Rapidash: Foundations of Side-Contract-Resilient Fair Exchange

Hao Chung Elisaweta Masserova Elaine Shi
Sri AravindaKrishnan Thyagarajan

Carnegie Mellon University

Abstract

Fair exchange is a fundamental primitive enabled by blockchains, and is widely adopted in applications such as atomic swaps, payment channels, and DeFi. Most existing designs of blockchain-based fair exchange protocols consider only the participating users as strategic players, and assume the miners are honest and passive. However, recent works revealed that the fairness of commonly deployed fair exchange protocols can be broken entirely in the presence of user-miner collusion. In particular, a user can bribe the miners to help it cheat — a phenomenon also referred to as Miner Extractable Value (MEV).

In this work, we provide the first formal treatment of *side-contract-resilient* fair exchange where users and miners may enter into arbitrary contracts on the side. We propose a new fair exchange protocol called Rapidash, and prove that the protocol is incentive compatible in the presence of user-miner collusion. In particular, we show that Rapidash satisfies a coalition-resistant Nash equilibrium absent external incentives. Further, even when there exist arbitrary but bounded external incentives, Rapidash still protects honest players and ensures that they cannot be harmed. Last but not least, our game-theoretic formulations also lay the theoretical groundwork for studying side-contract-resilient fair exchange protocols. Finally, to showcase the instantiability of Rapidash with a wide range of blockchain systems, we present instantiations of Rapidash that are compatible with Bitcoin and Ethereum while incurring only a minimal overhead in terms of costs for the users.

Contents

1 Introduction

Consider the following scenario between mutually distrusting Alice and Bob: Alice possesses something that Bob wants, and Bob possesses something that Alice wants. A fair exchange protocol enables an exchange between Alice and Bob such that either both of them get the desired item, or neither of them does. Fair exchange is a problem that has been studied for a long time [Mic03, Aso98, ASW97]. In particular, it has been shown that fair exchange is impossible to achieve without further assumptions [PG99, Mic03]. One way to circumvent this limitation is to rely on a trusted third party such as a blockchain [BBSU12, Her18, MMS+, vdM19, MD19, Max, CGGN, BDM16, Fuc, BK, MES16, MMA, Bis, ZHL+19, JMM14, TYME21, PD]. Indeed, fair exchange is a fundamental primitive in blockchain applications [BBSU12, Her18, MMS+, vdM19, MD19, Max, CGGN, BDM16, Fuc, BK, MES16, MMA, Bis, ZHL+19, JMM14, TYME21, PD], and has been widely adopted in the form of atomic swaps [Her18, MMS+, vdM19, MD19], contingent payment [Max, CGGN, BDM16, Fuc, BK], payment channels [PD, DW15, GM, MMSH16, MBB+19, DFH18, DEFM19], or vaults [MES16, MMA, Bis, ZHL+19].

Most existing blockchain-based fair exchange protocols *consider only Alice and Bob as potentially strategic players, and the miners are assumed to be honest* [EFS20, DEF18, AHS22, CGJ+17, GKM+22, BK14]. Recently, however, the community has become increasingly concerned that potential user-miner collusion can completely break the fairness guarantees promised by fair exchange protocols [TYME21, WHF19, Bon16, MMS+, MHM18, JSZ+21, Ham] — a phenomenon commonly referred to as Miner Extractable Value (MEV). As a concrete example, a Hash Timelock Contract (HTLC) is one of the commonly employed mechanisms for realizing fair exchange in blockchain environments. Imagine that Alice has a secret s and she wants to sell it to Bob at a price $\$v$. A standard HTLC contract is parametrized with the hash of the secret $h = H(s)$, a timeout value T, and the price $\$v$. In a preparation phase, Bob deposits $\$v$ coins into the contract. The contract now allows Alice to redeem the $\$v$ coins by posting the secret s whose hash should be equal to h. However, if Alice fails to redeem $\$v$ by time T, Bob can get his deposit $\$v$ back. Since the HTLC contract can protect Bob from an offline Alice, we also say that it is *dropout resilient*.

Unfortunately, a number of recent works have pointed out that the standard HTLC is vulnerable to user-miner collusion. In particular, Bob may collude with some miners in an attempt to starve Alice's redeeming transaction. If Bob's coalition can suppress the transaction till the timeout T, then they can get the $\$v$ deposit back after learning the secret s! Various works have shown that such user-miner collusion is indeed possible in practice through bribery mechanisms [TYME21, WHF19, HZ20, MHM18, JSZ+21, Ham]. Such attacks can be instantiated in various ways [TYME21, WHF19, HZ20, MHM18, JSZ+21, Ham], e.g., by exploiting the decentralized smart contracts available in blockchain environments. Moreover, with some clever tricks, they can be instantiated in a fairly inexpensive manner [TYME21].

Therefore, the status quo raises a natural and important question:

Can we have a blockchain-based fair exchange protocol that resists user-miner collusion?

If a fair exchange protocol is incentive compatible even in the presence of user-miner coalitions, we also say that it is *side-contract-resilient*.

Tsabary et al. [TYME21] made a pioneering attempt at answering the above question. They proposed a new fair exchange mechanism called a *Mutual-Assured Destruction Hash Timelock Contract (MAD-HTLC)*. Unfortunately, their scheme secures only against a couple specific forms of bribery attacks. At the same time, it opens up some new attacks (see Section 2.2 for a more detailed discussion). Indeed, the authors of the MAD-HTLC paper acknowledge themselves that MAD-HTLC does not provide any provable guarantee in the presence of user-miner collusion [TYME21].

1.1 Our Results and Contributions

To the best of our knowledge, we are the *first to give a formal treatment of side-contract-resilient fair exchange*. Specifically, we make the following contributions:

Rapidash: a side-contract resilient fair exchange protocol. We propose a new, side-contract resilient *cross-chain atomic swap* protocol called Rapidash[1], which works atop any standard Proof-of-Work blockchain or a Proof-of-Stake blockchain where the next block proposer is selected on the fly with the probability proportional to the miner's stake. Rapidash allows two players to exchange one cryptocurrency token for another, while offering the following game-theoretic guarantees (informally stated[2]):

- **Cooperative strategy proofness (i.e., CSP fairness).** We prove that absent any external incentives, any coalition of players (that does not simultaneously contain Alice and Bob[3]) is incentivized to play honestly, as long as the rest of the world is playing honestly and the coalition does not control 100% of the mining power. In other words, honest behavior is a *coalition-resistant Nash equilibrium*.

- **Bounded maximin fairness.** We prove that our protocol protects honest players even when other players may have *arbitrary but bounded* external incentives that may encourage them to deviate from the honest protocol, a notion which we call "bounded maximin fairness". In particular, we show that as long as 1) the externally incentivized coalition plays rationally, 2) their external incentives are bounded, and 3) they do not control 100% mining power, then honest players will not get negative utility. In other words, simple-minded, non-strategic players can always feel safe participating as long as they believe that the present protocol has only bounded externalities.

- **Drop-out resilience.** An honest player is protected from loss even if the counterparty drops offline in the middle of the protocol (e.g., due to loss of password or network outage).

We argue why Rapidash disincentivizes coalitions of 100% mining power by analyzing the coalition formation process as a *meta-game*. Specifically, a 100% coalition is not an equilibrium in this meta-game which also justifies our modeling approach where we consider coalitions that does not wield 100% mining power.

Besides offering provable game-theoretic guarantees, our Rapidash atomic swap protocol enjoys good efficiency properties. We achieve a property called *optimistic responsiveness*, i.e., when both parties are honest and online, the protocol completes as soon as $c = O(1)$ number of blocks are mined on each chain. Specifically, we only need $c = 1$ to achieve CSP-fairness, and we need $c = 3$ if we additionally require bounded maximin fairness.

Definitional and conceptual contributions. Our work lays the formal groundwork for studying side-contract-resilient fair exchange protocols. In general, mechanism design in the blockchain world is complicated by the existence of decentralized smart contracts, which can be used to openly solicit coalitions, as well as implement potentially *arbitrary* side contracts among players.

Our modeling approach follows a line of recent works [PS17a, CGL+18, WAS22, CS21] and can capture *arbitrary* side contracts among coalitions of players. We assume that the goal of a rational coalition is to maximize its joint utility, i.e., the sum of the utilities of all coalition members.

[1] Rapiddash is a fire-type Pokémon who can control its flames such that its rider is not burnt. Our Rapidash contract incentivizes honest behavior and protects the players' collateral from being burnt.

[2] For Proof-of-Stake, the percentage of mining power is measured in terms of stake.

[3] If Alice and Bob were in the same coalition, they would not need the fair exchange.

Equivalently, we assume that there is some *binding* side contract that allows the coalition to split their joint gains among themselves, and moreover the enforcement of this side contract is ensured. Besides capturing arbitrary binding side contracts between different players, we also capture the coalitions that are naturally formed when the same player controls multiple pseudonyms such as public keys. For example, Alice or Bob may well be the pseudonym of some miner (*c.f.* the assumption made in earlier works [TYME21], that Bob must not be a miner, is clearly unenforceable in the real world).

We adopt two main incentive-compatibility notions.

1. The first one, CSP-fairness (or equivalently, coalition-resistant Nash Equilibrium), was proposed and adopted in a line of recent works [PS17a, CGL+18, WAS22, CS21]. This notion implicitly assumes that that the players do not have external incentives outside the current protocol instance, and thus a strategic individual or coalition is incentivized to act in a way that maximizes its profit.

2. Our second notion, *bounded maximin fairness*, is a new notion that may be of independent interest in a broader context, especially in cases where external incentives may lead players to behave maliciously from the perspective of the current protocol. However, security against arbitrarily malicious behavior is difficult to attain. In this respect, the strongest possible game-theoretic notion one can hope for is that even when the strategic coalition can be arbitrarily malicious, honest players should not be harmed. Indeed, this notion was termed as *maximin fairness* in a line of recent works [PS17a, CGL+18, WAS22]. In maximin fairness, we implicitly assume that the external incentives may be arbitrary and unbounded, and this is why the strategic players' behavior may appear arbitrarily malicious from the perspective of the present protocol. Maximin fairness, however, is a very stringent requirement, and insisting on such a strong notion of fairness may severely constrain the design space or even lead to impossibility results in some applications.

   In fair exchange, we currently do not know how to achieve maximin fairness. Instead, we suggest a meaningful relaxation called *bounded maximin fairness*. Unlike the stringent notion, in bounded maximin fairness, we assume that the external incentives may still be arbitrary, but the total amount is bounded. We want to guarantee that honest participation will not lead to negative utility as long as the other strategic players behave rationally in light of the arbitrary but bounded external incentives.

Finally, our definitional approach models players and coalitions as interactive Turing Machines which can send and receive a special type of variable called money. This allows us to capture a most general strategy space, i.e., deviating players can not only send arbitrary messages, but also post new smart contracts on the fly during protocol execution.

Instantiation atop Bitcoin and Ethereum. For ease of understanding, we first describe Rapidash assuming the existence of generic smart contracts. We then go on to give instantiations of the atomic swap protocols (for both CSP-fairness and bounded maximin fairness) that are compatible even with the limited scripting language of Bitcoin. To achieve this, we only make use of some of the most commonly used scripts in Bitcoin and exploit the transaction model of Bitcoin, where coins in addresses can be spent exactly once, and excess coins in a transfer are treated as a transaction fee for the miners. We also instantiate our atomic swap protocols in Solidity [Eth22], Ethereum's smart contract language, and deploy them on the Goerli testnet [goe22]. Additionally, we implement and evaluate a CSP-fair Rapidash contract which allows a user to sell a secret, and provide a comparison of this contract to HTLC, MAD-HTLC, and He-HTLC [WSZN], which aim to achieve a similar functionality.

1.2 Additional Related Work

We now review additional related work besides those already mentioned. Our work is related to financially fair protocols [BK14, MB17, BZ17, CGJ+17, KB16, KMS+16], where it is common to use collateral and penalty mechanisms to incentivize honest behavior. For some specific applications such as lottery and leader election, a few works showed that collateral is in fact not necessary to achieve game-theoretic fairness [MB17, BZ17, CCWS21]. To the best of our knowledge, almost all prior works consider only the users as potentially strategic players, and the miners are assumed to be honest.

Miner Extractable Value (MEV) is among the often debated problems in the cryptocurrency community today. In MEV, the miner leverages its unique position, i.e., its ability to decide what to include in the block, to profit from blockchain applications. The original motivation of our work is possible user-miner collusion, which is a form of MEV. Interestingly, inspired by MAD-HTLC [TYME21], we leverage MEV itself to defend against MEV — specifically, to thwart such user-miner collusion, Rapidash allows honest miners to extract the value should any cheating behavior take place.

Concurrent work. The concurrent work Helium [WSZN] has results that are very closely related to ours. Both works were initially completed in May 2022, and since then both works have undergone several revisions. The main differences between the two works are the following:

- Helium [WSZN] considers only the knowledge-coin exchange primitive which is the warmup of our paper. As we show, the main technical challenges arise in the construction of the atomic swap. In particular, we show that contrary to the common belief, directly composing two knowledge-coin instances does not yield a secure atomic swap protocol. This exposes intriguing technicalities regarding the composition of game-theoretically secure primitives.

- Helium [WSZN] considers only a fairness notion similar to our CSP fairness, but they provide no guarantee against external incentives. As stated above, one of our main contributions is the new formulation of bounded maximin fairness along with an efficient protocol that can be proven in the model.

2 Technical Roadmap

2.1 Problem Statement and Assumptions

Our final goal is to realize *cross-chain atomic swap*, where two parties wish to exchange one cryptocurrency with another. However, as a starting point, we first consider a simpler primitive henceforth called *knowledge-coin exchange*. In a knowledge-coin exchange protocol, Bob wants to buy a secret s of Alice at a price of $\$v$ coins. Notice that knowledge-coin exchange itself is typically not the end goal in real-world scenarios. However, this primitive has been proven to be useful in many applications. In particular, existing cross-chain atomic swap constructions [Her18, MMS+, vdM19, MD19] work by composing two instances of knowledge-coin exchange (although known constructions do not provide side-contract resilience).

Henceforth we assume that Alice loses value $\$v_a$ for revealing s, and Bob gains value $\$v_b$ if he learns s. If $\$v_a < \$v < \$v_b$, both parties benefit from Alice selling s to Bob at price $\$v$.

Throughout this paper, we will assume that Alice or Bob may collude with a subset of the miners, and the coalition may adopt arbitrary probabilistic polynomial-time strategies to maximize its joint utility. The only restriction we impose on the strategy space is that the coalition does not

perform a consensus-level or network-level attack. For example, we do not consider 51% attacks that aim to make profit through double-spending. There is an orthogonal line of work that focuses on consensus security [GKL15, PSS17, PS17b].

Therefore, we assume an idealized mining process. In every time step, an ideal functionality picks the next winning miner at random with probability proportional to its mining power (or stake). The winning miner may choose a set of transactions to include in the next block. We assume that the network delay is 0, i.e., any message posted by Alice, Bob or any new block mined will be immediately seen by other players.

In Section 3.1, we will formalize the notion of transactions and a smart contract execution model.

2.2 Strawman and Prior Approaches

Throughout the paper, we will write smart contracts using pseudocode. Since all existing and new smart contracts in this paper use simple logic, it does not necessitate a general smart contract language like Ethereum to instantiate them — even Bitcoin's restricted scripting language works, as we show in Section 8.

Naïve protocol. The most straightforward idea is for Alice and Bob to agree on a smart contract which knows $h_s = H(s)$ where $H(\cdot)$ denotes a cryptographic hash function. Moreover, Bob deposits $\$v$ into the contract upfront. The smart contract's logic is very simple:

> On receiving s from Alice such that $H(s) = h_s$, send $\$v$ to Alice.

The honest protocol is also simple: Alice sends s to the smart contract as soon as Bob has deposited $\$v$ into the contract, and Bob does nothing. Honest miners always include all outstanding transactions in any block they mine.

Somewhat surprisingly, even this naïve protocol satisfies CSP-fairness, i.e., any coalition consisting of either Alice or Bob as well as an arbitrary subset of the miners have no incentive to deviate from the honest protocol; and following the honest strategy maximizes the coalition's utility.

