miners (and vice versa), they need to exchange through a fair exchange mechanism which guarantees that Bob gets the bribe if and only if the miner gets pre_b. To focus on the analysis, we defer the implementation of fair exchange to Section 3.5.

We assume the exchange is instantaneous and that a miner cannot resell pre_b to other miners. For this section, we further assume that Bob will only accept bribes in round t_{pub}. So a miner can obtain pre_b in three ways:

1) through reverse bribery with Bob in round t_{pub};
2) when Bob redeems *MH-Dep* via *dep-B*;
3) when some other miner M_j, $j \neq i$, reveals pre_b on the blockchain, e.g., when redeeming *MH-Dep* via *dep-M* or *MH-Col* via *col-M*.

3.3.3. Utility.

For each player, the utility function $u_i : \mathit{Action} \times (\mathbb{Z}, \mathit{States}) \to \mathbb{R}$ is the number of tokens they can earn at the end of the game. We refer to Alice's utility as u_A, Bob's utility as u_B, and miner M_i's utility as u_i. To account for the stochastic nature of the game, we consider expected utilities at the end of the game.

We refer to player j's utility in G when action $\bar{s}$ is taken by j as $u_j(\bar{s}, G)$. Further, we refer to the maximum utility a player can end up with in G as $\bar{u_j}(G)$. We assume that the utility from a block containing only unrelated transactions is 0. Thus, if a transaction related to *MAD-HTLC* with utility x is included in exchange for an unrelated transaction, it would have a utility of $x - f$.

3.4. MAD-HTLC Incentive Incompatibility

We will now show that if there is a single actively rational miner M_i, then M_i and Bob will prefer engaging in reverse bribery to following the *MAD-HTLC* protocol (Theorem 1).

Lemma 1. *In subgame $G^{\mathit{ind}}(T, \cdot)$, if chosen to create block b_T, a miner with knowledge of pre_b will redeem MH-Col via path col-M and a miner without knowledge of pre_b will redeem MH-Col via path col-B.*

Proof: Note that *MH-Col* cannot be redeemed before round T by definition so redeeming *MH-Col* is a viable action in round T. By assumption, $\mathit{tx}_B^{\mathit{col}}$ is always available in round T. Since $v^{\mathit{col}} > f_B^{\mathit{col}} > f$, if M_i knows pre_b then including $\mathit{tx}_M^{\mathit{col}}$ in b_T is strongly dominant for M_i. Also, note that we assume pre_a has been revealed in $t_{\mathit{pub}} < T$.

If M_i does not know pre_b in round T, we will argue that M_i will have no chance to redeem *MH-Col* in the future, so including $\mathit{tx}_B^{\mathit{col}}$ in b_T is the best strategy. We have argued that miners have three ways to learn pre_b (in Section 3.3.2). If M_i does not know pre_b in round T, it follows that M_i would not learn pre_b from Bob in the future (since Bob only accepts bribery in t_{pub} by assumption, and Bob will not volunteer pre_b since pre_a has been revealed.) The only other possibility for M_i to learn pre_b is from other miners. But since any miner can redeem *MH-Col*, by the time pre_b is revealed, *MH-Col* would already have been redeemed.

Consequently, since $f_B^{col} > f$, including tx_B^{col} in b_T is strongly dominant for M_i if it does not know pre_b. □

Lemma 2. *In subgame $G^{ind}(T, red)$, if chosen to create block b_T, a miner with knowledge of pre_b will redeem MH-Dep via path dep-M and a miner without knowledge of pre_b will redeem MH-Dep via path dep-A.*

Proof: The argument is analogous to Lemma 1, given that $v^{dep} > f_A^{dep} > f$. □

Lemma 3. *In any subgame $G^{ind}(k, red)$, where $k \geq t_{pub}$ and $k < T$, if chosen to create block b_k, a miner with knowledge of pre_b will redeem MH-Dep via path dep-M.*

Proof: Let M_i be some miner in round k with knowledge of pre_b chosen to create b_k. From Lemma 1, it follows that *MH-Col* will be redeemed in round T. Consequently, M_i's expected utility with respect to the redemption of *MH-Col* is given by $\lambda_i(v^{col} - f)$ independent of how many other miners know pre_b. In particular, revealing pre_b by including tx_M^{dep} in block b_k before round T does not negatively affect M_i's expected utility with respect to the redemption of *MH-Col*. Given no negative effects with respect to the later redemption of *MH-Col*, it follows from $v^{dep} > f_A^{dep} > f$ that including tx_M^{dep} in block b_k is strongly dominant for M_i. □

Lemma 4. *In subgame $G^{ind}(t_{pub}, red)$, when all other miners are passively rational, offering a $br_i < \lambda_i(v^{dep} - f_A^{dep}) + \lambda_i(v^{col} - f_B^{col})$ strongly dominates any other action available to a single actively rational M_i.*

Proof: Let M_i be an actively rational miner in subgame $G^{ind}(t_{pub}, red)$ facing the decision of whether or not to offer br_i to Bob. If M_i chooses to bribe, supposing Bob accepts, it follows from Lemma 2 and Lemma 3 that M_i will redeem *MH-Dep* via path *dep-M* if chosen to create a block in any subgame $G^{ind}(k, red)$, where $k \geq t_{pub}$.

Note that M_i's expected utility with respect to the redemption of *MH-Dep* is lowest if all passively rational miners try to redeem *MH-Dep* via path *dep-A* in all subgames $G^{ind}(k, red)$, where $k \geq t_{pub}$. In this case, M_i's expected utility from knowing pre_b with respect to the redemption of *MH-Dep* is given by $\lambda_i(v^{dep} - f)$. From Lemma 1, it follows that M_i's expected utility from knowing pre_b with respect to the redemption of *MH-Col* is always given by $\lambda_i(v^{col} - f)$. Consequently, M_i's total expected utility from offering br_i to Bob is lower bounded by:

$$\lambda_i(v^{dep} - f) + \lambda_i(v^{col} - f) - br_i \tag{3}$$

Now suppose M_i chooses not to bribe. Then, since all other miners are passively rational, no one will bribe. In this case, pre_b will never be revealed. In this case, since $f_A^{dep} > f$ and since there is no benefit from waiting for pre_b to be revealed, in a perfect information game, every passively rational miner and M_i will redeem *MH-Dep* via path *dep-A* if chosen to create a block in any subgame $G^{ind}(k, red)$, where $k \geq t_{pub}$. Consequently, M_i's expected utility with respect to the redemption of *MH-Dep* if choosing not to bribe is given by $\lambda_i(f_A^{dep} - f)$. It further follows from Lemma 1 that M_i's expected utility with respect to the redemption of *MH-Col* is given by $\lambda_i(f_B^{col} - f)$ in this case. Therefore, M_i's total expected utility from choosing not to bribe in round t_{pub} is given by:

$$\lambda_i(f_A^{dep} - f) + \lambda_i(f_B^{col} - f) \ . \tag{4}$$

Given the expected utilities in Eq. (3) and Eq. (4), we can see that when no other miner bribes, for M_i offering br_i to Bob strictly dominates not bribing in round t_{pub} as long as $br_i < \lambda_i(v^{dep} - f_A^{dep}) + \lambda_i(v^{col} - f_B^{col})$. □

Lemma 5. *In subgame $G^{ind}(t_{pub}, red)$, when all miners except M_i are passively rational, Bob will strictly have higher utility from accepting M_i's bribe as long as $br_i > (1 - (1 - \lambda_i)^{T - t_{pub} + 1})(v^{col} - f_B^{col})$.*

Proof: Once Alice publishes pre_a in t_{pub}, the most Bob stands to gain from following the protocol is $v^{col} - f_B^{col}$.

If Bob shares pre_b with M_i, then it follows from Lemma 3 that M_i will try to mine a block containing tx_M^{dep} to redeem *MH-Dep* via path *dep-M*. Denote the event that M_i can successfully mine a block with tx_M^{dep} between time t_{pub} and T as E. If E happens, Bob's utility from *MH-Col* will be 0, since miners will take *col-M* using pre_b revealed by E. If E does not happen, Bob gets to redeem *MH-Col* unless M_i is elected in round T, since M_i is the only miner that can confiscate Bob's collateral. So Bob's utility is $Pr[\neg E](1 - \lambda_i)(v^{col} - f_B^{col}) + br_i$. $Pr[E]$ depends on M_i's mining power as well as the actions of other (passively rational) miners.

We consider the extreme case in which all passively rational miners only include unrelated transactions. This is the worst-case scenario for Bob since it maximizes the probability that M_i gets *dep-M*, and thus correspondingly some miner gets *col-M*. So E happens as long as M_i is elected in any round between t_{pub} and T. Therefore $Pr[\neg E] = (1 - \lambda_i)^{T - t_{pub}}$ and Bob's utility is $(1 - \lambda_i)^{T - t_{pub} + 1}(v^{col} - f_B^{col}) + br_i$.

In this case, Bob would have strictly higher utility from accepting M_i's bribe if $v^{col} - f_B^{col} < (1 - \lambda_i)^{T - t_{pub} + 1}(v^{col} - f_B^{col}) + br_i$ which is equivalent to: $(1 - (1 - \lambda_i)^{T - t_{pub} + 1})(v^{col} - f_B^{col}) < br_i$. □

Theorem 1. *Let M_i be the single active miner. Assuming that all miners are rational and non-myopic, then as long as $v^{col} - f_B^{col} < \frac{\lambda_i}{1 - \lambda_i}(v^{dep} - f_A^{dep})$, there always exists a value for br_i, such that M_i and Bob have higher expected utility from mounting SIRBA with br_i than from following the MAD-HTLC protocol, when all other miners are passive.*

Proof: From Lemma 5 it follows that in the case in which all other miners are passively rational, Bob will have strictly higher utility from accepting br_i if $br_i < (1-(1-\lambda_i)^{T-t_{pub}+1})(v^{col} - f_B^{col})$ independent of the action that passively rational miners choose before round T. From Lemma 4, we further know that M_i would have strictly higher expected utility from paying any $br_i < \lambda_i(v^{dep} - f_A^{dep}) + \lambda_i(v^{col} - f_B^{col})$ to Bob in exchange for pre_b. Clearly, a feasible bribe value that results in higher expected utility for both Bob and M_i will always exist if $(1-(1-\lambda_i)^{T-t_{pub}+1})(v^{col} - f_B^{col}) < \lambda_i(v^{dep} - f_A^{dep}) + \lambda_i(v^{col} - f_B^{col})$. It is simple to see that if $v^{col} - f_B^{col} < \frac{\lambda_i}{1-\lambda_i}(v^{dep} - f_A^{dep})$, such a value will always exist. □

Now that we know that participation in an *SIRBA* scheme is the action preferred by Rational Miners, we present a protocol that fairly achieves the fair exchange required for *SIRBA*.

3.5. Realising *SIRBA*

In this section, we present a practical implementation of the success independent attack. The challenge is to realize a fair exchange between Bob and a miner M_i, such that the M_i learns pre_b if and only if a payment to Bob of an agreed-upon amount is confirmed on the blockchain (the payment need not happen on the same blockchain as the MAD-HTLC). We show a solution using Trusted Execution Environments (TEEs).

We assume the bribing miner M_i and Bob have access to a TEE that guarantees integrity and confidentiality and supports remote attestation. We have seen one use of the same in Section 4.4 Further, we have assumed that they can access a secure Proof-of-Work based blockchain for payment (e.g., Bitcoin or Ethereum). We also assume that the difficulty does not vary between the time of bribe setup and the completion (e.g., the timeout of the HTLC). Our description below is specific to Bitcoin, but it can be adapted to any PoW blockchain.

Setup. Bob and M_i negotiate the details of the bribery, including Bob's address to receive bribe $Addr_{Bob}$, the amount of bribe *Amount*, the hash of pre_b Hash_{pre_b}, as well as a lower bound for PoW difficult diff_l used by the contingent decryption protocol below. This can happen well before the timeout of the the HTLC.

M_i runs the TEE code shown in Fig. 5. For ease of exposition the above parameters are hardcoded in Fig. 5, so they are covered by TEE attestation. M_i shares the code with Bob, who can review and verify its correctness.

To initialize, M_i calls `init`(λ) to generate a pair of keys protected by TEE. Specifically, TEE samples a key and returns pk along with an attestation σ_{TEE}, binding pk to the TEE code, while the secret key is kept in TEE so that the miner cannot access.

M_i sends $(\mathsf{pk}, \sigma_{\mathsf{TEE}})$ to Bob, who verifies that the attestation σ_{TEE} is consistent with the source code he has obtained from the miner (including the parameters hardcoded therein), and that pk is certified by σ_{TEE}.

```
Pseudocode of the TEE enclave for success-independent bribery
Hardcoded:
  $Addr_{Bob}$: Bob's address to receive bribe
  $Amount$: The amount of bribe
  $\mathsf{Hash}_{pre_b}$: the hash of $pre_b$
  $\ell$: number confirmations required (e.g., in Bitcoin $\ell = 6$)
  $\mathsf{diff}_l$: Difficulty lower bar
Function Init($\lambda$):
  $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KGen}(1^\lambda)$ // generate a pair of keys in TEE
  $\sigma_{\mathsf{TEE}} = \mathsf{TEE.attestation}(\mathsf{pk})$ // $\sigma_{\mathsf{TEE}}$ binds pk to the code
  return $(\mathsf{pk}, \sigma_{\mathsf{TEE}})$
Function VerifyMsgFromBob($m$):
  // Verify that encrypted message from Bob has the right preimage
  return $\mathsf{H}(\mathsf{Dec}_{\mathsf{sk}}(m)) = \mathsf{Hash}_{pre_b}$
Function Decrypt($\mathsf{TX}_{bribe}, MerkleProof, h_1, \ldots, h_\ell$):
  Assert $(h_1, \ldots, h_\ell)$ forms a valid blockchain
  Assert that the difficulty in $h_i$ is at least $\mathsf{diff}_l$ for all $i$
  Assert that $\mathsf{TX}_{bribe}$ is in $h_1$ by checking $MerkleProof$
  Assert that $\mathsf{TX}_{bribe}$ pays an amount of $Amount$ to $Addr_{Bob}$
  return $\mathsf{Dec}_{\mathsf{sk}}(m)$
```

Figure 5. TEE enclave program for *SIRBA*

Bob provisions pre_b to TEE. Having verified the correctness of pk, Bob sends pre_b encrypted to the miner in $c = \mathsf{Enc}_{\mathsf{pk}}(pre_b)$. M_i then calls `VerifyMsgFromBob` to verify that the ciphertext encrypts the correct preimage.

Contingent decryption. The key idea is that the TEE code enforces contingent decryption of c upon receiving a proof of payment to Bob. Specifically, upon receiving a Bitcoin transaction TX_{bribe}, a Merkle proof, and a sequence of block headers $(h_1, \ldots, h_n)$, the TEE code will verify that the block headers form a hash chain with sufficient difficulty, and that TX_{bribe} is included in one of the block h_j that has been buried sufficiently deep (e.g., $n - j \geq \ell$, where ℓ is the number of confirmations required for TX_{bribe} to be considered final with high probability), and that TX_{bribe} pays the agreed upon amount to Bob' specified address. If all checks pass, TEE will decrypt c and reveal pre_b. Therefore, to obtain pre_b, M_i makes the payment on-chain, waits for ℓ confirmations, and presents the required information to TEE.

Security arguments. Assuming that TEE guarantees integrity and confidentiality, and that the PoW difficulty does not increase significantly beyond diff_l before the timeout of the HTLC in question, and that bribe amount is smaller than the block rewards (6.25 BTC $\sim$ $\$236,000$ as of January 2022), we argue that the miner cannot learn pre_b without paying the bribe (or more).

First, via remote attestation, Bob establishes that he is interacting with an genuine TEE running the expected source code in Fig. 5. According to Fig. 5, the only way for M_i to obtain pre_b without paying the bribe is to feed the enclave with a forged chain of headers. Hardcoding the lower watermark for PoW difficulty prevents the miner from forking an old block with low difficulty. To pass the checks enforced by TEE, the miner must generate at least 6 blocks

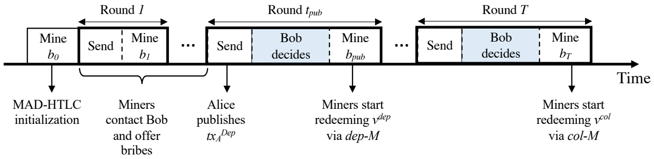

Round 1
Round t_{pub}
Round T
Mine b_0
Send
Mine b_1
...
Send
Bob decides
Mine b_{pub}
...
Send
Bob decides
Mine b_T
Time
MAD-HTLC initialization
Miners contact Bob and offer bribes
Alice publishes tx_A^{Dep}
Miners start redeeming v^{dep} via dep-M
Miners start redeeming v^{col} via col-M

Figure 6. The timeline of the *SDRBA* game is the same as that of *SIRBA* except that bribes can be offered in all rounds after t_{pub}.

that could have been accepted by the blockchain, which is prohibitively expensive and irrational since the bribe amount is smaller.

4. Success Dependent Reverse Bribery Attack

A primary limitation of *SIRBA* is that the bribe is paid to Bob *before* gaining any profit, so the attackers (miners) bear a risk. In fact, if multiple miners attempt to bribe Bob, except for the two miners (and potentially one) who manage to redeem *MH-Dep* and *MH-Col*, everyone else loses their bribe, even though they were better off in expectation. To remove this risk, we propose the success-dependent reverse bribery attack (*SDRBA*), which guarantees that the miner only pays the bribe if it successfully completes the attack, i.e., redeeming *MH-Dep* via path *dep*-M. Below we present a game-theoretical analysis, as well as two implementation options to enable exchange between the bribe and redemption by miners.

4.1. Game Setup

To analyze *SDRBA*, we construct a game G^{dep} where *dep* refers to success-dependent. The game setup is the same as that for *SIRBA* (Section 3.3) with one modification: in G^{dep} miners can offer bribes in all subgames $G^{dep}(k, red)$ for $k \geq t_{pub}$. Therefore all rounds $k > t_{pub}$ have an intermediate step where Bob decides if he accepts the offers. Figure 6 illustrates the timeline and the differences from that of *SIRBA*. For our analysis, we assume miners do not change their bribes br_i across rounds.

As before, to focus on the analysis, we assume a fair exchange mechanism between a bribing miner M_i and Bob, such that M_i pays Bob an agreed-upon bribe br_i if and only if M_i can include tx_M^{dep} (which contains pre_b) in a block. We construct such a fair exchange mechanism in Section 4.4.

Note that in a game $G^{dep}(k, st)$ with $st = irred\text{-}rev$, every miner has the information about pre_b, whereas in $st = irred\text{-}nrev$, no miner has the information about pre_b. This is different from Section 3.3, where some miners could have information about pre_b, whereas other would not.

4.2. MAD-HTLC Incentive Incompatibility

In this section, we will show that any active rational miner M_i and Bob will prefer engaging in *SDRBA* over following the *MAD-HTLC* protocol for a range of parameters depending on mining share of active rational miners.

Lemma 6. *In subgame $g = G^{dep}(T, st)$, for each miner $M_i \in \mathbf{M}$ with mining power λ_i, if $st = irred\text{-}rev$, $\bar{u}_i(g) = \lambda_i(v^{col} - f)$, and if $st = irred\text{-}nrev$, $\bar{u}_i(g) = \lambda_i(f_B^{col} - f)$.*

Proof: Since *MH-Dep* has been redeemed in both *irred-rev* and *irred-nrev*, the miner of round T can choose between three actions in round T have three actions: redeem *MH-Col* via *col*-B, redeem *MH-Col* via *col*-M, or do nothing (including unrelated transactions).

We discuss based on how *MH-Dep* might have been redeemed. Since *dep*-B cannot happen before round T, *MH-Dep* must have been redeemed through *dep*-M or *dep*-A.

Suppose the path *dep*-M was taken, i.e. $st = irred\text{-}rev$. As a result, pre_b has been publicly revealed to all miners. Consequently, since $v^{col} > f_B^{col} > f$, M_i will strictly prefer to redeem *MH-Col* via path *col*-M over any other action. Consequently, since every miner is trying to redeem *MH-Col* in this subgame, $\bar{u}_i(G^{dep}(T, irred\text{-}rev)) = \lambda_i(v^{col} - f)$.

Now suppose the path *dep*-A was taken i.e. $st = irred\text{-}nrev$. In this case, pre_b is only known to Bob. Since a miner can at most earn the value of *MH-Col* (i.e., v^{col}) through reverse bribery, it cannot bribe more than v^{col}. However, since Bob gets back v^{col} anyway, Bob has no incentive to accept the inferior bribes. So pre_b will never be revealed. Consequently, M_i will include tx_B^{col} (instead of unrelated transactions since $f_B^{col} > f$ by assumption). The utility is the fees, i.e., $\bar{u}_i(G^{dep}(T, irred\text{-}nrev)) = \lambda_i(f_B^{col} - f)$. □

Lemma 7. *For any subgame $g = G^{dep}(k, st)$ where $t_{pub} < k \leq T$ and $st \neq red$, $\bar{u}_i(g) = \bar{u}_i(G^{dep}(T, st))$.*

Proof: In all subgames in which *MH-Dep* has been redeemed, players can only expect additional utility from the redemption of *MH-Col*. As a consequence of Lemma 6, which shows the dominance of *MH-Col* redemption over waiting for unrelated transactions, we know that if *MH-Dep* is no longer redeemable, then *MH-Col* will be redeemed in round T.

Following the proof of Lemma 6, the expected utility from *MH-Col* solely depends on miners' knowledge of pre_b and mining power. Since *MH-Dep* has already been redeemed in some round prior to k, it follows that pre_b is either known to all miners if *MH-Dep* has been redeemed via path *dep*-M or will never be known if *MH-Dep* has been redeemed via path *dep*-A. By assumption, the mining power does not change. It follows that $\bar{u}_i(g) = \bar{u}_i(G^{dep}(T, st))$. □

Lemma 8. *In $G^{dep}(T, red)$, a passively rational miner will redeem MH-Dep and MH-Col via path dep-A and col-B.*

Proof: The proof is similar to lemmas 1 and 2 where the passively rational miner does not know pre_b in round T. □

Lemma 9. *In $G^{dep}(T, red)$, for an actively rational miner, paying a bribe br_i to Bob strongly dominates any other available action if $br_i < v^{dep} + v^{col} - f_A^{dep} - f_B^{col}$. In this case, $\bar{u}_i(G^{dep}(T, red)) = \lambda_i(v^{dep} + v^{col} - 2f - br_i)$.*

Proof: Let M_i be some actively rational miner in $G^{dep}(T, red)$. In this game, both *MH-Dep* and *MH-Col* are redeemable. First, note that the immediate utility to M_i from round T if chosen to create block b_T is strictly greater from including tx_A^{dep} and tx_B^{col} than from including only unrelated transactions. However, if M_i chose to include only unrelated transactions in block b_T, then *MH-Dep* and *MH-Col* could still be redeemed in a later round. Clearly, given that M_i would need to be chosen again to propose a block in some later round, which happens with probability $\lambda_i < 1$, and since the reward would be the same, it is strictly inferior for M_i to redeem *MH-Dep* and *MH-Col* by including tx_A^{dep} and tx_B^{col} in a later round.

Now suppose miner M_i were to bribe br_i. Then M_i's expected utility from bribery, conditional on Bob's acceptance of br_i, is given by

$$\lambda_i(v^{dep} - f + v^{col} - f - br_i) \tag{5}$$

since M_i can earn $v^{dep} + v^{col}$ with probability? λ_i. On the other hand, M_i's utility from not bribing (i.e., including transactions from Alice and Bob instead) is $\lambda_i(f_A^{dep} - f + f_B^{col} - f)$. Thus, if $br_i < v^{dep} - f_A^{dep} + v^{col} - f_B^{col}$, then M_i prefers paying bribe br_i to Bob. □

Lemma 10. *In any subgame $G^{dep}(k, red)$ where $t_{pub} \le k < T$, as long as $br_i < v^{dep} - f_A^{dep} + \lambda_i(v^{col} - f_B^{col})$, for an actively rational miner, bribing Bob and redeeming MH-Dep via path dep-M in round k strongly dominates redeeming MH-Dep via path dep-A.*

Proof: Let M_i be some actively rational miner chosen to create a block in round k where $t_{pub} \le k < T$. If M_i chooses to bribe br_i, and Bob accepts the bribe, then M_i's utility in subgame $G^{dep}(k, red)$ is given by $v^{dep} - f - br_i + \bar{u}_i(G^{dep}(k+1, irred\text{-}rev))$. By Lemma 6 and Lemma 7, this expression becomes:

$$v^{dep} - f - br_i + \lambda_i(v^{col} - f) \tag{6}$$

If M_i chooses to include tx_A^{dep}, then *MH-Dep* will be redeemed via path *dep-A*. As argued in proof of Lemma 7, we argue that in this case pre_b will never be released. Hence, M_i's expected utility from choosing to include tx_A^{dep} if chosen to create block b_k is given by $f_A^{dep} - f + \bar{u}_i(G^{dep}(k+1, irred\text{-}nrev))$. By Lemma 6 and Lemma 7, this expression becomes:

$$f_A^{dep} - f + \lambda_i(f_B^{col} - f) \tag{7}$$

From Eq. (6) and Eq. (7), if $br_i < v^{dep} - f_A^{dep} + \lambda_i(v^{col} - f_B^{col})$, then M_i's expected utility from bribing is strictly higher than from redeeming *MH-Dep* via path *dep-A*. □

Lemma 11. *In any subgame $G^{dep}(k, red)$ where $t_{pub} \le k < T$, as long as $br_i < v^{dep} - f_A^{dep}$, for an actively rational miner, bribing Bob and redeeming MH-Dep via path dep-M in round k strongly dominates including only unrelated transactions.*

Proof: Let M_i be an actively rational miner creating a block in round k where $t_{pub} \le k < T$. As in Lemma 10, its utility from choosing to pay bribe br_i to Bob in round k is given by $v^{dep} - f - br_i + \lambda_i(v^{col} - f)$ (Eq. (6)).

Consider the case where the best action for M_i changes to bribe Bob in some round $\ge k+1$ and $< T$, then if M_i gets chosen to create a block in that round and *MH-Dep* is still redeemable, the utility earned in that round would still be given by $v^{dep} - f - br_i + \lambda_i(v^{col} - f)$. Since the miner would get chosen in such a round with probability < 1, it follows that bribing Bob in round k strictly dominates by deferring the bribe to some later round $< T$. Consequently, the only other action for M_i would be to consider including only unrelated transactions till timeout T.

Suppose M_i includes only unrelated transactions until the timeout, then there are three possible cases to consider depending on the game state in the timeout round:

Case 1: $G^{dep}(T, red)$. In this case, *MH-Dep* has not been redeemed until the timeout round. M_i's expected utility is thus given by Eq. (5) and equal to $\lambda_i(v^{dep} + v^{col} - 2f - br_i)$.

Case 2: $G^{dep}(T, irred\text{-}rev)$. In this case, *MH-Dep* has been redeemed by some other miner via path *dep-M*. From Lemma 6 it follows that M_i's expected utility is given by $\lambda_i(v^{col} - f)$.

Case 3: $G^{dep}(T, irred\text{-}nrev)$. In this case, *MH-Dep* has been redeemed by some other miner via path *dep-A*. From Lemma 6 it follows that M_i's expected utility is given by $\lambda_i(f_B^{col} - f)$.

Consequently, M_i's expected utility from choosing to include only unrelated transactions before the timeout is upper bounded by $\max(\lambda_i(v^{dep} + v^{col} - 2f - br_i), \lambda_i(v^{col} - f))$. Clearly, when $br_i < v^{dep} - f$, M_i's expected utility from bribery in round k is strictly higher than from including an unrelated transaction. □

Lemma 12. *In any subgame $G^{dep}(k, \cdot)$ where $k \ge t_{pub}$, Bob will have higher utility from accepting any $br_i > v^{col} - f_B^{col}$ than from following the MAD-HTLC protocol.*

Proof: By the same argument as in Lemma 5, once Alice publishes pre_a in t_{pub}, Bob's highest achievable utility from following the protocol at this point is $v^{col} - f_B^{col}$. Given the game setup, Bob will only release pre_b if the attack succeeds. In this case, by Lemma 6 and Lemma 9, Bob will lose v^{col}. Consequently, Bob will not accept any bribe $br_i \le v^{col} - f_B^{col}$. On the other hand, Bob will have strictly higher utility at the end of the game from accepting $br_i > v^{col} - f_B^{col}$ than from following the *MAD-HTLC* protocol. □

Theorem 2. *If $v^{col} - f_B^{col} < v^{dep} - f_A^{dep}$ then there exists a bribe value for every actively rational miner M_i, such that in any subgame $G^{dep}(k, red)$, where $k \ge t_{pub}$, both M_i and Bob have higher expected utility from SDRBA redeeming MH-Dep via path dep-M than from following the MAD-HTLC protocol.*

Proof: Let M_i be some actively rational miner in $G^{dep}(k, red)$. When $br_i < v^{dep} - f_A^{dep}$, we observe that bribing Bob dominates (i) all other actions in round T (Lemma 9), (ii) including Alice's transaction (*dep*-*A*) in round $< T$ (Lemma 10), (iii) including unrelated transactions in round $< T$ (Lemma 11).

By Lemma 12, we further know that Bob will have higher utility from accepting br_i than from following the *MAD-HTLC* protocol if $br_i > v^{col} - f_B^{col}$.

Consequently, for bribery to result in higher utility for both M_i and Bob, we would need $v^{col} - f_B^{col} < br_i < v^{dep} - f_A^{dep}$. Thus, we know that there always exists a value for br_i that meets both constraints as long as $v^{col} - f_B^{col} < v^{dep} - f_A^{dep}$. □

We have shown that for all active miners it is better to bribe as soon as given a chance to mine a block if $v^{col} - f_B^{col} < v^{dep} - f_A^{dep}$. However, this does not mean that *MAD-HTLC* is safe given that $v^{col} - f_B^{col} > v^{dep} - f_A^{dep}$. Next, we show the constraints required for *MAD-HTLC* to be safe against *SDRBA* attacks.

4.3. Safety of *MAD-HTLC* against *SDRBA*

Theorem 3. *If* $v^{col} - f_B^{col} > \frac{1}{1-\lambda_{max}} v^{dep}$ *and* $f_A^{dep} \geq \lambda_{ac}{}^{T-t_{pub}+1}(v^{dep} - f_A^{dep})$*, following MAD-HTLC protocol would be preferred by all miners over bribing Bob*

Proof: From Lemma 10 and Lemma 12, it follows that bribery would be preferred over tx_A^{dep} before round T only if $v^{col} - f_B^{col} < v^{dep} - f_A^{dep} + \lambda_i(v^{col} - f_B^{col})$, else tx_A^{dep} is preferred over bribing Bob. Thus, if $v^{col} - f_B^{col} > \frac{1}{1-\lambda_{max}} v^{dep}$, then no miner M_i would choose to bribe Bob in the current round.

From Lemma 9 and Lemma 12 it follows that if *MH-Dep* is still redeemable in round T, bribery will be strictly preferred for both an actively rational miner and Bob if $v^{col} - f_B^{col} < v^{dep} - f_A^{dep} + v^{col} - f_B^{col}$. Since $v^{dep} - f_A^{dep} > 0$, it follows that bribery will always be preferred at that time. In case all active miners choose to not include related transactions till T, the probability that the game will reach the game $G^{dep}(T, red)$ is given by $\lambda_{ac}{}^{T-t_{pub}}$. Further, probability for any active miner to win in round T would be given by λ_{ac}. Thus, the probability that any active miner receives utility from including unrelated transactions till round T is $\lambda_{ac}{}^{T-t_{pub}+1}$ and the utility is given by $\lambda_{ac}{}^{T-t_{pub}+1}(v^{dep} - f_A^{dep} + v^{col} - f_B^{col} - br_i$ and since $br_i > v^{col} - f_B^{col}$, the utility is upper bounded by $\lambda_{ac}{}^{T-t_{pub}+1}(v^{dep} - f_A^{dep})$. If utility from including tx_A^{dep} exceeds the same, then all active miners would be better off following *MAD-HTLC* protocol. □

4.4. Realizing *SDRBA*

The key property of *SDRBA* is success-dependence in that a miner only pays Bob if she can successfully attack, i.e., having tx_M^{dep} confirmed on-chain. At a high level, in order to construct tx_M^{dep}, the miner needs to know pre_b, and the miner needs to pay Bob for an agreed-upon bribe. However, performing this exchange *fairly* is challenging: as soon as Bob releases pre_b to M_i, it is in the best interest of M_i to not pay Bob, and vice versa. On the other hand, though, if Bob does not release pre_b, how could M_i construct tx_M^{dep} and have it confirmed on-chain?

Our key observation is that, in most blockchain implementations, miners need not know the content of transactions to mine a block that includes them. Specifically, PoW mining is typically done over a block header, which only includes a compact representation of the transaction (e.g., a Merkle root). Therefore a miner can start mining knowing only the hashes of all transactions (from which the Merkle root can be calculated). In PoS, transactions bind to the block header via a signature from M_i, which again can be generated from transaction hashes.

Protocol skeleton. With this idea in mind, we can achieve fair exchange as follows: Bob prepares a transaction tx_M^{dep} that redeems *MH-Dep* for M_i, and sends $h = \mathsf{H}(tx_M^{dep})$ to the miner, along with a proof π showing its correctness. For instance, Bob and the miner can agree on a transaction template with pre_b missing, and in π, Bob proves that the hash of the template filled with a particular pre_b matches h. Such proofs can be produced with zero-knowledge proofs (e.g., [24]) or Trusted Executed Environment (TEE) such as Intel SGX [25]. The miner verifies the proof and mines a partial block B including: 1) the hash of tx_M^{dep} (again, the miner only needs the hash h for mining), 2) a bribing transaction that pays Bob *br* tokens from the coinbase. (Paying Bob from the coinbase ensures that the validity of the payment does not depend on other transactions.) and 3) any other transactions from the mempool. The miner sends B to Bob, who verifies that the block includes intended transactions, fills in tx_M^{dep}, and broadcasts the completed block. For ease of argument, we assume v^{dep} is smaller than the block reward (current 6.25BTC in Bitcoin), to disincentivize M_i from forking B (e.g., replacing it with a block without payment to Bob). In practical uses of HTLC, v^{dep} is typically much smaller than block rewards (the average Lightning capacity is 0.045BTC [5]).

An instantiation with TEEs. Below we present a concrete instantiation using TEEs. We assume Bob has access to a TEE (e.g., Intel SGX) that guarantees integrity and supports remote attestation. However, we do not require confidentiality guarantees (i.e., it only requires transparent execution environments [26]) since Bob knows the secret anyway. There is extensive literature on SGX and we refer readers to [27] for background. For ease of exposition, we state the protocol assuming Bitcoin, but it can be easily adapted to other PoW or PoS blockchains.

1) Setup: Bob and M_i negotiate the details of the bribery, including the miner's address to receive the redemption of *MH-Dep*, $addr_{Miner}$, the amount of bribe *Amount*, Hash_{pre_b}, and Hash_{pre_a}. Bob instantiates a TEE running code in Fig. 7. For ease of exposition, the bribery parameters are hardcoded in Fig. 7, so they are covered by TEE attestations. Bob shares the source code with the

```
Pseudocode of the TEE enclave for success-dependent bribery
Hardcoded:
  $addr_{Miner}$: Miner's address to receive $tx_M^{dep}$
  $\mathsf{Hash}_{pre_b}$: the hash of $pre_b$
  $\mathsf{Hash}_{pre_a}$: the hash of $pre_a$
Function GetHashOfTxn($tx_M^{dep}$):
  Assert that $tx_M^{dep}$ is redemption to $addr_{Miner}$
  Assert that $tx_M^{dep}$ contains ($pre_a$, $pre_b$), s.t.
      $H(pre_b) = \mathsf{Hash}_{pre_b}$, $H(pre_a) = \mathsf{Hash}_{pre_a}$
  $h = \mathsf{H}(tx_M^{dep})$
  $\sigma_{\text{TEE}} =$ TEE.attestation($h$) // $\sigma_{\text{TEE}}$ binds $h$ to the code
  return $(h, \sigma_{\text{TEE}})$
```

Figure 7. TEE enclave program used in *SDRBA* implementation

miner who can verify its correctness. This can happen well before the timeout T of *MAD-HTLC*.

2) Bob: When Alice releases pre_a, Bob constructs the redemption transaction tx_M^{dep}, and calls `GetHashOfTxn` in TEE (Fig. 7) to compute a hash $h = \mathsf{H}(tx_M^{dep})$ along with an attestation σ_{TEE} proving that h is computed by the specific code that Bob shared with the miner earlier. Bob sends (h, σ_{TEE}) to the miner.
3) Miner: Miner verifies σ_{TEE} against its copy of the source code and checks that h is certified by σ_{TEE}. Then the miner builds a Merkle tree as described before. Then M_i starts mining. After finding a valid block B, M_i sends B to Bob
4) Bob: After receiving B, Bob verifies that 1) B includes a proper payment to him and then completes B with tx_M^{dep} to the peer-to-peer network.

Security arguments. Under the assumption that TEE protects integrity and supports remote attestation, the attestation σ_{TEE} guarantees that the provided hash is valid. When Bob receives and verifies a partial block B from the miner, he can choose whether to fill in the missing transaction and broadcast it or not. If he chooses not to broadcast, he forgoes the bribe. Thus, Bob would only receive back v^{col}, which as we showed in Theorem 2, is smaller than the bribe he receives if he chooses to broadcast. It is important to note that Bob cannot withhold the block but only send the bribe transaction because the bribe transaction is only valid if B is confirmed on-chain, as it spends the coinbase of B. We consider a few variants of *SDRBA*, replacing the proposed TEE instantiation with the following constructions

ZKPs in place of TEEs. Instead of TEEs, Bob can prove the correctness of h using zero-knowledge proof [24]. This removes the reliance on TEE security (although we emphasize that the presented protocol only requires integrity, not confidentiality), but ZKPs are typically orders of magnitude slower than equivalent implementation using TEEs.

A protocol with TEE on the miner's side. We can also adapt the construction in [15] to get another implementation of *SDRBA* where the miner hosts the TEE. Briefly, Bob sends encrypted pre_b to a TEE, which decrypts it upon receiving a complete proof-of-work block satisfying certain criteria. Then the miner completes the block and broadcasts it. This variant necessarily relies on the confidentiality guarantee of TEEs and works with PoW blockchains. The upside, however, is that the miner does not rely on Bob's rationality for broadcasting. A similar construction appeared in MEV-SGX [15] although in a different context.

5. Hybrid Delay-Reverse Bribery Attack

In this section, we present *HyDRA*, the third attack on *MAD-HTLC* that combines a delay attack and *SDRBA*. Not only does *HyDRA* maintain the no-risk feature of *SDRBA*, but it also works regardless of relative sizes of v^{dep} and v^{col} with constant probability. We present the attack in this section with a "pay per block" delay strategy, under which for each block, Bob pays the miner, in expectation, an amount greater than the transaction fee offered by Alice.

5.1. The *HyDRA* attack

There are two steps to the attack: censoring Alice's transaction and an *SDRBA* attack between Bob and miner. Below we go through the two steps. Figure 9 specifies the protocol more formally.

Step 1: Censoring tx_A^{dep}: As soon as Alice posts the redemption transaction tx_A^{dep} in t_{pub}, Bob sets up a contract that issues *promised rewards* to miners who censor tx_A^{dep} (Fig. 8). The intuition is that given a promise of receiving enough amount in the future, miners would prefer to not include tx_A^{dep}.

The attack can be facilitated with a smart contract. We outline a possible implementation in Fig. 8. At a high level, for each subsequent block, after tx_A^{dep} is published, its miner can call `getToken` to get special tokens that can be redeemed only after the second step of the attack succeeds (at which point Bob pays the cost of censoring tx_A^{dep} using the bribe he receives from the miner). The total cost of censoring tx_A^{dep} is denoted by C_{delay}.

Step 2: Mounting *SDRBA*: The second part of the attack is similar to the *SDRBA* attack (Section 4.4) except that the fair exchange is slightly modified to guarantee that the miner M_i pays the bribe if and only if both *MH-Dep* and *MH-Col* are redeemed by M_i in the same block (whereas in the original *SDRBA* attack, M_i pays bribe if and only if *MH-Dep* is redeemed).

Following a successful *SDRBA* attack, anyone can call `redeemToken` to trigger the distribution of payouts. `redeemToken` will verify that *SDRBA* indeed succeeded (i.e., both *MH-Dep* and *MH-Col* have been redeemed after timeout via *dep*-M), and then redeem tokens issued earlier at a pre-specified exchange rate (c_1 for tokens issued before the timeout T, and c_2 otherwise as specified in Fig. 8), and send the remaining balance to Bob. The reason for choosing the specific values will become clear shortly in the analysis.

$\mathcal{SC}_{HyDRA}$: delay contract for *HyDRA*

```
Variables:
    MH-Dep, MH-Col: Address of MAD-HTLC contracts
    $n_1 = 0$ // total # of $T_1$ tokens issued
    $n_2 = 0$ // total # of $T_2$ tokens issued
    $Bal_1 = \{\}$ // mapping from address to balances of $T_1$
    $Bal_2 = \{\}$ // mapping from address to balances of $T_2$
    // $c_i$ is the exchange rate between $T_i$ and native currency.
    $c_1 = \frac{f_A^{dep}}{1-(1-\lambda_{ac})^{1/\lambda_{ac}}} + \epsilon_1$ // $\epsilon_i$ is any small positive number.
    $c_2 = \frac{f_A^{dep}}{\lambda_{ac}} + \epsilon_2$
Setup:
    Bob deposits $v^{dep}$ in the contract

Function getToken():
    Abort if the caller is not the miner of the current block
    Abort if this function has been called in this block
    Let $i$ be the current block number
    if $i < T$
        $n_1 \leftarrow n_1 + 1$
        $Bal_1[caller] \leftarrow Bal_1[caller] + 1$
    else
        $n_2 \leftarrow n_2 + 1$
        $Bal_2[caller] \leftarrow Bal_2[caller] + 1$

Function redeemToken():
    Assert that $v^{dep} - n_1 \cdot c_1 - n_2 \cdot c_2 > 0$
    Check that MH-Dep and MH-Col were redeemed through
dep-M and col-M in this block.
    for (addr, bal) in $Bal_1$
        Send $c_1 \cdot bal$ native tokens to addr
    for (addr, bal) in $Bal_2$
        Send $c_2 \cdot bal$ native tokens to addr
    // Send leftover to Bob
    Send $v^{dep} - n_1 \cdot c_1 - n_2 \cdot c_2$ to Bob

Function refundToken():
    If MH-Dep has been redeemed by Alice through dep-A, send $v^{dep}$ to Bob.
```

Figure 8. A smart contract (sketch) that facilitates the delay phase of *HyDRA*. Before the timeout T, the contract issues token T_1 to miners who censor tx_A^{dep}, after the timeout it issues token T_2. Tokens T_1 and T_2 can be redeemed to native tokens (e.g., Ether on Ethereum) only after the *HyDRA* attack succeeds. Each T_i will be redeemed to c_i native tokens.

Protocol for *HyDRA*

Setup and Init:

Bob sets up a facilitating smart contract $\mathcal{SC}_{HyDRA}$ as described in Fig. 8

Before round T:

Miners of blocks between t_{pub} and T censor tx_A^{dep} and call `getToken` in $\mathcal{SC}_{HyDRA}$ to get tokens.

In and after round T:

If a miner is active, bribe Bob with $br = v^{col} + n_1 \cdot c_1 + n_2 \cdot c_2$ to mount the modified *SDRBA* attack (see the description of step 2). If the attack succeeds, call `redeemToken` to distribute payouts.

If a miner is passive, keep censoring tx_A^{dep} (same as above) until *SDRBA* succeeds.

Figure 9. Protocol followed during *HyDRA* attack

5.2. Security Analysis

Game setup. The game for *HyDRA* is the same as that Section 3.3 except that the action to censor tx_A^{dep} is always available for all rational miners—active or passive—in all rounds (including rounds before and after the timeout T). In rounds $t > T$, active miners can choose to mount *SDRBA*.

Analysis. We analyse the incentives of the protocol proposed in Fig. 9, which we refer to as *HyDRA*. To analyze the incentive compatibility of *HyDRA*, we first show that given a choice, an active miner would always choose to mount *SDRBA* (Step 2) over censoring tx_A^{dep} (Step 1). The intuition behind this can be explained as follows. Since Bob is always receiving a fixed amount more than the v^{col} and C_{delay} he incurs, it is essentially the active miner who mounts the *SDRBA* paying every other miner to censor the transaction. If the miner chooses to get paid by the contract, it would either be paying itself or some other miner would share some profit with the miner. We consider every miner to be rational and some of them are active ($\lambda_{ac} > 0$). The values of c_1 and c_2 are described in the contract Fig. 8.

Lemma 13. *For an active rational miner in round $t \geq T$, where mounting SDRBA is an available action (i.e., MH-Dep and MH-Col are still redeemable) and $v^{dep} - (T - t_{pub} - 1) \cdot c_1 - (t - T) \cdot c_2 > f_A^{dep}$, mounting SDRBA (Step 2) dominates censoring tx_A^{dep} (Step 1).*

Proof: The total available amount for any miner of round t is $v^{dep} - (T - t_{pub} - 1) \cdot c_1 - (t - T) \cdot c_2$. This is also the utility for the miner who chooses to perform *SDRBA* in the current round. If the miner instead chooses to censor tx_A^{dep} and get c_2 bribe, and is chosen to build the next block in round t', the expected utility would be at most $c_2 + v^{dep} - (T - t_{pub} - 1) \cdot c_1 - (t' - T) \cdot c_2$, even considering that no other active miner between time t and t' chooses to take Step 2. Since $t' \geq t+1$, this utility is lower than or equal to the utility earned by the active miner choosing to mount *SDRBA* in round t. Thus, given that *SDRBA* is available, all active rational miners will take that action, instead of potentially sharing the revenue with other miners. □

Next, we show that no rational miner would ever include tx_A^{dep} in presence of such a contract, before or after the timeout.

Lemma 14. *For any rational miner of round $t \geq T$, where MH-Dep and MH-Col are not yet redeemed and $v^{dep} - (T - t_{pub} - 1) \cdot c_1 - (t - T) \cdot c_2 > f_A^{dep}$, censoring tx_A^{dep} and accepting delay bribe (Step 1) dominates over including tx_A^{dep}.*

Proof: In each round $t \geq T$, given the choice between including tx_A^{dep} and censoring it, a rational miner, chosen to mine the current round, would have a utility of f_A^{dep} if it chooses to include tx_A^{dep}, whereas a miner who chooses to censor tx_A^{dep}, can call the contract with proof of not including the transaction and get a future amount of c_2, which we have set to be $> f_A^{dep}/\lambda_{ac}$ (Fig. 8), with a potential to earn more in the future rounds. With probability λ_{ac}, the next block would be mined by an active miner, who as shown in Lemma 13, will choose to perform *SDRBA* step and make the attack successful. Thus, the probability of receiving the future amount of c_2 is at least λ_{ac}. Therefore, utility from censoring the transaction is given by $u > \lambda_{ac} \cdot (f_A^{dep}/\lambda_{ac})$. Thus, it is rational for all miners to censor tx_A^{dep} than to include it on-chain. □

With the above Lemma 13 and Lemma 14, we have established the value of c_2 and looked at the actions available for all rational miners after timeout T. Next, we need to show that it would be rational for miners before timeout to censor tx_A^{dep}, and establish the amount (c_1) required to be promised to do so. Since after timeout T, the probability of active miner mining a block is λ_{ac}, the expected number of blocks required after timeout would be $1/\lambda_{ac}$.

Lemma 15. *For any rational miner of round $t < T$, where MH-Dep has not yet been redeemed and $v^{dep} - (T - t_{pub} - 1) \cdot c_1 + (1/\lambda_{ac}) \cdot c_2 > f_A^{dep}$, censoring tx_A^{dep} and accepting delay bribe (Step 1) dominates over including tx_A^{dep}.*

Proof: In each round, given the choice between including tx_A^{dep} and censoring it, a rational miner chosen to mine the current round would have a utility of f_A^{dep} if it chooses to include tx_A^{dep}, whereas a miner who chooses to censor tx_A^{dep}, can call the contract with proof of not including the transaction and get a future amount of c_1. Since the expected number of blocks the attack lasts after timeout is given by $1/\lambda_{ac}$, the probability with which the attack succeeds until that round is given by $1 - (1 - \lambda_{ac})^{1/\lambda_{ac}}$. We have set $c_1 > \frac{f_A^{dep}}{1-(1-\lambda_{ac})^{1/\lambda_{ac}}}$ (Fig. 8), which gives the miner a utility greater than f_A^{dep}. Thus, it is rational for all miners to censor tx_A^{dep} than to include it on-chain. □

Theorem 4. *All rational miners are better off from following strategy outlined in HyDRA, than behaving honestly and redeeming transaction tx_A^{dep} for Alice, given $v^{dep} - (T - t_{pub} - 1) \cdot c_1 + (1/\lambda_{ac}) \cdot c_2 > f_A^{dep}$.*

Proof: From lemmas 14 and 15, if $v^{dep} - (T - t_{pub} - 1) \cdot c_1 + (1/\lambda_{ac}) \cdot c_2 > f_A^{dep}$, censoring tx_A^{dep} dominates including tx_A^{dep}. After timeout T, the censoring bribe needs to be paid for an expected $\frac{1}{\lambda_{ac}}$ blocks. With the same condition $v^{dep} - (T - t_{pub} - 1) \cdot c_1 - \frac{1}{\lambda_{ac}} \cdot c_2 > f_A^{dep}$ by Lemma 13, all active miners prefer to participate in *SDRBA* over censoring tx_A^{dep} and by extension including tx_A^{dep}. Thus, all rational miners will follow the *HyDRA* strategy. □

Success probability. As shown in Theorem 4, the success probability is $1 - (1 - \lambda_{ac})^{1/\lambda_{ac}}$, since at least one block needs to be mined by an active rational miner in $\frac{1}{\lambda_{ac}}$ blocks. However, *HyDRA* is incentive-compatible regardless—all rational miners are better off from following *HyDRA* as long as c_1 and c_2 are set accordingly.

Cost of defending the hybrid attack. One way to defend against *HyDRA* is for Alice to pay a high fee, and publish tx_A^{dep} early. We provide a rough estimation of the cost. Consider a HTLC contract with capacity $v^{dep} = 2$ BTC and timeout $T = 30$ days (typical configuration in Lightning network [5]). Estimating λ_{ac} accurately is hard, but the adoption rate of MEV-geth [28] can provide a ballpark reference because only active miners (by our definition) will prefer MEV-geth over geth. As of March 2022, the adoption rate is about 85% [29]. If Alice closes the channel one week prior to its timeout, then $T - t_{pub} = 7 \times 24 \times 60/10 = 1008$ blocks and thus the transaction fee cost f_A^{dep} to Alice in order to violate the condition in Theorem 4 must be $f_A^{dep} = v^{dep} - (T - t_{pub} - 1) \cdot c_1 - \frac{1}{\lambda_{ac}} \cdot c_2$, where $c_1 = f_A^{dep}/0.9, c_2 = f_A^{dep}/0.85$, which makes $f_A^{dep} \approx 0.0018$ BTC which is about 818× the average closing cost of 2.2e-6 BTC [9]. The utilization of the channel lifetime is lowered to $\sim 77\%$.

Optimizing the attack. We have shown a basic version of a delay attack. Using attacks shown in [8], this bound can be significantly reduced. We can directly use an attack that relies on delaying till round T and use our strategy above; alternatively delay till $T + \frac{1}{\lambda_{ac}}$ while allowing *SDRBA* action. Further, even if the timed delay phase expires, we argue that the following lemma holds for rational miners who have been promised some utility in the delay phase.

Lemma 16. *In round $t \geq T$, if a passively rational miner M_i with utility $C_i > f_A^{dep} - f$ conditional on the success of SDRBA is chosen to create the block in round t, M_i will only include transactions unrelated to MH-Dep and MH-Col.*

Thus, they would still not choose to include tx_A^{dep}, even if they are not able to receive any further delay bribe.

6. He-HTLC: An Incentive Compatible HTLC

We now present He-HTLC, a protocol where all parties are incentivized to follow the HTLC spec. Our protocol is inspired by the learnings of our attacks on *MAD-HTLC*. In particular, we ensure that (i) the miners are not overcompensated when acting as enforces, and (ii) separate the redemption of v^{dep} and v^{col} some blocks apart. Since we have presented a detailed intuition behind these changes in Section 2.3, we proceed with describing the protocol.

Our protocol He-HTLC$[\ell]$ is parameterized by ℓ, the number of blocks between redemption of v^{dep} and v^{col}. He-HTLC$[\ell]$ consists of two contracts: *He-Dep* and *He-Col*, to which Bob deposits v^{dep} and v^{col} respectively. Similar to *MAD-HTLC*, v^{dep} is intended to be paid to Alice, if Alice reveals a secret before the timeout T, otherwise to Bob; and v^{col} is refunded to Bob in case the exchange occurs *honestly*. More formally, the redemption paths of *He-Dep* and *He-Col* are specified in Eqs. (8) and (9).

***He-Dep*:**

$$\begin{aligned} &(pre_a, sig_a) && \text{// } \textit{dep-A}\text{: redemption by Alice} \\ &(pre_b, sig_b, t \geq T) && \text{// } \textit{dep-B}\text{: refunding Bob after } T \end{aligned} \tag{8}$$

***He-Col*:**

$$\begin{aligned} &(sig_b, t \geq T, \pi_{A \vee B}[\ell]) && \text{// } \textit{col-B}\text{: refunding Bob after } T \\ &(pre_a, pre_b, \pi_B) && \text{// } \textit{col-M}\text{: Anyone can redeem} \\ & && v^{col} - v^{dep}\text{, burning the rest.} \end{aligned} \tag{9}$$

where,

- $\pi_{A \vee B}[\ell]$ is a proof showing *He-Dep* has been redeemed through either *dep-A* or *dep-B*, ℓ blocks ago.
- π_B is a proof showing *He-Dep* has been redeemed through *dep-B* (perhaps in the same block).

At a high level, the protocol will proceed through different paths as follows:

- *dep-A*: Under an honest execution, when Alice reveals a preimage pre_a of the hash lock, *He-Dep* is redeemed for Alice.
- *dep-B*: In case Alice does not reveal pre_a within timeout, Bob gets back the deposit by revealing pre_b.
- *col-B*: Bob receives a refund of collateral by showing proof that *He-Dep* was redeemed ℓ blocks ago.
- *col-M*: In case both pre_a and pre_b are known, the miner can get $v^{col} - v^{dep}$ from *He-Col*, by proving *He-Dep* has been redeemed. We require $v^{col} > v^{dep}$.

In what follows, we prove the incentive compatibility of He-HTLC. We leave the implementation for future work.

6.1. Security Analysis

We will show that He-HTLC is incentive-compatible for any $v^{col} \in (v^{dep}, 2v^{dep}]$ (we require this throughout this section). Note that the lower v^{col} the more desirable He-HTLC is for Bob since he needs to put down less collateral. To be a bit more specific, we will show for any $0 < \epsilon < 1$ and $v^{col} = (1+\epsilon)v^{dep}$, there always exists an sufficiently large ℓ such that all parties are incentivized to follow He-HTLC[ℓ], as long as $v^{col} - v^{dep} > f$.

Claim 1. *Suppose Bob has redeemed He-Dep through dep-B in round $r \geq T$ and paths col-B or col-M have not been taken. Then, for all miners in subsequent rounds, unless it is bribed with an amount $> v^{col} - v^{dep}$, it has a higher utility in taking col-M in comparison to including unrelated transactions.*

Proof: Since *He-Dep* was redeemed through *dep-B*, subsequent miners know pre_b and can produce π_B. If it follows path *col-M* in Eq. (9), it receives a utility of $v^{col} - v^{dep}$. If the miner includes unrelated transactions, it would earn a fee f. Since we assume $v^{col} - v^{dep} > f$, all subsequent miners would prefer *col-M*.[3] □

Now, we show an upper bound on how much Bob can pay after *dep-B*.

Claim 2. *Once Bob redeems He-Dep through dep-B, Bob would never choose to bribe more than a sum of v^{col}.*

Proof: This follows from the fact that the Bob can only earn v^{col} through the path *col-B*. Paying a higher amount as a bribe strictly reduces Bob's utility. □

Lemma 17. *Suppose Alice publishes tx_A^{dep} to redeem He-Dep via dep-A in round $t_{pub} < T$. Suppose Bob has redeemed He-Dep via dep-B. Suppose the parameter ℓ is such that the expected number of distinct miners within ℓ blocks is κ. Then, among these distinct miners, there exists a miner who has a higher utility in taking the path col-M when $v^{col} \geq (1 + \frac{1}{\kappa-1})v^{dep}$.*

Proof: Since both pre_a and pre_b are available, path *col-M* can be taken. In addition, if Bob provides sig_b, path *col-B* can be taken ℓ blocks after *dep-B*. There are κ distinct miners deciding between choosing *col-M*, *col-B*, or not taking either path. From Claim 1, we know that *col-B* can only be taken if each of the κ miners is offered a bribe $> v^{col} - v^{dep}$. Moreover, from Claim 2, we know that Bob can offer a maximum bribe of v^{col} distributed among all the κ miners. At least one of these bribes to some miner needs to be $\leq \frac{v^{col}}{\kappa} = v^{col} - v^{dep}$. Let M_i be the first such miner with the lowest round number. All miners before M_i cannot take *col-B* (since they are within ℓ blocks from *dep-B*). As per Claim 1, M_i is incentivized to take *col-M*. □

Theorem 5. *Assuming all parties are (actively or passively) rational, and $(1+\frac{1}{\kappa-1})v^{dep} \leq v^{col} \leq 2v^{dep}$ for some integer $\kappa \geq 2$, ℓ is set such that expected number of miners across ℓ blocks is κ, our protocol He-HTLC[ℓ] provides the following guarantees:*

1) *If (pre_a, sig_a) is available to miners at round $t < T$, then all parties prefer taking dep-A followed by col-B.*
2) *If (pre_a, sig_a) is not available to miners in any round $t < T$, then all parties prefer dep-B followed by col-B.*

Proof: We prove two guarantees separately.

Part 1. If (pre_a, sig_a) is available to miners at round $t < T$, the following two options are possible.

1) Bob does not reveal pre_b.
2) Bob reveals pre_b to miners.

First, we will show that in this case, Bob will always prefer the first option over the second. If Bob does not reveal pre_b to the miners, Alice redeems v^{dep} through *dep-A* (since $f_A^{dep} > f$), and eventually Bob redeems v^{col} through *col-B*.

[3] Observe that there may exist external mechanisms where *all* miners and Bob together enter into an agreement where *col-M* is never taken and together they receive and share $v^{dep} + v^{col}$; if a miner does not follow the agreement, it penalizes the miner. However, due to the permissionlessness of the system, adhering to such an agreement cannot be enforced. In particular, a miner can always mine using a new key pair and obtain $v^{col} - v^{dep}$ through *col-M*.

The utility of Alice and Bob are v^{dep} and v^{col} respectively. Each of the miners can earn a fee f_A^{dep} and f_B^{col} respectively.

In the second option, Bob reveals pre_b to miners. Observe that choosing this option offers no additional benefit for Bob. Given that pre_a is already known to miners, if *He-Dep* were to be redeemed for Bob via path *dep*-B it follows from Lemma 17 that *He-Col* would be redeemed by some miner via path *col*-M. In this case, the sum of Bob's and all miners' expected payoff would only be v^{col} (which is the same as Bob's expected payoff in the honest case). If miners choose to still redeem *He-Dep* for Alice via path *dep*-A even though both pre_a and pre_b are known, only then would Bob's expected payoff be v^{col}, which he could also have received from not revealing pre_b in the first place. Therefore, Bob will always choose not to reveal pre_b when pre_a is already known.

Thus, if (pre_a, sig_a) is available to miners at round $t < T$, then all parties are incentivized to take paths *dep*-A followed by *col*-B. Note that even if pre_b were known, contrary to *MAD-HTLC*, a miner could not redeem *He-Col* via path *col*-M if *He-Dep* was redeemed via path *dep*-A.

Part 2. Let us consider that Alice does not reveal pre_a at any time $< T$. Observe that if Alice does not have access to pre_a, then the only available path is *dep*-B followed by *col*-B. Thus, we now assume Alice does know pre_a at round $< T$. Then the following options are possible:

1) Bob reveals pre_b to miners at some round $\geq T$. Alice never reveals pre_a.
2) Bob reveals pre_b to miners at some round $\geq T$. Alice reveals pre_a in some round $\geq T$ possibly in exchange for a bribe from a miner.

Observe that in the first option, the only available paths are *dep*-B followed by *col*-B. Alice, Bob, and miners have a utility of 0, $v^{dep} + v^{col}$ and transactions fees respectively.

In the second option, hypothetically if Alice had revealed pre_a at round $< T$, she would earn a utility of v^{dep}. Thus, the only reason to delay revealing pre_a is if she can obtain a bribe $> v^{dep}$. This may be possible if Alice is colluding with miners to cheat Bob. However, due to the need for π_M in redeeming *col*-M for $v^{col} - v^{dep} < v^{dep}$, the paths *dep*-$A$ and *col*-M are exclusive. Thus, since the miner's earnings through this path is $v^{col} - v^{dep} < v^{dep}$, it is not incentive compatible for them to pay a bribe $> v^{dep}$ to Alice.

Thus, if pre_a is known to Alice, sharing it with the miners yields her the highest utility of v^{dep}. Hence, if (pre_a, sig_a) is not available to miners in any round $t < T$, it must be the case that Alice does not have access to it either. Thus, the only available path is *dep*-B followed by *col*-B. □

7. Related Work

Incentive attacks in blockchains. Bribery attacks specifically against HTLC have been proposed. [8] proposes temporary censorship attacks where attackers can censor transactions from a victim until a timeout. Their attacks apply to HTLC and would enable Bob to censor Alice's transaction and revert the payment. Attacks in [8], require expressive smart contracts languages such as Solidity in Ethereum. [9] proposed a variant that works with Bitcoin script, along with other improvements.

In general, the incentive compatibility of blockchain protocols has been studied extensively. E.g., selfish mining attack [30] show that Bitcoin is not incentive-compatible and miners may gain more by strategically withholding blocks. External forces can also affect miners' behaviors. Bonneau et al. [10] shows how to *bribe* miners, possibly using the blockchain itself, to take over the system for a short duration. McCorry et al. [11] use smart contacts to enable expressive bribery attacks, such as censoring transactions and even rewriting the history. Other ways to implement bribery have been proposed. E.g., [31] shows that one can embed bribes in transaction fees to incentivize history rewrite attacks.

However, these works did not consider the possibility that miners may be incentivized to initiate reverse bribery attacks. Also, our attacks require a more sophisticated bribery mechanism, since miners initiate the attack.

Fair exchanges solutions using blockchains. We devised new fair exchange protocols to enable reverse bribery. Compared to various fair exchange protocols proposed in other contexts [19]–[21], what our attacks require is more challenging to achieve because of the power asymmetry between the participants of the exchanges: miners have much more control of the blockchain than Bob. First, proposed fair exchange protocols focus on the fair exchanges between two non-mining parties, whereas in our setting the exchange takes place between Bob and miners. Second, existing solutions typically rely on miners as trusted parties to facilitate the exchange, while in our setting rational miners may try to compromise the exchange in order to maximize their gain.

Actively rational miners and MEV. Miners are typically modeled as rational agents that pick the most profitable transactions to include in blocks (e.g., [9]–[11], [31]). We show that they can do better if they actively create more opportunities. In a sense, our model is a generalization of that in [32], which points out that miners can gain significant profit by manipulating transaction orderings and mounting, e.g., frontrunning attacks. This surplus is referred to as Miners Extractable Value or MEV. In practice, miners do initiate protocols on top of regular mining to get MEV, e.g., by joining MEV extraction networks such as Flashbots and Eden Network [15], [16] or offer direct RPC channels for a fee [17]. The reverse bribery attacks we show are a different avenue of MEV from influencing other (non-mining) participants.

8. Discussion and Future Work

In this section, we discuss our model, attacks, and our solution, in terms of parameter choices, implications in practice, and interesting open questions.

On actively rational miners. Miners are typically modeled as rational agents, but our key observation is that we must consider different aspects of rationality. *MAD-HTLC*, for example, only considers *passively* rational miners who optimize over passively available opportunities, whereas this paper shows that they can do better if they actively create more opportunities. As discussed above, reverse bribery can be viewed as another avenue of MEV [32] through actively influencing other (non-mining) participants.

On the other hand, assuming all miners are actively rational is perhaps too strong and does not reflect reality. If that were true, all miners would extort any entity trying to get a refund from an HLTC. Clearly, we do not live in such a world *currently*. Nonetheless, we do observe some miners engaging in active MEV extraction. Identifying how reality corresponds to the actively rational model is an interesting open question.

Comparing the three attacks and our solution. We present three different attacks. The first attack *SIRBA* serves as a warmup showing there exists *a* strategy that would be preferred by Bob and an actively rational miner compared to following *MAD-HTLC*. *SDRBA* improves upon this by removing uncertainties associated with miner election — the attack takes place *only if* the actively rational miner is elected for the appropriate block. However, the attack works only when the collateral amount is small. With *HyDRA*, we remove this restriction since a "winning" miner can redeem v^{col} and v^{dep} simultaneously after T. However, the miner and Bob together need to compensate other miners with the amount C_{delay}. Thus, the preferred attack depends on the values of v^{col}, v^{dep}, and C_{delay}.

Our solution He-HTLC forces Bob to get the refund in multiple steps, which intuitively increases the cost C_{delay} to a large value where each miner needs to be paid $> v^{col} - v^{dep}$ (the amount they would receive otherwise). By making this total cost larger than v^{col}, we ensure Bob cannot afford to bribe. In that sense, our solution can be viewed as a complement to the *HyDRA* attack. We present a solution for $v^{col} = 2v^{dep}$; however, this can be extended to $v^{col} = (1 + \epsilon)v^{dep}$ (where $\epsilon \cdot v^{dep}$ is larger than the transaction fees) by increasing ℓ (and the number of distinct miners κ) appropriately. For Bob, this can be viewed as a trade-off between providing high collateral and receiving a refund quickly (small ℓ) and providing small collateral and waiting for a long time (large ℓ). Extending our solution to have $v^{col} \leq v^{dep}$, the regime where *SDRBA* works is an open question.

Assumption on the absence of forks. For simplicity, we assume that the blockchain does not fork. This is reasonable for our attacks since we only need to show that there exist scenarios where the attacks are successful. For our solution He-HTLC, we introduce the path *col-M* where miners can earn $v^{col} - v^{dep}$. However, if Alice and Bob are rational, miners would never have the opportunity to use this path. Thus, the presence of forks does not influence a miner's utility w.r.t. our contracts, and conversely, since miners cannot redeem through *col-M*, they will not create forks to achieve higher gains through this contract.

Practicality considerations. In our attacks, miners and Bob need to communicate through some off-chain channel to negotiate offers and run the exchange protocol. Though they do not exist today, such channels can be built in several ways. E.g., like [17], miners could open up channels for Bob to indicate interests. Alternatively, platforms like [28] might emerge to connect miners and malicious Bob.

Fair exchange as a solution? We use fair exchange protocols to facilitate reverse bribery but can setting up fair exchanges between Alice and the miner solve bribery attacks against HTLC to begin with? This does not work since the fair exchange does not solve the fundamental misalignment of incentives. Here is a specific attack even we assume miners do not learn pre_a. Even though the knowledge of pre_a is important to the miner while negotiating a bribe, it is not required to redeem *MH-Dep* and *MH-Col*; after the timeout, they can be redeemed using only pre_b and sig_b. This allows for a hybrid attack where Bob gets v^{dep} and v^{col} and pays the miner some bribe.

Extending reverse bribery to other systems. We considered specific attacks against *MAD-HTLC*, but whenever application-level protocols rely on miners (e.g., as enforcers), the possibility of miners' actively participating need to be considered. Thus, reverse bribery might be a concern in not just HTLC-like contracts, and there might be other attacks when miners engage in application-level logic in addition to reverse bribery.

References

[1] S. Nakamoto *et al.*, "Bitcoin," *A peer-to-peer electronic cash system*, 2008.

[2] V. Buterin *et al.*, "A next-generation smart contract and decentralized application platform," *white paper*, vol. 3, no. 37, 2014.

[3] Hash Time Locked Contracts - Bitcoin Wiki. [Online]. Available: https://en.bitcoin.it/wiki/Hash_Time_Locked_Contracts

[4] J. Poon and T. Dryja, "The bitcoin lightning network: Scalable off-chain instant payments," 2016.

[5] Lightning network statistics — 1ML - Lightning network search and analysis engine - Bitcoin mainnet. [Online]. Available: https://1ml.com/statistics

[6] M. Campanelli, R. Gennaro, S. Goldfeder, and L. Nizzardo, "Zero-knowledge contingent payments revisited: Attacks and payments for services," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, 2017, pp. 229–243.

[7] M. Herlihy, "Atomic cross-chain swaps," in *Proceedings of the 2018 ACM symposium on principles of distributed computing*, 2018, pp. 245–254.

[8] 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)*. IEEE, 2019, pp. 357–366.

[9] I. Tsabary, M. Yechieli, A. Manuskin, and I. Eyal, "Mad-htlc: because htlc is crazy-cheap to attack," in *2021 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2021, pp. 1230–1248.

[10] J. Bonneau, "Why buy when you can rent?" in *International Conference on Financial Cryptography and Data Security*. Springer, 2016, pp. 19–26.

[11] P. McCorry, A. Hicks, and S. Meiklejohn, "Smart contracts for bribing miners," in *International Conference on Financial Cryptography and Data Security*. Springer, 2018, pp. 3–18.

[12] V. Buterin. The $p + \epsilon$ attack — Ethereum Foundation blog. [Online]. Available: http://web.archive.org/web/20220121082632/https://blog.ethereum.org/2015/01/28/p-epsilon-attack/

[13] L. Breidenbach, C. Cachin, B. Chan, A. Coventry, S. Ellis, A. Juels, F. Koushanfar, A. Miller, B. Magauran, D. Moroz *et al.*, "Chainlink 2.0: Next steps in the evolution of decentralized oracle networks," 2021.

[14] P. Daia, T. Kell, I. Miers, and A. Juels. On-chain vote buying and the rise of dark daos. [Online]. Available: https://hackingdistributed.com/2018/07/02/on-chain-vote-buying/

[15] Flashbots. (2021, 05) MEV-SGX: A sealed bid MEV auction design. [Online]. Available: https://ethresear.ch/t/mev-sgx-a-sealed-bid-mev-auction-design/9677

[16] Eden Network. [Online]. Available: https://explorer.edennetwork.io/

[17] Ethermine's MEV Relay. [Online]. Available: https://ethermine.org/mev-relay

[18] W. Foxley, "Ethermine adds front-running software to help miners offset eip 1559 revenue losses," *CoinDesk*, May 2021. [Online]. Available: https://www.coindesk.com/markets/2021/03/17/ethermine-adds-front-running-software-to-help-miners-offset-eip-1559-revenue-losses/

[19] G. Maxwell, "Zero knowledge contingent payment," 2015. [Online]. Available: https://en.bitcoin.it/wiki/Zero_Knowledge_Contingent_Payment

[20] I. Bentov, Y. Ji, F. Zhang, L. Breidenbach, P. Daian, and A. Juels, "Tesseract: Real-time cryptocurrency exchange using trusted hardware," in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, 2019, pp. 1521–1538.

[21] M. Herlihy, "Atomic cross-chain swaps," in *Proceedings of the 2018 ACM symposium on principles of distributed computing*, 2018, pp. 245–254.

[22] I. Anati, S. Gueron, S. Johnson, and V. Scarlata, "Innovative Technology for CPU Based Attestation and Sealing," in *HASP*, 2013.

[23] F. McKeen, I. Alexandrovich, A. Berenzon, C. V. Rozas, H. Shafi, V. Shanbhogue, and U. R. Savagaonkar, "Innovative instructions and software model for isolated execution," in *HASP*, 2013.

[24] scipr-lab/libsnark: C++ library for zksnarks. [Online]. Available: https://github.com/scipr-lab/libsnark

[25] M. Hoekstra, R. Lal, P. Pappachan, V. Phegade, and J. Del Cuvillo, "Using innovative instructions to create trustworthy software solutions," in *HASP*, 2013.

[26] F. Tramèr, F. Zhang, H. Lin, J. Hubaux, A. Juels, and E. Shi, "Sealed-glass proofs: Using transparent enclaves to prove and sell knowledge," in *EuroS&P*, 2017.

[27] V. Costan and S. Devadas, "Intel SGX explained." *IACR Cryptol. ePrint Arch.*, vol. 2016, no. 86, pp. 1–118, 2016.

[28] Flashbots' MEV Explore. [Online]. Available: https://explore.flashbots.net/

[29] Flashbots. (2022) Transparency Dashboard. [Online]. Available: https://dashboard.flashbots.net/

[30] I. Eyal and E. G. Sirer, "Majority is not enough: Bitcoin mining is vulnerable," in *International conference on financial cryptography and data security*. Springer, 2014, pp. 436–454.

[31] K. Liao and J. Katz, "Incentivizing blockchain forks via whale transactions," in *International Conference on Financial Cryptography and Data Security*. Springer, 2017, pp. 264–279.

[32] P. Daian, S. Goldfeder, T. Kell, Y. Li, X. Zhao, I. Bentov, L. Breidenbach, and A. Juels, "Flash boys 2.0: Frontrunning in decentralized exchanges, miner extractable value, and consensus instability," in *2020 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2020, pp. 910–927.