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 such as payment channels, atomic swaps, etc. Unfortunately, HTLC is incentive-incompatible and is vulnerable to bribery attacks. The state-of-the-art solution is MAD-HTLC (Oakland'21), which proposes an elegant idea that leverages miners' profit-driven nature to defeat bribery attacks.**

In this paper, we show that MAD-HTLC is still vulnerable as it only considers a somewhat narrow set of *passive* strategies by miners. Through a family of novel *reverse-bribery* attacks, we show concrete *active* strategies that miners can take to break MAD-HTLC and profit at the loss of MAD-HTLC users. For these attacks, we present their implementation and game-theoretical profitability analysis.

Based on the learnings from our attacks, we propose a new HTLC realization, He-HTLC,[1] that is provably secure against all possible strategic manipulation (passive and active). In addition to being secure in a stronger adversary model, He-HTLC achieves other desirable features such as low and user-adjustable collateral, making it more practical to implement and use that proposed schemes. We implemented He-HTLC on Bitcoin and the transaction cost of He-HTLC is comparative to average Bitcoin transaction fees.

I. 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 *bribery attacks*. 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, violating the contract terms in HTLC and causing Alice to lose the tokens.

Tsabary et al. [9] proposed an elegant solution called MAD-HTLC that aims to defend HTLC from bribery attacks. The key idea is to require the payer to deposit collateral, and any misbehavior by the payer will lead to the collateral being confiscated by miners. MAD-HTLC ensures that rational miners will always penalize the cheating payer, thus deterring the payer from misbehaving.

However, while MAD-HTLC treats miners as rational and strategic agents (instead of assuming they are honest), it only considers *a narrow set of possible strategies*. Specifically, it assumes miners will confirm the most profitable transactions, but will not engage in other activities. This assumption, however, is often violated in reality. Due to their special position in the ecosystem, miners might be able to reap additional gains by taking strategic actions on top of mining. A notable example is miners' fast-growing involvement in MEV extraction [10]. In the year 2021, almost all dominant Ethereum miners (99.9% of Ethereum hashrate according to [11]) started to engage in for-profit "private relay" services [12]–[14] as an additional source of revenue [15].

We refer to miners' strategic actions beyond mining as *actively rational* strategies—in contrast we refer to miners performing only standard mining as *passive* since they passively pick the best opportunity made available to them, instead of creating better ones by themselves.

Motivated by the rise of actively rational strategies in real-world blockchains (e.g., [12]–[14]), we aim to understand their security implications for HTLC. Our results are twofold:

- We discover a family of attacks that render the state-of-the-art bribery-resistant realization of HTLC, MAD-HTLC, insecure with the presence of actively rational miners.
- We propose a new HTLC realization, He-HTLC, that is provably secure against all possible strategic manipulation by actively rational miners.

A. Reverse bribery attacks against MAD-HTLC

The damaging power of active rational miners is showcased through a family of novel *reverse bribery attacks* against MAD-HTLC. In these attacks, the miner bribes the payer—hence the bribery is "reverse"—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 actively rational actions because miners initiate the attack.

Key intuition. To understand the intuition behind our attacks, we briefly review the design of MAD-HTLC. It has the same

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

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, the miners will agree with Bob on the following deal: Bob will divulge certain information to enable miners to confiscate v^{dep} and v^{col}, but the miners will compensate Bob with $v^{col} + \epsilon$ 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, the above intuition assumes there was only one miner, but 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. We need to establish whether there exists a feasible range of values such that both Bob and the miner are better off through rigorous game-theoretic analyses.

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. Unlike other blockchain-based fair exchange problems (such as [16]–[18]) where miners are trusted to facilitate the exchange, we must not trust miners. To address this, we must carefully balance the power between Bob and miners to achieve desired properties.

B. Fixing HTLC with He-HTLC

Since the state-of-the-art HTLC scheme is proven insecure in the presence of actively rational miners, whether it is possible to realize HTLC securely in this stronger adversary model becomes an open question. With He-HTLC, we answer the question positively.

Intuition behind He-HTLC. Our attacks exploit two critical 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, we can execute an exchange between Bob and the miners even if they do not trust each other.

To design our incentive-compatible protocol He-HTLC, we need to ensure that a combination of (i) Alice and miners, or (ii) Bob and miners, are not incentivized to deviate from the protocol. To address the first combination, He-HTLC does not allow Alice and miners together to retrieve an amount greater than v^{dep}; this elicits an honest execution from Alice. To address the second combination, the key challenge is to ensure that neither bribery from Bob to miners (for which MAD-HTLC was introduced) nor reverse-bribery from miners to Bob (introduced in this work) is possible.

To disincentivize reverse bribery, we reduce the amount of tokens that miners can confiscate as enforcers of the contract. Thus, miners do not earn enough to divide up the loot with Bob. However, as is, this does not solve the concern for bribery attacks (for which MAD-HTLC was introduced) — the HTLC specification requires that Bob can spend *all* of the deposited tokens after time T. Thus, there is an imbalance between what Bob can earn in this situation (i.e., $v^{dep} + v^{col}$), and the largest amount miners can confiscate. Consequently, miners would still be receptive to bribes higher than the confiscation amount. To disincentivize bribery, we break the ability to atomically perform a fair exchange and utilize a combination of distrust between the miners and Bob and the ability of each of the miners to confiscate, albeit a lower amount, as enforcers. In particular, we require that Bob reveals the secret required for miners to confiscate, several blocks before he can spend the amount. Now, even if he bribes miners larger than the confiscation amount, if the number of blocks in the interim is sufficiently large, at some point he would run out of budget to bribe all of them. This ensures that Bob would follow the HTLC specification.

Practical benefits. Not only is our protocol secure in a stronger adversary model with actively rational miners, but it also achieves several other desirable features. Even setting aside the strong security guarantees, these improvements alone make it more practical to implement and deploy than other proposed schemes (e.g., MAD-HTLC and Ponyta [19]). First, our protocol specifies how much collateral the payer has to put down to render the protocol secure, which helps payers to avoid making heuristic decisions. Furthermore, it provides a knob (looking ahead, the ℓ parameter) that the payer can turn to trade-off between the collateral amount v^{col} and the time duration that the collateral must be locked for. Choosing a longer lock period allows the payer to lock less collateral, and vice versa. Thanks to this, the required collateral can be made much lower than v^{dep} which is friendly to users with low capital (whereas [19] requires collateral to be $2 \times v^{dep}$). Finally, stronger security is not achieved through much more complexity. Our protocol is simple to understand and its security analysis is clean. It can be implemented on current blockchains without requiring any changes. We report on the implementation in Bitcoin and the experiment results show that the transaction costs are below average.

C. 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 [12]–[14], miners' strategic actions besides mining have not been captured by existing models, leaving a gap in game theoretic analyses that make our attacks possible.
- We introduce novel *reverse bribery attacks* which allows actively rational miners to profitably deviate from MAD-HTLC. We analyze the profitability of our attacks in a rigorous game-theoretic model and outline their implementation using trusted execution environments or zero-knowledge proofs. We further show that reverse bribery can

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

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. In addition to stronger security guarantees, He-HTLC also achieves desirable features such as low and user-adjustable collateral. We implemented He-HTLC on Bitcoin and the transaction costs are comparable to average Bitcoin transaction fees.

II. Overview of Results

A. 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 $(\mathsf{pk}_A, \mathsf{pk}_B, v^{dep}, \mathit{pre}_a, T)$, where pk_A and pk_B are public keys of Alice and Bob, pre_a is a secret value 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., a refund mechanism 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.

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 1 depicts one of the attacks, 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 an off-chain channel and Alice releases tx_A^{dep} (paying a transaction fee f_A^{dep}) at some time before T. As shown in Fig. 1, 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. 1, 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.)

MAD-HTLC. To make HTLC bribery-resistant, Tsabary et al. [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 $(\mathit{pre}_a, \mathit{pre}_b)$. 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.

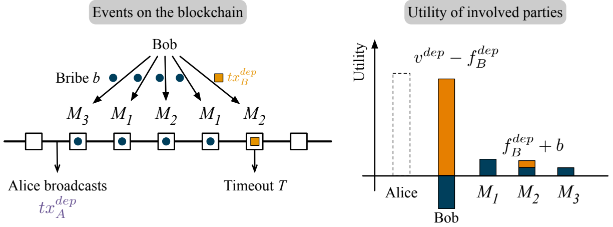


Figure 1: A delaying attack by Bob on HTLC. Bob bribes miners to censor tx_A^{dep} until the timeout, and then broadcasts tx_B^{dep}. In terms of utility, miners earn fees and bribes, and Bob earns v^{dep} at the expense of Alice (dashed rectangle denotes Alice's expected gain in honest executions).

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):

$$\begin{aligned} &\mathit{tx}_A^{dep} = (\mathit{pre}_a, \mathit{sig}_a) && \textit{// dep-A}\text{: Alice can spend with } \mathit{pre}_a \\ &\mathit{tx}_B^{dep} = (\mathit{pre}_b, \mathit{sig}_b, t \geq T) && \textit{// dep-B}\text{: Bob can spend after } T \\ &\mathit{tx}_M^{dep} = (\mathit{pre}_a, \mathit{pre}_b) && \textit{// dep-M}\text{: Anyone can spend with} \\ & && \quad \text{both pre-images} \end{aligned} \tag{1}$$

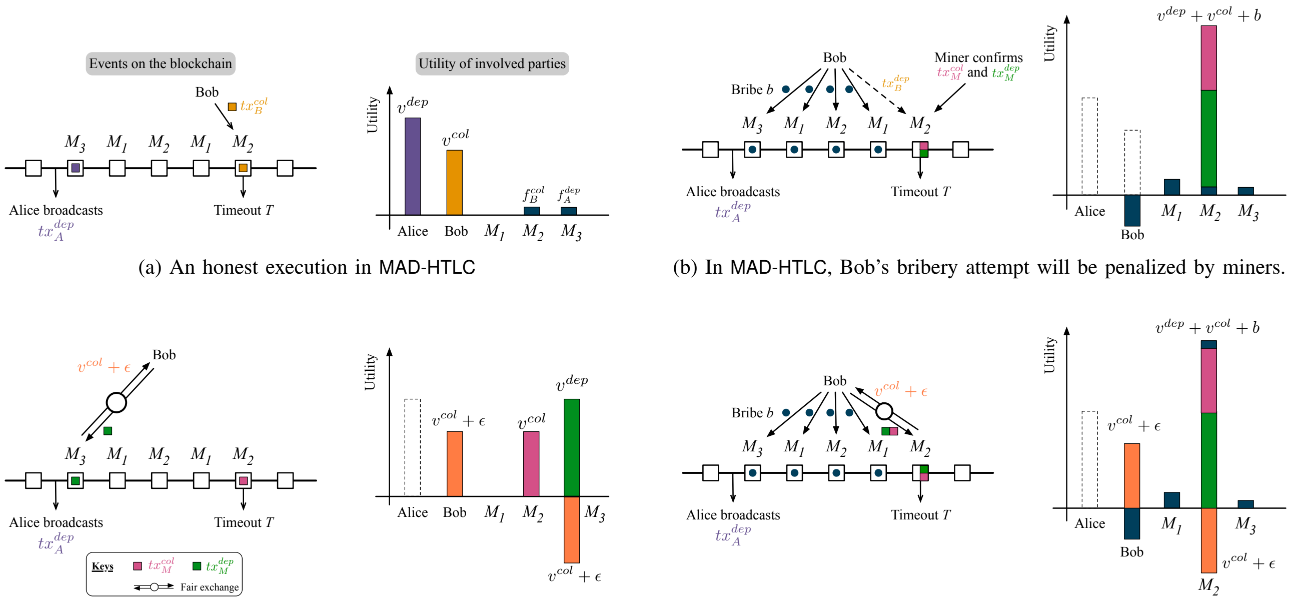
Bob's collateral v^{col} in contract *MH-Col* [9] can be redeemed in two ways:

$$\begin{aligned} &\mathit{tx}_B^{col} = (\mathit{sig}_b, t \geq T) && \textit{// col-B}\text{: Bob can spend after } T \\ &\mathit{tx}_M^{col} = (\mathit{pre}_a, \mathit{pre}_b, t \geq T) && \textit{// col-M}\text{:Anyone can spend} \\ & && \quad \text{with both pre-images} \end{aligned} \tag{2}$$

Figures 2a and 2b present two example scenarios in MAD-HTLC. In Fig. 2a, Bob is honest and he broadcasts tx_B^{col} after the timeout to get back the collateral. Figure 2b 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.

B. 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.


Events on the blockchain
Utility of involved parties
Bob
tx_B^{col}
M_3 M_1 M_2 M_1 M_2
Alice broadcasts tx_A^{dep}
Timeout T
Utility
v^{dep}
v^{col}
f_B^{col} f_A^{dep}
Alice Bob M_1 M_2 M_3
Bribe b
Miner confirms tx_M^{col} and tx_M^{dep}
$v^{dep}+v^{col}+b$
$v^{col}+\epsilon$
Keys tx_M^{col} tx_M^{dep} Fair exchange

(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 2: Example execution scenarios in MAD-HTLC where tx_A^{dep} is broadcast and reverse bribery attacks against MAD-HTLC.

To intuitively see why this is possible, compare the two scenarios in Fig. 2a and Fig. 2b, 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 two different attacks to achieve such reverse bribery.

***SDRBA*: Success-dependent reverse bribery attack.** Our first attack essentially shows that when $v^{dep} > v^{col}$, Bob and miners can successfully engage in a reverse bribery attack. We call this a *success-dependent* reverse bribery attack (in Section III-B) since in this attack a miner will only pay a bribe to Bob if it has the opportunity to redeem v^{dep} through *dep-M*. However, in the process, pre_b is revealed when v^{dep} is redeemed for a miner; thus, all miners will engage in a competition to redeem v^{col} at time T. As shown in Fig. 2c, 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) [20], [21] and the second uses zero-knowledge proofs [22].

***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. 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 second attack (Section IV) 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. 2d, 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 cost for the original delay attack (e.g., those in [8]) is $< v^{dep} - \epsilon$.

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

Finally, we devise a new HTLC protocol, called He-HTLC, that is incentive-compatible even against actively rational miners. Intuitively, any such solution needs to protect against both bribery attacks and reverse bribery attacks. More precisely, the solution needs 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. He-HTLC introduces the following two contracts to achieve incentive compatibility.

Description of He-HTLC[ℓ] between Alice and Bob

***He-Dep*:**

dep-A: If Alice reveals pre_a, send v^{dep} to Alice and v^{col} to Bob

dep-B: If Bob reveals pre_b after T, send $v^{dep} + v^{col}$ to *He-Col*

***He-Col*:**

col-B: After ℓ blocks of the funding of *He-Col*, send $v^{dep} + v^{col}$ to Bob

col-M: If both pre_a, pre_b are revealed, send v^{col} to miner and v^{dep} to $\bot$

At a high level, our solution breaks the atomicity of redemption by Bob, and for some paths, burns some tokens. To address miners' high earnings (through *dep-M* + *col-M* in MAD-HTLC), in He-HTLC, we ensure that miners can never confiscate the complete amount, $v^{dep} + v^{col}$. We partially burn some of these tokens so that they earn only up to v^{col} tokens (through path *col-M* in He-HTLC). Thus, Bob has no incentive to obtain a reverse bribe from miners. 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 reveal the secret pre_b (through path *dep-B*) for some ℓ blocks (where ℓ is a parameter) before he can receive *dep-B* + *col-B* (through path *col-B*). This separation provides two guarantees. First, if pre_a is not available, Bob would indeed receive v^{dep} + v^{col} after ℓ blocks of delay. Second, if pre_a is known, then miners have the option of either (i) confiscating v^{col} through *col-M* or (ii) letting Bob receive v^{dep} + v^{col} and potentially earning more than v^{col} through a bribe from Bob. By setting the value of the delay ℓ appropriately, and since every distinct miner would have to make this choice, we ensure that Bob cannot bribe all the miners. Such a situation where Bob attempts to bribe miners is depicted in Fig. 3; the net earnings for Bob are lower than the honest execution path. Thus, it is advantageous for Bob to just follow the protocol.

Finally, to achieve constraint (iii), i.e., Alice and miners combined cannot cheat Bob, we ensure that when $v^{col} \leq v^{dep}$, 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.

III. Reverse Bribery Attacks

In this section, we present the system model and the success-dependent reverse bribery attack (*SDRBA*), and analyze its feasibility through game-theoretical analysis.

A. System Model

Our system model is similar to that in MAD-HTLC [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 $\mathbf{M} = \{M_1, \ldots, M_n\}$.

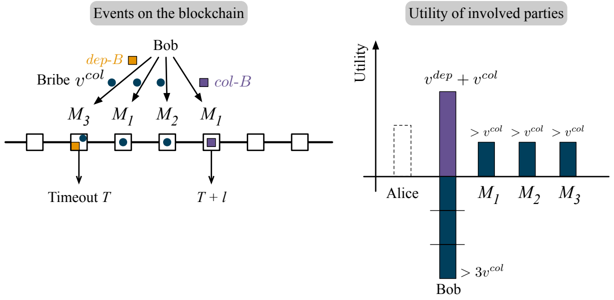


Figure 3: An attempted bribery attack by Bob on He-HTLC. Bob reveals pre_b through *dep-B* at time T, and attempts to incentivize not taking *col-M* by bribing an amount $> v^{col}$ to each miner until time $T + \ell$. Since Bob earns $v^{dep} + v^{col}$ but loses $> 3v^{col}$, Bob's attack fails.

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 VII). 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}.

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 [14], miners could open up channels for Bob to indicate interests. Alternatively, platforms like [23] might emerge to connect miners and malicious Bob.

B. Success Dependent Reverse-Bribery Attack

The intuition behind *SDRBA* is described in Section II-B. We now describe its realization in detail.

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., [22]) or Trusted Executed Environment (TEE) such as Intel SGX [24]. 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 the Lightning network, there have been only 4 (past) payment channels with capacity greater than 6.25 BTC [25]).

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.

```
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.
        $H(pre_b) = \mathsf{Hash}_{pre_b} \wedge H(pre_a) = \mathsf{Hash}_{pre_a}$
    $h = \mathsf{H}(tx_M^{dep})$
    $\sigma_{\mathsf{TEE}} = \mathsf{TEE.attestation}(h)$ // $\sigma_{\mathsf{TEE}}$ binds $h$ to the code
    return $(h, \sigma_{\mathsf{TEE}})$
```

Figure 4: TEE enclave program for *SDRBA* implementation

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. 4. For ease of exposition, the bribery parameters are hardcoded in Fig. 4, so they are covered by TEE attestations. Bob shares the source code with the 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. 4) 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 $(h, \sigma_{\mathsf{TEE}})$ 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 2) completes B with tx_M^{dep} to the peer-to-peer network.

The above protocol realizes *SDRBA*. We argue that the above protocol realizes *SDRBA*, i.e., a bribing miner only pays Bob if and only if tx_M^{dep} is confirmed on-chain. The "if" direction holds because both tx_M^{dep} and the bribe transaction are included in B. Since the miner is disincentivized to fork B, if tx_M^{dep} is confirmed, so will the bribery.

The "only if" direction holds as follows. Under the assumption that TEE protects integrity and the remote attestation scheme is secure, the attestation σ_{TEE} guarantees that the provided hash is indeed a hash of tx_M^{dep}. Therefore if B is broadcast, then the miner will receive v^{dep} from tx_M^{dep}. Moreover, 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. Finally, we just need to argue that Bob will broadcast B when bribed for more than v^{col}, which follows from the rationality assumption (if not, Bob forges the bribe and only gets v^{col}).

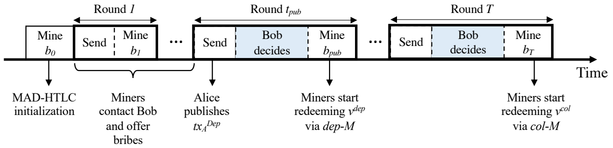

Round 1
Round t_{pub}
Round T
Mine b_0
Send
Mine b_1
...
Send
Bob decides
Mine b_{pub}
...
Send
Bob decides
Mine b_T
Time
MAD-HTLC initialization
Miners contact Bob and offer bribes
Alice publishes tx_A^{Dep}
Miners start redeeming v^{dep} via *dep-M*
Miners start redeeming v^{col} via *col-M*

Figure 5: Timeline of the *SDRBA* game.

In the next sections, we analyze the feasibility of *SDRBA* through a rigorous game-theoretical analysis.

C. Game Setup

To derive the condition under which *SDRBA* is feasible, we model it as a game G^{dep} between Alice, Bob, and miners $M_1, M_2, \ldots, M_n$ as follows.

1) Timeline of the 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 5 visualizes the timeline of the *SDRBA* game. Every round before 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 all rounds after 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 $\{red, irred\text{-}nrev, irred\text{-}rev\}$.

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

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 $t_{pub} \in [1, T)$ to publish tx_A^{dep} (c.f., Eq. (1)) with a fee f_A^{dep} of her choice. Note that $f < f_A^{dep} < v^{dep}$ is necessary for Alice's transaction to outbid all other transactions in the mempool.

After tx_A^{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 tx_B^{col} offering a fee $f_B^{col} > f$ of his choice. For ease of exposition, we assume Bob always publishes tx_B^{col} in round T even if he accepts bribes (see below). This does not change the game because tx_B^{col} does not reveal any information about pre_b. Another possible action is for Bob to engage in reverse bribery with actively rational miners, as we will describe shortly.

Passively rational miners. We recall the definition of passively rational miners from Section III-A. Passively rational miners will strategically choose transactions to mine to maximize utility. The transactions available to miners depend on the round k as well as miners' knowledge of pre_a and pre_b.

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

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

Actively rational miners and reverse bribery. We recall the definition of actively rational miners from Section III-A. In this game, actively rational miners can 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 allowing it to redeem *MH-Dep*. 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}.

Bob can independently decide whether or not to accept each bribe br_i, however, he receives only one of these bribes in case the corresponding miner, M_i, is able to mine the block (due to the fairness guarantee of *SDRBA*). For the game theoretic analysis we show that a fair exchange 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} containing pre_b in a block) is a dominant strategy over following MAD-HTLC protocol.

As a corollary of the fairness guarantee, the fair exchange process ensures the privacy of pre_b until it is revealed on the blockchain[3]. Therefore either all miners know the pre_b or none of them does. Formally, in 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 does.

3) Utility: For each player, the utility function $u_i : Action \times (\mathbb{Z}, States) \to \mathbb{R}$ 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

[3] Otherwise a miner can abort the fair exchange protocol, avoiding paying the bribe, yet is still able to redeem v^{dep} using pre_b

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 $x - f$.

D. Incentive incompatibility of MAD-HTLC.

Having presented *SDRBA* and its game theoretical model G^{dep}, we are ready to analyze the feasibility of *SDRBA*.

In this section, we will show in Theorem 1 that if $v^{col} - f_B^{col} < v^{dep} - f_A^{dep}$, all active miners are incentivized to mount *SDRBA* against MAD-HTLC as soon as given a chance to mine a block. Below we state a series of lemmas leading to the theorem, while deferring the proofs of the lemma and the theorem to Appendix B due to space constraints.

In Lemma 1, we show that in round T, if *MH-Dep* has not been redeemed by Alice, then bribing Bob with $br_i < v^{dep} + v^{col} - f_A^{dep} - f_B^{col}$ yields more utility than any other actions. Intuitively, if the miner bribes and is chosen to create a block in round T, she can redeem both v^{dep} and v^{col} together.

Lemma 1. *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, $\bar{u}_i(G^{dep}(T, red)) = \lambda_i(v^{dep} + v^{col} - 2f - br_i)$.*

In any round before T, suppose *MH-Dep* has not been redeemed yet, if a miner M_i bribes and is chosen to mine a block, she can redeem v^{dep} immediately and redeem v^{col} in round T with probability λ_i. By comparing to this expected utility, we can derive bounds on the bribe value so that bribing Bob dominates other actions.

Lemma 2. *In any subgame $G^{dep}(k, red)$ where $t_{pub} \leq k < T$, 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.*

Lemma 3. *In any subgame $G^{dep}(k, red)$ where $t_{pub} \leq k < T$, 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.*

Finally, we observe that Bob will accept any bribe value higher than $v^{col} - f_B^{col}$, the amount he will gain by following the MAD-HTLC protocol.

Lemma 4. *In any subgame $G^{dep}(k, \cdot)$ where $k \geq t_{pub}$, Bob will have higher utility from accepting any $br_i > v^{col} - f_B^{col}$ than from following the MAD-HTLC protocol.*

With above lemmas, we are ready to show that if $v^{col} - f_B^{col} < v^{dep} - f_A^{dep}$, all active miners are incentivized to mount *SDRBA* than following MAD-HTLC.

Theorem 1. *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 $k \geq t_{pub}$, 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 1), (ii) including Alice's transaction (*dep-A*) in round $< T$ (Lemma 2), (iii) including unrelated transactions in round $< T$ (Lemma 3).

By Lemma 4, 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 $v^{col} - f_B^{col} < v^{dep} - f_A^{dep}$. ■

However, this does not mean that MAD-HTLC is safe given that $v^{col} - f_B^{col} > v^{dep} - f_A^{dep}$. We refer to Appendix B for the safety constraints required for MAD-HTLC.

IV. Hybrid Delay-Reverse Bribery Attack

In this section, we present the *HyDRA* attack against MAD-HTLC. *HyDRA* combines a classic delay attack (e.g., [8], [28]) and *SDRBA*. *HyDRA* is a stronger attack than *SDRBA* because it works regardless of relative sizes of v^{dep} and v^{col}. We present the attack in this section with a "pay per block" delay strategy similar to [8] but with simplified upper bound bribe, under which for each block, Bob offers miner a bribe which offers greater utility than the transaction fee offered by Alice.

A. The HyDRA attack

Let's say a MAD-HTLC is established between Bob and Alice, with Bob depositing v^{dep} and v^{col} to *MH-Dep* and *MH-Col* respectively. Bob tries to attack Alice (who we assume knows pre_a, and can thus redeem the deposit contract) in order to receive any amount greater than v^{col}. There are two steps to Bob's attack: censoring Alice's transaction and an *SDRBA* attack between Bob and miner. Below we go through the two steps. Figure 7 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. 6). 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.[4] We outline a possible implementation in Fig. 6. 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). The total cost of censoring tx_A^{dep} is denoted by C_{delay}. This step continues until some miner takes Step 2.

Step 2: Mounting *SDRBA*: After time T, any miner can mount an *SDRBA* to move the attack to the second step. The second part of the attack is similar to the *SDRBA* attack (Section III-B) except that the fair exchange is slightly modified

[4] This does not mean that the HTLC needs to be on the same chain. Bribery on one chain can be facilitated by smart contracts on another chain through cross-chain bridges, assuming passive miners are mining on both chains.

```
$\mathcal{SC}_{HyDRA}$: delay contract for HyDRA
Variables:
  MH-Dep, MH-Col: Address of MAD-HTLC contracts
  For $i \in \{1,2\}$, $n_i = 0$ // total # of $T_i$ tokens issued
  For $i \in \{1,2\}$, $Bal_i = \{\}$ // mapping from address to balances of $T_i$
  // $c_i$ 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 6: A smart contract (sketch) that facilitates the delay phase of *HyDRA*.

so that a 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).

Note that Step 1 is a continuous process and lasts until a miner takes Step 2. 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. 6), and send the remaining balance to Bob. The reason we choose the specific values will become clear shortly in the analysis.

B. Incentive incompatibility of MAD-HTLC

Game setup. The game for *HyDRA* is the same as that Section III-C 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 $t \geq T$, active miners can choose to mount *SDRBA*. 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. 6.

Protocol for *HyDRA*

Setup and Init:

Bob sets up a facilitating smart contract $\mathcal{SC}_{HyDRA}$ as described in Fig. 6

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 7: Protocol followed during *HyDRA* attack

Analysis. To analyze the incentive compatibility of *HyDRA*, we first show that an active miner would always choose to mount *SDRBA* (moving to Step 2) over censoring tx_A^{dep} (remaining in Step 1) if chosen to create a block. The intuition is that ultimately the cost of censoring tx_A^{dep} is covered by the active miner who successfully mounts an *SDRBA* attack. Thus, stopping the censorship attack as soon as possible is in the best interest of the said miner. We prove the statements claimed in lemmas in Appendix D

Lemma 5. *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).*

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

Lemma 6. *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}.*

With the above Lemma 5 and Lemma 6, we have 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}. Since after timeout T, the probability of active miner mining a block is λ_{ac}, the expected number of blocks required for an active miner to create a block after timeout is $1/\lambda_{ac}$.

Lemma 7. *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}.*

With above lemmas, we can show that *HyDRA* will be preferred by all rational miners unless Alice pays a transaction fee for tx_A^{dep} higher than $v^{dep} - (T - t_{pub} - 1) \cdot c_1 + (1/\lambda_{ac}) \cdot c_2$.

Theorem 2. *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 6 and 7, 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 5, 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 2, 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 an HTLC contract with a complete capacity of the payment channel $v^{dep} = 2$ BTC. Estimating λ_{ac} accurately is hard, but the adoption rate of MEV-geth [23] among Ethereum miners 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 78% [29]. We conservatively set λ_{ac} to 50%. If the channel is closed one day prior to its timeout, then $T - t_{pub} = 24 \times 60/10 = 144$ blocks and thus the transaction fee cost f_A^{dep} to Alice in order to violate the condition in Theorem 2 must be $f_A^{dep} = v^{dep} - (T - t_{pub} - 1) \cdot c_1 - \frac{1}{\lambda_{ac}} \cdot c_2$, where $c_1 = f_A^{dep}/0.75, c_2 = f_A^{dep}/0.5$, which makes $f_A^{dep} \approx 0.01$ BTC which is about 4550× the average closing cost of 2.2e-6 BTC [9].

V. He-HTLC: An Incentive Compatible HTLC

We now present He-HTLC, an incentive-compatible implementation of 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) we break the atomicity by separating so that v^{dep} and v^{col} cannot be redeemed in one block.

An overview of the protocol has been presented in Section II-C. In this section, we describe the protocol in detail in Section V-A. The incentive compatibility of He-HTLC is proven in Section V-B. Finally, in Section V-C, we report on a Bitcoin implementation.

A. The Protocol

Our protocol He-HTLC$[\ell]$ is parameterized by ℓ, the number of blocks between the redemption of v^{dep} and v^{col}. This parameter adjusts the tradeoff between Bob's collateral amount and the time duration that the collateral must be locked for. Setting a large ℓ allows Bob to lock less collateral but for a longer period of time, and vice versa. He-HTLC$[\ell]$ involves two contracts, *He-Dep*, and *He-Col*, as described below.

He-Dep. To initiate an HTLC with Alice, Bob deposits $v^{dep} + v^{col}$ tokens to *He-Dep*. As with MAD-HTLC, v^{dep} is intended to be paid to Alice and v^{col} is Bob's collateral. *He-Dep* stipulates that the tokens can be spent in one of the two ways:

- *dep-A*: If Alice knows pre_a, then she can transfer v^{col} to Bob and rest to herself by revealing pre_a.
- *dep-B*: If tokens are still unspent after the timeout T, Bob can transfer $v^{dep} + v^{col}$ to the collateral contract *He-Col*, as specified below, by revealing pre_b.

He-Col. The collateral contract *He-Col* stipulates that the tokens can be spent in one of the two ways:

- *col-B*: Bob can spend $v^{dep} + v^{col}$ after ℓ blocks of the formation of *He-Col*.
- *col-M*: Anyone revealing both pre_a and pre_b can spend v^{col} tokens while rendering the rest v^{dep} permanently unspendable (aka burnt).

We would show that if Alice is rational and knows the secret pre_a, then she would reveal pre_a before the timeout expires (Lemma 8). In the case that Alice fails to reveal the secret until the timeout, Bob can get refunded in two steps. First, Bob will transfer $v^{dep} + v^{col}$ tokens to the collateral contract *He-Col* (through path *dep-B*). Then, after ℓ blocks, Bob can spend $v^{dep} + v^{col}$.

B. Security Analysis

We analyze the security of He-HTLC in the model described in Section III-A.

We will show that He-HTLC is incentive-compatible for any $v^{col} \in (f, v^{dep}]$ (we require this throughout this section). To be specific, we will show for any $0 < \epsilon \leq 1 - \frac{f}{v^{dep}}$ and $v^{col} = \epsilon v^{dep} + f$, there always exists a sufficiently large ℓ such that all parties are incentivized to follow He-HTLC$[\ell]$. In the analysis below, we assume that Alice pays a transaction fee f_A^{dep} slightly higher than f, but to reduce clutter we replace $f_A^{dep} = f$.

First, we show that the scheme is safe if Bob is following the protocol, whereas Alice who knows pre_a is trying to maximize her profits.

Lemma 8. *Suppose Alice knows pre_a in some round $r < T$. Withholding pre_a until after T gives Alice a strictly lower utility than revealing pre_a in a round $t_{pub} < T$ to miners.*

Proof: In case Alice withholds pre_a until after Bob reveals pre_b, Alice can get an amount of v^{col} shared with the miner if the miner takes the path *col-M*, out of which the miner takes a fee of at least f in order to compensate for the block space. However, revealing earlier leads to her receiving $v^{dep} - f$ by taking the path *dep-A*. Since $v^{col} \leq v^{dep}$, Alice would prefer to reveal pre_a before Bob reveals pre_b. Since Bob can reveal pre_b at any time after T in accordance with the protocol, Alice would reveal pre_a before T. ■

In case Alice does not know pre_a, the only path that can be taken is *dep-B* followed by *col-B* and Alice cannot take any action. Now that we have ensured that if Bob acts honestly, Alice cannot gain a utility greater than her honest execution, we show that if Alice acts honestly, then Bob cannot gain any more utility than his honest execution.

Lemma 9. *Suppose pre_a has been revealed to the miners; Bob successfully funds He-Col through dep-B in round $r \geq T$, and*

He-Col redemption paths are not yet taken. Then, each miner in round $> r$, unless bribed with an amount $> v^{col} - f$, has a higher utility in taking col-M in comparison to including unrelated transactions.

Proof: Since Alice has revealed pre_a to the miner and Bob has revealed pre_b through *dep-B*, all the subsequent miners know both pre_a and pre_b. If the miner follows path *col-M* instead of including unrelated transactions, it receives a utility of $v^{col} - f$. Since $v^{col} > f$, the miner would prefer *col-M* than including unrelated transactions unless bribed with an amount $> v^{col} - f$ to include unrelated transactions. ■

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

Lemma 10. *Once He-Col is funded from He-Dep through dep-B, Bob would never choose to bribe more than a sum of $v^{dep} + v^{col} - f$ to the miners in order to incentivize them into not taking path col-M.*

Proof: This follows from the fact that Bob can only earn $v^{dep} + v^{col}$ through the path *col-B*, paying a fee of f to the miner in the process. Thus, paying a higher amount as a bribe strictly reduces Bob's utility. ■

Lemma 11. *Suppose pre_a and pre_b have been revealed to the miners and He-Col has been funded in some round $r > T$; 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 $v^{col} \geq (\frac{v^{dep}}{\kappa - 1}) + f$.*

Proof: Since both pre_a and pre_b are available to the miner, path *col-M* can be taken which leads to burning the deposit. Alternatively, path *col-B* can be taken ℓ blocks after *dep-B*.

There are κ distinct miners in expectation deciding between choosing *col-M*, *col-B* or not taking either path. From Lemma 9, we know that *col-B* can only be taken if each of the κ miners is offered a bribe $> v^{col} - f$. Moreover, from Lemma 10, we know that Bob can offer a maximum bribe of $v^{dep} + v^{col} - f$ distributed among all the κ miners. By Pigeon-Hole principle, at least one of these bribes to some miner needs to be $\leq \frac{v^{dep} + v^{col} - f}{\kappa} \leq v^{col} - f$. Let M_i be the first such miner. All miners before M_i cannot take *col-B* (since they are within ℓ blocks from the funding of *He-Col*). Since the bribe is not large enough, as per Lemma 9, M_i is incentivized to take *col-M*. ■

Theorem 3. *Assuming all parties are (actively or passively) rational, and $(\frac{1}{\kappa - 1})v^{dep} + f \leq v^{col} \leq v^{dep}$ for $\kappa \geq 2$, ℓ is set such that expected number of distinct miners across ℓ blocks is κ, He-HTLC$[\ell]$ guarantees:*

1) *If Alice reveals pre_a via path dep-A at round $t < T$, then all parties prefer taking dep-A.*
2) *If Alice does not reveal pre_a in any round $t < T$, then all parties prefer dep-B followed by col-B.*

Proof: We prove the two guarantees separately.

Part 1. If pre_a is available to miners at round $t < T$, Bob can take one of the two options.

1) Bob never reveals pre_b.
2) Bob reveals pre_b to miners in some round.

First, we will show that Bob will always prefer the first option over the second.

If Bob never reveals pre_b to the miners, Alice redeems v^{dep} through *dep-A*, and Bob gets back v^{col}. The utility of Alice and Bob are $v^{dep} - f$ and v^{col} (Alice pays the transaction fee), respectively. The miner can earn a fee of f.

In the second option, Bob reveals pre_b to miners. The miner can either take path *dep-A* or *dep-B* if a path amongst them has not already been taken. The path *dep-B* can be followed by *col-B* after ℓ blocks or by *col-M* possibly in the same block. With path *dep-B*, we know that the miner gets a utility $\geq v^{col}$ by taking both *dep-B* and *col-M* in the same block (transaction fee f and the utility for taking the path *col-M*) whereas it will get f from *dep-A*. Since $v^{col} > f$, *dep-B* would be preferred. Since pre_a has been revealed and $(\frac{1}{\kappa - 1})v^{dep} + f \leq v^{col}$ (by assumption), it follows from Lemma 11 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 amount received would be v^{col}. Since the minimum amount miner expects is f, the maximum amount Bob can expect is $v^{col} - f$ (which is less than Bob's expected utility in case 1). Therefore, Bob will always choose not to reveal pre_b when pre_a is already known.

Thus, if pre_a is available to miners at round $t < T$, then all parties are incentivized to take the path *dep-A*.

Part 2. Let us consider that Alice does not reveal pre_a at any time $< T$. By Lemma 8, we know that if Alice knows pre_a, then she would have revealed it in a round $< T$. Thus, if pre_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*. ■

C. Implementation in Bitcoin

Implementing He-HTLC for blockchains with smart contract support (e.g., Ethereum) is straightforward. In this section, we report on the implementation of He-HTLC for Bitcoin.

Bitcoin scripts. An He-HTLC contract is instantiated as a pair of Bitcoin UTXOs (UTXO$_{\text{dep}}$ and UTXO$_{\text{col}}$ respectively) with scripts shown in Table I. Note that currently Bitcoin script only enforces *when* a UTXO can be spent, it cannot, however, enforce *how* a UTXO will be spent in the future (e.g., an authorized spender of a UTXO can send it to any address she wishes; Covenant [30] will change this but it is not yet available in Bitcoin). Therefore, our implementation combines Bitcoin script with pre-signed transactions to enforce that UTXO$_{\text{dep}}$ and UTXO$_{\text{col}}$ can only be spent in specific ways. Namely, spending UTXO$_{\text{dep}}$ or UTXO$_{\text{col}}$ requires signatures from both Alice and Bob, so that both parties can ensure that the funds can only be spent in the following ways.

- UTXO$_{\text{dep}}$ can be spent by two transactions: tx_A^{dep} spends it and sends v^{col} to Bob, and v^{dep} to Alice; tx_B^{dep} spends it and creates UTXO$_{\text{col}}$ with amount $v^{dep} + v^{col}$.
- UTXO$_{\text{col}}$ can be spent by two transactions: tx_B^{col} spends it and sends $v^{dep} + v^{col}$ to Bob; tx_M^{col} spends it and creates a UTXO with amount v^{dep} and script `OP_RETURN` (the

Table I: Bitcoin scripts for *He-Dep* and *He-Col*

He-Dep	***He-Col***
2	2
pk_a	pk_a
pk_b	pk_b
2	2
OP_CHECKMULTISIGVERIFY	OP_CHECKMULTISIGVERIFY
OP_HASH160	OP_HASH160
dig_a	dig_a
OP_EQUAL	OP_EQUAL
OP_IF	OP_IF
OP_TRUE	OP_HASH160
OP_ELSE	dig_b
T	OP_EQUAL
OP_CHECKSEQUENCEVERIFY	OP_ELSE
OP_DROP	ℓ
OP_HASH160	OP_CHECKSEQUENCEVERIFY
dig_b	OP_DROP
OP_EQUAL	OP_TRUE
OP_ENDIF	OP_ENDIF

standard way to render v^{dep} provably unspendable). The rest will be taken by the miner as transaction fees.

Note that all of the transactions must be prepared and signed before a He-HTLC contract is instantiated, otherwise whoever acts first is at a disadvantage. For instance, if Bob creates UTXO$_{\text{dep}}$ before obtaining a tx_B^{dep} signed by Alice, then there is no guarantee that Alice will sign, thus Bob's money might get stuck.

Transaction pre-signing. The above transactions can be pre-signed securely as follows. First, Bob creates a transaction that deposits $v^{dep}+v^{col}$ (out of his pocket) to create UTXO$_{\text{dep}}$. He keeps the transactions private and creates $tx_A^{dep}, tx_B^{dep}, tx_B^{col}$, and tx_M^{col}. Bob attaches his signatures to tx_A^{dep} and tx_M^{col} and sends all of them to Alice. Note that transactions tx_B^{col} and tx_M^{col} rely on a UTXO to be created by tx_B^{dep}. To avoid malleability issues, tx_B^{dep} must be *SegWit* [31]. Alice checks that all transactions are properly formed, and attaches her signatures to tx_B^{dep}, tx_B^{col}, and tx_M^{col} and sends them back to Bob. To ensure that tx_M^{col} is accessible to all miners, Alice puts tx_M^{col} on chain using an OP_RETURN script. Alice then waits for UTXO$_{\text{dep}}$ to be created on-chain, at which point the timeout T starts. If UTXO$_{\text{dep}}$ does not appear on chain, the contract is canceled.

Operation. After Bob receives and verifies pre-signed transactions from Alice, he funds UTXO$_{\text{dep}}$. Post this Alice can take path *dep*-A by adding pre_a and her signature to tx_A^{dep} and broadcast. If Alice does not reveal the transaction until timeout, Bob can add pre_b and sig_b to tx_B^{dep} to create UTXO$_{\text{col}}$. After waiting for another ℓ blocks, Bob can spend UTXO$_{\text{col}}$.

Testnet deployment. We developed a prototype He-HTLC client in Golang [32] to automatically generate transactions. We deployed He-HTLC on testnet to verify the correctness of our implementation. Table II shows the example testnet transactions corresponding to four protocol paths. The last column shows the estimated transaction fees at various fee rates. Even at a generous rate of 20 Satoshi per vByte, transactions costs of He-HTLC is at below $1 (for reference, the average Bitcoin transaction fee fluctuates between $1 to $5 from August 2021 to August 2022).

Comparison with MAD-HTLC. We compare the transaction sizes of MAD-HTLC [9]. The following table compares the transaction cost incurred by each party. The row for miner stands for the cost of confiscation. Cells with a + sign mean that action requires two transactions. In He-HTLC, since Alice redeeming v^{dep} also refunds Bob with v^{col}, Bob does not pay a transaction fee.

	He-HTLC (vB)	MAD-HTLC (B)	Reduction
Alice	190	323	41%
Bob (refund v^{col})	0	248	100%
Bob (refund both)	172 + 152	322 + 248	43%
Miner	172 + 168	282 + 241	54%

In general, He-HTLC transactions are significantly smaller. It is imperative to note that MAD-HTLC transaction sizes can be reduced if implemented using SegWit transactions.

VI. 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 the Bitcoin script, along with other improvements.

In general, the incentive compatibility of blockchain protocols has been studied extensively. E.g., selfish mining attack [33] shows 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. [34] shows how to *bribe* miners, possibly using the blockchain itself, to take over the system for a short duration. McCorry et al. [28] 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., [35] 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 [16]–[18], 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], [28], [34], [35]). We

Table II: Example transactions on-chain result. Fees are calculated at 23,894 USD/BTC.

Path	Transaction ID	Size	Estimated fee (USD) @ 2/10/20 Sat/vB
dep-A	5dbb7c677b3177700a541ecc23604bbfdb5ddd5a463e924428ae096646214180	190 vbyte	\$0.09 / \$0.45 / \$0.91
dep-B	e80fcfb0a1ce936fcb9b514f03d2d78d4c8f06a0657e3a9f54411dee636d1ce6	172 vbyte	\$0.08 / \$0.41 / \$0.82
col-B	a3f2902a3e7d3bd81295fc020a2a211dbf00b428f7bc6e93c4cfc12f7a55a628	152 vbyte	\$0.07 / \$0.36 / \$0.73
col-M	78b7ec346dc7ef75295ba26d1705cb791da23a97ff1511e28321e2b62f5de689	168 vbyte	\$0.08 / \$0.40 / \$0.80

show that they can do better if they actively create more opportunities. In a sense, our model is a generalization of that in [10], 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 [12], [13] or offer direct RPC channels for a fee [14]. The reverse bribery attacks we show are a different avenue of MEV from influencing other (non-mining) participants.

Concurrent and independent work. Recently, there has been a concurrent work titled PONYTA [19], which identifies the same incentive compatibility concern with MAD-HTLC and proposes an incentive compatible solution. Our works differ in the following ways. First, they only present an intuition towards the existence of an attack in MAD-HTLC. We discuss the challenges of making such an attack work and present a game-theoretic proof and an implementation to implement such an attack using TEEs and ZKPs (Section III-B). Second, their solution to disincentivize collusion between Alice and miner or Bob and miner requires Alice to provide collateral of v^{dep} and Bob to provide collateral of $2v^{dep}$. In He-HTLC, Alice does not pay collateral, and Bob can choose small collateral of v^{dep}/κ where κ, chosen by Bob, determines how early he receives his collateral back after the timeout. On the other hand, their work does consider parties may start with some external incentives, which is not considered in our work. Note that their work does consider collusion between miners (even more than 50%) and one of the parties; in our work, such collusion only requires us to set the value of κ appropriately (under the same assumption that the coalition of miners will not mount consensus-level attacks).

VII. Discussion and Future Work

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 [10] 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 attacks and our solution. We present two different attacks. The first attack *SDRBA* shows there exists *a* strategy that would be preferred by Bob and every actively rational miner compared to following MAD-HTLC. 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 $> v^{col}$. By making this total cost larger than v^{dep}, 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 $v^{col} = \epsilon v^{dep} + f$, in which by increasing ℓ (and the number of distinct miners κ) appropriately, we can reduce ϵ which reduces the required collateral. 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).

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 path *col-*M where miners can earn v^{col}. However, if Alice and Bob are rational, miners can never 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.

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 if 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] 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.

[11] B. Weintraub, C. F. Torres, C. Nita-Rotaru, and R. State, "A flash (bot) in the pan: Measuring maximal extractable value in private pools," *arXiv preprint arXiv:2206.04185*, 2022.

[12] 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

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

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

[15] 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/

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

[17] 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.

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

[19] H. Chung, E. Masserova, E. Shi, and S. A. Thyagarajan, "Ponyta: Foundations of side-contract-resilient fair exchange," *Cryptology ePrint Archive*, 2022.

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

[21] 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.

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

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

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

[25] "Lightning channels - top capacity." [Online]. Available: https://1ml.com/channel?order=capacity

[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] 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.

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

[30] Checktemplateverify. [Online]. Available: https://github.com/bitcoin/bips/blob/master/bip-0119.mediawiki

[31] bips/bip-0141.mediawiki at master · bitcoin/bips. [Online]. Available: https://github.com/bitcoin/bips/blob/master/bip-0141.mediawiki

[32] Bitcoin in go. [Online]. Available: https://github.com/btcsuite

[33] 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.

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

[35] 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.

Appendix

A. Success Independent Reverse Bribery Attack

In this section, we present another attack on the MAD-HTLC protocol. It differs from Section III-B in that the miners pay a fee regardless of whether or not they are able to mine the block after t_{pub}. Formally, the exchange between the miner and Bob is br_i for the knowledge of pre_b. We assume that the miner does not sell this pre_b to other miners in the system.

We first 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 III-B 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.

```
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 8: TEE enclave program for *SIRBA*

M_i runs the TEE code shown in Fig. 8. For ease of exposition the above parameters are hardcoded in Fig. 8, 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 $\texttt{init}(\lambda)$ 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}.

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 $\texttt{VerifyMsgFromBob}$ 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. 8. According to Fig. 8, 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 that could have been accepted by the blockchain, which is prohibitively expensive and irrational since the bribe amount is smaller.

Analysis. 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 4). We borrow the model and setup from Section III to construct a game $G^{ind}(\cdot, \cdot)$ in which we change the fair exchange as defined in this section. For ease of exposition, we add another action for Bob. Bob always publishes tx_B^{col} in round T even if he accepts bribes. This does not change the game because tx_B^{col} does not reveal any information about pre_b.

Lemma 12. *In subgame $G^{ind}(T, \cdot)$, 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, tx_B^{col} is always available in round T. Since $v^{col} > f_B^{col} > f$, if M_i knows pre_b then including tx_M^{col} in b_T is strongly dominant for M_i. Also, note that we assume pre_a has been revealed in $t_{pub} < T$.

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 tx_B^{col} in b_T is the best strategy. 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 13. *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 12, given that $v^{dep} > f_A^{dep} > f$. ■

Lemma 14. *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 12, 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 15. *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 13 and Lemma 14 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 12, 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:

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

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 12 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:

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

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 16. *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 $v^{col} - f_B^{col}$. On the other hand, if Bob accepts a bribe br_i, his gain is at least br_i; moreover, with certain probability, Bob might also get v^{col}.

Observe that Bob gets to redeem v^{col} in round T when two events happens: E_1) pre_b is not revealed before round T (or else some miner—not necessarily M_i—will redeem v^{col} in round T), and E_2) the bribing miner M_i is not elected to mine in round T (or else M_i will redeem v^{col} herself). Bob's total income when accepting the bribe is therefore $br_i + Pr[E_1] \cdot Pr[E_2] \cdot (v^{col} - f_B^{col})$.

The probability depends on M_i's mining power as well as the actions of other (passively rational) miners. We consider the worst case to get a lower bound. pre_b is revealed only when M_i redeems v^{dep}, the probability of which is maximized when when all passively rational miners only include unrelated transactions. Therefore $Pr[E_1] \geq (1 - \lambda_i)^{T - t_{pub}}$. It is not hard to see $Pr[E_2] = 1 - \lambda_i$.

Overall, Bob's utility is $(1 - \lambda_i)^{T - t_{pub} + 1}(v^{col} - f_B^{col}) + br_i$. 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 4. *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 16 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 15, 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. ■

B. Analysis of SDRBA

We first define some additional helpful lemmas to prove the lemmas mentioned in the text.

Lemma 17. *Suppose v^{dep} has been redeemed in some round before T. Consider the subgame $g = G^{dep}(T, st)$ in round T. If $st = irred\text{-}nrev$, redeeming v^{col} through dep-B is a dominant strategy for both Bob and miners. If $st = irred\text{-}rev$, redeeming v^{col} through col-M is a dominant strategy for miners. Moreover, the utility of miner $M_i \in \mathbf{M}$ with mining power λ_i is given by*

$$\bar{u_i}(g) = \begin{cases} \lambda_i(v^{col} - f) & st = irred\text{-}rev \\ \lambda_i(f_B^{col} - f) & st = irred\text{-}nrev \end{cases}.$$

Proof: Since *MH-Dep* has been redeemed, the miner of round T can choose between three actions: to redeem *MH-Col* via *col-*B, to redeem *MH-Col* via *col-*M, or to include unrelated transactions. We analyze the utility of these options based on how *MH-Dep* might have been redeemed. Since *dep-*B is not possible before round T per HTLC spec, *MH-Dep* must have been redeemed through *dep-*M or *dep-*A.

Case 1: If the path *dep-*M was taken, then pre_b would have been revealed and the game will be in $st = irred\text{-}rev$. Given that $v^{col} > f_B^{col} > f$, M_i will strictly prefer to redeem *MH-Col* via path *col-*M over any other action. The utility is $\bar{u}_i(G^{dep}(T, irred\text{-}rev)) = \lambda_i(v^{col} - f)$.

Case 2: If the path *dep-*A was taken, then pre_b is only known to Bob (bribing miners will not learn pre_b directly through the fair exchange protocol, as discussed in Section III-C). 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, it is infeasible for miners to incentivize Bob to deviate from the honest protocol and reveal pre_b. 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 18. *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: State $st \neq red$ means that *MH-Dep* has been redeemed, so the utility change can only be gained from redemption of *MH-Col*. Following the proof of Lemma 17, the expected utility from *MH-Col* solely depends on miners' knowledge of pre_b and mining power. We now argue that neither changes between round k and T. If *MH-Dep* has been redeemed via path *dep-*M, then pre_b has been revealed on-chain. If *MH-Dep* has been redeemed via path *dep-*A, then pre_b will never be known to any miners (following a similar argument as the proof of Lemma 17). Thus the knowledge of pre_b will not change between round k and T. By assumption, the mining power does not change. ■

Lemma 19. *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 12 and 13 where the passively rational miner does not know pre_b in round T. ■

Now, we redefine and prove the lemma statements used in Section III

Lemma 1. *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 round T, M_i has the chance to redeem both v^{dep} and v^{col}, thus M_i can choose between i) including tx_A^{dep} and tx_B^{col}, ii) offering a bribe, or iii) including unrelated transactions. We first observe that including unrelated transactions realizes strictly less profit than including tx_A^{dep} and tx_B^{col}. Now we compare the utility from (ii) bribing and (iii).

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

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

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 2. *In any subgame $G^{dep}(k, red)$ where $t_{pub} \leq k < T$, 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 $t_{pub} \leq k < T$. 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 17 and Lemma 18, 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 18, 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 17 and Lemma 18, 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 3. *In any subgame $G^{dep}(k, red)$ where $t_{pub} \leq k < T$, 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 $t_{pub} \leq k < T$. As in Lemma 2, 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 $\geq k+1$ 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: $G^{dep}(T, \mathit{irred\text{-}rev})$. In this case, *MH-Dep* has been redeemed by some other miner via path *dep*-M. From Lemma 17 it follows that M_i's expected utility is given by $\lambda_i(v^{col} - f)$.

Case 3: $G^{dep}(T, \mathit{irred\text{-}nrev})$. In this case, *MH-Dep* has been redeemed by some other miner via path *dep*-A. From Lemma 17 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 4. *In any subgame $G^{dep}(k, \cdot)$ where $k \geq t_{pub}$, 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 16, 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 17 and Lemma 1, Bob will lose v^{col}. Consequently, Bob will not accept any bribe $br_i \leq v^{col} - f_B^{col}$. 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. ■

Safety of MAD-HTLC against *SDRBA*

Theorem 5. *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 2 and Lemma 4, 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 all rounds before T.

From Lemma 1 and Lemma 4 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. ■

C. Variants of the Protocol for Realizing SDRBA

We consider a few variants of the protocol discussed in Section III-B.

ZKPs in place of TEEs. Instead of TEEs, Bob can prove the correctness of h using zero-knowledge proof [22]. 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 [12] 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 [12] although in a different context.

D. Proof of Lemmas in HyDRA

Lemma 5. *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 amount remaining in the contract for any miner of round t to take a bribe of c_1 or c_2 for censorship is $v^{dep} - (T - t_{pub} - 1) \cdot c_1 - (t - T) \cdot c_2$. This is to be the utility for the miner $(v^{dep} + v^{col} - br)$ 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. ■

Lemma 6. *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. 6), 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 5, 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. ■

Lemma 7. *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. 6), 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. ■