The naïve protocol, however, has two drawbacks. 1) Alice can harm Bob without incurring any cost to herself. Simply by withholding s, Bob will lose its deposit $\$v$; and 2) even if Alice is well-meaning, she may accidentally drop offline (e.g., due to an unforeseen circumstance such as losing her password) in which case Bob also loses $\$v$. Note that the first drawback can be overcome by requiring Alice to make a deposit of $\$v'$ too besides Bob's deposit of $\$v$, such that Alice can claim $\$v + \$v'$ if she sends s to the smart contract. However, this does not fix the second problem, that is, the protocol is *not dropout resilient*.

Hash timelock Contract. Recognizing the drawback of the naïve approach, a line of work called hash timelock contracts (HTLCs) [atob, atoa, Her18, CGGN, Max] focus on achieving *dropout resilience*. HTLCs are prevalently deployed today, and indeed they work well absent miner-user collusion. Unfortunately, they do not provide resilience against miner-user collusion.

A standard HTLC works as follows, where Bob is still required to deposit $\$v$ into the contract upfront, and the contract is parametrized with a timeout T_1:

HTLC

- On receiving s from Alice such that $H(s) = h_s$, send $\$v$ to Alice.
- After T_1, on receiving `ok` from Bob: send $\$v$ to Bob.

In the above, the two activation points are mutually exclusive, i.e., only one of them can be activated and only once.

This contract allows Bob to recover its deposit if Alice drops offline. However, the HTLC-based protocol is *not CSP fair* in the presence of user-miner coalitions. Suppose there is no transaction fee, then a coalition of Bob and some miners should never include Alice's transaction: if Alice's transaction is starved till after T_1, then Bob's coalition can get both the secret s and the $\$v$ deposit back!

If Alice offers a transaction fee of $\$f$, then as long as Bob bribes each miner $\$f + \$\epsilon$ (for some small $\$\epsilon > 0$) for excluding Alice's transaction, rational miners will take the bribe [TYME21]. This bribery attack makes sense for Bob as long as $\$v > \$f \cdot T_1$. It may seem like HTLC is secure as long as $T_1 \cdot \$f$ is sufficiently large. However, Tsabary et al. [TYME21] showed new attacks where the cost to Bob is not dependent on T_1. We provide more details on Tsabary et al.'s attack [TYME21] in Section 7. In particular, using our terminology, such attacks are viewed as strategies in the *coalition forming meta-game*. The HTLC contract is undesirable because there exist meta-games where 100% of the miners taking the "bribe" is an equilibrium, thus encouraging 100% coalitions — see Section 7 for details.

MAD-HTLC. Tsabary et al. [TYME21] suggest a new contract called MAD-HTLC described below. Bob deposits $\$v$ upfront, and the contract works as follows[4]:

MAD-HTLC

- **On receive** pre_a from Alice such that $H(pre_a) = h_a$: send $\$v$ to Alice.
- **After** T, **on receive** pre_b from Bob such that $H(pre_b) = h_b$: send $\$v$ to Bob.
- **On receive** (pre_a, pre_b) from anyone P such that $H(pre_a) = h_a$ and $H(pre_b) = h_b$: send $\$v$ to P.

All activation points in the above contract are again mutually exclusive, and here we use the notation pre_a to denote the secret Alice wants to sell.

In MAD-HTLC, if Alice has disclosed pre_a and yet Bob still attempts to get his deposit back by posting pre_b, then the miner easily preempts Bob's transaction and claim $\$v$ itself by posting the pair (pre_a, pre_b). MAD-HTLC indeed defends against the simple bribery attack mentioned above as well as the attack of Tsabary et al. [TYME21] — or in our language, MAD-HTLC removes the undesirable 100%-colluding-equilibrium in the coalition forming meta-game (see Section 7). Unfortunately, as the authors acknowledge themselves, MAD-HTLC is NOT incentive compatible in the presence of *binding side contracts* between miners and users.

A binding side contract allows the coalition to split off their joint utility in a binding manner. For example, in MAD-HTLC, Bob can collude with some miners, and as soon as Alice posts pre_a, if the colluding miners happen to mine the next block, they can exclude Alice's transaction and redeem the $\$v$ coins for themselves by posting both pre_a and pre_b. Then, using the binding side contract, the coalition can split off the $\$v$ coins among its members. It could also be that Bob is a miner himself. In this case, if Bob happens to mine the next block after Alice posts pre_a, Bob can get the secret for free.

[4]MAD-HTLC has some extra logic to defend against a spiteful Bob which we omit for simplicity, as this logic does not mitigate the coalition attacks we point out.

2.3 Warmup: CSP-Fair Knowledge-Coin Exchange

To understand our atomic swap protocol, we first describe a warmup, that is, how to construct a fair knowledge-coin exchange protocol. Our atomic swap protocol will compose two instances of this primitive — at this moment, some non-trivial compositional issues arise which we will discuss later.

The single instance RAPIDASH protocol works as follows. At the very beginning, Bob deposits not just the intended payment $\$v$, but also an additional $\$c_b$ amount of collateral into the contract shown below.

Rapidash contract

/**parametrized with* $h_a, h_b, T_1, T_2, \$v, \$c_b, \$\epsilon$, Bob deposits $\$v + \$c_b$*/

P1: On receive pre_a from Alice such that $H(pre_a) = h_a$, send $\$v$ to Alice and $\$c_b$ to Bob.

P2: Time T_1 or greater: on receive pre_b from Bob such that $H(pre_b) = h_b$, do nothing.

C1: At least T_2 after P2 is activated: on receiving _ from anyone, send $\$v + \$c_b$ to Bob.

C2: On receive (pre_a, pre_b) from anyone P such that $H(pre_a) = h_a$ and $H(pre_b) = h_b$, send $\$\epsilon$ to player P. All remaining coins are burnt.

Then, Alice posts pre_a as soon as Bob's deposit takes effect. If Alice fails to post pre_a by time T_1, Bob posts pre_b to P2 at time T_1 to request a refund. Note that the activation point P2 merely allows Bob to express his intent to request a refund. The actual refund happens when T_2 time has passed since the activation of P2 — at this point, Bob sends _ to C1 which actually sends him the refund. An honest miner always includes all outstanding transactions in any block it mines. Importantly, if an honest miner has observed both pre_a and pre_b contained in the transactions posted, it will immediately post (pre_a, pre_b) to C2, and this transaction will always be ordered in front of others in the block it mines.

Intuition. The key insight in our construction is that the activation point C2 serves as a "*bomb*". Suppose that the honest Alice has posted pre_a. Now, should a strategic Bob-miner coalition ever post pre_b to P2 in an attempt to get refunded and thus get the secret pre_a for free, then both pre_a and pre_b would be publicly known. Observe that at this point, the coalition has to wait at least T_2 amount of time before the actual refund C1 can be activated. During this T_2 window, if any non-colluding miner mines a block, it will trigger the bomb by posting (pre_a, pre_b) to C2, which causes Bob to lose its collateral. In other words, if a Bob-miner coalition wishes to get the secret for free, it essentially has to take a gamble that it will be able to mine all blocks in the T_2 window.

In Section 4, we formally prove that the above protocol indeed satisfies CSP-fairness as long as the parameters respect the following constraints:

- $\$c_b > \$\epsilon$, and $\$v > \$\epsilon$: this former makes sure that a sufficient amount is burnt should the bomb C2 be triggered, such that activating P2 + C2 does not make sense for Bob; the latter makes sure that Alice prefers to activate P1 rather than C2.
- $\$\gamma^{T_2} \leq \frac{\$c_b}{\$c_b + \$x}$ where γ is an upper bound on the fraction of mining power controlled by the coalition: if the honest Alice posts pre_a to P1, this condition makes sure that it is not worth it for the Bob-miner coalition to take a gamble and try to invoke both P2 and C1 to get all of Bob's deposit back. As mentioned, once the Bob-miner coalition has invoked P2, both pre_a and pre_b become publicly known. At this moment, the coalition must mine *all* within the next T_2 window to guarantee that C1 is invoked, since any non-colluding miner who mines a block during this T_2 window will trigger the bomb C2.

For example, suppose we choose $\text{Ξ}c_b = \text{Ξ}x$. Then, we need to make sure $\gamma^{T_2} \leq \frac{1}{2}$. This means if $\gamma = 90\%$, we can set $T_2 = 7$; if $\gamma = 49.9\%$, we can set $T_2 = 1$. Asymptotically, for any $\gamma = O(1)$, T_2 is a constant. Increasing $\text{Ξ}c_b$ helps to make T_2 smaller. For CSP fairness to hold, $\text{Ξ}\epsilon$ can be arbitrarily small. However, as we discuss later when analyzing the coalition-forming meta-game (see Section 7), we may want that $\text{Ξ}\epsilon$ is not too small, such that 100% coalition is NOT an equilibrium in the coalition-forming meta-game. In practice, we can set $\text{Ξ}\epsilon$ to be slightly smaller than $\text{Ξ}x$.

2.4 CSP-Fair Atomic Swap Protocol

Henceforth, we will use Bitcoin and Ethereum as examples of the two cryptocurrencies being exchanged — however, our protocol can be used to exchange other cryptocurrencies too. Suppose that Bob wants to exchange his $\text{Ξ}x$ amount of Ethers with Alice's $\text{Ƀ}x'$ amount of Bitcoins.

2.4.1 Strawman: Direct Composition

Existing cross-chain atomic swap protocols [Her18, MMS+, vdM19, MD19] work by composing two instances of knowledge-coin exchange, one on each blockchain. Therefore, the first natural idea is to see whether this approach will work in our context.

The idea is the following. Imagine that we run one instance of the knowledge-coin exchange protocol on Ethereum henceforth called Rapidash, and its four activation points are henceforth called P1, P2, C1, and C2. We run another instance on Bitcoin called Rapidash', and its four activation points are henceforth called P1', P2', C1', and C2'. The two instances are connected as follows: they both use the same pre_a, in Rapidash Alice is the one posting it on blockchain, and in Rapidash' Bob is the one who must publish it (after obtaining it from Alice). Initially, Alice deposits $\text{Ƀ}x' + \text{Ƀ}c'_a$ into Rapidash' where $\text{Ƀ}x'$ is the intended exchange amount, and $\text{Ƀ}c'_a$ is some extra collateral. Similarly, Bob deposits $\text{Ξ}x + \text{Ξ}c_b$ into Rapidash: $\text{Ξ}x$ is the intended exchange amount, and $\text{Ξ}c_b$ is some extra collateral.

Now, Alice makes the first move by posting pre_a to the activation point P1 of Rapidash. In this way, Alice get Bob's $\text{Ξ}x$ amount of Ethers, and moreover, Bob gets his collateral $\text{Ξ}c_b$ back. At this moment, Bob learns pre_a, and therefore, he can post the same pre_a to the P1' activation point of the Rapidash' instance. This allows Bob to get Alice's $\text{Ƀ}x'$, and Alice also gets her collateral $\text{Ƀ}c'_a$ back.

If Alice drops out, Bob can ask for a refund by posting pre_b to P2. Similarly, Alice can ask for a refund by posting pre'_a to P2' in case Bob drops out. As before, once Alice has posted pre_a to P1, a strategic Bob-miner coalition is disincentivized from posting pre_b to P2, and once Bob has posted pre_a to P1', a strategic Alice-miner coalition is disincentivized from posting pre'_a to P2'.

Flaw in the strawman approach. Unfortunately, this strawman approach suffers from a major flaw. Intriguingly, *even though the underlying knowledge-coin exchange protocol is CSP-fair, the composed atomic swap protocol is NOT CSP-fair*! More generally, the technicality described below suggests that interesting subtleties arise during the composition of game-theoretically fair primitives. Indeed, the composibility of such primitives is a major open question.

The problem is that a strategic Alice-miner coalition can fail to post pre_a, and get refunded from Rapidash' by invoking P2'+C1'. Of course, the honest Bob will also try to get refunded by posting pre_b to P2. However, with some noticeable probability, Alice may be able to get refunded faster than Bob. At this moment, the coalition can attempt to invoke P1 with pre_a, and if successful, the coalition can essentially get Bob's $\text{Ξ}x$ for free!

Upon closer examination, the reason why composition breaks fairness is because when Alice has

got her deposit back from Bitcoin, the value pre_a loses its value! Recall that our earlier knowledge-coin exchange protocol is only proven to be fair assuming that pre_a has *timeless* value!

2.4.2 Our Atomic Swap Protocol

Our first idea is to punish Alice if she fails to post pre_a to P1 when she is supposed to, and then tries to invoke P2' with pre'_a. This can be achieved by allowing the bomb C2' to be triggered with the pair (pre_b, pre'_a), since the honest Bob will post pre_b to P2 if Alice fails to post pre_a. However, adding this extra trigger alone breaks the dropout resilience for Alice. Specifically, if the deposit transactions took too long to confirm, Bob should be allowed to abort by posting pre_b to P2. However, now Alice is blocked from posting pre'_a to P2' to get her deposit back, for fear of triggering the bomb C2'.

To remedy this situation, we introduce a "two-phase preparation" stage. Initially, P1 and C2 are locked with pre_c known only to Bob. Bob publishes the secret pre_c if the deposits into both contracts take effect in a timely manner. Once the secret pre_c is released, Alice is supposed to post pre_a immediately. Now, if, for some reason, Bob ever wants to back out by posting pre_b to P2, he will either help Alice get her deposit back or not, depending on the situation:

- If the honest Bob decides to back out because the deposit transactions took too long to confirm, in this case, before Bob posts pre_b to P2, he will post _ to P2' to help Alice get her deposit back — this resolves the aforementioned dropout resilience issue where Alice will not be able to get her deposit back once Bob has posted pre_b. Note that it is only safe for Bob to help Alice get refunded before getting refunded himself, since he has not opened the lock by releasing pre_c yet, and thus no one else can cash out his coins in Rapidash.
- If Bob has already opened the lock with pre_c, then, should the honest Bob ever post pre_b to P2, it must be due to Alice's failure to post pre_a to P1. In this case, Bob will not help Alice get her deposit back by posting _ to P2'. Since Alice is not honest in this case, we do not care about protecting Alice.

CSP-fair atomic swap. Based on these ideas, we construct the following atomic swap protocol.

Rapidash contract (on Ethereum)

/* *Parameters:* $(h_a, h_b, h_c, T_1, \tau, \ddot{\text{E}}x, \ddot{\text{E}}c_b, \$\ddot{\text{E}}\epsilon)$, Bob deposits $\ddot{\text{E}}x + \ddot{\text{E}}c_b$. */

P1: On receive pre_a from Alice and pre_c from Bob such that $H(pre_a) = h_a$ and $H(pre_c) = h_c$, send $\ddot{\text{E}}x$ to Alice and $\ddot{\text{E}}c_b$ to Bob.

P2: Time T_1 or greater: On receive pre_b from Bob such that $H(pre_b) = h_b$ or on receiving _ from Alice, do nothing.

C1: At least τ after P2 is activated: on receiving _ from anyone, send $\ddot{\text{E}}x + \ddot{\text{E}}c_b$ to Bob.

C2: On receive (pre_a, pre_b, pre_c) from anyone P such that $H(pre_a) = h_a$, $H(pre_b) = h_b$, and $H(pre_c) = h_c$ send $\ddot{\text{E}}\epsilon$ to player P. All remaining coins are burnt.

Rapidash' contract (on Bitcoin)

/* *Parameters:* $(h'_b, h'_a, T'_1, \tau', \ddot{\text{B}}x', \ddot{\text{B}}c'_a, \$\ddot{\text{B}}\epsilon')$, *Alice deposits* $\ddot{\text{B}}x' + \ddot{\text{B}}c'_a$, *Bob deposits* $\ddot{\text{B}}c'_b$ */

P1': On receiving pre'_b from Bob[a] such that $H(pre'_b) = h'_b$ or on receiving _ from Alice, send $\ddot{\text{B}}x' + \ddot{\text{B}}c'_b$ to Bob and send $\ddot{\text{B}}c'_a$ to Alice.

P2': Time T_1' or greater: on receiving pre_a' from Alice such that $H(pre_a') = h_a'$ or on receiving _ from Bob, do nothing.

C1': At least τ' after P2' is activated: on receiving _ from anyone, send ₿$x' +$ ₿c_a' to Alice and ₿c_b' to Bob

C2': On receiving (pre_b', pre_a') or (pre_a', pre_b) from anyone P such that $H(pre_b') = h_b'$, $H(pre_a') = h_a'$ and $H(pre_b) = h_b$, send ₿ϵ' to player P. All remaining coins are burnt.

[a] We set $h_b' = h_a$, and Bob will let pre_b' be the pre_a he learns in the RAPIDASH instance.

In the above, T_1 and T_1' denote the earliest times at which the refund paths P2 and P2' can be invoked, expressed in Ethereum time and Bitcoin time, respectively. It is important that Bitcoin time T_1' happens later than Etheurem time T_1, since otherwise, Alice could get refunded from the Bitcoin contract RAPIDASH before Bob even posts pre_b to P2, and then trigger P1 to get Bob's Ξx for free. Observe that when Alice and Bob are both honest, Alice will post pre_a to P1 immediately and then Bob will learn pre_a and post it to P1' immediately. Therefore, both players get their desired cryptocurrency and all their collateral back as soon as new block is confirmed on both Ethereum and Bitcoin — in this sense, the protocol satisfies *optimistic responsiveness*.

As mentioned, it is important that in the honest protocol, Bob differentiates the reason should he ever want to post pre_b to P2. If he has not posted pre_c yet, he will first help Alice get refunded first before getting his own refund. Otherwise, he need not help Alice get refunded. Also, honest miners will include all outstanding transactions in any block they mine. Moreover, whenever honest miners observe sufficient information to trigger either the C2 bomb or the C2' bomb, they will indeed trigger the bomb, and the corresponding transaction will be ordered ahead of all other transactions in the block they mine. We leave a full description of the honest protocol and parameters to Section 5, as well as the proof of CSP fairness.

Regarding the asymmetry in collateral. Last but not the least, there is one remaining subtlety in our protocol. Notice that the protocol is asymmetric in the sense that Alice has to put down collateral only in RAPIDASH', but Bob has to put down collateral in both RAPIDASH and RAPIDASH'. The worry is that Bob can fail to deposit into RAPIDASH and then try to trigger C2' to get ₿ϵ' from Alice for free. To defend against this, we require that Bob first deposit his collateral $\$c_b'$ into RAPIDASH' before Alice sends her deposit. In this way, Bob will be incentivized against triggering C2' even when he has not deposited into RAPIDASH. Indeed, the full honest protocol in Section 5 makes the order of the deposit transactions clear.

It would be more desirable if Bob did not have to put down collateral into RAPIDASH' which is the cryptocurrency he is trying to buy. However, currently we do not know any approach that is CSP-fair and both parties need not put down collateral in the other cryptocurrency that they want. We therefore leave this as an open question. We stress, however, that assuming that a party has some collateral in the other cryptocurrency is nonetheless reasonable in many real-world applications. For example, an analogy is that to mine Ether, one must already have some Ether as stake. Furthermore, our approach would also be a great fit for atomic swap between a client and an exchange. In this case, it is natural to expect that the exchange already has some collateral in the cryptocurrency that it is trying to buy.

Parameter choices. The parameter choices of both RAPIDASH and RAPIDASH' are similar to those in the knowledge-coin exchange protocol, with the additional contraint that ₿$c_b' >$ ₿ϵ, which makes sure that Bob cannot benefit from failing to deposit into RAPIDASH and triggering C2'.

2.5 Achieving Bounded Maximin Fairness

The notion of CSP-fairness assumes that players do not have external incentives outside the present protocol. We additionally want to achieve resilience against external incentives. In particular, external incentives may lead strategic players to behave non-rationally from the perspective of the current protocol. When the external incentive may be arbitrary and unbounded, strategic players can behave arbitrarily maliciously, which agrees with the standard "malicious adversary" assumption commonly adopted in the cryptography literature. The strongest game theoretic notion one can hope for is that honest players do not get harmed even when there exist arbitrarily malicious individuals or coalitions. This notion is called *maximin fairness* in some recent works [PS17a, CGL+18, WAS22].

Fairness against arbitrarily malicious behavior, however, can be too stringent and challenging to satisfy. We are not aware of any approach that protects honest players against arbitrarily malicious behavior. However, we are indeed able to provide a relaxed notion of fairness called *bounded maximin fairness*, that protects honest individuals from potential non-rational behavior stimulated by possibly *arbitrary* external incentives — as long as the external incentives are *bounded*. Since bounded maximin fairness is a new notion, we first define the notion, and then we explain our ideas to achieve it.

Defining bounded maximin fairness. Imagine that there is a set of players $\mathcal{C}'$ who have external incentives that might incentivize them to deviate from honest behavior. We want to argue that even when $\mathcal{C}'$ may have arbitrary external incentives, as long as the external incentives are bounded and $\mathcal{C}'$ is *rational*, then any group of players without external incentives should feel safe to participate honestly — as long as they participate honestly, their utility will not be negative.

Regarding how to define *rationality* of the externally incentivized coalition, some interesting technicalities arise due to the fact that we model players and contracts as PPT Interactive Turing Machines. A strawman idea is to assume that the externally incentivized coalition $\mathcal{C}'$ plays the *optimal* strategy that maximizes its expected utility, and recall that we want to ensure no harm for any honest group or individual. The problem with this definition is that it makes the possibly unrealistic assumption that $\mathcal{C}'$ knows the optimal strategy based on the external incentive function, even though finding the optimal strategy may be computationally hard since the external incentive function can take an arbitrary form.

Instead, we want to protect honest individuals and groups against any PPT strategy of the coalition $\mathcal{C}'$ as long as the strategy is not *blatantly non-rational*. We can often show that some strategies are *blatantly non-rational* without having to find the optimal strategy. Indeed, later in our formal proofs, we will show that a class of strategies are blatantly non-rational, since simple modifications of such strategies lead to better outcomes for $\mathcal{C}'$. We then show that as long as $\mathcal{C}'$ does not adopt a blantantly non-rational strategy, an honest individual or group is protected.

We therefore devise the following definition which is parametrized by a strategy space $\mathcal{R}$ that contains all possible PPT strategies except a set of blatantly non-rational strategies $\overline{\mathcal{R}}$.

Definition 2.1 (Bounded maximin fairness)**.** We say that a protocol satisfies α-bounded maximin fairness w.r.t. some strategy space $\mathcal{R}$, iff for any set of PPT players denoted $\mathcal{C}$ without external incentives, and any externally incentivized PPT coalition $\mathcal{C}'$ that is disjoint from $\mathcal{C}$, controlling at most α fraction of mining power, and playing any strategy $S_{\mathcal{C}'} \in \mathcal{R}$, there is a negligible function $\mathsf{negl}(\cdot)$ such that except with $\mathsf{negl}(\lambda)$ probability, it must be that

$$\mathsf{util}^{\mathcal{C}}(HS_{\mathcal{C}}, S_{\mathcal{C}'}, HS_{\mathcal{D}}) \geq 0$$

where $\mathcal{D}$ denotes all players not in $\mathcal{C} \cup \mathcal{C}'$.

Achieving bounded maximin fairness. We show that a variant of our earlier CSP-fair protocol can achieve bounded maximin fairness. The main difference from the CSP-fair protocol is that we have to suitably increase the collateral amount in light of the a-priori known bound on the external incentive. In our bounded maximin fair protocol, both parties have to put down some collateral on both blockchains. While the idea of using collateral accordingly to defend against external incentives is natural and even commonly adopted in real-world applications like proof-of-stake, to the best of our knowledge, *we are the first to provide provable guarantees of this general paradigm — prior to our work, it was not even clear how to define fairness* (except for the aforementioned maximin fairness notion that is typically too stringent in real-world applications). Therefore, we hope that our definitional approach can lend to the formal reasoning of a wider class of protocols that also use this "staking in" paradigm.

To prove bounded maximin fairness, our high-level idea is to show that if a strategy ever triggers any of the bombs, it must be *non-rational*. In particular, the strategic individual or coalition would end up losing more in collateral than the amount of external incentives. We then prove that for any strategy that does not trigger the bombs, the honest players' utilities must be non-negative (barring the negligible probability that cryptography fails).

We defer the full description of our bounded maximin fair protocol and its proofs to Section 6. Besides the increase in collateral, in the formal description of our bounded maximin-fair protocol, we also introduce a couple more activation points (namely B1, B2, A1', and A2') with the goal of disincentivizing Alice from posting pre'_a too early and similarly, disincentivizing Bob from posting pre_b or pre_c too early. These extra activation points are needed for technical reasons in the proof. In particular, it makes our proof easier if the following is true: if one of the bombs is ever triggered, we want to directly attribute it to some message sent by the strategic coalition. In this way, we can show that the corresponding strategy is non-rational by arguing that if the coalition simply drops the message that triggers the bomb, its utility will improve. It turns out that these extra activation points lend to this argument. We conjecture that even the protocol without these extra activation points satisfy bounded maximin fairness — however, the proof will likely become much more sophisticated and we leave an exploration of this to future work.

2.6 Coalition Forming Meta-Game

To prove that our protocol satisfies CSP and bounded maximin fairness, we used the assumption that the coalition does not wield 100% of the mining power. We justify this assumption by analyzing the coalition forming meta-game in Section 7. Consider the knowledge-coin exchange protocol. In the coalition forming meta-game, imagine that Bob posts a smart contract that promises to split the gains with any miner who helps him get the $\$v$ back, after Alice has posted pre_a. We argue that 100% miner participation is not an equilibrium in this meta-game. A miner has the option of not joining the coalition, in which case it has some probability (proportional to its mining power) of claiming the $\$\epsilon$ itself by posting (pre_a, pre_b) — should Bob ever decide to cheat. To incentivize the miners to join, Bob must offer more than the expected gain of the miner had it not joined the coalition. However, to do so, the cost incurred for Bob would be greater than the price $\$v$ of the secret. We defer the details to Section 7.

In Section 7, we also show that this meta-game approach is not only a good complement to the definition of CSP-fairness and bounded maximin-fairness, but also helpful for reasoning about the known bribery attacks [TYME21, WHF19, HZ20, MHM18, JSZ+21] that pertain to the standard HTLC contract.

3 Model and Definitions

3.1 Blockchain, Transaction, and Smart Contracts

Smart contracts and transactions. We assume that *smart contracts* are *ideal functionalities* that are 1) aware of money; and 2) whose states are publicly observable. A smart contract can have one or more *activation points*. Each *transaction* is associated with a unique identifier, and consists of the following information: 1) an arbitrary message, 2) some non-negative amount of money, and 3) which activation point of which smart contract it wants to be sent to. When the transaction is executed, the corresponding activation point of the smart contract will be invoked, and then, some arbitrary computation may take place accompanied by the possible transfer of money.

Money can be transferred from and to the following entities: *smart contracts* and *players' pseudonyms*. Without loss of generality, we may assume that players cannot directly send and receive money among themselves; however, they can send money to or receive money from smart contracts. The balance of a smart contract is the amount of money it has received minus the amount of money it has sent out. *The balance of any smart contract must always be non-negative.*

We assume that each smart contract has a unique name, and each player may have multiple pseudonyms — in practice, a pseudonym is encoded as a public key. A miner is also a special player who is capable of mining blocks.

Mining. In this paper, we do not consider strategies that involve consensus- or network-level attacks — there is an orthogonal and complementary line of work that focuses on this topic [GKL15, PSS17, PS17b]. For example, a 51% miner can possibly gain by performing a double spending attack.

For simplicity, we assume an idealized mining process, that is, in each time step t, an ideal functionality picks a winning miner with probability proportional to each miner's mining power (or amount of stake for Proof-of-Stake blockchains). The winning miner may choose to include a set of transactions in the block, and order these transactions in an arbitrary order. At this moment, a new block is mined, and all (valid) transactions contained in the block are executed. Any transaction that has already been included in the blockchain before is considered invalid and will be ignored. The above idealized mining process can capture standard Proof-of-Work blockchains and Proof-of-Stake blockchains where the next proposer is selected on the fly with probability proportional to the stake held by the miner.

3.2 Players and Strategy Spaces

There are three kinds of players in the model: Alice, Bob, and the miners. We also call Alice and Bob the users to differentiate from miners. We consider the following strategy space for players.

Anyone, including Alice, Bob, or the miners, is allowed to do the following at any point of time:

1. Post a *transaction* to the network at the beginning of any time step. We assume that the network delay is 0, such that transactions posted are immediately seen by all other users and miners. When miners pick which transactions to include in some time step t, they can see transactions posted by users for time step t.

2. Create an arbitrary smart contract and put an arbitrary amount of money into the smart contract. For example, a smart contract can say, "if the state of the blockchain satisfies *some predicate* at *some time*, send *some pseudonym some amount* of money, where the recipient and the amount of money can also be dependent on the state of the blockchain.

Additionally, the *miners* are allowed the following actions: whenever it is chosen to mine a block, it can choose to include an arbitrary subset of the outstanding transactions into the block, and order them arbitrarily. The miner can also create new transactions on the fly and include them in the mined block.

Coalition. Alice or Bob can form a coalition with some of the miners. When the coalition is formed, all members in the coalition share their private information. The coalition's strategy space is the union of the strategy space of each member in the coalition. Notice that once Alice and Bob are in the same coalition, they can exchange the secret s privately without using the blockchain. Thus, we do not consider the coalition consists of Alice and Bob.

3.3 Protocol Execution

In our paper, an honest protocol is always a *simple* protocol that does not create additional smart contracts in the middle of the execution (even though strategic are allowed to create smart contracts on the fly). A protocol execution involves Alice, Bob, and the miners who are modeled as interactive Turing machines who can send and receive a special type of variables called money. Additionally, the protocol may involve one or more *smart contracts* which can be viewed as ideal functionalities whose states are publicly visible to anyone. Ideal functionalities are also interactive Turing machines capable of sending and receiving money.

For the honest protocol, we want the miners' honest behavior to be consistent with their honest behavior in typical consensus protocols, i.e., *the miner's honest behavior should include all outstanding transactions in the mined block.*

Finally, since we consider probabilistic polynomial time (PPT) players, we assume that the protocol execution is parametrized by a security parameter λ.

3.4 Definition: Atomic Swap

Suppose that Bob has x amount of Ethers denoted $\text{Ξ}x$, and Alice as x' amount of Bitcoins denoted $\text{Ƀ}x'$. Bob wants to exchange his $\text{Ξ}x$ with Alice's $\text{Ƀ}x'$. The currencies exchanged may be other currencies but we shall use Bitcoin and Ether as an example.

We may assume that Alice and Bob are not in the same coalition. Therefore, we effectively consider the following three types of strategic players or coalitions: 1) Alice-miner coalition (including Alice alone); 2) Bob-miner coalition (including Bob alone); and 3) miner-only coalition.

Given some strategic player or coalition, we assume that it has some specific valuation of each unit of Bitcoin and each unit of Ether. For convenience, we use the notation $\$\mathsf{AV}(\cdot)$ to denote the valuation function of Alice of an Alice-miner coalition; specifically, $\$\mathsf{AV}(\text{Ξ}x + \text{Ƀ}x') = \$v_a \cdot x + \$v'_a \cdot x'$ where $\$v_a \geq 0$ and $\$v'_a \geq 0$ denote how much Alice or the Alice-miner coalition values each Ether and Bitcoin, respectively. Similarly, we use the notation $\$\mathsf{BV}(\cdot)$ to denote the valuation function of Bob or a Bob-miner coalition, and we use $\$\mathsf{MV}(\cdot)$ to denote the valuation function of a miner-only coalition. Throughout this section, we may make the following assumption which justifies why Alice wants to exchange her $\text{Ƀ}x'$ with Bob for $\text{Ξ}x$, and vice versa.

$$\textbf{Assumption:} \quad \$\mathsf{AV}(\text{Ξ}x - \text{Ƀ}x') > 0, \quad \$\mathsf{BV}(\text{Ƀ}x' - \text{Ξ}x) > 0$$

Utility. Let $\text{Ƀ}d'_a, \text{Ξ}d_a \geq 0$ be the cryptocurrencies that Alice or an Alice-miner coalition deposit into the smart contracts. Let $\text{Ƀ}r'_a, \text{Ξ}r_a \geq 0$ be the payment Alice or an Alice-miner coalition receive from the smart contracts during the protocol. Let $\$e_a(\ldots) \geq 0$ denote the external incentives of Alice or the Alice-miner coalition, where $\ldots$ means that the external incentives can depend

arbitrarily on the blockchain's state. Now, we can define the utility $\$u_a$ of Alice or the Alice-miner coalition as follows:

$$\$u_a = \$\mathsf{AV}(\text{₿}r'_a - \text{₿}d'_a + \text{Ξ}r_a - \text{Ξ}d_a) + \$e_a(\ldots)$$

Similarly, we can define the utility $\$u_b$ of Bob or a Bob-miner oalition, and the utility $\$u_m$ of a miner-only coalition as follows:

$$\$u_b = \$\mathsf{BV}(\text{₿}r'_b - \text{₿}d'_b + \text{Ξ}r_b - \text{Ξ}d_b) + \$e_b(\ldots),$$

$$\$u_m = \$\mathsf{MV}(\text{₿}r'_m - \text{₿}d'_m + \text{Ξ}r_m - \text{Ξ}d_m) + \$e_m(\ldots),$$

where $\text{₿}r'_b, \text{Ξ}r_b \geq 0$, denote the payment the Bob-miner coalition or Bob receives during the protocol, $\text{₿}d'_b, \text{Ξ}d_b \geq 0$ denote the deposit the Bob-miner coalition or Bob sends to any smart contract during the protocol, and $\$e_b(\ldots) \geq 0$ denote any external incentives for the Bob-miner coalition or Bob. The variables $\text{₿}r'_m, \text{Ξ}r_m, \text{₿}d'_m, \text{Ξ}d_m, \$e_m(\ldots) \geq 0$ are similarly defined but for the miner-only coalition.

Modeling time. In our cross-chain atomic swap application, since the two blockchains have different block intervals, we use the following convention for denoting time. Without loss of generality, we may assume that the moment the protocol execution begins, the current lengths of the Bitcoin and Ethereum chains are renamed to 0. We use the terminology Ethereum time T to refer to the moment the Ethereum chain reaches length T, and similarly, we use the terminology Bitcoin time T' to refer to the moment when the Bitcoin chain reaches length T'.

Regarding external incentives. For bounded maximin fairness, we care about players' external incentives. Our modeling of external incentives is general and can capture external incentives of any form. Precisely, any side contract where money is redistributed to players of the present protocol is considered external incentive if 1) the contract is *pre-existing*, i.e., created before the start of the present protocol; or 2) the contract is created any time, and an *outsider*, i.e., not a player of the present protocol, deposited money into it. As an example of the former, imagine that Alice was involved in some bet prior to the fair exchange protocol that bets on the state of a future block, and the outcome of the present protocol may affect the state. As an example of the latter, imagine that Alice is Mallory's competitor, and Mallory is offering external incentives for anyone who can cause financial loss to Alice, such that Alice can become bankrupt. We stress that for a pre-existing side contract, it does not matter who funded the contract — any money deposited into a pre-existing contract is sunk cost w.r.t. this protocol. For a similar reason, assuming that the external incentive is non-negative is without loss of generality.

In our protocol, honest players will not create new contracts on the fly and deposit money into them during the protocol execution. However, in our strategy space, we allow strategic players to create new contracts on the fly and deposit money into them. Such contracts that are initiated and funded solely by strategic players of the present protocol *after* the start of the protocol are *not* considered external incentives, since our utility model takes into account the gains from these contracts — we say that such contracts "belong to" the present protocol.

Jumping ahead, to prove our protocols game theoretically fair, we do not care where the external incentives are coming from — we just need an upper-bound on the maximum amount of external incentives possible. In practice, for high-value transactions, we may want to assume a larger upper bound on the amount of external incentives; and in this case, our protocol will require the players to place a larger collateral.

3.5 Definitions: Incentive Compatibility and Dropout Resilience

CSP fairness. We review the notion of cooperative strategy proofness (CSP fairness), formulated in earlier works [PS17a, CGL+18, WAS22]. Intuitively, CSP fairness says that a coalition that is profit-driven and wants to maximize its own utility has no incentive to deviate from the honest protocol, as long as all other players play by the book. In this sense, the honest protocol achieves a *coalition-resistant Nash Equilibrium*.

Definition 3.1 (CSP fairness)**.** We say that a protocol satisfies γ-cooperative-strategy-proofness (or γ-CSP-fairness for short), iff the following holds. Let $\mathcal{C}$ be any coalition that controls at most $\gamma \in [0, 1)$ fraction of the mining power, and possibly includes either Alice or Bob. Then, for any probabilistic polynomial-time (PPT) strategy $S_{\mathcal{C}}$ of $\mathcal{C}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that except with $\mathsf{negl}(\lambda)$ probability,

$$\mathsf{util}^{\mathcal{C}}(S_{\mathcal{C}}, HS_{-\mathcal{C}}) \leq \mathsf{util}^{\mathcal{C}}(HS_{\mathcal{C}}, HS_{-\mathcal{C}})$$

where we use HS to mean the honest strategy.

The atomic swap application involves two separate blockchains. In this case, the γ parameter above is an upper bound on the coalition's mining power in both chains.

Bounded maximin fairness. Bounded maximin fairness is a new notion we introduce to capture the resilience against strategic players with arbitrary but bounded external incentives. We formally defined bounded maximin fairness earlier in Section 2.5.

Dropout resilience. In practice, a dropout can happen due to mistakes, misconfiguration, or unforeseen circumstances, e.g., Alice may lose her hardware wallet. We define a notion called dropout resilience which requires the following. Suppose that at least $1/\mathsf{poly}(\lambda)$ fraction of the mining power is honest. Then, if either Alice or Bob plays honestly but drops out before the end of the protocol, then with $1 - \mathsf{negl}(\lambda)$ probability where $\mathsf{negl}(\cdot)$ is a negligible function, the other party's utility must be non-negative.

Definition 3.2 (Dropout resilience)**.** A protocol is said to be dropout resilient, iff the following holds: as long as at least $1/\mathsf{poly}(\lambda)$ fraction of the mining power is honest, then with $1 - \mathsf{negl}(\lambda)$ probability an honest Alice (or Bob) is guaranteed to have non-negative utility even when Bob (or Alice) is honest but may drop out in the middle of the protocol execution.

3.6 Convention for Writing Smart Contracts

We use the following style of pseudo-code to express smart contracts. Since the contracts employed in this paper use simple logic, it does not take a general smart contract language like Ethereum to instantiate them. We can even instantiate them using Bitcoin's limited scripting language. Below we give a toy contract to explain the notation.

A toy contract

- **Parameters:** T.
- **Preparation phase:** Alice and Bob each deposits $\$d_a$ and $\$d_b + \$d'_b$, respectively.
- **Execution phase:**

A1: **On receive** (msg, $\$c$) from Alice: send $\$d < \$d_a + \$d_b$ to Bob.

A2: **After** T, **on receive** (msg, $\$c$) from Bob: send $\$d_a + \$d_b - \$d$ to Alice.

B1: **On receive** (msg, $\$c$) from Bob: send $\$d'_b$ to Bob.

In our notational system, every activation point is given a unique name that consists of a letter followed by a number. The leading letter defines the *type* of the activation point. All activation points of the same *type* are *mutually exclusive*. For example, if A1 has been invoked, then neither A1 nor A2 can be invoked any more; however, B1 can still be invoked (as long as it has not been invoked yet). If an activation point constrained some time interval (e.g., after T), then any attempted invocation that happens outside the specified time interval is considered invalid and not counted.

Our example toy contract above has a standard *prepation phase* where Alice and Bob each deposits some coins into the contract. In a practical implementation, the contract should allow each player to withdraw its deposit if the other player has not made its deposit yet. However, once both players have made their deposits, the redistribution of money is only possible through the activation points of the execution phase. Later in our paper, the preparation phase may also have customized logic — in this case, we will spell out the logic of the preparation phase explicitly.

In our contract notation, we simply assume that there is an authenticated channel from each user to the smart contract. In practice, each party involved is actually identified by their public keys. The party can sign the message sent to the activation point to authenticate itself. In a practical instantiation, each party may also need to pay a typically small *transaction fee* for their transaction to be confirmed. For simplicity, we ignore the transaction fee in our theoretical model since we need not rely on transaction fees to achieve our game theoretic guarantees. Adding an ϵ-small transaction fee in a practical instantiation will only introduce $O(\epsilon)$-slack to our game theoretic guarantees.

4 CSP-fair Knowledge-Coin Exchange

In this section, we prove the warmup knowledge-coin protocol of Section 2.3 CSP-fair. Before that, we first give the formal utility definitions in 4.1.

4.1 Definitions

Imagine that Alice has some secret pre_a and Bob offers to pay Alice $\$v$ amount of coins in exchange for the secret. For example, pre_a may be a secret value that Bob can later use to unlock some other coins, e.g., through a smart contract.

We assume that the secret pre_a is worth $\$v_a$ and $\$v_b$ to Alice and Bob, respectively. That is, Alice will lose utility $\$v_a$ if pre_a is released to someone else, and Bob will gain $\$v_b$ if he learns pre_a. We assume that $\$v_b > \$v > \$v_a$, such that Alice wants to sell the secret pre_a to Bob at a price of $\$v$.

Players' utility. Let $\beta \in \{0, 1\}$ be an indicator such that $\beta = 1$ if and only if Bob outputs the secret pre_a at the end of the protocol. Let $\$d_a \geq 0$ and $\$d_b \geq 0$ be the amount of money Alice and Bob deposit into the smart contract, respectively. Let $\$r_a \geq 0$ and $\$r_b \geq 0$ be the payments that Alice and Bob obtain from all smart contracts during the protocol.

Then, Alice's utility, $\$u_a$, is defined as

$$\$u_a = -\$d_a + \$r_a - \beta \cdot \$v_a$$

and Bob's utility, $\$u_b$, is defined as

$$\$u_b = -\$d_b + \$r_b + \beta \cdot \$v_b$$

Similar to Alice and Bob, we can also define the utility for any miner. Fix some miner. Let $\$d_m$ be the money that the miner deposits into the smart contracts belonging to this protocol, and let $\$r_m$ be the payment received by the miner in the current protocol instance. A miner's utility, denoted $\$u_m$, is defined as

$$\$u_m = -\$d_m + \$r_m$$

Finally, the joint utility of the coalition is simply the sum of every coalition member's utility.

4.2 Proofs: CSP Fairness and Dropout Resilience

Lemma 4.1 (Alice-miner coalition)**.** *Let $\mathcal{C}$ be any coalition that consists of Alice and an arbitrary subset of miners*[5] *(possibly no miner). Then, for any (even unbounded) coalition strategy $S_{\mathcal{C}}$,*

$$\mathsf{util}^{\mathcal{C}}(S_{\mathcal{C}}, HS_{-\mathcal{C}}) \leq \mathsf{util}^{\mathcal{C}}(HS_{\mathcal{C}}, HS_{-\mathcal{C}})$$

where $HS_{-\mathcal{C}}$ denotes the honest strategy for everyone not in $\mathcal{C}$.

Proof. When the coalition $\mathcal{C}$ follows the protocol, they will send pre_a at $t = 0$, and P1 will be activated in the next block. In this case, the utility of $\mathcal{C}$ is $\$v - \$v_a$.

Now, consider the case that the coalition $\mathcal{C}$ deviates from the honest strategy. We may assume that the coalition does not post any new smart contract on the fly and deposit money into it[6] — if it did so, it cannot recover more than its deposit since any player not in $\mathcal{C}$ will not invoke the smart contract. There are two possibilities:

- First, P1 is activated at some point. In this case, nothing else can be activated. Thus, the utility of $\mathcal{C}$ is $\$v - \$v_a$, which is the same as the honest case.
- Second, P1 is never activated. The Alice-miner coalition cannot cash out from P2 or C1, it can only cash out ϵ from C2. However, when C2 is activated, pre_a is publicly known, so the utility of $\mathcal{C}$ is $\$\epsilon - \$v_a$, which is less than the honest case since $\$\epsilon < \$v$.

□

Lemma 4.2 (Bob-miner coalition)**.** *Let $\mathcal{C}$ be any coalition that consists of Bob and a subset of miners controlling at most γ fraction of mining power. Then, as long as $\gamma^{T_2} \leq \frac{\$c_b}{\$c_b + \$v}$, for any (even unbounded) coalition strategy $S_{\mathcal{C}}$, it must be that*

$$\mathsf{util}^{\mathcal{C}}(S_{\mathcal{C}}, HS_{-\mathcal{C}}) \leq \mathsf{util}^{\mathcal{C}}(HS_{\mathcal{C}}, HS_{-\mathcal{C}})$$

Proof. The honest Alice will always send pre_a to P1. Thus, when $\mathcal{C}$ follows the protocol, P1 will be activated in the next block, and the utility of $\mathcal{C}$ is $\$v_b - \$v$.

Now, suppose $\mathcal{C}$ may deviate from the protocol. As in Lemma 4.1, we may assume that the coalition does not post any new smart contract on the fly and deposit money into it. There are three cases.

[5] We assume that the coalition cannot break the underlying consensus layer. If the underlying consensus actually secures against, say, honest majority, then essentially the lemma holds for any coalition that wields minority of the mining power.

[6] However, the coalition $\mathcal{C}$ itself could be facilitated by smart contracts, our modeling of coalition already captures any arbitrary side contract within the coalition.

- First, neither P1 nor P2 is activated. Because P2 is not activated, C1 cannot be activated. The Bob-miner coalition can only get $\$\epsilon$ from C2. Thus, the coalition's utility is at most $\$v_b - \$v - \$c_b + \$\epsilon < \$v_b - \$v$ where the inequality is due to the constrant $\$c_b > \$\epsilon$.

- Second, P1 is activated. In this case, nothing else can be activated, and the utility of $\mathcal{C}$ is $\$v_b - \$v$, which the same as the honest case.

- Third, P2 is activated. Let $t^* \geq T_1$ be the time at which P2 is activated. There are two subcases. In the first subcase, the coalition also gets $\$\epsilon$ from C2 during $[t^*, t^* + T_2]$. In this case, the coalition's utility is at most $\$v_b - \$c_b - \$v + \$\epsilon$, and since $\$c_b > \$\epsilon$, this is less than the honest case. Henceforth, we may assume that the coalition does not invoke C2 after time t^* as after time $t^* + T_2$ it is always better to invoke C1. Since the honest Alice posts pre_a at $t = 0$ and $t^* \geq T_1$, both pre_a and pre_b are publicly known at t^*. Since all non-colluding miners are honest, after t^*, they will activate C2 themselves when they mine a new block if C2 has not already been activated before. If a non-colluding miner mines a new block during $(t^*, t^* + T_2]$, we say that the coalition loses the race. Otherwise, we say that the coalition wins the race. If the coalition loses the race, then it gets nothing from C1 or C2, and thus its utility is at most $\$v_b - \$c_b - \$v$. Else if it wins the race, then the coalition's utility is at most $\$v_b$. The probability p that the coalition wins the race is upper bounded by $p \leq \gamma^{T_2}$. Therefore, the coalition's expected utility is at most

$$(\$v_b - \$c_b - \$v) \cdot (1 - p) + \$v_b \cdot p.$$

  For $(\$v_b - \$c_b - \$v) \cdot (1 - p) + \$v_b \cdot p$ to exceed the honest utility $\$v_b - \$v$, it must be that $p > \frac{\$c_b}{\$c_b + \$v}$ which contradicts our assumption.

We thus conclude that $\mathcal{C}$ cannot increase its utility through any deviation. □

Theorem 4.3 (CSP fairness)**.** *Suppose that the hash function $H(\cdot)$ is a one-way function and that $\gamma^{T_2} \leq \frac{\$c_b}{\$c_b + \$v}$. Then, the* Rapidash *protocol satisfies γ-CSP-fairness.*

Proof. Lemmas 4.1 and 4.2 proved γ-CSP-fairness for the cases when the coalition consists of either Alice or Bob, and possibly some miners. Since by our assumption, Alice and Bob are not in the same coalition, it remains to show γ-CSP-fairness for the case when the coalition consists only of some miners whose mining power does not exceed γ. Since both Alice and Bob are honest, the coalition's utility is 0 unless C2 is activated. However, C2 requires that $\mathcal{C}$ to find pre_b on its own — the probability of this happening is negligibly small due to the one-wayness of the hash function $H(\cdot)$. □

We now prove that Rapidash is dropout resilient.

Theorem 4.4 (Dropout resilience)**.** *Suppose that $H(\cdot)$ is a one-way function and that all players are PPT machines.* Rapidash *is dropout resilient. In other words, suppose at least $1/\mathsf{poly}(\lambda)$ fraction of the mining power is honest. If either Alice or Bob plays honestly but drops out before the end of the protocol, then with $1 - \mathsf{negl}(\lambda)$ probability, the other party's utility should be non-negative.*

Proof. Throughout the proof, for any $X \in \{pre_a, pre_b\}$, we ignore the negligible probability that the miners can find the preimage X by itself if Alice and Bob have never sent X before.

We first analyze the case where Alice drops out. There are two possible case: 1) Alice drops out before posting a transaction containing pre_a; 2) Alice drops out after she already posted a transaction containing pre_a at $t = 0$. In the first case, as long as $1/\mathsf{poly}(\lambda)$ fraction of the mining power is

honest, Bob would activate P2 and C1 in polynomial time except with negligible probability, and his utility is 0 since he simply gets all his deposit back. In the second case, the honest Bob will not post pre_b to P2. An honest miner would include Alice's transaction and activate P1. As long as $1/\mathsf{poly}(\lambda)$ fraction of the mining power is honest, P1 will be activated in polynomial time except with negligible probability. As a result, Bob's utility is $\$v_b - \$v > 0$.

Next, we analyze the case where Bob drops out. In this case, Alice always posts a transaction containing pre_a, and except with negligible probability, P1 will always be activated. Thus, Alice's utility is always $\$v - \$v_a > 0$.

To sum up, in all cases, the utility of the remaining party is always non-negative except with negligible probability. □

5 Atomic Swap: Achieving CSP-Fairness

In Section 3.4, we defined the problem of cross-chain atomic swap. Further, we gave an overview of our CSP-fair protocol in Section 2.4. In this section, we formlly specify the detailed parameters and the protocol, and we then prove CSP fairness.

Contracts. Our atomic swap protocol involve two contracts, called Rapidash and Rapidash', executed on the Ethereum and Bitcoin blockchains, respectively. The specification of the contracts were presented in Section 2.4. Below, we specify the detailed parameter constraints and the full protocol.

Parameter constraints. We first specify the parameter constraints. The parameters ₿c'_a and ₿c'_b denote Alice and Bob's collateral into Rapidash', respective; and Ξc_b denotes Bob's collateral into Rapidash. Our protocol uses a few timeouts including T_0, T_1, and T expressed in Ethereum time (i.e., the length of the Ethereum blockchain), and T'_1 expressed in Bitcoin time. T is a timeout by which Alice and Bob should deposit into the Rapidash and Rapidash' contracts; if not, honest players would invoke an abort procedure. T_0 is a timeout for Bob to post pre_c to Rapidash—if Bob does not send pre_c in time, the honest Alice will invoke the abort procedure. Recall that pre_c is an explicit signal from Bob to proceed with contract execution. Only when Bob releases pre_c can the P1 and C2 be activated. T_1 and T'_1 denote the earliest times at which P2 and P2' can be activated, expressed in Ethereum time and Bitcoin time, respectively. As mentioned earlier, it is important that Bitcoin time T'_1 happens later than Etheurem time T_1, since otherwise, Alice could get refunded from the Bitcoin contract Rapidash before Bob even posts pre_b to P2, and then trigger P1 to get Bob's Ξx for free. The parameter τ denotes the window of time that must elapse between P2 and C1 of Rapidash, i.e., between when Bob expresses his intent to back out, and when the refund actually happens. The parameter τ' is similarly defined for Rapidash'. Below, we explain the parameter choices under which CSP-fairness holds.

Parameter Constraints for Atomic Swap

Constraints for Rapidash (on Ethereum):

- $h_a = H(pre_a)$, $h_b = H(pre_b)$ and $h_c = H(pre_c)$.
- $T_1 > T_0 > T > 0$.
- Ξ$0 <$ Ξ$\epsilon <$ Ξx, and Ξ$c_b >$ Ξϵ

Constraints for Rapidash' (on Bitcoin):

- $h'_b = H(pre_a) = h_a$ and $h'_a = H(pre'_a)$.
- Bitcoin time $T'_1 >$ Ethereum time T_1, i.e., the Bitcoin block of length T'_1 is mined after the Ethereum block of length T_1.[a]
- $\ddot{\mathrm{B}}0 < \ddot{\mathrm{B}}\epsilon'$, $\ddot{\mathrm{B}}c'_a > \ddot{\mathrm{B}}\epsilon'$ and $\ddot{\mathrm{B}}c'_b > \ddot{\mathrm{B}}\epsilon'$.

Choice of timeouts: *// γ is the coalition's fraction of mining power*

- $\tau \geq 1$, $\tau' \geq 1$.
- $\gamma^{\tau'} \leq \frac{\ddot{\mathrm{B}}c'_a}{\ddot{\mathrm{B}}c'_a + \ddot{\mathrm{B}}x'}$, $\gamma^{\tau} \leq \frac{\ddot{\mathrm{E}}c_b}{\ddot{\mathrm{E}}c_b + \ddot{\mathrm{E}}x}$

[a] In practice, this constraint should be respected except with negligible probability despite the the variance in inter-block times.

Intuitively, the constraint $\ddot{\mathrm{E}}\epsilon < \ddot{\mathrm{E}}x$ makes sure that Alice, who does not have collateral in Rapidash, always prefers P1 to the bomb C2. The constraint $\ddot{\mathrm{E}}c_b > \ddot{\mathrm{E}}\epsilon$ makes sure that if Bob gets Alice's $\ddot{\mathrm{B}}x'$ and triggers the bomb C2, he still loses to the honest case, and the constraint $\ddot{\mathrm{B}}c'_a > \ddot{\mathrm{B}}\epsilon'$ serves a similar purpose. The condition $\ddot{\mathrm{B}}c'_b > \ddot{\mathrm{B}}\epsilon'$ makes sure that Bob does not want to trigger the bomb C2' even when he can get all of his deposit into Rapidash refunded. Finally, the constraint $\gamma^{\tau} < \frac{\ddot{\mathrm{E}}c_b}{\ddot{\mathrm{E}}c_b + \ddot{\mathrm{E}}x}$ makes sure that the window between P2 and C1 is sufficiently long such that once the honest Alice has posted pre_a, it is not worth it for Bob to take a gamble to trigger P2 and C1. In particular, if during the τ window, any honest miner mines a block, then the bomb C2 will be triggered and Bob will lose his collateral. The condition $\gamma^{\tau'} < \frac{\ddot{\mathrm{E}}c'_a}{\ddot{\mathrm{E}}c'_a + \ddot{\mathrm{E}}x'}$ serves a similar purpose, but now for Alice and Rapidash'.

We give some typical parameter choices below. For example, suppose we choose $\ddot{\mathrm{E}}c_b = \ddot{\mathrm{E}}x$. Then, we need to make sure $\gamma^{\tau} \leq 1/2$. This means if $\gamma = 90\%$, we can set $\tau = 7$; if $\gamma = 49.9\%$, we can set $\tau = 1$. Asymptotically, for any $\gamma = O(1)$, τ is a constant. Increasing $\ddot{\mathrm{E}}c_b$ helps to make τ smaller. A similar calculation also works for τ' and $\ddot{\mathrm{B}}c'_a$.

Full protocol. We describe the full atomic swap protocol below, which specifies Alice, Bob, and the miners' honest behavior.

- *Miner.* The miner's honest protocol is described below.
  - The miner watches all transactions posted to P1, P2, C1, C2, P1', P2', C1', and C2' (i.e., all the P-type and C-type activation points for both contracts), to see if they contain a valid $pre_a = pre'_b$, pre_b, pre'_a, and pre_c.
  - As soon as the miner has observed pre_a, pre_b and pre_c, it posts (pre_a, pre_b, pre_c) to C2; as soon as the miner has observed both pre'_a and pre'_b, it posts (pre'_a, pre'_b) to C2'; as soon as the miner has observed pre'_a and pre_b, it posts (pre'_a, pre_b) to C2'.
  - Whenever the miner mines a block, it always includes its own transactions ahead of others.
- *Alice and Bob.* Below, we define the honest protocol for Alice and Bob. The moment that both contracts have been posted and take effect is defined to be the start of the execution (i.e. $t = 0$). We define Ethereum time 0 and Bitcoin time 0 to be the length of Ethereum and Bitcoin when the execution starts, respectively.

Atomic Swap Protocol — Alice and Bob

Preparation Phase:

1. At $t = 0$, Alice sends the deposit transaction of $\text{₿}x' + \text{₿}c'_a$ to Rapidash' and sends the collateral transaction of $\text{Ξ}c_a$ to Rapidash. Similarly, Bob sends the deposit transaction of $\text{Ξ}x + \text{Ξ}c_b$ to Rapidash and sends the collateral transaction of $\text{₿}c'_b$ to Rapidash'.
2. As soon as both Rapidash and Rapidash' enter the execution phase, Bob sends pre_c to P1 and enters the execution phase. As soon as Bob sent pre_c to P1, Alice enters to the execution phase.
3. At Ethereum time T, if either Rapidash or Rapidash' has not entered the execution phase, Alice and Bob go to the abort phase.
4. At Ethereum time T_0, if Bob has not sent pre_c to P1, Alice enters to the abort phase.

Execution Phase:

1. Alice sends pre_a to P1. As soon as P1 has been activated, Alice sends an empty message _ to P1'.
2. Before Ethereum time T_1, as soon as Alice sends pre_a to P1, Bob sends $pre'_b = pre_a$ to P1'. If Alice does not send pre_a to P1 before Ethereum time T_1, Bob sends pre_b to P2 at Ethereum time T_1.
3. If τ *Ethereum time* has passed since P2 is activated, Alice and Bob send _ to C1. As soon as C1 is activated, Bob sends _ to P2'.
4. If τ' *Bitcoin time* has passed since P2' is activated, Alice and Bob send _ to C1'.

Abort Phase:

1. At Ethereum time T_0, Bob sends _ to P2' and pre_b to P2; Alice sends _ to P2.
2. At Ethereum time T_1, if Bob has not sent _ to P2', Alice sends pre'_a to P2'.
3. If τ' *Bitcoin time* has passed since P2' is activated, Alice and Bob send _ to C1'; similarly, if τ *Ethereum time* has passed since P2 is activated, Alice and Bob send _ to C1.

5.1 Proofs

Before proving CSP fairness of the protocol, we give some useful lemmas. CSP fairness is formally proven by Theorem 5.5.

We define the *net profit* of $\mathcal{C}$ from Rapidash to be the coins that $\mathcal{C}$ gets from Rapidash *minus* the coins that $\mathcal{C}$ deposits into Rapidash. The net profit of $\mathcal{C}$ from Rapidash' is defined similarly. Notice that the net profit might be negative, which means $\mathcal{C}$ deposits more coins than what it gets.

Lemma 5.1. *Suppose* Rapidash *and* Rapidash' *both enter the execution phase. Suppose the coalition $\mathcal{A}$ consists of Alice and an arbitrary $\gamma \in [0, 1]$ fraction of the mining power. If $\text{₿}c'_a > \text{₿}\epsilon'$, the utility of $\mathcal{A}$ can be more than the honest case, that is, $\$\mathsf{AV}(\text{Ξ}x - \text{₿}x')$, only if one of the following holds*

- *P1, P2' and C1' are activated;*
- *C2, P2' and C1' are activated.*

Proof. First, we prove that either P1 or C2 is the necessary condition for the utility of $\mathcal{A}$ to be more than the honest case. For the sake of reaching a contradiction, suppose neither of P1 or C2 is activated. Because $\mathcal{A}$ cannot get any coin from P2 or C1, the net profit from Rapidash is at most 0. However, because $\text{₿}c'_a > \text{₿}\epsilon'$, we have $\text{₿}\epsilon' - \text{₿}x' - \text{₿}c'_a < -\text{₿}x' < 0$. Thus, the net profit from Rapidash' is also at most 0. Consequently, the utility of $\mathcal{A}$ is at most zero, which is less than $\$\mathsf{AV}(\text{Ð}x - \text{₿}x')$. Thus, one of P1 and C2 must be activated.

Next, we prove that P2' and C1' must be activated for the utility of $\mathcal{A}$ to be more than the honest case. For the sake of reaching a contradiction, suppose one of them is not activated. Because $\text{₿}c'_a > \text{₿}\epsilon' > 0$, the net profit from Rapidash' is at most $-\text{₿}x'$ since P2' or C1' is not activated. However, the net profit from Rapidash is at most $\text{Ð}x$. Thus, the utility of $\mathcal{A}$ is at most $\$\mathsf{AV}(\text{Ð}x - \text{₿}x')$, which is the same as the honest case. Thus, both of P2' and C1' must be activated. □

Lemma 5.2 (Alice-miner coalition)**.** *Suppose that the hash function $H(\cdot)$ is a one-way function. Let $\mathcal{A}$ be any coalition that consists of Alice and $\gamma \in [0,1]$ fraction of mining power. Then, as long as $\gamma^{\tau'} \leq \frac{\text{₿}c'_a}{\text{₿}c'_a + \text{₿}x'}$, for any PPT coalition strategy $S_{\mathcal{A}}$, except with negligible probability, it must be*

$$\mathsf{util}^{\mathcal{A}}(S_{\mathcal{A}}, HS_{-\mathcal{A}}) \leq \mathsf{util}^{\mathcal{A}}(HS_{\mathcal{A}}, HS_{-\mathcal{A}}),$$

where $HS_{\mathcal{A}}$ and $HS_{-\mathcal{A}}$ denotes the honest strategy for coalition $\mathcal{A}$ and everyone not in $\mathcal{A}$, respectively.

Proof. Recall that the utility of $\mathcal{A}$ is $\$\mathsf{AV}(\text{Ð}x - \text{₿}x') > 0$ under an honest execution. Now, suppose $\mathcal{A}$ may deviate from the protocol. We may assume that the coalition does not post any new smart contract on the fly and deposit money into it (see the definition of strategy space in Section 3.2) — if it did so, it cannot recover more than its deposit since any player not in $\mathcal{A}$ will not invoke the smart contract. We analyze the possible cases depending on which phase Bob enters.

Bob enters the abort phase. If Rapidash never enters the execution phase, the net profit of $\mathcal{A}$ from Rapidash is at most zero. Now, assume Rapidash enters the execution phase. When Bob enters the abort phase, he never sends any transaction containing pre_c. Ignoring the negligible probability that $\mathcal{A}$ finds pre_c by itself, P1 or C2 can never be activated. Because Alice does not get any coin from P2 or C1, the net profit of $\mathcal{A}$ from Rapidash is at most zero. On the other hand, because $\text{₿}c'_a > \text{₿}\epsilon'$, the net profit of $\mathcal{A}$ from Rapidash' is at most zero, no matter whether Rapidash' enters the execution phase or not.

To sum up, except with negligible probability, the utility of $\mathcal{A}$ is at most zero, which is less than the honest case.

Bob enters the execution phase. If Bob enters the execution phase, both Rapidash and Rapidash' must enter the execution phase. By Lemma 5.1, the utility of $\mathcal{A}$ can exceed the honest case only when (P1 + P2' + C1') or (C2 + P2' + C1') are activated. Henceforth, we assume either (P1 + P2' + C1') or (C2 + P2' + C1') are activated. Notice that in either case, P2' must be activated. When Bob enters the execution phase, P2' can be activated only either 1) by Bob sending _ to P2' after C1 has been activated, or 2) by Alice sending pre'_a to P2'. Consider the first scenario. In this case, since C1 has been activated, Alice can't get any money from Rapidash. However, from Rapidash', Alice can get at most zero. Thus, the utility of $\mathcal{A}$ is less than the honest case. Now consider the second case. Suppose that P2' is activated at Bitcoin time $t^* \geq T'_1$, so pre'_a is publicly known after Bitcoin time t^*.

Now, notice that if P1 or C2 is activated, $\mathcal{A}$ has to send a transaction containing pre_a.

- *Case 1: $\mathcal{A}$ sends a transaction containing pre_a to P1 or C2 before Ethereum time T_1.* Since Ethereum time T_1 is earlier than Bitcoin time T_1', pre_a and pre_a' are both publicly known at Bitcoin time t^*. Recall that $pre_a = pre_b'$. Thus, during Bitcoin time $(t^*, t^* + \tau']$, any miner in $-\mathcal{A}$ will activate C2' if it wins a block. We say $\mathcal{A}$ loses the race if a non-colluding miner mines a new block during Bitcoin time $(t^*, t^* + \tau']$. Otherwise, we say $\mathcal{A}$ wins the race. If $\mathcal{A}$ loses the race, it gets nothing from C1' or C2', and its utility is at most $\$\mathsf{AV}(\ddot{\mathrm{E}}x - \ddot{\mathrm{B}}x' - \ddot{\mathrm{B}}c_a')$. Else if $\mathcal{A}$ wins the race, then its utility is at most $\$\mathsf{AV}(\ddot{\mathrm{E}}x)$, which can be achieved by activating P2', C1' and P1. The probability p that $\mathcal{A}$ wins the race is upper bounded by $p \leq \gamma^{\tau'}$. Therefore, the expected utility of $\mathcal{A}$ is upper bounded by

$$\$\mathsf{AV}((\ddot{\mathrm{E}}x - \ddot{\mathrm{B}}x' - \ddot{\mathrm{B}}c_a') \cdot (1-p) + \ddot{\mathrm{E}}x \cdot p).$$

  Since $p \leq \gamma^{\tau'} \leq \frac{\ddot{\mathrm{B}}c_a'}{\ddot{\mathrm{B}}c_a' + \ddot{\mathrm{B}}x'}$, we have

$$\$\mathsf{AV}((\ddot{\mathrm{E}}x - \ddot{\mathrm{B}}x' - \ddot{\mathrm{B}}c_a') \cdot (1-p) + \ddot{\mathrm{E}}x \cdot p) < \$\mathsf{AV}(\ddot{\mathrm{E}}x - \ddot{\mathrm{B}}x').$$

- *Case 2: $\mathcal{A}$ does not send any transaction containing pre_a to P1 or C2 before Ethereum time T_1.* In this case, the honest Bob will send pre_b to P2 at Ethereum time T_1. Because P2' is activated at Bitcoin time $t^* \geq T_1'$, which is later than Ethereum time T_1, pre_a' and pre_b are both publicly known at Bitcoin time t^*. Thus, during Bitcoin time $(t^*, t^* + \tau']$, any miner in $-\mathcal{A}$ will activate C2' if it wins a block. By the same calculation as the previous case, since $p \leq \gamma^{\tau'} \leq \frac{\ddot{\mathrm{B}}c_a'}{\ddot{\mathrm{B}}c_a' + \ddot{\mathrm{B}}x'}$, we have $\$\mathsf{AV}((\ddot{\mathrm{B}}x' - \ddot{\mathrm{B}}c_a' + \ddot{\mathrm{E}}x) \cdot (1-p) + \ddot{\mathrm{E}}x \cdot p) < \$\mathsf{AV}(\ddot{\mathrm{E}}x - \ddot{\mathrm{B}}x')$.

□

Lemma 5.3. *Suppose* Rapidash *and* Rapidash' *both enter the execution phase. Suppose the coalition $\mathcal{B}$ consists of Bob and an arbitrary $\gamma \in [0,1]$ fraction of the mining power. If $\ddot{\mathrm{E}}c_b > \ddot{\mathrm{E}}\epsilon$ and $\ddot{\mathrm{B}}c_b' > \ddot{\mathrm{B}}\epsilon'$, the utility of $\mathcal{B}$ can be more than the honest case, that is, $\$\mathsf{BV}(\ddot{\mathrm{B}}x' - \ddot{\mathrm{E}}x)$, only if P2, C1 and P1' are activated.*

Proof. First, note that P1 and P2 are mutually exclusive, and neither C1 nor C2 can be activated after P1 because not enough money is available in the contract. Moreover, C1 and C2 are mutually exclusive. Thus, all the possible cases for the net profit of Bob's coalition from Rapidash can be summarized as shown in Table 1.

which is activated	net profit of Bob's coalition
none or only P2	$-\ddot{\mathrm{E}}x - \ddot{\mathrm{E}}c_b$
P1	$-\ddot{\mathrm{E}}x$
P2 + C1	0
C2 or P2 + C2	$\leq \ddot{\mathrm{E}}\epsilon - \ddot{\mathrm{E}}x - \ddot{\mathrm{E}}c_b$

Table 1: The net profit of Bob's coalition from Rapidash, assuming that Rapidash enters the execution phase.

Similarly, if P1' is activated, no other activation points of Rapidash' can be activated. Moreover, C1' and C2' are mutually exclusive. Thus, all the possible cases for the net profit of Bob's coalition from Rapidash can be summarized as shown in Table 2.

Suppose the coalition $\mathcal{C}$ consists of the miners and Bob. If $\mathcal{C}$ follows the protocol, P1 and P1' will be activated, and the utility of $\mathcal{C}$ is $\$\mathsf{BV}(\ddot{\mathrm{B}}x' - \ddot{\mathrm{E}}x) > 0$. When P2, C1 and P1' are activated,

which is activated	net profit of Bob's coalition
none or only P2'	$-\ddot{\mathbb{B}}c'_b$
P1'	$\ddot{\mathbb{B}}x'$
P2' + C1'	0
C2' or P2' + C2'	$\leq \ddot{\mathbb{B}}\epsilon' - \ddot{\mathbb{B}}c'_b$

Table 2: The net profit of Bob's coalition from Rapidash', assuming that Rapidash' enters the execution phase.

$\mathcal{C}$'s utility is $\$\mathsf{BV}(\ddot{\mathbb{B}}x')$. Now, we will show that it is the only scenario for $\mathcal{C}$'s utility to exceed the honest case. For the sake of reaching a contradiction, suppose $\mathcal{C}$'s utility is strictly greater than $\$\mathsf{BV}(\ddot{\mathbb{B}}x' - \ddot{\mathbb{E}}x)$, while one of P2, C1 and P1' is not activated. There are two subcases.

- **Subcase 1: P1' is not activated.** Because $\ddot{\mathbb{B}}c'_b > \ddot{\mathbb{B}}\epsilon'$, we have $\ddot{\mathbb{B}}\epsilon' - \ddot{\mathbb{B}}c'_b < 0$. Thus, if P1' is not activated, the net profit from Rapidash' is at most 0. Because $\ddot{\mathbb{E}}c_b > \ddot{\mathbb{E}}\epsilon$, we have $\ddot{\mathbb{E}}\epsilon - \ddot{\mathbb{E}}x - \ddot{\mathbb{E}}c_b < -\ddot{\mathbb{E}}x$. Thus, the net profit from Rapidash is also at most 0. Consequently, the utility of $\mathcal{C}$ is at most zero, which is less than $\$\mathsf{BV}(\ddot{\mathbb{B}}x' - \ddot{\mathbb{E}}x)$.
- **Subcase 2: P2 or C1 is not activated.** Because $\ddot{\mathbb{E}}c_b > \ddot{\mathbb{E}}\epsilon \geq 0$, the net profit from Rapidash is at most $-\ddot{\mathbb{E}}x$ since P2 or C1 is not activated. However, the net profit from Rapidash' is at most $\ddot{\mathbb{B}}x'$. Thus, the utility of $\mathcal{C}$ is at most $\$\mathsf{BV}(\ddot{\mathbb{B}}x' - \ddot{\mathbb{E}}x)$, which is the same as the honest case.

Therefore, we conclude that if $\mathcal{C}$'s utility is strictly greater than $\$\mathsf{BV}(\ddot{\mathbb{B}}x' - \ddot{\mathbb{E}}x)$, P2, C1 and P1' must be activated. □

Lemma 5.4 (Bob-miner coalition)**.** *Suppose that the hash function $H(\cdot)$ is a one-way function. Let $\mathcal{B}$ be any coalition that consists of Bob and a subset of miners controlling at most $\gamma \in [0,1]$ fraction of mining power. Then, as long as $\gamma^\tau \leq \frac{\ddot{\mathbb{E}}c_b}{\ddot{\mathbb{E}}c_b + \ddot{\mathbb{E}}x}$, for any PPT coalition strategy $S_{\mathcal{B}}$, except with negligible probability, it must be*

$$\mathsf{util}^{\mathcal{B}}(S_{\mathcal{B}}, HS_{-\mathcal{B}}) \leq \mathsf{util}^{\mathcal{B}}(HS_{\mathcal{B}}, HS_{-\mathcal{B}}),$$

where $HS_{\mathcal{B}}$ and $HS_{-\mathcal{B}}$ denotes the honest strategy for coalition $\mathcal{B}$ and everyone not in $\mathcal{B}$, respectively.

Proof. Recall that the utility of $\mathcal{B}$ is $\$\mathsf{BV}(\ddot{\mathbb{B}}x' - \ddot{\mathbb{E}}x) > 0$ under an honest execution. Now, suppose $\mathcal{B}$ may deviate from the protocol. We may assume that the coalition does not post any new smart contract on the fly and deposit money into it — if it did so, it cannot recover more than its deposit since any player not in $\mathcal{B}$ will not invoke the smart contract. We analyze the two possible cases depending on which phase Alice enters.

Alice enters the abort phase. If Rapidash' never enters the execution phase, the net profit of $\mathcal{B}$ from Rapidash' is at most zero. Now, assume Rapidash' enters the execution phase. When Alice enters the abort phase, she never sends any transaction containing $pre_a = pre'_b$. Ignoring the negligible that $\mathcal{B}$ finds pre'_b by itself, P1' can never be activated. Because $\ddot{\mathbb{B}}c'_b > \ddot{\mathbb{E}}\epsilon'$, the net profit of $\mathcal{B}$ from Rapidash' is at most zero. On the other hand, because $\ddot{\mathbb{E}}x > \ddot{\mathbb{E}}\epsilon$, the net profit of $\mathcal{B}$ from Rapidash is at most zero, no matter Rapidash enters the execution phase or not.

To sum up, except with negligible probability, the utility of $\mathcal{B}$ is at most zero, which is less than the honest case.

Alice enters the execution phase. By Lemma 5.3, the utility of $\mathcal{B}$ can be more than the honest case only if P2, C1 and P1' are activated, so we assume it is the case. Therefore, we may assume that P2 is activated at Ethereum time $t^* \geq T_1$, and pre_b is publicly known after Ethereum time t^*. If Alice enters the execution, Bob must have sent pre_c before Ethereum time T_0. Moreover, Alice sends pre_a to P1 at Ethereum time T_0 and $T_0 < T_1$. Therefore, pre_a, pre_b and pre_c are all publicly known at Ethereum time t^*. Thus, during Ethereum time $(t^*, t^* + \tau]$, any miner in $-\mathcal{B}$ will activate C2 if it wins a block. We say $\mathcal{B}$ loses the race if a non-colluding miner mines a new block during Ethereum time $(t^*, t^* + \tau]$. Otherwise, we say $\mathcal{B}$ wins the race. If $\mathcal{B}$ loses the race, it gets nothing from C1 or C2, and its utility is at most $\$\mathsf{BV}(₿x' - Ðx - Ðc_b)$ which can be achieved if P1' is activated. Else if $\mathcal{B}$ wins the race, then its utility is at most $\$\mathsf{BV}(₿x')$ which can be achieved by activating P2, C1 and P1'. Since $p \leq \gamma^\tau \leq \frac{Ðc_b}{Ðc_b + Ðx}$, we have

$$\$\mathsf{BV}((₿x' - Ðx - Ðc_b) \cdot (1-p) + ₿x' \cdot p) < \$\mathsf{BV}(₿x' - Ðx).$$

□

Theorem 5.5 (CSP fairness)**.** *Suppose that the hash function $H(\cdot)$ is a one-way function. For any $\gamma \in [0, 1]$, if the parameters satisfy the constraints specified in Section 5, then, the atomic swap protocol satisfies γ-CSP-fairness.*

Proof. In Lemma 5.2 and Lemma 5.4, we show that the atomic swap protocol satisfies γ-CSP-fairness when the coalition consists of Alice or Bob, and possibly with some miners. Because we assume that Alice and Bob are not in the same coalition, it remains to show γ-CSP-fairness when the coalition $\mathcal{C}$ consists only of miners controlling at most γ fraction of the mining power.

Henceforth, we assume Alice and Bob are both honest. It is clear from the protocol that the honest Alice and honest Bob always make the same decision whether to enter the execution phase or abort phase. We may assume that the coalition does not post any new smart contract on the fly and deposit money into it — if it did so, it cannot recover more than its deposit since any player not in $\mathcal{B}$ will not invoke the smart contract.

Next, when $\mathcal{C}$ follows the protocol, its utility is always zero. Suppose $\mathcal{C}$ may deviate from the protocol. Notice that the utility of $\mathcal{C}$ can be positive only when C2 or C2' is activated. There are two possible cases.

- *Case 1: both Alice and Bob enter the execution phase.* In this case, Alice always sends pre_a to P1, and she never sends any transaction containing pre'_a. Ignoring the negligible probability that $\mathcal{C}$ finds pre'_a by itself, C2' can never be activated. Moreover, Alice always sends pre_a to P1 at latest at Ethereum time T_0, and thus Bob will not post any transaction containing pre_b. Ignoring the negligible probability that $\mathcal{C}$ finds pre_b by itself, C2 can never be activated. To sum up, except the negligible probability, the utility of $\mathcal{C}$ is at most zero, which is the same as the honest case.

- *Case 2: both Alice and Bob enter the abort phase.* In this case, Alice never sends any transaction containing pre_a. Ignoring the negligible probability that $\mathcal{C}$ finds pre_a by itself, C2 can never be activated, and C2' can be activated only by (pre'_a, pre_b). However, Bob always sends _ to P2' and pre_b to P2 at Ethereum time T_0, so Alice never sends any transaction containing pre'_a. Ignoring the negligible probability that $\mathcal{C}$ finds pre'_a by itself, C2' cannot be activated by (pre'_a, pre_b). To sum up, except with negligible probability, the utility of $\mathcal{C}$ is at most zero, which is the same as the honest case.

□

Theorem 5.6 (Dropout resilience of atomic swap)**.** *Suppose that $H(\cdot)$ is a one-way function and that all players are PPT machines. Then, the atomic swap protocol is dropout resilient.*

Proof. Throughout the proof, for any $X \in \{pre_a, pre_b, pre_c, pre'_a\}$, we ignore the negligible probability that the miners can find the preimage X by itself if Alice and Bob have never sent X before.

We first analyze the cases where Alice drops out. There are three possible cases.

- *Case 1: Bob enters the abort phase.* In this case, Bob will send pre_b to P2 and _ to P2' at Ethereum time T_0. When τ Ethereum time has passed since P2 is activated, Bob sends _ to C1; when τ' Bitcoin time has passed since P2' is activated, Bob sends _ to C1'. When Bob enters the abort phase, he never sends any transaction containing pre_c, and thus Alice never enters the execution phase and never sends any transaction containing pre_a no matter when she drops out. Because Bob sends _ to P2' at Ethereum time T_0, Alice never sends any transaction containing pre'_a. Without knowing pre_a, pre_c and pre'_a, the miner cannot activate P1, C2, P1' and C2'.

  As long as $1/\mathsf{poly}(\lambda)$ fraction of the mining power is honest, P2, C1, P2' and C1' must be activated in polynomial time except with negligible probability, and Bob's utility is 0 since he simply gets all his deposit back.

- *Case 2: Bob enters the execution phase, and Alice sent pre_a before Ethereum time T_1.* In this case, Bob will send $pre'_b = pre_a$ to P1' at Ethereum time T_1 at latest. Moreover, Alice and Bob never send any transaction containing pre_b and pre'_a. Without knowing pre_b and pre'_a, the miner cannot activate P2, P2', C2 and C2'. If P2 and P2' are not activated, C1 and C1' cannot be activated either.

  As long as $1/\mathsf{poly}(\lambda)$ fraction of the mining power is honest, P1 and P1' must be activated in polynomial time except with negligible probability, and Bob's utility is $\$\mathsf{BV}(₿x' - Ξx) > 0$.

- *Case 3: Bob enters the execution phase, while Alice drops out before sending pre_a.* In this case, Bob will send pre_b to P2 at Ethereum time T_1. When τ Ethereum time has passed since P2 is activated, Bob sends _ to C1. As soon as C1 is activated, Bob sends _ to P2'. When τ' Bitcoin time has passed since P2' is activated, Bob sends _ to C1'. Without knowing pre_a and pre'_a, the miner cannot activate P1, C2, P1' and C2'.

  As long as $1/\mathsf{poly}(\lambda)$ fraction of the mining power is honest, P2, C1, P2' and C1' must be activated in polynomial time except with negligible probability, and Bob's utility is 0 since he simply gets all his deposit back.

Next, we analyze the case where Bob drops out. There are two cases.

- *Case 1: Alice enters the abort phase.* If Alice enters the abort phase, Bob must drop out before Ethereum time T_0, so Bob has not sent pre_b to P2. Then, Alice will send _ to P2 at Ethereum time T_0, and pre'_a to P2' at Ethereum time T_1. When τ Ethereum time has passed since P2 is activated, Alice sends _ to C1; when τ' Bitcoin time has passed since P2' is activated, Alice sends _ to C1'. If Alice enters the abort phase, she never sends any transaction containing pre_a. Without knowing pre_a and pre_b, the miner cannot activate P1, C2, P1' and C2'.

  As long as $1/\mathsf{poly}(\lambda)$ fraction of the mining power is honest, P2, C1, P2' and C1' must be activated in polynomial time except with negligible probability, and Alice's utility is 0 since she simply gets all her deposit back.

- *Case 2: Alice enters the execution phase.* In this case, Bob must have sent pre_c to P1. Alice will send pre_a to P1 before Ethereum time T_1, and thus Bob never sends any transaction containing pre_b. As soon as P1 is activated, she will send _ to P1'. In the execution phase, Alice never sends any transaction containing pre'_a. Without knowing pre_b and pre'_a, the miner cannot activate P2, P2', C2 and C2'. If P2 and P2' are not activated, C1 and C1' cannot be activated either.

  As long as $1/\mathsf{poly}(\lambda)$ fraction of the mining power is honest, P1 and P1' must be activated in polynomial time except with negligible probability, and Alice's utility is $\$\mathsf{AV}(\ddot{\mathrm{E}}x - \ddot{\mathrm{B}}x') > 0$.

□

6 Atomic Swap: Achieving Bounded Maximin Fairness

So far, we have seen how to achieve CSP fairness. However, when there are some external incentives, the strategic player may be encouraged to deviate from the protocol, and the honest player could be harmed. In this section, we specify how to modify the contracts, the protocol, and the parameters, so that we can also achieve bounded maximin fairness. Henceforth, let $\alpha \in [0, 1 - 1/\mathsf{poly}(\lambda)]$ denote the maximum fraction of mining power controlled by the set of externally incentived players, and let $\$E$ be an upper bound on any individual or coalition's valuation of the total possible external incentive.

6.1 Constructions

Parameter constraints. Compared to the constraints in Section 5, we further require Alice to put the collateral $\ddot{\mathrm{E}}c_a$ into Rapidash, so that Alice and Bob both put the collateral into Rapidash and Rapidash'. The choices of the time out T, T_0, T_1 on Ethereum and and T'_1 on Bitcoin subject to the same constraints as in Section 5.

Parameter Constraints for Atomic Swap

Constraints for Rapidash (on Ethereum):

- $h_a = H(pre_a)$, $h_b = H(pre_b)$ and $h_c = H(pre_c)$.
- $T_1 > T_0 > T > 0$.
- $\ddot{\mathrm{E}}\epsilon > \ddot{\mathrm{E}}0$, $\ddot{\mathrm{E}}c_a > \ddot{\mathrm{E}}\epsilon$, and $\ddot{\mathrm{E}}c_b > \ddot{\mathrm{E}}\epsilon$.
- $\$\mathsf{AV}(\ddot{\mathrm{E}}c_a) > \frac{\$\mathsf{AV}(\ddot{\mathrm{B}}x' + \alpha\ddot{\mathrm{E}}x) + \$E}{1-\alpha}$ and $\$\mathsf{BV}(\ddot{\mathrm{E}}c_b) > \frac{\$\mathsf{BV}(\ddot{\mathrm{B}}x' + \alpha\ddot{\mathrm{E}}x) + \$E}{1-\alpha}$

Constraints for Rapidash' (on Bitcoin):

- $h'_b = H(pre_a) = h_a$ and $h'_a = H(pre'_a)$.
- Bitcoin time $T'_1 >$ Ethereum time T_1, i.e., the Bitcoin block of length T'_1 is mined after the Ethereum block of length T_1.[a]
- $\ddot{\mathrm{B}}\epsilon' > \ddot{\mathrm{B}}0$, $\ddot{\mathrm{B}}c'_a > \ddot{\mathrm{B}}\epsilon'$, and $\ddot{\mathrm{B}}c'_b > \ddot{\mathrm{B}}\epsilon'$.
- $\$\mathsf{AV}(\ddot{\mathrm{B}}c'_a) > \frac{\$\mathsf{AV}(\ddot{\mathrm{E}}x + \alpha\ddot{\mathrm{B}}x') + \$E}{1-\alpha}$ and $\$\mathsf{BV}(\ddot{\mathrm{B}}c'_b) > \frac{\$\mathsf{BV}(\ddot{\mathrm{E}}x + \alpha\ddot{\mathrm{B}}x') + \$E}{1-\alpha}$.

Choice of timeouts:

- $\tau \geq 1$, $\tau' \geq 1$.

[a] In practice, this constraint should be respected except with negligible probability despite the the variance in inter-block times.

Intuitively, if the externally incentivized individual or coalition tries to take advantage of the honest player, the $1 - \alpha$ fraction of honest miners may trigger the bomb. Thus, the collateral amounts $\ddot{\mathrm{E}}c_a, \ddot{\mathrm{E}}c_b, \ddot{\mathrm{B}}c'_a, \ddot{\mathrm{B}}c'_b$ should be large enough relative to the mining power it controls so that a coalition involving Alice or Bob would never want to take any gamble that would risk getting their collateral (partially) burnt even if they are compensated by $\$E$.

Rapidash contract (on Ethereum)

/* parametrized with (h_a, h_b, h_c, T_1, τ, $\ddot{\mathrm{E}}x$, $\ddot{\mathrm{E}}c_b, \$\ddot{\mathrm{E}}\epsilon$)*/

Preparation phase: Bob deposits $\ddot{\mathrm{E}}x + \ddot{\mathrm{E}}c_b$, and Alice deposits $\ddot{\mathrm{E}}c_a$. Once both parties make the correct amount of deposits, the preparation phase ends and the execution phase starts.

Execution phase:

B1: On receiving _ from anyone, do nothing.

B2: On receiving pre_c from anyone P such that $H(pre_c) = h_c$, send $\ddot{\mathrm{E}}\epsilon$ to player P. All remaining coins are burnt.

P1: On receive pre_a from Alice such that $H(pre_a) = h_a$ and pre_c from Bob such that $H(pre_c) = h_c$, send $\ddot{\mathrm{E}}x + \ddot{\mathrm{E}}c_a$ to Alice and $\ddot{\mathrm{E}}c_b$ to Bob.

P2: Time T_1 or greater: On receive pre_b from Bob such that $H(pre_b) = h_b$ or on receiving _ from Alice, do nothing.

C1: At least τ after P2 is activated: on receiving _ from anyone, send $\ddot{\mathrm{E}}x + \ddot{\mathrm{E}}c_b$ to Bob and $\ddot{\mathrm{E}}c_a$ to Alice.

C2: On receive (pre_a, pre_b, pre_c) from anyone P such that $H(pre_a) = h_a$, $H(pre_b) = h_b$, and $H(pre_c) = h_c$ send $\ddot{\mathrm{E}}\epsilon$ to player P. All remaining coins are burnt.

Rapidash' contract (on Bitcoin)

/* parametrized with ($h'_b, h'_a, T'_1, \tau', \ddot{\mathrm{B}}x', \ddot{\mathrm{B}}c'_a, \$\ddot{\mathrm{B}}\epsilon'$)*/

Preparation phase: Alice deposits $\ddot{\mathrm{B}}x' + \ddot{\mathrm{B}}c'_a$, and Bob deposits $\ddot{\mathrm{B}}c'_b$. Once both parties make the correct amount of deposits, the preparation phase ends and the execution phase starts.

Execution phase:

A1': Time T'_1 or greater: on receiving _ from Alice or Bob, do nothing.

A2': On receiving pre'_a from anyone P such that $H(pre'_a) = h'_a$, or on receiving pre_b from anyone P such that $H(pre_b) = h_b$, send $\ddot{\mathrm{B}}\epsilon'$ to player P. All remaining coins are burnt.

P1': On receiving pre'_b from Bob[a] such that $H(pre'_b) = h'_b$ or on receiving _ from Alice, send $\ddot{\mathrm{B}}x' + \ddot{\mathrm{B}}c'_b$ to Bob and send $\ddot{\mathrm{B}}c'_a$ to Alice.

P2': Time T'_1 or greater: on receiving pre'_a from Alice such that $H(pre'_a) = h'_a$ or on receiving _ from Bob, do nothing.

C1': At least τ' after P2' is activated: on receiving _ from anyone, send ₿$x' +$ ₿c'_a to Alice and ₿c'_b to Bob.

C2': On receiving (pre'_b, pre'_a) from anyone P such that $(pre'_b) = h'_b$ and $H(pre'_a) = h'_a$, or on receiving (pre'_a, pre_b) from anyone P such that $H(pre'_a) = h'_a$ and $H(pre_b) = h_b$, send ₿ϵ' to player P. All remaining coins are burnt.

[a] Bob will let pre'_b be the pre_a he learns in the RAPIDASH instance.

Recall that the activation points of the same type are mutually exclusive. Once B1 has been activated, B2 cannot be activated. Thus, the purpose of activating B1 is to "defuse" the bomb of B2. Similarly, once A1' has been activated, A2' cannot be activated, so activating A1' is to defuse the bomb of A2'.

The atomic swap protocol. We describe the atomic swap protocol below.

- *Miner.* The miner's honest protocol is described below.
  - The miner watches all transactions posted to B1, B2, P1, P2, C1, C2, A1', A2', P1', P2', C1', and C2' (i.e., all the activation points for both contracts), to see if they contain a valid $pre_a = pre'_b$, pre_b, pre'_a, and pre_c.
  - If C1 has not been activated, as soon as the miner has observed pre_a, pre_b and pre_c, it posts (pre_a, pre_b, pre_c) to C2. Similarly, if C1' has not been activated, as soon as the miner has observed both pre'_a and pre'_b, it posts (pre'_a, pre'_b) to C2'; as soon as the miner has observed pre'_a and pre_b, it posts (pre'_a, pre_b) to C2'.
  - If B1 has not been activated, as soon as the miner has observed pre_c, it posts pre_c to B2. If A1' has not been activated, as soon as the miner has observed pre'_a or pre_b, it posts pre'_a or pre_b to A2'.
  - Whenever the miner mines a block, it always includes its own transactions ahead of others.
- *Alice and Bob.* Below, we define the honest protocol for Alice and Bob. The moment that both contracts have been posted and take effect is defined to be the start of the execution (i.e. $t = 0$). We define Ethereum time 0 and Bitcoin time 0 to be the length of Ethereum and Bitcoin when the execution starts, respectively.

Atomic Swap Protocol — Alice and Bob

Preparation Phase:

1. At $t = 0$, Alice sends the deposit transaction of ₿$x' +$ ₿c'_a to RAPIDASH' and sends the collateral transaction of Ξc_a to RAPIDASH. Similarly, Bob sends the deposit transaction of Ξ$x +$ Ξc_b to RAPIDASH and sends the collateral transaction of ₿c'_b to RAPIDASH'.
2. As soon as both RAPIDASH and RAPIDASH' enter the execution phase, Bob sends _ to B1. As soon as B1 is activated, Bob sends pre_c to P1 and enters the execution phase. As soon as Bob sent pre_c to P1, Alice enters to the execution phase.
3. At Ethereum time T, if either RAPIDASH or RAPIDASH' has not entered the execution phase, Alice and Bob go to the abort phase; otherwise, Bob sends _ to B1. During Ethereum time $[T, T_0)$, as soon as B1 is activated, Bob sends pre_c to P1.
4. At Ethereum time T_0, if Bob has not sent pre_c to P1, Alice and Bob go to the abort phase; otherwise, both parties to the execution phase.

Execution phase:

1. Alice sends pre_a to P1. As soon as P1 has been activated, Alice sends an empty message _ to P1'.
2. As soon as Alice sends pre_a to P1, Bob sends $pre'_b = pre_a$ to P1'. If Alice does not send pre_a to P1 before Ethereum time T_1, Bob sends _ to A1'.
3. If Alice does not send pre_a to P1 before Ethereum time T_1, Bob sends pre_b to P2 as soon as A1' is activated.
4. If τ *Ethereum time* has passed since P2 is activated, Alice and Bob send _ to C1. As soon as C1 is activated, Bob sends _ to P2'.
5. If τ' *Bitcoin time* has passed since P2' is activated, Alice and Bob send _ to C1'.

Abort Phase:

1. If Rapidash (Rapidash', resp.) has not entered the execution phase, Alice and Bob withdraw their deposits from Rapidash (Rapidash', resp.).
2. At Bitcoin time T'_0, Bob sends _ to P2' and _ to A1'; Alice sends _ to P2 and and _ to A1'.
3. If Bob has not sent _ to P2' by Bitcoin time T'_1, Alice waits until A1' is activated and sends pre'_a to P2'. Similarly, if Alice has not sent _ to P2 by Bitcoin time T'_1, Bob waits until A1' is activated and sends pre_b to P2.
4. If τ' *Bitcoin time* has passed since P2' is activated, Alice and Bob send _ to C1'; similarly, if τ *Ethereum time* has passed since P2 is activated, Alice and Bob send _ to C1.

Observe that when Alice and Bob are both honest, Bob will send _ to B1 immediately when both Rapidash and Rapidash' enter the execution phase, and send pre_c to P1 as soon as B1 is activated. Then, Alice will post pre_a to P1 immediately and then Bob will learn pre_a and post it to P1' immediately. Therefore, both players get their desired cryptocurrency and all their collateral back as soon as a new block is confirmed on both Ethereum and Bitcoin — in this sense, the protocol satisfies optimistic responsiveness.

Theorem 6.1 shows that the above modified version of the atomic swap protocol still satisfies CSP fairness absent external incentives.

Theorem 6.1 (CSP fairness)**.** *Suppose that the hash function $H(\cdot)$ is a one-way function. Suppose the choice of the parameters satisfy the constraints in Section 6.1, and further satisfy $\gamma^{\tau'} \leq \frac{\ddot{B}c'_a}{\ddot{B}c'_a + \ddot{B}x'}$ and $\gamma^{\tau} \leq \frac{\ddot{E}c_b}{\ddot{E}c_b + \ddot{E}x}$. Then, the atomic swap protocol satisfies γ-CSP-fairness.*

Proof. Deferred to Appendix A. □

Next, we show that the atomic swap protocol satisfies bounded maximin fairness. Formally, we prove the following theorem.

Theorem 6.2 (Bounded maximin fairness)**.** *Suppose that $H(\cdot)$ is a one-way function and suppose the parameters $\ddot{E}c_a, \ddot{B}c'_a, \ddot{E}c_b, \ddot{B}c'_b, \alpha \in [0, 1 - 1/\mathsf{poly}(\lambda)]$ satisfy the relations specified in Section 6.1. Then, the atomic swap protocol satisfies α-bounded maximin fairness against external incentives. In other words, for any set of PPT players denoted $\mathcal{C}$ without external incentives, and any externally incentivized PPT coalition $\mathcal{C}'$ that is disjoint from $\mathcal{C}$, controlling at most α fraction of mining power,*

and playing any strategy $S_{\mathcal{C}'} \in \mathcal{R}$, there is a negligible function $\mathsf{negl}(\cdot)$ *such that except with* $\mathsf{negl}(\lambda)$ *probability, it must be that*

$$\mathsf{util}^{\mathcal{C}}(HS_{\mathcal{C}}, S_{\mathcal{C}'}, HS_{\mathcal{D}}) \geq 0$$

where $\mathcal{D}$ denotes all players not in $\mathcal{C} \cup \mathcal{C}'$.

The entire Section 6.2 dedicates to proving Theorem 6.2.

6.2 Proof for Bounded Maximin Fairness

Proof Roadmap. Conceptually, because Alice and Bob both put the collateral on both RAPIDASH and RAPIDASH', none of them wants to trigger any of the bomb (B2, C2, A2' and C2'). In Section 6.2.1, we analyze players' utilities when the bomb is triggered. In Section 6.2.2, we define a set of "bad events" that leads to the activation of the bomb. The bad events are defined such that if any of the bad events is about to happen, it must be that a strategic invidual or coalition is about to send a transaction that does not follow from the honest protocol. In Section 6.2.3, we show that whenever the Bob-miner coalition $\mathcal{B}$ is about to send a message that makes a bad event happen, $\mathcal{B}$'s expected utility can be strictly improved if $\mathcal{B}$ simply stops sending any messages (including the message that is about to trigger the bad event) from that moment on (Lemma 6.5). Hence, any strategy that makes the bad event happen is a blatantly non-rational strategy for $\mathcal{B}$, so a rational player would never make the bad events happen. Then, we show that as long as none of the bad events happens, the honest Alice's utility is never negative (Lemma 6.4 and Lemma 6.6). A similar argument can be made for the case of an Alice-miner coalition (see Section 6.2.4). Finally, in Section 6.2.5, we combine all the arguments above, and prove that the atomic swap protocol achieves bounded maximin fairness.

6.2.1 Utilities When Bombs Are Triggered

Normally, when Alice and Bob follow the protocol, P1 and P1' will be activated, and they exchange the coins successfully. However, if one of the parties drops out, the other party will trigger (P2 + C1) and (P2' + C1') to get refunded. Finally, B2, C2, A2' and C2' are the bombs, and both Alice and Bob lose their collateral when a bomb is triggered. We define the following events.

- Normal: P1 is activated.
- Refund: either (P2 + C1) are activated, or one of Alice and Bob withdraws their deposits from RAPIDASH successfully before RAPIDASH enters the execution phase.
- Bomb: B2 or C2 is activated.
- Normal': P1' is activated.
- Refund': either (P2' + C1') are activated, or one of Alice and Bob withdraws their deposits from RAPIDASH' successfully before RAPIDASH' enters the execution phase.
- Bomb': A2' or C2' is activated.

Lemma 6.3. *Suppose the coalition $\mathcal{B}$ consists of Bob and possibly some miners. Let $\$E$ be an upper bound on $\mathcal{B}$'s valuation of the total possible external incentive. If $\text{Ł}c_b > \text{Ł}\epsilon$, $\text{Ƀ}c'_b > \text{Ƀ}\epsilon'$, then, the following statements hold.*

- *The utility of $\mathcal{B}$ is at most* $\$\mathsf{BV}(\text{Ƀ}x') + \$E$.

- *If* Bomb *is activated by an honest miner* $\notin \mathcal{B}$*, the utility of* $\mathcal{B}$ *is at most* $\$\mathsf{BV}(\text{₿}x' - \text{Ξ}x - \text{Ξ}c_b) + \$E$.
- *If* Bomb′ *is activated by an honest miner* $\notin \mathcal{B}$*, the utility of* $\mathcal{B}$ *is at most* $\$\mathsf{BV}(-\text{₿}c'_b) + \$E$.

Similarly, suppose the coalition $\mathcal{A}$ *consists of Alice and possibly some miners. Let* $\$E$ *be an upper bound on* $\mathcal{A}$*'s valuation of the total possible external incentive. If* $\text{Ξ}c_a > \text{Ξ}\epsilon$, $\text{₿}c'_a > \text{₿}\epsilon'$, *then, the following statements hold.*

- *The utility of* $\mathcal{A}$ *is at most* $\$\mathsf{AV}(\text{Ξ}x) + \$E$.
- *If* Bomb *is activated by an honest miner* $\notin \mathcal{A}$*, the utility of* $\mathcal{A}$ *is at most* $\$\mathsf{AV}(-\text{Ξ}c_a) + \$E$.
- *If* Bomb′ *is activated by an honest miner* $\notin \mathcal{A}$*, the utility of* $\mathcal{A}$ *is at most* $\$\mathsf{AV}(\text{Ξ}x - \text{₿}x' - \text{₿}c'_a) + \$E$.

Proof. First, consider the coalition $\mathcal{B}$ consisting of Bob and possibly some miners. Note that since $\text{Ξ}c_b > \text{Ξ}\epsilon$ the best option for Bob when considering Rapidash separately (not considering how this potentially affects Rapidash') is to obtain his money back via Refund (i.e., his utility in Rapidash is zero). In Rapidash', as $\text{₿}c'_b > \text{₿}\epsilon'$, Bob can obtain at most $\text{₿}x'$ (via Normal′). Thus, besides the external incentive, Bob can obtain at most $\text{₿}x'$ from both contracts together.

Next, if Bomb is activated by an honest miner $\notin \mathcal{B}$, Bob loses all money he desposited into Rapidash, i.e., $\text{Ξ}x + \text{Ξ}c_b$. Thus, besides the external incentive, in total Bob can obtain at most $\text{₿}x' - \text{Ξ}x - \text{Ξ}c_b$ from both contracts (again using Normal′ in Rapidash').

Then, if Bomb′ is activated by an honest miner $\notin \mathcal{B}$, Bob loses all money he deposited into Rapidash', i.e., $\text{₿}c'_b$. Thus, besides the external incentive, Bob's utility is at most $-\text{₿}c'_b$ in this case (using Refund in Rapidash).

Now, consider the coalition $\mathcal{A}$ consisting of Alice and possibly some miners. Since $\text{Ξ}c_a > \text{Ξ}\epsilon$, Alice can obtain at most $\text{Ξ}x$ in Rapidash (via Normal). In Rapidash', as $\text{₿}c'_a > \text{₿}\epsilon'$, the best option for Alice is to obtain her money back via Refund′. Thus, besides the external incentive, Alice can obtain at most $\text{Ξ}x$ from both contracts together.

If Bomb is activated by an honest miner $\notin \mathcal{A}$, Alice loses all money she deposited into Rapidash, i.e., $\text{Ξ}c_a$. Thus, besides the external incentive, in total Alice's utility is at most $-\text{Ξ}c_a$.

Finally, if Bomb′ is activated by an honest miner $\notin \mathcal{A}$, Alice loses all money she deposited into Rapidash', i.e., $\text{₿}x' + \text{₿}c'_a$. Thus, besides the external incentive, in total Alice can obtain at most $\text{Ξ}x - \text{₿}x' - \text{₿}c'_a$ from both protocols. □

6.2.2 Non-Rational Strategies

Terminologies. We define a family of PPT strategies denoted $\overline{\mathcal{R}}$, and we will show given any strategy $S \in \overline{\mathcal{R}}$, we can give a simple modification of S, resulting in a new PPT strategy which makes the externally incentivized coalition better off — in this sense, the strategy space $\overline{\mathcal{R}}$ is blatantly non-rational.

Consider an externally incentivized coalition Alice-miner $\mathcal{A}$ (the case for Bob-miner coalition $\mathcal{B}$ is similarly defined). Given an activation point X, we say that X is *guaranteed to be activated* at some time t, iff either X was already activated before t, or a colluding miner has been chosen as the winning miner at time t, and it activates X in the new block it mines. We say Rapidash is *guaranteed to enter the execution phase* at some time t, iff either Rapidash was already entered the execution phase before t, or a colluding miner has been chosen as the winning miner at time t, and it includes Alice's and Bob's deposit transactions for Rapidash in the new block it mines. The

case of Rapidash' is defined similarly. We say Alice is *guaranteed to withdraw her deposit from* Rapidash at some time t, iff either Alice already withdrew her deposit from Rapidash before t, or all the following conditions hold.

- Rapidash is still in the preparation phase before time t.
- At time t, a colluding miner is chosen as the winning miner, and Alice's withdrawal transaction is included in the block at time t.

The cases that Alice is guaranteed to withdraw her deposit from Rapidash', and Bob is guaranteed to withdraw his deposit from Rapidash or Rapidash' are defined similarly.

Non-rational strategies. The set $\overline{\mathcal{R}}$ of non-rational strategies for the externally incentivized Bob-miner coalition (including Bob alone) $\mathcal{B}$ is the set of strategies such that with non-negligible probability, any of the following happens:

$\mathbf{E}_1$: Before B1 is guaranteed to be activated and before Bob is guaranteed to withdraw his deposit from Rapidash, anyone in $\mathcal{B}$ sends any transaction containing pre_c to (B2, P1 or C2) and the deposit transaction to Rapidash.

$\mathbf{E}_2$: Rapidash and Rapidash' are guaranteed to enter the execution phase before Ethereum time T. Additionally, anyone in $\mathcal{B}$ sends pre_c to (B2, P1 or C2) before Ethereum time T_0, and sends pre_b to (P2, C2, A2' or C2') before (P1 or C1) is guaranteed to be activated.

$\mathbf{E}_3$: Before A1' is guaranteed to be activated and before Bob is guaranteed to withdraw his deposit from Rapidash', anyone in $\mathcal{B}$ sends any transaction containing pre_b to (P2, C2, A2' or C2') and the deposit transaction to Rapidash'.

$\mathbf{E}_4$: Alice enters the abort phase, and Bob does not send _ to P2' before Bitcoin time T_1'. Additionally, at Bitcoin time T_1' or later, anyone in $\mathcal{B}$ sends any transaction containing pre_b to (P2, C2, A2' or C2') before (P1' or C1') is guaranteed to be activated.

The set $\overline{\mathcal{R}}$ of non-rational strategies for the externally incentivized Alice-miner coalition (including Alice alone) $\mathcal{A}$ is the set of strategies such that with non-negligible probability, any of the following happens:

$\mathbf{E}_5$: Bob sends pre_c to P1 before Ethereum time T_0, and Alice does not send pre_a to P1 until Ethereum time T_1. However, at Ethereum time T_1 or later, anyone in $\mathcal{A}$ sends a transaction containing pre_a to (P1, C2, P1' or C2') before (P1 or C1) is guaranteed to be activated.

$\mathbf{E}_6$: Before A1' is guaranteed to be activated and before Alice is guaranteed to withdraw her deposit from Rapidash', anyone in $\mathcal{A}$ sends any transaction containing pre_a' to (A2', P2' or C2') and the deposit transaction to Rapidash'.

$\mathbf{E}_7$: Anyone in $\mathcal{A}$ sends any transaction containing pre_a to (P1, C2, P1' or C2') and any transaction containing pre_a' to (A2', P2' or C2') before (P1' or C1') is guaranteed to be activated.

$\mathbf{E}_8$: Any one of the conditions holds.

- Bob enters the execution phase, and Alice does not send pre_a to P1 before Ethereum time T_1. Additionally, at Bitcoin time T_1' or later, anyone in $\mathcal{A}$ sends any transaction containing pre_a' to (A2', P2' or C2') before (P1' or C1') is guaranteed to be activated.
- Bob enters the abort phase, and Alice does not send _ to P2 before Bitcoin time T_1'. Additionally, at Bitcoin time T_1' or later, anyone in $\mathcal{A}$ sends any transaction containing pre_a' to (A2', P2' or C2') before (P1' or C1') is guaranteed to be activated.

6.2.3 Against Externally Incentivized Bob-Miner Coalition

Lemma 6.4. *Let $\mathcal{B}$ be the coalition consisting of Bob and miners controlling no more than α fraction of the mining power where $\alpha \in [0, 1 - 1/\mathsf{poly}(\lambda)]$. Suppose Alice and at least $1 - \alpha$ fraction of mining power are honest. For any PPT strategy by $\mathcal{B}$, except with negligible probability, as long as none of $\mathbf{E}_1, \mathbf{E}_2, \mathbf{E}_3, \mathbf{E}_4$ happens, then, one and only one of the following statement holds.*

1. Normal *and* Normal′ *happen in polynomial time.*
2. Normal *and* Refund′ *happen in polynomial time.*
3. Refund *and* Refund′ *happen in polynomial time.*

Proof. First, we are going to show that one of Normal and Refund will happen. There are two cases.

- *Case 1: Alice enters the execution phase.*

  We are going to show that P1 must be activated in polynomial time except with negligible probability. Once P1 is activated, Normal happens. In the execution phase, Alice always sends pre_a to P1. If B2, P2 and C2 are not activated, the honest miners will include Alice's transaction, pre_a to P1, once they mine a block. Thus, it suffices to show that B2, P2, C2 (all activations points that P1 is mutually exclusive with) cannot be activated except with negligible probability. First, because $\mathbf{E}_1$ does not happen, $\mathcal{B}$ never sends pre_c before B1 is guaranteed to be activated. Thus, B2 can never be activated.

  Next, because Alice enters the execution phase, Rapidash and Rapidash' must enter the execution phase, and Bob already sent pre_c to P1 before Ethereum time T_0. Because $\mathbf{E}_2$ does not happen, no one in $\mathcal{B}$ sends pre_b to C2 before P1 or C1 is guaranteed to be activated. Thus, C2 cannot be activated.

  It remains to show that P2 cannot be activated. In the execution phase, Alice never sends _ to P2, so P2 can be activated only if Bob sends pre_b to P2. Because $\mathbf{E}_2$ does not happen, Bob never sends pre_b to P2 before P1 or C1 is guaranteed to be activated. If P1 is guaranteed to be activated, P2 cannot be activated as they are mutually exclusive. On the other hand, if C1 is guaranteed to be activated, P2 must have been activated $\tau \geq 1$ Ethereum time before. Thus, by the time P2 is activated, C1 has not been guaranteed to be activated. However, and Bob never sends pre_b to P2 before C1 is guaranteed to be activated. Therefore, P2 can never be activated.

- *Case 2: Alice enters the abort phase.* In this case, Alice will send the withdrawal transaction to Rapidash, and _ to P2. Notice that when Rapidash is still in the preparation phase, the honest miner will include Alice's withdrawal transactions once they mine a block. Thus, except with negligible probability, in polynomial time, either Rapidash enters the execution phase, or Alice successfully withdraws her deposit from Rapidash. If Alice withdraws her deposit from Rapidash, Refund happens.

  Henceforth, we assume Rapidash enters the execution phase. Because $\mathbf{E}_1$ does not happen, $\mathcal{B}$ never sends pre_c before B1 is guaranteed to be activated. Thus, B2 can never be activated. Then, notice that Alice never sends pre_a when she enters the abort phase. Ignoring the negligible probability that $\mathcal{B}$ finds pre_a by itself, C2 can never be activated.

  Thus, since $1 - \alpha$ fraction of mining power is honest, either P1 or P2 will be activated in polynomial time. If P1 is activated, Normal happens. If P2 is activated, Alice will send _ to C1 when τ Ethereum time has passed since P2 is activated. Again, the honest miner will

include Alice's transaction _ to C1, once they mine a block. Thus, Refund will happen in polynomial time.

Next, we are going to show that one of Normal′ and Refund′ will happen. There are two cases.

- *Case 1: Alice enters the execution phase.* As we have shown, when Alice enters the execution phase, P1 must be activated in polynomial time. Then, Alice will send _ to P1' as soon as P1 is activated. Because $\mathbf{E}_3$ does not happen and Alice never sends pre'_a when in the execution phase, A2' cannot be activated. Ignoring the negligible probability that $\mathcal{B}$ finds pre'_a by itself, C2' cannot be activated. Because $1-\alpha$ fraction of mining power is honest, either P1' or P2' will be activated in polynomial time. If P1' is activated, Normal′ happens, If P2' is activated, Alice will send _ to C1'. Again, the honest miner will include a transaction _ to C1', once they mine a block. Thus, C1' will be activated in polynomial time, and Refund′ happens.

- *Case 2: Alice enters the abort phase.* In this case, Alice will send the withdrawal transaction to Rapidash', and _ to A1'. If Rapidash' is still in the preparation phase, the honest miner will include Alice's withdrawal transactions once they mine a block. Thus, in polynomial time, either Rapidash' enters the execution phase, or Alice successfully withdraws her deposit from Rapidash', and so Refund′ happens.

  Henceforth, we assume Rapidash' enters the execution phase in polynomial time. Because $\mathbf{E}_3$ does not happen, $\mathcal{B}$ never sends pre_b before A1' is guaranteed to be activated. As A1' and A2' are mutually exclusive, A2' can not be activated via pre_b. Since Alice sends pre'_a to P2' only after A1' is activated, 1) A2' can not be activated via pre'_a, and 2) C2' can not happen before A1' is activated. If either P1' or C1' are activated, Normal′ or Refund′ happens. Otherwise, the honest miner will include the transaction _ to A1' once they mine a block. Because $1-\alpha$ fraction of mining power is honest, A1' will be activated in polynomial time.

  When Alice is in abort phase, either Bob sends _ to P2' before Bitcoin time T'_1 or Alice sends pre'_a to P2' when A1' is activated. If Bob sends _ to P2' before Bitcoin time T'_1, Alice never sends pre'_a, and thus C2' can not be activated. When Alice enters the abort phase, she never sends pre_a, and thus P1' can not be activated. As we showed before, A2' can not be activated either, and thus the honest miner will include Bob's transaction, _ to P2', once they mine a block, so P2' will be activated in polynomial time.

  On the other hand, suppose Bob does not send _ to P2' before Bitcoin time T'_1, so Alice sends pre'_a to P2'. