Decentralized Exchange that Mitigate a Bribery Attack

Nitin Awathare
IIT Jodhpur
nitina@iitj.ac.in

***Abstract*—Despite the popularity of Hashed Time-Locked Contracts (HTLCs) because of their use in wide areas of applications such as payment channels, atomic swaps, etc, their use in exchange is still questionable. This is because of its incentive incompatibility and susceptibility to bribery attacks.**

State-of-the-art solutions such as MAD-HTLC (Oakland'21) and He-HTLC (NDSS'23) address this by leveraging miners' profit-driven behaviour to mitigate such attacks. The former is the mitigation against passive miners; however, the latter works against both active and passive miners. However, they consider only two bribing scenarios where either of the parties involved in the transfer collude with the miner.

In this paper, we expose vulnerabilities in state-of-the-art solutions by presenting a miner-collusion bribery attack with implementation and game-theoretic analysis. Additionally, we propose a stronger attack on MAD-HTLC than He-HTLC, allowing the attacker to earn profits equivalent to attacking naive HTLC.

Leveraging our insights, we propose DEMBA**, a game-theoretically secure HTLC protocol resistant to all bribery scenarios.** DEMBA **employs a two-phase approach, preventing unauthorized token confiscation by third parties, such as miners. In Phase 1, parties commit to the transfer; in Phase 2, the transfer is executed without manipulation. We demonstrate** DEMBA**'s efficiency in transaction cost and latency via implementations on Bitcoin and Ethereum.**

***Index Terms*—component, formatting, style, styling, insert.**

I. Introduction

Cryptocurrencies such as Bitcoin [1] and Ethereum [2], which are based on blockchain technology, allow for the secure transfer of tokens without the need for a central authority. They facilitate straightforward internal token transactions and allow for the implementation of sophisticated smart contracts for the conditional transfer of tokens. Smart contracts are created and interacted with through transactions, and blocks containing these transactions are generated by entities known as miners. Miners construct and publish blocks, forming the blockchain, validating transactions, and driving the system forward. The system's state is derived by sequentially processing transactions in the order they appear within the blocks, starting from the genesis block to the most recent one.

The Hashed TimeLocked Contract (HTLC) [3] is a commonly used smart contract that is compatible with both Ethereum and Bitcoin. It plays a significant role in the Lightning network [4] [5], enabling secure payment routing through multiple payment channels, as well as facilitating contingent payments [6]–[10], atomic swaps [12]–[16], vault [17]–[20], and other applications. Essentially, an HTLC is defined by two parameters: a timelock T and a hash lock H, which together ensure the conditional transfer of v tokens from the payer (Bob) to the payee (Alice) [1]. Specifically, Alice can spend the v tokens by submitting a transaction containing a pre-image of H to the blockchain before the T timeout, while Bob can spend the tokens after the T time has elapsed.

Regrettably, the Hashed TimeLocked Contract (HTLC) is susceptible to incentive manipulation attacks known as bribery attacks. This is because HTLC assumes that miners will promptly add Alice's transaction to the blockchain before the T timeout. However, as profit-driven rational agents, miners may not conform to the expected behaviour when incentivized by a malicious Bob. For instance, Winzer et al. [21] have demonstrated that Bob can bribe miners using specific smart contracts to disregard Alice's transactions until the timeout is reached. Likewise, Harris and Zohar [22] indicate that Bob can delay the confirmation of Alice's transaction by overloading the system with his own transactions. Both of these strategies enable Bob to acquire the HTLC tokens while denying Alice access to them, even if Alice has already published the preimage.

MAD-HTLC is an ingenious solution put forth by Tsabary et al. [23] to safeguard HTLC from bribery attacks. This proposal involves requiring the payer to provide collateral, and any misconduct by the payer will result in the miners seizing the collateral. However, MAD-HTLC focuses solely on passive mining strategies, whereby miners only conduct standard mining by selecting the best available transaction rather than creating better ones themselves. Due to this limitation, Wadhwa et al. [24] put forward an alternative approach called He-HTLC, which is demonstrably secure against both passive and active rational strategies of the miner. This includes strategic actions beyond mining, thereby offering a more comprehensive solution.

However, He-HTLC [24] as well as MAD-HTLC [23] don't consider the bribery scenario where a miner bribes another miner and collude to confiscate the HTLC token[2], which is prominent in the real-world as miners are rational entities in the network. We demonstrated it by proposing an attack called Miner to Miner Bribery Attack (M2MBA).

[1]Please note that throughout this paper, we will use "Alice" and "Payee" interchangeably, as well as "Bob" and "Payer

[2]**Note that a miner bribing another miner differs from bribery by other entities, such as Alice or Bob, because miners are not direct participants in the underlying exchange.**

M2MBA exploits He-HTLC when miners bribe another miner and collude to confiscate the tokens involved in Bob's collateral. In He-HTLC, Exchange tokens [3] are transferred to Alice, and collateral is transferred to Bob if Alice reveals his preimage before time T. On the other hand, after time T, Bob can redeem both exchange tokens and collateral by revealing his preimage. However, if a miner has both the preimages, then they can confiscate Bob's collateral and the exchanged tokens are burned. In M2MBA, miners would wait for Bob to reveal his preimage, say pre_B, even if Alice reveals his preimage, say pre_A and hence confiscate Bob's collateral once Bob reveals pre_B. Note that a miner can execute this attack alone However, by bribing and colluding with other miners-as done in M2MBA-the miner can achieve a more stable income (demonstrated in Appendix D).

Additionally, we propose an attack on MAD-HTLC called the Bribery-Based Block Broadcasting Attack (B3A), which is more severe than the SDRBA and HyDRA attacks proposed by He-HTLC. In B3A, Bob gains more tokens than he would through SDRBA or HyDRA, providing even stronger incentives to launch the attack. In fact, our analysis shows that Bob can extract nearly the same amount of profit as he would by attacking the naive HTLC.

To address these issues, we propose DEMBA, which ensures that miners cannot confiscate tokens as long as neither Alice nor Bob deviates from the protocol. Simply removing the miner's confiscation power would reintroduce vulnerabilities of naive HTLCs. Instead, DEMBA mitigates this by requiring both parties to post collateral, shifting accountability to the participants themselves. This design is based on the key insight that the core vulnerability in MAD-HTLC and He-HTLC stems from granting enforcement power to an uninvolved third party—the miner.

DEMBA operates in two phases. In the first phase, each party redeems their respective collateral by revealing their preimage, thereby committing to the exchange. Once this commitment is made–by broadcasting a transaction that reveals a specific preimage–they can no longer back out. Specifically, Alice can commit in one of three ways, depending on when and which preimages she reveals (pre_A and/or pre'_A). First, revealing pre_A transfers the exchange amount v^{dep} to Alice. Second, revealing only pre'_A returns v^{dep} to Bob. Lastly, revealing both pre_A and pre'_A burns v^{dep}. Alice makes this final commitment only if she previously revealed pre_A, as a means to punish Bob for bribing miners to censor her commitment. This mechanism deters bribery by ensuring that any attempt to suppress a valid transaction leads to a loss for Bob.

On the other hand, Bob has one way to commit–by revealing his preimage pre_B. Once both Alice and Bob have made their respective commitments, the protocol proceeds to the second phase, where the exchange is executed. Our game-theoretic analysis in Section VI shows that any deliberate false commitment results in a loss for the deviating party.

[3]The tokens that Bob sends to Alice

Lastly, We implemented and evaluated DEMBA on Bitcoin and Ethereum. On Bitcoin, DEMBA reduces costs by 42% vs. MAD-HTLC but incurs 17% more than He-HTLC for Alice's token claim, while lowering Bob's refund cost for both his collateral and exchange tokens by 42% vs. He-HTLC and 150% vs. MAD-HTLC. Similar trends hold on Ethereum, though with smaller margins: DEMBA saves 38% vs. MAD-HTLC and costs 7% more than He-HTLC for Alice, while reducing Bob's refund cost by 23% and 71% vs. He-HTLC and MAD-HTLC, respectively.

Furthermore, We compared DEMBA with He-HTLC and MAD-HTLC in terms of time to complete (TTC), defined as the duration from transaction initiation to completion. DEMBA matches or exceeds their performance while incurring lower costs.

In summary we made following contributions:

- We demonstrate two attacks on MAD-HTLC and He-HTLC, showing that Bob can still exploit MAD-HTLC to earn the same tokens as in an attack on naive HTLC.
- We propose DEMBA, the first HTLC scheme addressing all bribery scenarios—Alice bribing miners, Bob bribing miners, and miner-to-miner bribery—unexplored in prior work. Our claims are supported through game-theoretic analysis.
- We implemented and evaluated DEMBA on Bitcoin and Ethereum, demonstrating its superiority over state-of-the-art protocols in most cases.

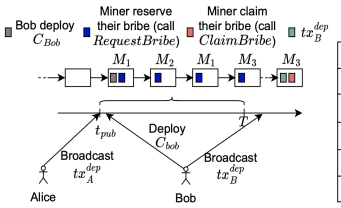
The rest of the paper is organized as follows: §II discusses existing work that aims to enhance naive HTLC to resist bribery attacks. The motivation is explained in §III through the demonstration of attacks on existing work. §IV gives an overview of DEMBA. The game-theoretic analysis of the M2MBA attack and DEMBA are presented in §V and §VI, respectively. DEMBA's implementation, evaluation, and results are discussed in §VII. §VIII reviews related work. Finally, the paper concludes with a discussion and future directions in §A (In the Appendix)

II. Background

To facilitate comprehension and provide context, we will begin by revisiting Hash Time-Locked Contracts (HTLC), outlining the issues associated with the current HTLC protocol, and examining prior attempts to address them before delving into our findings.

A. Naive HTLC and Bribery attack on it

A naive Hash Time-Locked Contract (HTLC), $C^{dep}_{B->A}$, involves Bob paying Alice v^{dep} tokens is defined by the tuple (pk_A, pk_B, v^{dep}, pre_A, T). The $C^{dep}_{B->A}$ indicates that Bob's deposit of v^{dep} tokens which either transferred to Alice when Alice broadcast signed transaction tx^{dep}_A that reveals pre_A or Bob can redeem by broadcasting a signed transaction tx^{dep}_B after the timeout period T. The later one serves as a refund mechanism if Alice remains inactive till time T. It is important to note that the preceding description omits implementation-



Users	Honest	Bribery
M_1	f_A^{dep}	$2 \cdot br > f_A^{dep}$
M_2	0	$br > f_A^{dep}$
M_3	0	$f_B^{dep} + br > f_A^{dep}$
Alice	$v^{dep} - f_A^{dep}$	0
Bob	0	$v^{dep} - \left(5 \cdot br + f_B^{dep} + f_B^{C_{bob}}\right)$

Fig. 1: Bobs' attempt to make an bribery attack on the naive HTLC. Bob deploy contract committing a bribe br to the miners for censoring tx_A^{dep}, which is ultimately send to miners at time $t > T$

and application-specific details, enabling us to concentrate on the fundamental concerns.

Winzer et al. [21] and Tsabary et al. [23] demonstrated attacks on the naive HTLC by illustrating a scenario where Bob bribes miners to withhold tx_A^{dep} until time T. Once T is reached, Bob can claim v^{dep} using tx_B^{dep} and offering a transaction fee of f_B^{dep}. The profit gained from this attack can be calculated as follows. Let t_{pub} be the time when Alice broadcast tx_A^{dep} offering a transaction fee of f_A^{dep} and k be the number of block generated between time t_{pub} and T. It is further assumed that Bob offers a bribe $br > f_A^{dep}$ to the miner, per block, for not including tx_A^{dep}. Consequently, Bob gains $v^{dep} - [(k+1) * br + f_B^{dep} + f_B^{C_{Bob}}]$ after bribing the miner, as shown in Figure 1, where $k = 4$. We have given the sample smart contract, say C_{Bob}, to perform the above mentioned attack in Algorithm 1. In the forthcoming sections, we will demonstrate our proposed attacks on the existing remedies for the pitfalls of the naive HTLC using the modified version of the same contract. Given the contract C_{Bob}, the attack proceed as follows:

- Bob deploy a smart contract C_{Bob}, described in Algorithm 1, and initialize it with v^{dep}. This costs Bob the fee of $f_B^{C_{Bob}}$.
- Miners mining the block between time t_{pub} and T call *RequestBribe* to reserve their potential bribe for censoring tx_A^{dep}.
- After time T Bob broadcast tx_B^{dep} to claim v^{dep} (refund), which miners include in their block. This costs Bob the transaction fee of f_B^{dep}.
- After tx_B^{dep} is included in the block, any miner can call $claimBribe$ with a pre_A, which when executed send the reserved bribe to all the miners who has censored the tx_A^{dep}. The miner who include the transaction calling $claimBribe$ will receive the fee of br, which is the extra br in the equation $v^{dep} - [(k+1) * br + f_B^{dep} + f_B^{C_{Bob}}]$.

B. Existing work overcoming the bribery attack on naive HTLC (MAD-HTLC and He-HTLC)

To address the issue of HTLC bribery, Tsabary et al. [23] proposed MAD-HTLC, which features two key modifications. Firstly, it adds a second hash lock with preimage pre_B, along with an extra redemption path that allows miners to redeem v^{dep} if they possess both pre_A and pre_B. Secondly, MAD-HTLC introduces a collateral contract called *MH-Col*, initiated by Bob, that contains collateral tokens v^{col} that miners can also seize if they know (pre_A, pre_B). These changes are intended to discourage Bob from revealing pre_B unless he genuinely requires a refund, particularly when pre_A has not yet been released.

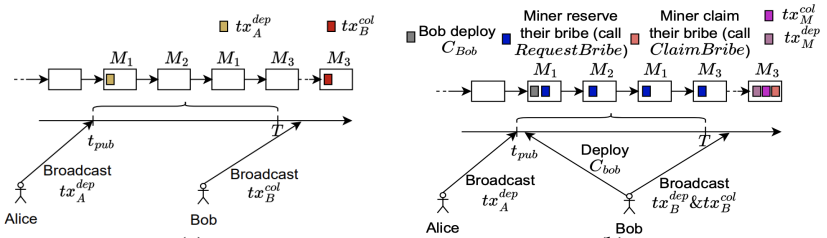
In MAD-HTLC, the payment of v^{dep} tokens made by Bob is stored in a contract called MH-Dep, which outlines three redemption paths: 1) *dep-A:* where Alice can redeem v^{dep} by broadcasting a signed transaction tx_A^{dep} revealing pre_A, 2) *dep-B:* where Bob can redeem v^{dep} by broadcasting a signed transaction tx_B^{dep} after time T, and 3) *dep-M:* where anyone (e.g., a miner) can redeem v^{dep} by broadcasting the transaction tx_M^{dep} revealing both pre_A and pre_B. Bob's collateral v^{col} is stored in contract MH-Col, which can be redeemed in two ways: 1) *col-B:* where Bob can redeem by broadcasting a signed transaction tx_B^{col} after T, and 2) *col-M:* where anyone (miner) can redeem by broadcasting the transaction tx_M^{col} revealing both pre_A and pre_B.

As depicted in Figure 2b, rational miners at time $t > T$ will not allow Bob to spend v^{dep} using tx_B^{dep} if they know pre_A. Instead, they will confiscate both v^{dep} and v^{col} through *dep-M* and *col-M*, respectively. Consequently, Bob loses not only v^{col} and v^{dep} but also $f_B^{C_{Bob}}$, which the fee charged by the miners to deploy contract C_{Bob}. On the other hands, the miner who confiscates (M_3 in our example) earns v^{dep} and v^{col}, as illustrated in Figure 2c.

However, MAD-HTLC does not address the possibility of miners actively seeking out other users in the system and bribing them to obtain a higher utility, called as reverse bribery. This is because the contract allows the miners and Bob to redeem $v^{col} + v^{dep}$. A successful attack could split this total gain such that Bob and miners are individually better off than following the protocol. [24] presents two such attacks that attempt reverse bribery: 1) Success-dependent reverse bribery attack (SDRBA), and 2) Hybrid delay-reverse bribery attack (HyDRA). The details of SDRBA and HyDRA is presented in Appendix C.

The SDRBA and HyDRA rely on two main factors. First, miners collude with either Bob or Alice to earn a high profit, $v^{dep} + v^{col}$, through *dep-M + col-M*. For instance, a miner can collude with Alice to earn $v^{col} - \epsilon$ or with Bob to earn $v^{dep} - \epsilon$, which they could have earned $f_A^{dep} + f_B^{col}$ by acting honestly. Second, the redemption of v^{dep} and v^{col} can occur atomically in a single transaction or in two separate transactions within a block. Considering these observations [24] introduces the He-HTLC to overcome the the above-mentioned attacks (SDRBA and HyDRA). He-HTLC achieve this by proposing two contracts 1) *He-Dep*, and 2) *He-Col*, like MAD-HTLC. The payment of v^{dep} and v^{col} is stored by Bob in *He-Dep*, which can be redeemed in two ways: 1) *dep-A:* Transfers v^{dep} to Alice and v^{col} to Bob, when Alice reveals pre_A. 2) *dep-B:* Deposit $v^{dep} + v^{col}$ to *He-Col* after time T, if Bob reveals pre_B. Similarly, *He-Col* can be redeemed in two ways: 1) *Col-B:* Transfer $v^{dep} + v^{col}$ to Bob after time $T + l$, and 2) *Col-M:* Transfer v^{col} to the miner producing both pre_A and pre_B while v^{dep} are burned.

With the above amendments, miners high earning is pre-



Users	Honest	Bribery
M_1	f_A^{dep}	0
M_2	0	0
M_3	f_B^{col}	$v^{dep}+v^{col}>f_A^{dep}$
Alice	v^{dep}	0
Bob	v^{col}	$-\left(v^{col}+f_B^{C_{Bob}}\right)$

(c)

Fig. 2: (a)Honest execution of MAD-HTLC, (b)Bob perform the bribery attack mentioned above on the MAD-HTLC, (c)Utility gained by each party during the honest and malicious execution of MAD-HTLC

vented by burning v^{dep} and hence allowing them to confiscate only v^{col} instead of $v^{dep}+v^{col}$. On the other hands, Bob's high earning is prevented by requiring him to disclose the secret pre_B (through the *dep-B* path) for l blocks before he can receive $v^{dep}+v^{col}$ via the *col-B* path. This separation provides two key benefits. Firstly, if pre_A is unavailable, Bob will receive $v^{dep}+v^{col}$ after a delay of l blocks. Secondly, if pre_A is known, miners can choose between two options: either they can confiscate v^{col} through *col-M*, or they can allow Bob to receive $v^{dep}+v^{col}$ and potentially earn more than v^{col} through a bribe from Bob. The authors prove that this bribe is avoidable by selecting an appropriate value of l.

III. Motivation (Attacks on the existing work)

We present two attacks: the Bribery-based Block Broadcasting Attack (B3A) and the Miner-to-Miner Bribery Attack (M2MBA). B3A targets MAD-HTLC, allowing Bob to extract nearly the same collateral as in a naive HTLC attack. In contrast, M2MBA targets He-HTLC, showing that miners profit more by attacking, regardless of Alice's or Bob's honesty.

A. Bribery based Block Broadcasting Attack (B3A)

The SDRBA and HyDRA protocols rely on the assumption of a fair exchange between the miner and Bob, wherein the miner only pays Bob a bribe if they can redeem v^{dep} and/or v^{col}. To implement this, [24] proposed the following approach: Bob creates a transaction tx_M^{dep} that redeems MH-Dep for M_i, sends $h=H(tx_M^{dep})$ to M_i, and provides a proof π to demonstrate its correctness. The miner, M_i, then verifies the proof and mines a partial block B. This block includes: 1) The hash of tx_M^{dep}, which is noteworthy as mining is typically performed on a block header that contains a compact representation of the transaction (e.g., a Merkle root that hashes all transactions), and 2) A bribing transaction that pays Bob br tokens from the coinbase, along with any other transactions from the mempool. Finally, the miner sends B to Bob, who checks that the block contains the intended transactions, completes B by adding tx_M^{dep}, and broadcasts the finalized block.

By broadcasting the completed block, Bob gains leverage to perform the bribery attack and extract additional collateral—matching what he would gain from bribing a naive HTLC. He finalizes and broadcasts the block only if it includes either of the following, along with unrelated transactions from the pool:

- case -1: a) The bribing coinbase transaction that pays Bob a bribe of $v^{dep}-br$, b) Hash of tx_M^{dep} which redeems v^{dep} for miner, c) tx_B^{col}, which redeem v^{col} for Bob at cost of paying a transaction fee of f_B^{col} to the miner, d) Transaction calling *claimBribe* of C_{Bob}.
- case -2: a) The bribing coinbase transaction that pays Bob a bribe of $v^{dep}+v^{col}-2br$, b) Hash of tx_M^{dep} and tx_M^{col} which redeems v^{dep} and v^{col} for miner, respectively, c) Transaction calling *claimBribe* of C_{Bob}.

Note that the above-mentioned block may appear after the timeout T. As a result of the attack, Bob must pay a bribe of $k\cdot br$ to each miner who mines a block between t_{pub} and T, and an additional br to the miner who includes the relevant block after T. Moreover, the miner should receive br for including tx_M^{dep}, and either f_B^{col} for including tx_B^{col} (Case 1), or br for including tx_M^{col}, compensating for the fees they forgo by not including unrelated transactions or tx_A^{dep} and/or tx_B^{col}. Consequently, Bob's net gain is calculated as $v^{dep}-[(k+2)*br+(br \text{ or } f_B^{col})+f_B^{C_{Bob}}]$ where k is the number of blocks mined in time $T-t_{pub}$. Assuming that $f_A^{col}=f_B^{dep}$, Bob can carry out the bribery attack with an additional cost of br compared to the attack on the naive HTLC.

It is important to note that B3A differs from the Success Independent Reverse Bribery Attack described in He-HTLC, which assumes that the miner refrains from sharing pre_B with any miner other than the briber. In contrast, B3A operates without making this assumption.

B. Miner to Miner Bribery Attack (M2MBA)

The above solutions proposed in He-HTLC guard when either party (Alice or Bob) colludes with miner. More precisely they guards against following two situations:

- Alice has colluded with a miner to receive greater incentives than she would by behaving honestly. This has been accomplished through the use of $v^{col}<v^{dep}$.
- Bob has conspired with a miner to receive greater incentives than he would through honest behavior. This is accomplished by permitting the miner to seize only v^{col} and incinerate v^{dep} when both pre_A and pre_B are available.

However, although Alice and Bob may act honestly, the possibility of bribery scenarios among miners where one miner bribes others and shares the high earnings gained from the attack is often overlooked in the proposed solutions. Here we will demonstrate that an miner can collaborate with other miners to earn more than they would have earned through honest behavior. If the miner chooses not to disclose pre_A on the chain, other miners may be willing to do the same, even if Alice discloses pre_A, as long as there is a potential for greater profit. The detailed attack steps are outlined as follows:

- Alice will reveal pre_A to the miner. However, the miner won't include the respective transaction (tx_A^{dep}) in the block, hence hiding pre_A. It also promises bribes to the other miners for not including tx_A^{dep} in the block.
- Miner waits until time T, after which Bob will broadcast tx_B^{dep} that reveals pre_B [4].
- Miner will follow the path *col-M* as they know pre_A and pre_B to confiscate the v^{col} and send the promised bribe to the other miners who censored tx_A^{dep} between t_{pub} and T. We called the miner who confiscate v^{col} a *bribing miner*.

By behaving honestly each miner can earn f_A^{dep}, or f_B^{col}, as shown in Figure 3a and 3b respectively. The later one is the case when Alice doesn't send tx_A^{dep} revealing pre_A. However, performing the above-mentioned attack the bribing miner, say M_i, will be able to gain $v^{col} - (k - k_{M_i}) * br$, where $br > f_A^{dep}$ and k and k_{M_i} is the total number of blocks mined and number of blocks mined by M_i in time $T - t_{pub}$, respectively. In the example showed in the Figure 3, where $k = 4$, and $k_{M_3} = 1$, M_3 is the bribing miner, who confiscate v^{col}. M_3 also promise the bribe to other miners for censoring tx_A^{dep} by locking v^{col} in contract, say C_{M2M}, details of which is explained further in section V-B. Given this, M_3 gained $v^{col} - 3 * br > f_B^{col}$, which is depicted in Figure 4. While other miners, say M_j, will gain $br * k_{M_j}$, where k_{M_j} is the number of blocks mined by M_j during time t_{pub} to T.

On the other hands, if $v^{col} - 3 * br < f_B^{col}$, then M_3 (bribing miner) would prefer to include tx_B^{dep} and earn f_B^{col}. However, because we assumed that $v^{col} > f_A^{dep}$ and $v^{col} > f_B^{col}$, it would rarely be the case. This is a legitimate assumption because the transaction fee is much lower than the transaction value.

Here we have distributed the bribe, where the bribing miner, M_i, would get $v^{col} - (k - k_{M_i}) * br$ and other miners would receive $br * k_{M_j}$. Another way to distribute bribes is equally distributing v^{col} among all involved in the attack, i.e. bribe miners and miners responsible for censoring tx_A^{dep}. With such distribution each miner would earns $v^{col} * k_{M_i}/k$. For ease of explanation in our analysis in section V-B, we used this (equal) bribe distribution. To overcome these attacks, we propose DEMBA, which we discuss next.

[4]Note that in this scenario, Alice and Bob is honest and adheres to the protocol. Since the malicious miner does not reveal any information about pre_A on-chain, Bob will proceed as specified and honestly release pre_B according to the protocol.

IV. DEMBA Overview

We begin this section by asking the question "Can we close the doors for attack using DEMBA?". We provide a positive response to the preceding question by introducing a novel HTLC protocol, referred to as DEMBA, which is designed to be incentive-compatible for actively rational miners, taking into account all possible bribing tactics. The primary concern with the previous protocols ([23] and [24]) is, they allow the third entity (in our case miner), which is not a part of exchange, to confiscate the tokens (v^{dep} and/or v^{col}). Which give them a upper hand and hence enough power to perform the attack that breaks the exchange (transfer).

We propose the solution, which revoke the power of token confiscation from the miner, in case the parties deviate from the protocol. The intuition is both the parties, Alice and Bob, involved in the exchange deposit their token in the collateral contract in addition to the contract that performs exchange. Let C^{dep} be the contract that the Bob deploy to perform the exchange and deposit tokens v^{dep} in it. Furthermore, let C_A^{col} and the C_B^{col} be the collateral contract for Alice and Bob where they deposit tokens v_A^{col} and v_B^{col}, respectively. In addition to that, C_A^{col} and C_B^{col} will output the respective values that Alice and Bob used to redeem v_A^{col} and v_B^{col}, respectively. Unlike the existing solution, v^{dep} can't be redeemed manually, meaning by sending transaction that discloses the pre_A and/or pre_B. Instead C^{dep} will read the secrete from C_A^{col} and C_B^{col} to make a decision on whom to transfer v^{dep}, either Alice or Bob. Given this, each contract can be redeemed/unlocked in the following ways:

- C^{dep}: 1) dep-A ($tx_{pre_A}^{dep}$): Transfer v^{dep} to Alice, if C_A^{col} output only pre_A and C_B^{col} output pre_B 2) dep-B ($tx_{pre'_A}^{dep}$): Transfer v^{dep} to Bob after time T, if C_A^{col} output only $pre_A^{'}$ and C_B^{col} output pre_B. 3) dep-Burn ($tx_{pre'_{AA}}^{dep}$): Burn v^{dep} after time T, if C_A^{col} output both pre_A and $pre_A^{'}$, irrespective of whether C_B^{col} output pre_B before time T.
- $C_{A'}^{col}$: Alice has the option to redeem using pre_A and/or $pre_A^{'}$. If she redeems before time T, she must use pre_A ($tx_{pre_A}^{col}$). However, if she has already disclosed the transaction revealing pre_A, which was not recorded on the blockchain due to bribery till time T, she can choose to redeem using both pre_A and $pre_A^{'}$ ($tx_{pre'_{AA}}^{col}$), else she redeem with just $pre_A^{'}$ after time T ($tx_{pre'_A}^{col}$). Note that, when C_A^{col} is redeemed with $tx_{pre'_{AA}}^{col}$, Alice will earn $v_A^{col} - v^{ded}$.
- C_B^{col}: Bob can unlock it at anytime using pre_B ($tx_{pre_B}^{col}$). However, After time T, v_B^{col} is deducted by some coins, say v^{ded}.

In addition to the redemption mechanisms discussed above, DEMBA distinguishes between the transaction fee paid by the sender and the fee actually earned by miners for including a transaction in a block. Let $f_{pre_A}^{col}$, $f_{pre'_A}^{col}$, $f_{pre'_{AA}}^{col}$, and $f_{pre_B}^{col}$ denote the transaction fees paid by Alice and Bob for transactions $tx_{pre_A}^{col}$, $tx_{pre'_A}^{col}$, $tx_{pre'_{AA}}^{col}$, and $tx_{pre_B}^{col}$, respectively. Similarly,

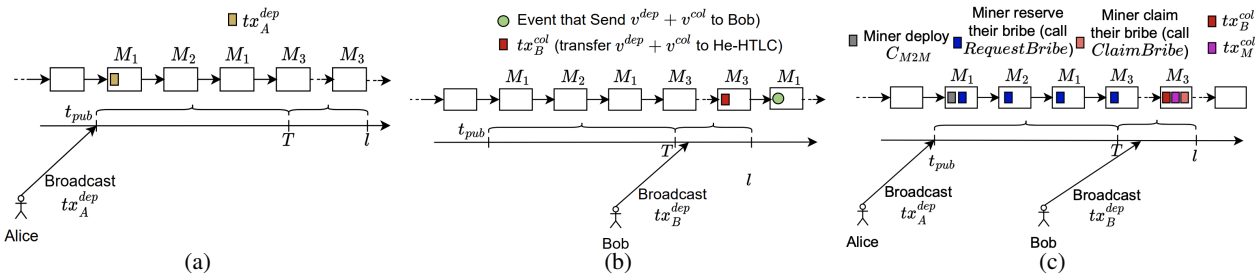


Fig. 3: (a) Honest execution of He-HTLC when Alice release the pre_A before time T, (b) Honest execution of He-HTLC when Bob release the pre_B post time T, and (c) Miner to miner bribery attack (M2MBA) on He-HTLC.

Users	Honest (Only Alice release pre_a)	Honest (Only Bob release pre_b)	M2MBA
M_1	f_A^{dep}	0	$2 \cdot br > f_A^{dep}$
M_2	0	0	$br > f_A^{dep}$
M_3	0	f_B^{col}	$v^{col} - \left(3 \cdot br + f_M^{C_{M2M}}\right) > f_B^{col}$
Alice	$v^{dep} - f_A^{dep}$	0	0
Bob	v^{col}	$v^{dep} + v^{col} - f_B^{col}$	0

Fig. 4: Utility gain by each involved party during honest execution of He-HTLC and during M2MBA attack.

let $fm_{pre_A}^{col}$, $fm_{pre'_A}^{col}$, $fm_{pre'_{AA}}^{col}$, and $fm_{pre_B}^{col}$ represent the portion of those fees actually earned by miners for including the respective transactions in a block.

To incentivize honest behavior, DEMBA enforces the following conditions:

$$f_{pre_A}^{col} < f_{pre'_A}^{col} < f_{pre'_{AA}}^{col} \tag{1}$$

$$fm_{pre_A}^{col} > fm_{pre'_A}^{col} > fm_{pre'_{AA}}^{col} \tag{2}$$

For $tx_{pre_B}^{col}$, miners earn a higher fee if they include it before time T; after T, their earnings decrease. However, Bob pays a higher fee if the transaction is included after T. The increasing gap between fees paid and fees earned by miners is burned, discouraging protocol deviations. For the honest behavior the fee paid by the party and the fee earn by the miner is same, i.e. difference is zero. For example, $f_{pre_A}^{col} = fm_{pre_A}^{col}$, and $f_{pre_B}^{col} = fm_{pre_B}^{col}$, if tx_B^{col} is included in the block before T. Such fee structure can be easily implemented using condition-based UTXOs in UTXO-based blockchains. In smart contract-based blockchains, it becomes even simpler by incorporating the condition directly within the relevant contract methods.

Given this knowledge, we can conclude that when all parties act honestly, Alice redeems C_A^{col} using pre_A, while Bob redeems C_B^{col} using pre_B. This provides sufficient information to unlock C^{dep} and transfer v^{dep} to Alice. On the other hand, if Alice fails to redeem C_A^{col} before time T, she can still redeem it using pre'_A after T, allowing Bob to refund v^{dep}. However, if Alice reveals pre_A before T but it is not included in the block, she will publish both pre_A and pre'_A to redeem C_A^{col}. In this case, if C^{dep} reads both pre_A and pre'_A from C_A^{col}, it will permanently lock or burn v^{dep}.

Bob's bribery attack mentioned above doesn't make him gain more than what he would have earned by behaving honestly, i.e. if he bribe miners for not including $tx_{pre_A}^{dep}$ in the block, then after time T, Alice will redeem C_A^{col} with both pre_A and pre'_A. So C^{dep} will burn v^{dep} and Bob end up earning $v_B^{col} - k * br$, where $br > f_A^{dep}$, and k is the number of blocks mined between time t_{pub} and T.

Furthermore, If Bob don't release tx_B^{col}, revealing pre_B, before T, he would earn $v_B^{col} - v^{ded}$, neither he gets any share in v^{dep}. Similarly, if Alice doesn't release pre_A, then he will end up loosing v^{dep}.

With this brief understanding of the proposed attacks and DEMBA next, we discuss the detailed analysis of the proposed M2MB attack on the He-HTLC and provide the game theoretic argument to show the gain that miners get by deviating from the protocol. Furthermore, we will provide the analysis of DEMBA.

V. M2MBA ANALYSIS

This section outlines our system model for analyzing the proposed M2MBA on the He-HTLC and our protocol. Following this, we will delve into the analysis of both.

A. System Model

The system model we adopt bears resemblance to the one employed in MAD-HTLC [23] and He-HTLC [24]. Our model considers a blockchain-based cryptocurrency that enables token transactions among a group of participants (entities). These participants include Alice, Bob, other users, and a predetermined group of n miners, which we refer to as $M = M_1, ..., M_n$.

Each entity (including miners) in the system can create transaction that deploy a contract. The contract deposit tokens and defines the predicate on the token transfer. Furthermore, contract can output specific value, which the other contract can read. Each entity can create transaction that transfer token to other entities. A transaction transferring the token must supply input values such that the contract predicate evaluated over them is true, after which tokens are transferred to the specific entity defined in the contract, and contract is said to be *redeemed*. Transaction must satisfy specifically the following predicates, in addition to the other predicates: 1) digital signature sig provided by the transaction matches a public key pk specified in the contract, 2) preimage pre provided by the transaction matches a hash digest dig specified in the contract, i.e., that $H(pre) = dig$.

For the predicate to be imposed correctly we assumed that every participant has access to a digital signature scheme [25] with a security parameter of μ and a hash function, denoted as $H : \{0,1\}^* \rightarrow \{0,1\}^\mu$, which maps arbitrary-length inputs to outputs of length μ. The security parameter μ has been selected such that the standard cryptographic assumptions are satisfied, which implies that the digital signature scheme is resistant to existential forgery attacks [25], [26], and the hash function H is resistant to preimage attacks [27].

Invalid transactions with a negative predicate value will not be included in any block and will be ignored. For a transaction to be considered valid, the token transferred must not exceed the amount deposited in the contract at the time of deployment. The transaction fee is calculated as the difference between the transferred token amount and the deposited amount.

The created transactions are submitted to the miners. Miners add the received valid transaction in the data structure called *mempool*. The transaction in *mempool* are called as *unconfirmed* transactions, which miner can confirm by including them in the block. Miners extends the blockchain by creating the new block. Each miner i can create a block with a specific rate denoted by λ_i, such that $\sum_n^{i=1} \lambda_i = 1$. The block appended at the j^{th} height is referred to as a b_j. We further assume that block-creation as a discrete-time, memoryless stochastic process. We also assume that, only one miner mines a block at specific time instance, hence there doesn't exist a transient inconsistencies in our system.

Furthermore, We assume there are two types of miners: *active miners and passive miners*. Active miners proactively seek out other participants to participate in external protocols, say by deploying contracts, which we are going to see next section (§V-B). By doing so, these miners can identify and capitalize on opportunities that maximize their utility in terms of the number of tokens earned. They do so by dividing the gained utilities among the colluding parties. Active rational miners are considered more formidable than their passive counterparts which only uses the transaction information available in the mempool to maximize their utility.

Out of n miners, say col miners are active miners. The consolidated mining power of the active miners is represented as λ_{col}. Active miners agree to share the rewards proportional to their individual mining power. The probability that a specific miner M_i within the colluding group gets the reward, given that one of the colluding miners mines the block, is proportional to their mining power. In other words, the probability that miner M_i receives a reward, given that one of the active colluding miners mines the block, is given by: P(Miner M_i receives reward/one colluding miner mines) $= \lambda_i/\lambda_{col}$, where P represents probability function.

For the resemblance with the production blockchain, we assume that there exists other users which generate the transaction that are not related to the DEMBA, we called such transaction as an *unrelated* transaction. Each transaction offers a fee to the miner for their inclusion in the block. We further assume that transaction related to the DEMBA offers a higher transaction fee compared to the *unrelated* transactions, unless specified otherwise. Since we assume that forks do not exist, it is assumed that a transaction is confirmed as soon as it is included in a block.

Beyond the previous assumptions, we model the system as a game among various entities, including miners. All entities are rational, risk-neutral, and non-myopic, with ties broken randomly. They aim to maximize expected utility, and no discounting is applied—i.e., a payment of x tokens holds equal utility regardless of time. These assumptions are consistent with those in MAD-HTLC and He-HTLC.

B. M2MBA on He-HTLC

In this section, we will delve into the M2MBA on the He-HTLC. Initially, we will discuss about the attack strategy of active miners. The basic idea is that some miner will deploy the contract C_{M2M}, and each miner M_i will lock their collateral of v^{col} in it. Locked collateral acts as a promise for getting bribed for censoring tx_A^{dep}. The miner who confiscates the v^{col} following *col-M* will send the promised bribe to the other miners for censoring tx_A^{dep}. On the other hand, for passive miners, the attack strategy is simply to exclude tx_A^{dep} from the block until time T and include tx_M^{col} as soon as Bob releases tx_B^{dep}. Next, we discuss the M2MB attack in detail.

1) M2MBA Details: : Let's say a He-HTLC is established between Bob and Alice, where they both deposits v^{dep} and v^{col}, respectively. The attack proceed in three phases First, a miner deploy the contract C_{M2M} represented in Algorithm 2, followed by which each miner, lock the amount of v^{col} in the contract by calling *LockCollateral*. They can do this before or immediately after Alice publishes tx_A^{dep}. Second, each miner mining the block between time t_{pub} to T lock their bribe (bribe they expected to earn in future for censoring tx_A^{dep}) by calling *RequestBribe*. The idea behind this is that if miners are assured of receiving a profitable amount in the future, they are more likely to choose not to include tx_A^{dep} in the block.

Lastly, immediately after time T (i.e. at time $T+1$), when Bob discloses pre_B via tx_B^{dep}, transferring $v^{dep} + v^{col}$ to He-Col, miners vie for its inclusion in the block. Note that since Bob is honest, he discloses pre_B through tx_B^{dep} at time $T+1$. Since every miner is aware of both pre_A and pre_B, they also compete to add tx_M^{col} to the same block and confiscate v^{col}. The first miner, say M_{brb} who confiscate the v^{col} from ***He-Col*** by following col-M (i.e. by adding tx_M^{col} in the block) will send the promised bribe to the miners who mine the blocks between time t_{pub} to T for censoring tx_A^{dep}. The amount locked in the first step for other miners, except M_{brb}, will be unlocked and sent back to them. However, the amount remains after paying bribe is sent to M_{brb}, depicted at line 41-43 in Algorithm 2.

2) Game Details: We model the M2MBA as a game, ζ^{mba} between Alice, Bob, and miners $M = \{M_1, \cdots, M_n\}$. The game runs for $T+1$ rounds, where each round j is represented by a block b_j. Like in [24], without loss of generality, we say the game begins when He-Dep and He-Col contracts are initiated in some block b_0. Furthermore, the round during which Alice releases pre_A is identified as t_{pub}.

In every round k, each party (Alice, Bob, and Miners) observes the following four game states: 1) *red*: *He-Dep* remains redeemable; 2) *nred-nrev*: He-Dep has already been redeemed and forwarded to He-col, but the miners are unaware of pre_A; 3) *nred-rev*: *He-Dep* has already been redeemed and dispatched to He-col, and some miners are cognizant of pre_A, and 4) *nred-A*: He-Dep has already been redeemed with pre_A. We denote this states set as S =*red, nred-nrev, nred-rev, nred-A*.

Each round, $k \in [t_{pub}, T]$, corresponds to the subgame $\zeta^{mba}(k, s)$, where s represents the state and denotes that $T-k$ more blocks must be produced following this subgame. Hence we look the game as the series of subgames. We use "." as a placeholder when referring to subsets of games, such as $\zeta^{mba}(., red)$, which denotes the collection of all subgames in which *He-Dep* can still be redeemed. Within each subgame, the participating parties may execute certain *actions*, which have the potential to alter the game state. Next we will discuss the set of action that each participating entity can take.

Alice complies with the He-HTLC protocol, whereby she has the option to select a round t_{pub} to release tx_A^{dep}, offering a fee f_A^{dep} of her preference. Once Alice broadcasts tx_A^{dep}, i.e., exposes pre_A, and the miner includes tx_A^{dep} in the chain by mining the block, the protocol is completed. Bob's actions do not impact the protocol execution since exposing tx_A^{dep} transfers v^{dep} to Alice and v^{col} to Bob. Alternatively, in the scenario where tx_A^{dep} is not disclosed on-chain, tx_B^{dep} revealed by Bob will be placed on-chain after time T. This transaction will transfer the sum of v^{dep} and v^{col} to the *He-Col*, which will ultimately (after l rounds) be transferred to Bob if pre_A is not revealed, either by honest or malicious behaviour from Alice.

The actions of miners involve strategically selecting transactions to be mined in order to maximize their utility. The transactions available for miners to choose from are dependent on the current round k as well as their knowledge of pre_A and pre_B. In any given subgame $\zeta^{mba}(.,.)$, a miner M_i can include unrelated transactions from the mempool and earn a fee of f. However, in subgames $\zeta^{mba}(k, red)$, where $T \geq k \geq t_{pub}$, M_i has the option to include tx_A^{dep} for a fee of f_A^{dep} or censor it with the expectation of the higher profit ($> f_A^{dep}$) in future. Similarly, in subgames $\zeta^{mba}(k, red)$ where $k > T$, M_i can choose to include tx_B^{dep} for a fee of f_B^{dep}, but only if tx_A^{dep} has not already been included.

Furthermore, miners action is limited to include only the unrelated transactions to maximize their utility in the subgame $G^{dep}(k,$
nred-nrev). However, in subgames $G^{dep}(k,$*nred-rev*), where $k > T$, M_i can create and include a transaction tx_M^{col} that redeems He-Col for itself via path *col-M*. Furthemore, as soon as Alice reveals pre_A, a miner can decide to pay a pre-agreed br to other miners in exchange for allowing him to redeem He-col in some future block (after round T). For ease of exposition, we assume that miners have reached an agreement about the value of br before or immediately after t_{pub}. Miner can independently decide whether or not to accept each bribe br, however, each miner receives only one of these bribes, i.e., the bribe from the miner who redeem *He-Col*. By undertaking these actions, each participant in the game accumulates certain utilities, which we will delve into in the following discussion.

Each entity's utility, u_i, in the game is determined by the tokens they earn at the end of the game, i.e., after T blocks have been created. Specifically, for each entity i, the utility function u_i : Action X $(\mathbb{Z},$ States$) \rightarrow \mathbb{R}$ maps to the number of tokens entity i earns at the end of the game. The utility of player i when action a is taken in the game ζ^{mba} is denoted as $u_i(a, \zeta^{mba})$. The maximum utility that player i can achieve in game ζ^{mba} is referred to as $u_i^{max}(\zeta^{mba})$. We assume that the utility obtained from a block containing only unrelated transactions is f. Therefore, if a transaction related to *He-HTLC* with utility x is included in exchange for an unrelated transaction, it would have a utility of $x - f$.

Given above, our game-theoretic analysis demonstrates that it is a dominant strategy for a miner, M_j, is to accept a bribe from a bribing miner, if he couldn't redeem *He-Col*. Else the dominant strategy is to redeem *He-Col* than following the *He-HTLC* protocol.

3) Analysis: As discussed earlier, M2MBA proceeds in three phases. In the first (setup) phase, miners agree on the bribe br and each active miner locks a collateral of value v^{col}. Let λ_{col} denote the mining power of miners who locked their collateral via *LockCollateral*.

Given the above game setup, we prove that for all miners, censoring tx_A^{dep} and accepting the bribe is a dominant strategy when $t \leq T$. Lemma E shows that at round $T+1$, invoking He-Col to confiscate v^{col} is also dominant. Thus, attacking dominates honest participation (Theorem 1). Lemma proofs are deferred to Appendix E.

Lemma 1. *In $\zeta^{mba}(k, red)$, where $k \leq T$, if $(T - t_{pub}) * br * \lambda_i/\lambda_{col} > f_A^{dep}$, then it is a dominant move for active miners to accept the bribe by censoring the transaction tx_A^{dep} instead of including it in the block.*

Lemma 2. *In $\zeta^{mba}(k, red)$, where $k \leq T$, if $v^{col} * \lambda_i/\lambda col + (T - t_{pub}) * br * (2 * \lambda_i/\lambda_{col} - 1) > f_A^{dep}$, then it is a dominant move for active miners to offer a bribe to other miners to censor the transaction tx_A^{dep} and perform the attack rather than including it in the block.*

Lemma 3. *In $\zeta^{mba}(k, red)$, where $k \leq T$. If $v_{col} * \lambda_i > f_A^{dep}$, it is the dominant move for passive miners to censor tx_A^{dep} rather than include it in the block.*

Lemma 4. *In $\zeta^{mba}(k, red)$, where $k > T$. Let z be the net earning of the miner by confiscating v^{col}. If $z > f_A^{dep}$, and Bob release transaction tx_B^{dep} and hence release secret pre_B at $T+1$, it is the dominant move for all miners to mine the block that includes tx_M^{col}, at $k = T+1$. (transient transition from $\zeta^{mba}(k, red)$ to $\zeta^{mba}(k, nred\text{-}rev)$ by including tx_B^{dep} and then including tx_M^{col} in the same block)*

Lemma 5. *We made the valid assumption that for an active*

miner, inequality $v^{col}\lambda_i/\lambda_{col} + (T - t_{pub}) * br * (2\lambda_i/\lambda_{col} - 1) > f_A^{dep}$ *holds. Similarly, for a passive miner, Assumption* $v^{col} * \lambda_i > f_A^{dep}$ *is also valid.*

Theorem 1. *It is dominant strategy for all the rational miners to perform the M2MBA than following the He-HTLC protocol, if* $v^{col} * \lambda_i > f_A^{dep}$ *and* $v^{col}\lambda_i/\lambda_{col} + (T - t_{pub}) * br * (2\lambda_i/\lambda_{col} - 1) > f_A^{dep}$ *.*

Proof. We will divide the proof into two cases: the behavior of passive miners and that of active miners.

Passive Miners: The potential strategies for passive miners are as follows: a) Include tx_A^{dep} in the block. b) Wait until round $T+1$, when Bob releases tx_B^{dep}, and include both tx_B^{dep} and tx_M^{dep} in the same block in round $T+1$ (this constitutes an attack). c) Act honestly and include only tx_B^{dep}. Given that $f_A^{dep} = f_B^{dep}$, under choices a) and c), the miner would earn f_A^{dep}. However, according to Lemma E and Lemma E, choice b) dominates both a) and c), making the attack the most profitable strategy for passive miners.

Active Miners: The possible moves for active miners are: a) Include tx_A^{dep} in the block. b) Accept a bribe from other active miners to censor tx_A^{dep}. c) Offer a bribe to other active miners to censor tx_A^{dep} until round $T+1$, when Bob releases tx_B^{dep}, and then include both tx_B^{dep} and tx_M^{dep} in the same block. d) Act honestly and include only tx_B^{dep}. Again, assuming $f_A^{dep} = f_B^{dep}$, following options a) and d), the miner would earn f_A^{dep}. However, from Lemma E, option b) dominates both a) and d). Furthermore, from Lemma E and Lemma E, option c) also dominates a) and d). It is important to note that by accepting the bribe, the miners participate in the attack.

In conclusion, based on the analysis above, performing the attack is the dominant strategy for both passive and active miners, rather than following He-HTLC."

□

VI. DEMBA Analysis

In this section, we present a detailed analysis of DEMBA. The game setup remains consistent with that outlined in the previous section (Section V-B2). Let S represent the set of state of the game, where $S = \{all\text{-}red, nred\text{-}AB, nred\text{-}A'B, nred\text{-}AA'B, nred\text{-}ABT, nred\text{-}A'BT, nred\text{-}AA'BT\}$. The description of each state is as follows: 1) $all\text{-}red$ - indicates that all smart contracts are redeemable, 2) $nred\text{-}AB$ - C_A^{col} is redeemed with pre_A and C_B^{col} is redeemed before T, 3) $nred\text{-}A'B$ - C_A^{col} is redeemed with pre'_A and C_B^{col} is redeemed before T, 4) $nred\text{-}AA'B$ - C_A^{col} is redeemed with both pre_A and pre'_A, and C_B^{col} is redeemed before T. Similarly, $nred\text{-}ABT$, $nred\text{-}A'BT$, $nred\text{-}AA'BT$ follow the same description except C_B^{col} is redeemed after T.

Given this, we demonstrate that under DEMBA, no party-including miners-can profit by deviating from the protocol; in the state $nred\text{-}AB$, all involved parties earn more by adhering to it. Due to space constraints, all lemma proofs are provided in the appendix F.

Lemma 6. *For* $v^{ded} > 0$*, it is dominant move for the Alice to follow* DEMBA *than deviating from it.*

Lemma 7. *For* $v^{ded} > 0$*, it is dominant move for the Bob to follow* DEMBA *than deviating from it.*

Lemma 8. *For* $\alpha < 1$*, it is dominant move for the miners to follow* DEMBA *than deviating from it.*

Theorem 2. *It is dominant strategy for all the parties (Alice, Bob, and Miners) to follows the protocol than deviating from it.*

Proof. From Lemmas 6, 7, and 8, the intersection of states where each party maximizes its earnings reveals that $nred\text{-}AB$ is the common state offering high rewards for all. Thus, adhering to DEMBA emerges as the dominant strategy for Alice, Bob, and the miners, since any deviation yields lower payoffs.

Next, we demonstrate that no two parties can collaborate to earn more than they would by following DEMBA. There are three possible collaborations: (1) Alice-Bob, (2) Alice-Miner, and (3) Bob-Miner. Collaboration between Alice and Bob can be ruled out since they are counter-parties in the transaction.

If Alice behaves honestly, her expected earnings are $v^{dep} + v_A^{col}$. Since no additional tokens are created in the exchange process (except for mining rewards granted by the consensus protocol), Alice cannot earn more than $v^{dep} + v_A^{col}$ even by collaborating with the Miner. Similarly, the Miner cannot earn more than the transaction fee and mining reward, which they are already entitled to by behaving honestly.

Finally, consider collaboration between Bob and the Miner. In this case, Alice retains the deciding move on whether v^{dep} is returned to Bob. Thus, Bob cannot earn more than v_B^{col}, the amount he would receive by behaving honestly. Consequently, collaboration offers no advantage to Bob or the Miner.

□

VII. DEMBA Evaluation

In this section, we present the evaluation of DEMBA in comparison to state-of-the-art approaches. We begin by outlining the implementation details, followed by a description of the evaluation setup. Finally, we conclude with a discussion of the evaluation results.

A. Implementation Details

We demonstrate the effectiveness of DEMBA through its evaluation on both Bitcoin and Ethereum. While deploying DEMBA on smart contract-enabled blockchains like Ethereum and Hyperledger Fabric is relatively straightforward, this section focuses primarily on detailing our implementation of DEMBA for Bitcoin.

The Bitcoin scripting language lacks a built-in mechanism for storing and retrieving arbitrary data, such as interacting with variables or state, as is possible with Ethereum smart contracts. This limitation affects the communication between C^{dep}, C_A^{col}, and C_B^{col}. In Bitcoin, each smart contract is

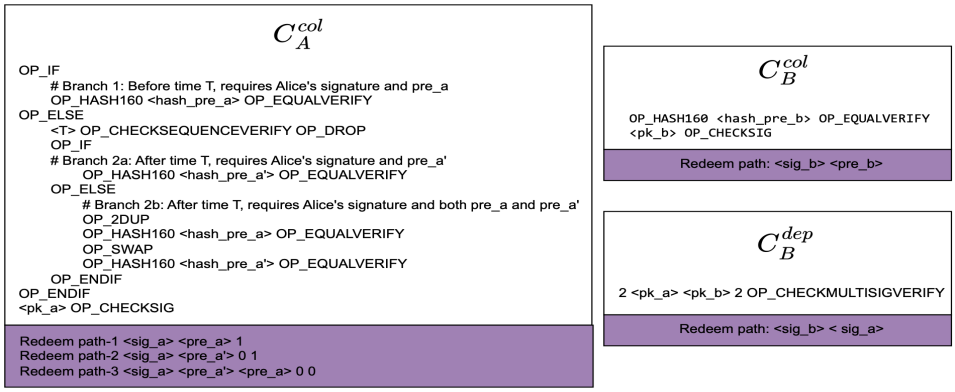


Fig. 5: Bitcoin implementation (locking script) for DEMBA contracts

represented as a UTXO, with respective names denoted as $UTXO^{C^{dep}}$, $UTXO^{C_A^{col}}$, and $UTXO^{C_B^{col}}$.

Alice creates $UTXO^{C_A^{col}}$, while Bob creates $UTXO^{C^{dep}}$ and $UTXO^{C_B^{col}}$. The Bitcoin locking scripts for these UTXOs are illustrated in Figure 5. As described earlier, $UTXO^{C_A^{col}}$ can be redeemed through three distinct paths:

- Using pre_A via the transaction $tx^{col}_{pre_A}$,
- Using pre'_A via the transaction $tx^{col}_{pre'_A}$, and
- Using both pre_A and pre'_A via the transaction $tx^{col}_{pre'_{AA}}$.

Each of these transactions must be signed by Alice.

In contrast, $UTXO^{C_B^{col}}$ has only one redeem path, which involves revealing pre_B through the transaction $tx^{col}_{pre_B}$, signed by Bob.

The final set of transactions transfers the funds to the respective parties by redeeming $UTXO^{C^{dep}}$. These transactions depend on how $UTXO^{C_A^{col}}$ is redeemed, leading to three possible paths:

- $tx^{dep}_{pre_A}$, which uses the outputs of $tx^{col}_{pre_A}$ and $tx^{col}_{pre_B}$,
- $tx^{dep}_{pre'_A}$, which uses the outputs of $tx^{col}_{pre'_A}$ and $tx^{col}_{pre_B}$, and
- $tx^{dep}_{pre'_{AA}}$, which uses the outputs of $tx^{col}_{pre'_{AA}}$ and $tx^{col}_{pre_B}$.

All these transactions require signatures from both Alice and Bob. This structured transaction mapping ensures the correct ordering of events, as each tx^{dep} transaction takes inputs from the relevant tx^{col} transactions. A detailed graph illustrating this process is provided in Figure 6.

This design addresses Bitcoin's inability to natively pass data between transactions, unlike Ethereum. Although Bitcoin provides the OP_RETURN opcode for embedding arbitrary data in transactions, Bitcoin Script cannot directly read such outputs. Instead, external mechanisms, such as Transaction Indexing Services or Custom Off-Chain Logic, can be employed to achieve this functionality.

The following steps outline the protocol that each party follows:

- Alice creates $UTXO^{C_A^{col}}$ along with the transactions $tx^{col}_{pre_A}$, $tx^{col}_{pre'_A}$, and $tx^{col}_{pre'_{AA}}$, and shares these transactions with Bob. Note that Alice does not share the signed versions of these transactions with Bob.

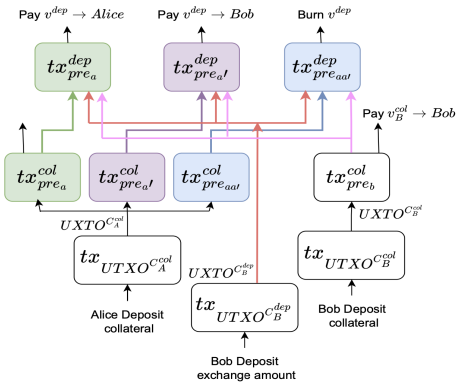


Fig. 6: Transaction dependency with respect to UTXO in Bitcoin implementation of DEMBA

- Bob creates $UTXO^{C_B^{col}}$ and $tx^{col}_{pre_B}$, then shares the unsigned version of $tx^{col}_{pre_B}$ with Alice. The first two steps are essential to verify that the construction of all tx^{dep}, as described in the following step, is correct.
- Bob further creates $UTXO^{C^{dep}}$ and the transactions $tx^{dep}_{pre_A}$, $tx^{dep}_{pre'_A}$, and $tx^{dep}_{pre'_{AA}}$. He signs these transactions and shares them with Alice. However, Bob keeps $UTXO^{C^{dep}}$ secret at this stage, i.e not publish it onto the blockchain.
- Alice signs $tx^{dep}_{pre_A}$, $tx^{dep}_{pre'_A}$, and $tx^{dep}_{pre'_{AA}}$, and sends the signed versions back to Bob, ensuring that both parties possess duly signed transactions.
- After receiving the signed versions of $tx^{dep}_{pre_A}$, $tx^{dep}_{pre'_A}$, and $tx^{dep}_{pre'_{AA}}$, Bob publishes $UTXO^{C^{dep}}$. This step is crucial because if Bob publishes $UTXO^{C^{dep}}$ before obtaining Alice's signatures for $tx^{dep}_{pre'_A}$ (or the other transactions), his tokens may remain locked in C^{dep} (i.e., in $UTXO^{C^{dep}}$ in a UTXO-based model) indefinitely if Alice withholds her signatures.

Once the above steps are done, Alice will send the transaction to the Blockchain, i.e., it send to a miner, which will broadcast it to the network. Next, we will discuss our network setup.

Furthermore, we have also implemented and evaluate the

DEMBA for Ethereum. For the ease of implementation[5], we haven't implemented signature verification in it. Moreover to make the comparison fair, we didn't implement the signature verification for He-HTLC and MAD-HTLC as well, with which we have compared DEMBA.

B. Evaluation Setup

We set up a private blockchain network consisting of 50 Oracle Virtual Machines to measure the protocol completion time—i.e., the time required to successfully complete the exchange—in a real network environment with potential blockchain forks [6]. In this setup, each VM is provisioned with a 2.19 GHz dual-core CPU, 8 GB RAM, and a 128 GB HDD. All VMs run Ubuntu 20.04 and are configured with download and upload bandwidths of 1 Gbps and 100 Mbps, respectively. Each VM hosts a single blockchain node.

Throughout the experiment, we control the block mining difficulty to maintain an average block generation time of 15 seconds for Ethereum and 10 minutes for Bitcoin, reflecting the time required to solve the Proof-of-Work (PoW) puzzle. As a result, the average block inter-arrival time in our experiment equals the PoW solving time (15 seconds or 10 minutes), plus the block propagation delay, block creation time, and block validation time. While Ethereum 2.0 [41], based on Proof-of-Stake (PoS), was already released at the time of our experiments, we deliberately focus on the earlier Proof-of-Work (PoW) version to ensure a consistent comparison with Bitcoin.

We allocate mining power to each node in our 50-node setup based on the mining power distribution of the top 50 miners in the Ethereum network [38]. Collectively, these miners account for approximately 99.98

Additionally, we replicate the geographical distribution of these miners by assigning our nodes locations consistent with the data from [38]. We configure inter-node latencies using Linux's `tc` tool to match the real-world ping delays corresponding to these geographic locations [39], effectively simulating the network delays of Ethereum's mainnet.

For network topology, inspired by the Bitcoin network-where node degrees follow a power-law distribution [40]-we design our experiment such that each node connects to a random subset of peers, ensuring the degree distribution adheres to a power-law.

C. Evaluation Result

In this section, we present the evaluation results. Specifically, we use two key metrics—transaction cost and time to completion—to compare the performance of DEMBA against state-of-the-art protocols.

The results for Bitcoin are shown in Figure 7, where DEMBA reduces Alice's redemption cost by 42% compared to MAD-HTLC but incurs 17% higher cost than He-HTLC, as it requires two transactions instead of one. In contrast, for Bob's refund, DEMBA lowers the cost by 42% relative to He-HTLC and by 150% compared to MAD-HTLC. Specifically, the transaction size in DEMBA is 48 bytes smaller than in He-HTLC and 294 bytes smaller than in MAD-HTLC.

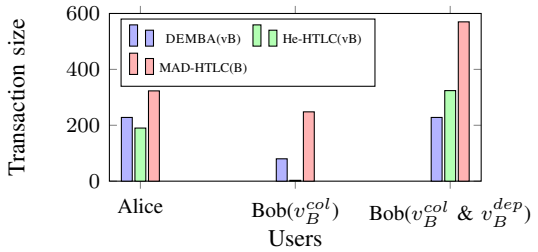


Fig. 7: Comparison of DEMBA, He-HTLC, and MAD-HTLC with respect to the transaction cost (size) on the Bitcoin Network

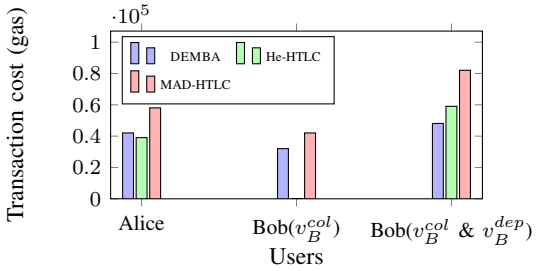


Fig. 8: Comparison of DEMBA, He-HTLC, and MAD-HTLC with respect to the transaction cost (gas used) on the Ethereum Blockchain Network

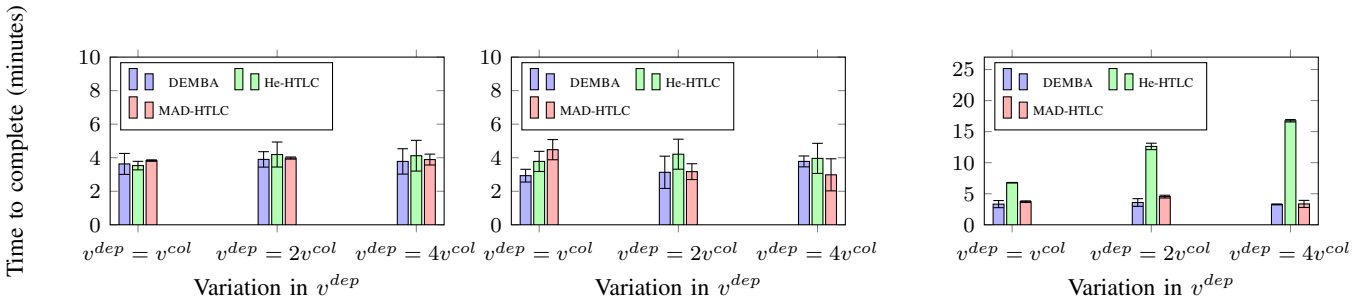
Figure 8 presents the Ethereum results, showing trends similar to Bitcoin but with smaller margins. DEMBA reduces Alice's claim cost by 38% compared to MAD-HTLC, but it is 7% costlier than He-HTLC, as it requires two transactions instead of one. For Bob's refund, DEMBA lowers the cost by 23% vs. He-HTLC and 71% vs. MAD-HTLC. Furthermore, upon closer analysis, we conclude that the observed results stem from the same reason—DEMBA requires Alice to send two transactions to redeem v^{dep}, whereas He-HTLC requires only one.

We further compared DEMBA with He-HTLC and MAD-HTLC in terms of time to complete (TTC), defined as the time from transaction initiation to final transfer. Results were averaged over 5 different runs, using $f_A^{dep} = f_B^{dep} = 0.0001 \times v^{dep}$, and reported with 95% confidence intervals.

Our observations are as follows: when Alice redeem v^{dep} and Bob redeem v^{col} (or v_B^{col} in the case of DEMBA), the TTC for all protocols, including DEMBA, shows negligible variation with changes in the exchange amount v^{dep}, as depicted in Figure 9a and 9b However, when Bob redeems both v^{dep} and v^{col}, all protocols except He-HTLC continue to show negligible variation in TTC. He-HTLC, on the other hand, exhibits an increase in completion time because its parameter l depends on v^{dep}. Specifically, to satisfy the condition $v^{col} \geq \left(\frac{v^{dep}}{\kappa - 1}\right) + f$, an increase in v^{dep} necessitates a

[5]You can find the Solidity implementation of DEMBA here https://github.com/nitinawathare14/DEMBAEth.git

[6]It is worth noting that, for comparing DEMBA in terms of transaction costs, deploying and sending transactions on a single-node blockchain network is sufficient.



(a) Variation in protocol completion time when Alice redeem v^{dep}

(b) Variation in protocol completion time when Bob redeem v^{col} (or v_B^{col})

(c) Variation in protocol completion time when Bob redeems v^{col} (or v_B^{col}) and v^{dep}

Fig. 9: Comparison of DEMBA, HTLC, He-HTLC, and MAD-HTLC with time to completion on the Bitcoin Blockchain Network

corresponding increase in κ, which is directly proportional to the parameter l. Figure 9c demonstrates this behavior.

From this, we conclude that DEMBA's performance in terms of protocol completion time is on par with, if not better than, the existing state-of-the-art protocols under consideration. Moreover, DEMBA achieves this while incurring lower costs compared to them.

VIII. Related Work

This section reviews related work in three areas: bribery attacks and countermeasures, mechanisms for fair exchange exploited in our approach, and strategies for optimizing transaction placement to maximize incentives—an alternative bribery attack.

Bribery Attacks: There have been proposals for bribery attacks aimed specifically at HTLC. In [21], temporary censorship attacks are suggested where attackers can censor transactions from a victim until a timeout, potentially allowing Bob to censor Alice's transaction and reverse the payment. However, their attack focus primarily on collaboration of one of the parties involved in the exchange to the consensus nodes (miners).

The incentive compatibility of blockchain protocols has received extensive research attention. For example, the selfish mining attack [31] demonstrates that Bitcoin is not incentive-compatible, and miners could gain more by selectively withholding blocks. Bonneau et al. [32] demonstrate how miners can be bribed, potentially utilizing the blockchain itself, to gain control of the system temporarily. McCorry et al. [33] employ smart contracts to facilitate sophisticated bribery attacks, such as transaction censorship and even the modification of the blockchain's history. Other approaches to implementing bribery have also been proposed. For instance, Liao [34] shows that bribes can be embedded in transaction fees to incentivize historical rewrites. According to [24], the reverse bribery attack involves miners bribing the parties participating in the exchange to split the profits earned. This results in both parties receiving more collateral than they would have if they had acted honestly.

HaoChung et al. proposed a solution named PONYTA [37] to prevent miner collusion with Alice or Bob to gain more collateral. To achieve this, they required Alice and Bob to each deposit c_a and $c_b + v$ into the contract, which is hash locked with h_a and h_b respectively. Here, $c_a > v$ and $c_b > 2v$, where v is the exchange amount. If any party, denoted by P, sends (pre_a, pre_b) such that $H(pre_a) = h_a$ and $H(pre_b) = h_b$, then the solution sends v to party P, while all the remaining coins are burnt. In PONYTA, the protocol requires Alice and Bob to deposit collateral exceeding the value of the tokens being exchanged, making it less favorable for users.

Dimitris et al. [42] examines bribing attacks in Proof-of-Stake (PoS) distributed ledgers through a game-theoretic lens. However, It does not explicitly model long-term strategic behaviors beyond the immediate bribing rounds.

From this, we conclude that previous works have not explored the scenario where a miner bribes other miners to execute an attack—a key focus of our study. [36] analyzed the Bitcoin blockchain and found that selfish miners engaged in bribing lower-status compliant miners to maximize their profits.

Fair exchange of collateral over a blockchain The authors of [24] introduced novel fair exchange protocols that enable reverse bribery. Unlike previous studies that propose fair exchange solely between the two non-mining parties, this approach achieves fair exchange between the asymmetrical parties involved, namely the miner and Bob.

However, The approach presented in [24] grants Bob more authority by allowing him to broadcast the completed block, which we leverage in our study to devise a novel attack on MAD-HTLC. Our findings demonstrate that Bob can earn a similar amount of collateral as he would by attacking a naive HTLC.

Transaction placement by the miner to create a block: Miners are usually portrayed as rational actors who prioritize transactions with higher fees to include in their blocks. However, according to [24], they can achieve better outcomes by actively generating more opportunities, such as by collaborating with certain parties involved in the exchange and sharing the net profit. These miners, who create such opportunities, are referred to as active rational miners. [35] shows one such way to create an opportunity by manipulating transaction orderings and mounting, e.g., frontrunning attacks. This surplus is referred to as *Miners Extractable Value (MEV)*.

In contrast, DEMBA is resistant to such attacks, as the potential rewards for dishonest behavior are lower than those earned by acting honestly.

REFERENCES

[1] Nakamoto, Satoshi and others (2008). Bitcoin: A peer-to-peer electronic cash system. Working Paper.

[2] Vitalik Buterin (2015). A NEXT GENERATION SMART CONTRACT & DECENTRALIZED APPLICATION PLATFORM. In .

[3] (). Hash Time Locked Contracts - Bitcoin Wiki.. https://en.bitcoin.it/wiki/Hash Time Locked Contracts.

[4] Joseph Poon and Thaddeus Dryja (2016). The Bitcoin Lightning Network:Scalable Off-Chain Instant Payment. https://lightning.network/lightning-network-paper.pdf.

[5] (). Real-Time Lightning Network Statistics.. https://1ml.com/statistics.

[6] Matteo Campanelli and Rosario Gennaro and Steven Goldfeder and Luca Nizzardo (2017). Zero-Knowledge Contingent Payments Revisited: Attacks and Payments for Services. https://eprint.iacr.org/2017/566.

[7] G Maxwell (). The first successful zero-knowledge contingent payment.. https://bitcoincore.org/en/2016/02/26/zero-knowledge-contingent-payments-announcement/.

[8] Wacław Banasik and Stefan Dziembowski and Daniel Malinowski (2016). Efficient Zero-Knowledge Contingent Payments in Cryptocurrencies Without Scripts. https://eprint.iacr.org/2016/451.

[9] Fuchsbauer, Georg (2019). WI Is Not Enough: Zero-Knowledge Contingent (Service) Payments Revisited. In Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security.

[10] Bursuc, Sergiu (2019). Contingent Payments on a Public Ledger: Models and Reductions for Automated Verification. In Computer Security – ESORICS 2019.

[11] Awathare, Nitin and Das, Sourav and Ribeiro, Vinay J. and Bellur, Umesh (2021). RENOIR: Accelerating Blockchain Validation using State Caching. In Proceedings of the ACM/SPEC International Conference on Performance Engineering.

[12] Herlihy, Maurice (2018). Atomic Cross-Chain Swaps. https://arxiv.org/abs/1801.09515.

[13] Giulio Malavolta and Pedro Moreno-Sanchez and Clara Schneidewind and Aniket Kate and Matteo Maffei (2018). Anonymous Multi-Hop Locks for Blockchain Scalability and Interoperability. https://eprint.iacr.org/2018/472.

[14] Meyden, Ron van der (2019). On the specification and verification of atomic swap smart contracts (extended abstract). In 2019 IEEE International Conference on Blockchain and Cryptocurrency (ICBC).

[15] Mahdi H. Miraz and (2019). Atomic Cross-chain Swaps: Development, Trajectory and Potential of. .

[16] Zie, Jean-Yves and Deneuville, Jean-Christophe and Briffaut, Jérémy and Nguyen, Benjamin (2019). Extending Atomic Cross-Chain Swaps. In .

[17] Möser, Malte (2016). Bitcoin Covenants. In Financial Cryptography and Data Security.

[18] McCorry, Patrick (2018). Why Preventing a Cryptocurrency Exchange Heist Isn't Good Enough. In Security Protocols XXVI.

[19] Bryan Bishop (). Bitcoin vaults with anti-theft recovery/clawback mechanisms.. https://lists.linuxfoundation.org/pipermail/bitcoin-dev/2019-August/017231.html.

[20] Zamyatin, Alexei and Harz, Dominik and Lind, Joshua and Panayiotou, Panayiotis and Gervais, Arthur and Knottenbelt, William (2019). XCLAIM: Trustless, Interoperable, Cryptocurrency-Backed Assets. In 2019 IEEE Symposium on Security and Privacy (SP).

[21] Winzer, Fredrik and Herd, Benjamin and Faust, Sebastian (2019). Temporary Censorship Attacks in the Presence of Rational Miners. In 2019 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW).

[22] Harris, Jona and Zohar, Aviv (2020). Flood & Loot: A Systemic Attack on The Lightning Network. In Proceedings of the 2nd ACM Conference on Advances in Financial Technologies.

[23] Tsabary, Itay and Yechieli, Matan and Manuskin, Alex and Eyal, Ittay (2021). MAD-HTLC: Because HTLC is Crazy-Cheap to Attack. In 2021 IEEE Symposium on Security and Privacy (SP).

[24] Sarisht Wadhwa and Jannis Stoeter and Fan Zhang and Kartik Nayak (2022). He-HTLC: Revisiting Incentives in HTLC. https://eprint.iacr.org/2022/546.

[25] Christian Badertscher and Ueli Maurer and Daniel Tschudi and Vassilis Zikas (2017). Bitcoin as a Transaction Ledger: A Composable Treatment. https://eprint.iacr.org/2017/149.

[26] Goldwasser, Shafi and Micali, Silvio and Rivest, Ronald L. (1988). A Digital Signature Scheme Secure against Adaptive Chosen-Message Attacks. Society for Industrial and Applied Mathematics.

[27] Stefan Dziembowski and Lisa Eckey and Sebastian Faust (2018). FairSwap: How to fairly exchange digital goods. https://eprint.iacr.org/2018/740.

[28] (). Ethereum. (2020) Solidity language. https:.

[29] (). etherscan transaction fee. https://etherscan.io/txs.

[30] Mathieu Baudet and Avery Ching and Andrey Chursin and George Danezis and François Garillot and Zekun Li and Dahlia Malkhi and Oded Naor and Dmitri Perelman and Alberto Sonnino (2019). State Machine Replication in the Libra Blockchain. In .

[31] Ittay Eyal and Emin Gun Sirer (2013). Majority is not Enough: Bitcoin Mining is Vulnerable. .

[32] Bonneau, Joseph (2016). Why Buy When You Can Rent?. In Financial Cryptography and Data Security.

[33] McCorry, Patrick and Hicks, Alexander and Meiklejohn, Sarah (2018). Smart Contracts for Bribing Miners. In Financial Cryptography and Data Security: FC 2018 International Workshops, BITCOIN, VOTING, and WTSC, Nieuwpoort, Curaçao, March 2, 2018, Revised Selected Papers.

[34] Liao, Kevin (2017). Incentivizing Blockchain Forks via Whale Transactions. In Financial Cryptography and Data Security.

[35] Daian, Philip and Goldfeder, Steven and Kell, Tyler and Li, Yunqi and Zhao, Xueyuan and Bentov, Iddo and Breidenbach, Lorenz and Juels, Ari (2020). Flash Boys 2.0: Frontrunning in Decentralized Exchanges, Miner Extractable Value, and Consensus Instability. In 2020 IEEE Symposium on Security and Privacy (SP).

[36] Roi Bar-Zur and Danielle Dori and Sharon Vardi and Ittay Eyal and Aviv Tamar (2023). Deep Bribe: Predicting the Rise of Bribery in Blockchain Mining with Deep RL. https://eprint.iacr.org/2023/472.

[37] Hao Chung and Elisaweta Masserova and Elaine Shi and Sri AravindaKrishnan Thyagarajan (2022). Ponyta: Foundations of Side-Contract-Resilient Fair Exchange. https://eprint.iacr.org/2022/582.

[38] (). Mining Power Distribution of Ethereum. https://investoon.com/charts/mining/eth.

[39] (). Global Ping Latency. https://wondernetwork.com/pings.

[40] Baumann, Annika and Fabian, Benjamin and Lischke, Matthias (2014). Exploring the Bitcoin Network. In .

[41] Ethereum Foundation (2020). Ethereum 2.0 Documentation. .

[42] Dimitris Karakostas and Aggelos Kiayias and Thomas Zacharias (2024). Blockchain Bribing Attacks and the Efficacy of Counterincentives. https://arxiv.org/abs/2402.06352.

APPENDIX A
DISCUSSION & FUTURE WORK

Different bribery scenarios: Bribery can involve any parties in a protocol, with miners actively seeking bribes or acting rationally based on their transaction pools. Existing models like MAD-HTLC and He-HTLC overlook miner collusion, focusing only on bribery between miners and a single party. Considering these overlooked scenarios, we propose the M2MBA attack, where miners collaborate to share bribery profits, and B3A, a stronger attack on MAD-HTLC allowing Bob to earn rewards comparable to attacking naive HTLC.

More broadly, any application-level protocol that relies on miners as enforcers must account for the risk of active miner participation in bribery and manipulation of application logic. Exploring such miner-driven application-level attacks presents a promising direction for future research.

Our solution, i.e. DEMBA**:** Our proposed solution, DEMBA, mitigates bribery among miners by removing their control over token confiscation and structuring the exchange into two distinct phases. In the first phase, the participating parties commit to either proceed with the exchange or abort it—returning both

the exchange tokens and collateral to Bob. In the second phase, this commitment is executed, and the actual token transfer occurs.

While DEMBA addresses bribery scenarios within a single blockchain network, its effectiveness in cross-chain exchanges remains uncertain. Even when following DEMBA, inter-blockchain exchanges may still be vulnerable to bribery attacks. Adapting DEMBA to securely handle cross-chain exchanges is an open research challenge.

On Assumptions: In our analysis, we make two primary assumptions: (i) only one block is mined per round, and (ii) token values remain constant throughout the protocol execution. The first assumption is reasonable for our attacks, as we only need to demonstrate the existence of scenarios where the attacks succeed. For DEMBA, since miners no longer have the ability to confiscate tokens, they gain no additional advantage by forking the blockchain. The second assumption holds for exchanges within a single blockchain network. However, when DEMBA is applied to inter-blockchain exchanges, analyzing the impact of token price volatility becomes a crucial research direction, as different tokens may exhibit varying growth rates.

APPENDIX B
SMART CONTRACT TO PERFORM BRIBERY ATTACK ON NAIVE HTLC

In this section, we provide a sample smart contract (Algorithm 1) that demonstrates a bribery attack on a naive HTLC. We will extend this contract to implement the M2MBA attack. The contract is initialized by the attacker (Bob, in our case) using the *init* method, with a deposit of v^{dep}.

The contract consists of two primary methods: *requestBribe* and *claimBribe*. The *requestBribe* method allows miners to reserve their future bribe by committing to censor tx_A^{dep} before the attack succeeds. Once the attack is successful, *claimBribe* facilitates the actual bribe transfer from Bob's deposit to the participating miners.

Notably, calling *claimBribe* requires pre_A as input to prove that Alice has revealed it. If Alice remains silent until time T, miners cannot claim the bribe, and Bob is refunded his full deposit of v^{dep}.

Algorithm 1 Bribery Contract (C_{Bob})

Variables:
$v^{dep} \leftarrow 0$; $balLeft \leftarrow 0$;
$Bal \leftarrow \{\}$ //mapping from address to balances

Constants:
$br = f_A^{dep} + \epsilon$

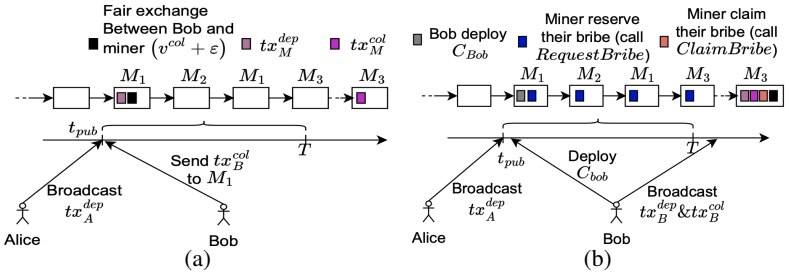
Functions:
//Called during contract deployment
procedure INIT(val)
  $v^{dep} \leftarrow val$; $balLeft \leftarrow v^{dep}$
end procedure

//Called by the miners to lock their bribe
procedure REQUESTBRIBE( )
  if caller is not the miner of the current block **or** function is called more than once in the block **then**
    return
  end if
  if height of the current block is $\leq T$ **then**
    $Bal[caller] \leftarrow Bal[caller] + 1$
    $balLeft \leftarrow balLeft - br$
  end if
end procedure

//Anyone can call to claim their bribe
procedure CLAIMBRIBE($preImage$)
  $count \leftarrow 0$
  if $preImage$ is not a pre_A **or** tx_A^{dep} is included in the block **or** tx_B^{dep} is not included in the block till the current block **or** $balLeft - (br + f_B^{dep} + f_B^{C_{Bob}}) < 0$ **then**
    return
  end if
  for Each $(addr, bal) \in Bal$ **do**
    Send $br * bal$ token of bribe to $addr$
    $count \leftarrow count + 1$
  end for
  //Gain to the caller
  send br to the caller (miner).
  //Send the remaining tokens to Bob
  send $v_{dep} - (count * br + br)$ to Bob
  $Bal \leftarrow \{\}$ //Reset the values
  $v^{dep} \leftarrow 0$
end procedure
//Refund to the bob if miner include tx_A^{dep}
procedure REFUNDTOBOB(val)
  if $tx_A^{dep} \in$ block with block height $\leq T$ **then**
    send v^{dep} to Bob.
  end if
end procedure

APPENDIX C
THE SUCCESS-DEPENDENT REVERSE BRIBERY ATTACK (SDRBA) AND HYBRID DELAY-REVERSE BRIBERY ATTACK (HYDRA).

The success-dependent reverse bribery attack (SDRBA) demonstrates that a reverse bribery attack can be successful when $v^{dep} > v^{col}$. In SDRBA, a miner, M_1 in our example in Figure 10a, pays a bribe ($> v^{col}$) to Bob only if it can redeem v^{dep} through *dep-M*. However, M_1 must reveal the pre_A and pre_B while redeeming v^{dep}, so all miners have access to pre_A and pre_B, and they compete to redeem v^{col} after timeout T. The probability of miner M_1 mining the block and redeeming v^{col} is λ_{M_1}/λ, where λ_{M_1} is M_1's block generation rate. Therefore, in the worst case of SDRBA, where another miner redeems v^{col} through *col-M*, the bribing miner (M_1) earns $v^{dep} - (v^{col} + \epsilon)$, as depicted in Figure 10c. Consequently, if $v^{dep} \leq v^{col}$, the attack may not be advantageous.



Users	SDRBA	HyDRA
M_1	$v^{dep}-\left(v^{col}+\varepsilon\right)$	$2\cdot br > f_A^{dep}$
M_2	0	$br > f_A^{dep}$
M_3	v^{col}	$v^{dep}+br-\varepsilon > f_A^{dep}$
Alice	0	0
Bob	$v^{col}+\varepsilon$	$v^{col}+\varepsilon-\left(5\cdot br+f_B^{C_{Bob}}\right) > v^{col}$

(c)

Fig. 10: (a)SDRBA on MAD-HTLC, (b)HyDRA on MAD-HTLC, (c)Utility gained by each party after performing SDRBA and HyDRA on MAD-HTLC

To overcome the disadvantage of SDRBA, the HyDRA is proposed which is designed to work for any values of v^{dep} and v^{col}. In HyDRA, Bob initiates the process by bribing miners to exclude transaction tx_A^{dep} until time T, at a cost of $k*br$, where k is the number of blocks mined in time $T-t_{pub}$. After time T, a miner (For example, M_3 in the Figure 10b) engages in SDRBA, where it exchanges $v^{col}+\epsilon$ with Bob and redeem $v^{dep}+v^{col}$. So Bob end up earning $v^{col}+\epsilon-[(k+1)*br+f_B^{C_{Bob}}]$, where $\epsilon > [(k+1)*br+f_B^{C_{Bob}}]$. The example depicted in the Figure 10 has $k=4$, so it requires $\epsilon > 5*br$. Alice can also collude with a miner to perform HyDRA by deploying the contract, say C_{Alice}, and earn $v^{dep}+\epsilon-[(k+1)*br+f_A^{C_{Alice}}]$ in a similar manner.

APPENDIX D
SOLO MINING VS POOL MINING

This section explains why miners prefer pool mining over solo mining. The key factor is reward variance—pool mining offers lower variance, leading to more stable and predictable income. In probabilistic systems like cryptocurrency mining, this stability is often valued over higher but inconsistent solo rewards. We begin by outlining the underlying assumptions.

Assumptions and Notations: Let E_{solo} and E_{pool} denote the expected rewards from solo and pool mining, respectively. Let H be the total network hash rate, h the miner's hash rate, N the number of miners in the pool, R the fixed block reward, and f the pool fee. Let λ and λ_i be the average mining rates of the network and miner i, respectively. Given these, we analyze a miner's rewards under solo and pool mining.

Rewards earned by the miner in solo mining and pool mining: The individual miner's probability of successfully mining a block X in a given time period follows:

$$X \sim \text{Poisson}(\lambda_i), \quad \text{where } \lambda_i = \frac{h}{H}\cdot\lambda.$$

So the expected reward for the miner in solo mining using the expected value of a Poisson distribution ($E[X]=\lambda_i$) is $E_{\text{solo}} = \lambda_i \cdot R = \frac{h}{H}\cdot\lambda\cdot R$.

Similarly, in pool mining, miners combine their hash rates to form a larger effective hash rate. The pool's combined hash rate is $N\cdot h$, and the probability of the pool mining k blocks follows:

$$\lambda_{\text{pool}} \sim \text{Poisson}(\lambda_{\text{pool}}), \quad \text{where } \lambda_{\text{pool}} = \frac{N\cdot h}{H}\cdot\lambda.$$

Each miner in the pool receives a share of the reward proportional to their contribution h. The expected reward for an individual miner in the pool, after accounting for the pool fee f, is:

$$E_{\text{pool}} = (1-f)\cdot\left(\frac{h}{N\cdot h}\cdot\lambda_{\text{pool}}\cdot R\right) = (1-f)\cdot\frac{h}{H}\cdot\lambda\cdot R.$$

Given this we are all set to find a relation between E_{pool} and E_{solo}. To compare E_{solo} and E_{pool}, we compute the ratio:

$$\frac{E_{\text{solo}}}{E_{\text{pool}}} = \frac{\frac{h}{H}\cdot\lambda\cdot R}{(1-f)\cdot\frac{h}{H}\cdot\lambda\cdot R} = \frac{E_{\text{solo}}}{E_{\text{pool}}} = \frac{1}{1-f}.$$

Since $f>0$, we have $\frac{1}{1-f}>1$, implying $E_{\text{solo}} > E_{\text{pool}}$. Thus, the expected reward for solo mining is higher in absolute terms, but this does not account for variance. Most miners are risk-averse, meaning they prefer smaller, steady rewards over infrequent, larger payouts. A risk-averse miner values stability and is willing to sacrifice a small portion of their expected reward (pool fee) for predictable income. This is because of the higher variance in the rewards in case of solo mining. So next we will consider the variance.

For the further discussion, let Var_{solo} and Var_{pool} be the variance in case of solo mining and pool mining, respectively. So its easy to conclude, $\text{Var}_{\text{pool}} = \frac{\text{Var}_{\text{solo}}}{N}$, where N is the number of miners in the pool.

As we have already discussed, a risk-averse miner values stability, represented by a utility function $U(W)$, where W is the miner's reward. A common choice is the negative exponential utility function, $U(W) = -e^{-\alpha W}$, where $\alpha > 0$ represents the miner's degree of risk aversion.

For small variances, the expected utility can be approximated using a Taylor expansion: $E[U(W)] \approx U(E[W]) + \frac{1}{2}U''(E[W])\cdot\text{Var}(W)$, where $U''(E[W]) = -\alpha^2 e^{-\alpha E[W]}$. Thus,

$$E[U(W)] \approx -e^{-\alpha E[W]}\left(1-\frac{\alpha^2\text{Var}(W)}{2}\right).$$

Given this, we will next calculate the utility for a solo mining and a pool mining

$$E[U(W_{\text{solo}})] \approx -e^{-\alpha E_{\text{solo}}}\left(1-\frac{\alpha^2 \text{Var}_{\text{solo}}}{2}\right).$$

$$E[U(W_{\text{pool}})] \approx -e^{-\alpha E_{\text{pool}}}\left(1-\frac{\alpha^2 \text{Var}_{\text{pool}}}{2}\right) = -e^{-\alpha(1-f)E_{\text{solo}}}\left(1-\frac{\alpha^2 \text{Var}_{\text{solo}}}{2N}\right)$$

Now we have utility functions for solo mining and pool mining, i.e. $E[U(W_{\text{solo}})]$ and $E[U(W_{\text{solo}})]$. Next we will compare these utility functions to reach to the conclusion. We compare the utilities by calculating the difference,

$\Delta U = E[U(W_{\text{pool}})] - E[U(W_{\text{solo}})]$

$$= -e^{-\alpha(1-f)E_{\text{solo}}}\left(1-\frac{\alpha^2 \text{Var}_{\text{solo}}}{2N}\right) - -e^{-\alpha E_{\text{solo}}}\left(1-\frac{\alpha^2 \text{Var}_{\text{solo}}}{2}\right)$$

In the above equation, expanding $e^{-\alpha(1-f)E_{\text{solo}}}$ around E_{solo} for small f, i.e, $e^{-\alpha(1-f)E_{\text{solo}}} \approx e^{-\alpha E_{\text{solo}}}(1+\alpha f E_{\text{solo}})$, will gives the approximate value of ΔU after simplification.

$$\Delta U \approx -e^{-\alpha E_{\text{solo}}}\left[\alpha f E_{\text{solo}} - \frac{\alpha^2 \text{Var}_{\text{solo}}}{2} + \frac{\alpha^2 \text{Var}_{\text{solo}}}{2N}\right].$$

We can interpret the above equation as follows:

- **First Term** ($\alpha f E_{\text{solo}}$): Represents the utility reduction due to the pool fee f. This term reduces the miner's expected utility in pool mining.
- **Second Term** ($-\frac{\alpha^2 \text{Var}_{\text{solo}}}{2}$): Represents the penalty for high variance in solo mining.
- **Third Term** ($\frac{\alpha^2 \text{Var}_{\text{solo}}}{2N}$): Represents the reduced penalty due to variance in pool mining. For large N, this term is small, making pool mining more stable.

Given this, we can conclude that, If f is small, the utility reduction due to the pool fee is negligible. Furthermore, For large N, the variance penalty in pool mining becomes significantly smaller than in solo mining ($\frac{1}{N}$-factor reduction). Lastly, Risk-averse miners prioritize stability, and the reduced variance in pool mining ensures that $\Delta U > 0$ for reasonable values of f, N, and Var_{solo}. Thus, over time, the stability from lower variance outweighs the small reduction in expected rewards due to the pool fee, making pool mining more lucrative for risk-averse miners.

APPENDIX E
M2MBA ANALYSIS LEMMA'S PROOFS

Lemma 1 In $\zeta^{mba}(k, red)$, where $k \leq T$, if $(T-t_{pub}) * br * \lambda_i/\lambda_{col} > f_A^{dep}$, then it is a dominant move for active miners to accept the bribe by censoring the transaction tx_A^{dep} instead of including it in the block.

Proof. By including tx_A^{dep} miner would earn f_A^{dep}. On the other hands, the expected bribe that a miner would earn by censoring the transaction for a block is $br * \lambda_i/\lambda_{col}$. $T - t_{pub}$ rounds produce $T - t_{pub}$ blocks. So the total expected bribe that a miner earn is $(T-t_{pub}) * br * \lambda_i/\lambda_{col}$. According to our assumption this value exceeds f_A^{dep}. So accepting the bribe is the dominant move. □

Lemma 2 In $\zeta^{mba}(k, red)$, where $k \leq T$, if $v^{col} * \lambda_i/\lambda col + (T-t_{pub}) * br * (2 * \lambda_i/\lambda_{col} - 1) > f_A^{dep}$, then it is a dominant move for active miners to offer a bribe to other miners to censor the transaction tx_A^{dep} and perform the attack rather than including it in the block.

Proof. The attack consists of two phases. In the first phase, a bribe is offered to censor the transaction tx_A^{dep}. In the second phase, the miner attempts to mine the block that includes the transaction tx_M^{col} in round $T+1$, where Bob releases tx_B^{dep} by revealing pre_B. The miner, referred to as the confiscating miner M_i, will earn a expected reward of $v^{col} * \lambda_i/\lambda col$ by successfully mining the block that includes tx_M^{col}.

The confiscating miner M_i offers a bribe to other miners, including himself, to censor the transaction tx_A^{dep} for round $T - t_{pub}$. The bribe that M_i offers to himself is $(T-t_{pub}) * br * \lambda_i/\lambda col$. On the other hands, the bribe that M_i offers to other active miners is $(T-t_{pub}) * br * (1-\lambda_i/\lambda col)$.

So the net gain of the miner by offering a bribe and perform the attack is $v^{col} * \lambda_i/\lambda col + (T-t_{pub}) * br * \lambda_i/\lambda col - (T-t_{pub}) * br * (1-\lambda_i/\lambda col)$, which boils down to $v^{col} * \lambda_i/\lambda col + (T-t_{pub}) * br * (2 * \lambda_i/\lambda_{col} - 1)$. According to our assumption this value exceeds f_A^{dep}, which is the incentive that miner earns by including tx_A^{dep} in the block. So offering a bribe and performing the attack is the dominant move.

□

Lemma 3 In $\zeta^{mba}(k, red)$, where $k \leq T$. If $v_{col} * \lambda_i > f_A^{dep}$, it is the dominant move for passive miners to censor tx_A^{dep} rather than include it in the block.

Proof. As passive miner will silently wait for the Bob to release tx_B^{dep} that reveals pre_B. Passive miners are not involved in the either accepting bribe or offering a bribe. So the net gain of the passive miner, M_i, is $\lambda_i * v^{col}$. According to our assumption this value exceeds f_A^{dep}, which is the incentive that miner earns by including tx_A^{dep} in the block. So censoring tx_A^{dep} and performing the attack is the dominant move for a passive miner. □

Lemma 4 In $\zeta^{mba}(k, red)$, where $k > T$. Let z be the net earning of the miner by confiscating v^{col}. If $z > f_A^{dep}$, and Bob release transaction tx_B^{dep} and hence release secret pre_B at $T+1$, it is the dominant move for all miners to mine the block that includes tx_M^{col}, at $k = T+1$. (transient transition from $\zeta^{mba}(k, red)$ to $\zeta^{mba}(k, nred\text{-}rev)$ by including tx_B^{dep} and then including tx_M^{col} in the same block)

Proof. If miner M_i decide to defer the inclusion of tx_M^{col} in the block till round t_y in future. M_i must ensure either of the following two things:

- M_i must mine all the block from round $T+1$ to t_y.
- If M_i couldn't mine all the block between round $T+1$ to t_y, they should insure that other miner who mined the block shouldn't include tx_M^{col}, which they can ensure by bribing them more than z.

In the former scenario, the probability of block mined by the miner M_i decreases exponentially with increase of difference

between t_y and $T+1$ and hence the expected gain would also decreases exponentially, i.e. $z * x^{[t_y-(T+1)]}$, where x is the miner's mining power, which is λ_i for passive miner and λ_i/λ_{col} for active miners.

On the other hands, in the later scenario, M_i has to bribe more than z to the other miner for not including tx_M^{col}. The expected bribe that M_i needs to pay is $z*(1-x^{[t_y-(T+1)]})$, which will also increase with increase of difference between t_y and $T+1$. Note that the second condition, where miners offers a bribe greater than z, is specific to the active miners.

Combining the above two conditions, we can claim that it is dominant move for all the miner to mine the block that includes tx_M^{col} at $T+1$, if tx_B^{dep} is released, i.e., pre_B is revealed.

□

Lemma 5 We made the valid assumption that for an active miner, inequality $v^{col}\lambda_i/\lambda_{col} + (T-t_{pub}) * br * (2\lambda_i/\lambda_{col} - 1) > f_A^{dep}$ holds. Similarly, for a passive miner, Assumption $v^{col} * \lambda_i > f_A^{dep}$ is also valid.

Proof. We will prove this using a combination of theoretical analysis and empirical evidence. We will divide the proof into two parts: the first will focus on the active miners, and the second on the passive ones.

- $Case_1 - Active\ miners$: As we discussed earlier the active miners receives a bribe from the bribing miner, i.e. the miner who confiscate v^{col}, which offers a bribe to other active miners. We know that the value $v^{col}\lambda_i/\lambda_{col} + (T-t_{pub}) * br * (2\lambda_i/\lambda_{col} - 1)$ indicates the amount the bribing miner gets after paying bribe to other active miners, including himself. So to prove this value is greater than f_A^{col}, its sufficient to prove that the bribe that every active miner gets from the bribing miner is greater than f_A^{col} and the value left with bribing miner after paying bribe to other active miners, including himself, is greater than 0. This we are going to prove next.
  If the game is in $\zeta^{mba}(k, red)$ state, the expected number of blocks that the active miner, say M_i, with mining power λ_i, mines among $T-t_{pub}$ blocks is $(T-t_{pub}) * \lambda_i/\lambda_{col}$.
  This scenario can be viewed as a series of $T-t_{pub}$) Bernoulli trials with success probability λ_i/λ_{col}. The number of successes is therefore Binomially distributed, and the expected number of blocks miner M_i creates is $(T-t_{pub}) * \lambda_i/\lambda_{col}$.
  According to the contract in Algorithm 2 for each block miner would receive the bribe of $f_A^{dep} * \frac{\lambda_{col}}{\lambda_i} * \frac{1}{T-t_{pub}} + \epsilon$ for censoring tx_A^{dep}. So the net bribe that an active miner with mining power λ_i earn is $(\frac{\lambda_i}{\lambda_{col}} * (T-t_{pub})) * (f_A^{dep} * \frac{\lambda_{col}}{\lambda_i} * \frac{1}{T-t_{pub}} + \epsilon)$. This equation boils down to $f_A^{dep} + (\frac{\lambda_i}{\lambda_{col}} * (T-t_{pub})) * \epsilon$. So for any $\epsilon > 0$ miner would earn more than f_A^{dep}.
  Furthermore, the contract in the algorithm2 (line 26) ensures that the amount left with the bribing miner after paying bribe to all the active miners, *including* himself is positive. Hence the resulting gain for the miner would always be greater than f_A^{dep}.
- $Case_2 - Passive\ miners$: For passive miners, we'll support our claim with empirical evidence. The Ethereum fee is less than 0.01% of the transaction value [29]. Additionally, the miner with the lowest mining power among the top 50 miners from the production Ethereum network—who collectively contribute 99.98% of the total mining power—has a mining power of 0.015 [11]. Even the miner with the lowest mining power would have expected earning more than the transaction fee by censoring tx_A^{dep}, so our claim that $v^{col} * \lambda_i > f_A^{dep}$ holds, even for the miner with the smallest share of mining power.

□

APPENDIX F
DEMBA ANALYSIS LEMMA'S PROOFS

Lemma 6 For $v^{ded} > 0$, it is dominant move for the Alice to follow DEMBA than deviating from it.

Proof. The possible moves for Alice to redeem C_A^{col} are as follows: 1) Release pre_A before round T, where she would earn v^{dep} and v_A^{col} (potential transition to $nred\text{-}AB$ or $nred\text{-}ABT$, based on when Bob reveal pre_B), 2) Release pre'_A after round T, where she would earn v_A^{col}, but ends up paying more transaction fee (from eq. 1) and 3) Release both pre'_A and pre_A after round T where she would earn $v_A^{col} - v^{ded}$ (potential transition to $nred\text{-}AA'B$ or $nred\text{-}AA'BT$.

The second scenario occurs when Alice is genuinely offline and misses the deadline T. In this case, DEMBA still imposes a penalty by requiring her to pay a higher fee for the transaction that reveals pre'_A after round T. The third scenario involves potential malicious behavior by Alice, where she pretends to have already released pre_A despite not having done so, intending to deliberately burn v^{dep}. However, it becomes evident that this approach results in Alice earning an v^{ded} less than if she had acted honestly.

Based on the above discussion, we conclude that in the states $nred\text{-}AB$ or $nred\text{-}ABT$, Alice's earnings are higher compared to other states in the game.

□

Lemma 7 For $v^{ded} > 0$, it is dominant move for the Bob to follow DEMBA than deviating from it.

Proof. The possible moves for Bob to redeem C_B^{col} is to release pre_B. However, Bob can do so anytime. So possible malicious behavior from Bob is to deliberately delay this release. We know that delaying it beyond round T would make him earn $v_B^{col} - v^{ded}$ which is less than v_B^{col}, i.e. the value he would have earned by releasing it before T.

Based on the above discussion, we conclude that in the states $nred\text{-}AB$, $nred\text{-}A'B$ and $nred\text{-}AA'B$, Bob's earnings are higher compared to other states in the game. □

Lemma 8 For $\alpha < 1$, it is dominant move for the miners to follow DEMBA than deviating from it.

Proof. A potential malicious behavior by the miner is to exclude Alice's and Bob's transactions, which reveal pre_A and pre_B, respectively, from the block—either indefinitely or until round T. However, since DEMBA transactions offer higher fees than unrelated transactions (by assumption), the case of indefinite exclusion is unlikely. Moreover, from Eq. 2, it is evident that the miner earns less by including DEMBA transactions after round T.

We can quantify this further as, for $t > T$, the miner earns $\alpha^t \cdot f^{col}_{pre_A}$ by including $tx^{col}_{pre_A}$ and $\alpha^t \cdot f^{col}_{pre_B}$ for $tx^{col}_{pre_B}$. Thus, timely inclusion of transactions ensures greater profitability for the miner.

Based on the above discussion, we conclude that in the states $nred\text{-}AB$, $nred\text{-}A'B$ and $nred\text{-}AA'B$, miner's earnings are higher compared to other states in the game. □

APPENDIX G
M2MBA SMART CONTRACT

This section describes a smart contract designed for executing M2MBA, where the initialization method in Algorithm 1 is replaced with *lockCollateral*. Each participating miner locks their collateral, which is refunded either upon the successful completion of the attack or when *refundToMiners* is invoked after the timeout T (as specified in line 37 of Algorithm 2).

Similar to Algorithm 1, the *requestBribe* method allows miners to reserve their future bribe by committing to censor tx^{dep}_A before the attack succeeds. Upon successful completion of the attack, the *claimBribe* method enables the actual bribe transfer from Bob's deposit to the participating miners.

Algorithm 2 M2MBA Contract (C_{M2M})

1: **Variables:**
2: He-Col: Address of He-col contracts in He-HTLC
3: $v^{col} \leftarrow 0$, $Bal_A \leftarrow \{\}$, $count \leftarrow 0$.
4: $LockCol \leftarrow \{\}$ //maps from balance to address
5:
6: **Constants:**
7: $br = f^{dep}_A * \frac{\lambda_{col}}{\lambda_i} * \frac{1}{T - t_{pub}} + \epsilon$
8:
9: **Functions:**
10: //Called while contract deployment and amount (v^{col}) locking
11: **procedure** LOCKCOLLATERAL(val)
12: $LockCol[Caler] \leftarrow val$
13: **end procedure**
14:
15: //Called by the miners to lock their bribe
16: **procedure** REQUESTBRIBE( )
17: **if** Caller is not the miner of the current block **or** Function is called more than once in the block **then**
18: **return**
19: **end if**
20: $Bal_A[caller] \leftarrow Bal_A[caller]+1$, $count \leftarrow count+1$
21: **end procedure**
22:
23: //Anyone can call to claim their bribe
24: **procedure** CLAIMBRIBE($preImage_A$)
25: $Addr_M$ address of miner who redeem He-Col
26: **if** $H(preImage_A)$ is not a $H(pre_A)$ **or** tx^{dep}_A is included in the block $\leq T$ **or** He-Col is not redeemed till the current block through Col-M **or** $v^{col} - count * br + Bal_A[Addr_M] * br < 0$ **then**
27: **return**
28: **end if**
29: //Bribe for censoring tx^{dep}_A
30: **for** Each $(addr, bal) \in Bal_A$ **do**
31: Send $br * bal$ token of bribe to $addr$
32: $LockCol[Addr_M] -= br * bal$
33: **end for**
34: //Incentive to the caller for calling ClaimBribe
35: send $br > f$ to caller , $LockCol[Addr_M] -= br$
36: //Send remaining and locked collateral to miners
37: **for** Each $(addr, bal) \in LockCol$ **do**
38: Send bal token to $addr$
39: **end for**
40: $LockCol \leftarrow \{\}$, $Bal \leftarrow \{\}$, $v^{col} \leftarrow 0$ //Reset values
41: **end procedure**
42:
43: //Refund locked collaterals to the miners
44: **procedure** REFUNDTOMINERS(val)
45: **if** $tx^{dep}_A \in$ block with block height $\leq T$ **then**
46: **for** Each $(addr, bal) \in LockCol$ **do**
47: Send bal token to $addr$
48: **end for**
49: **end if**
50: **end procedure**
51: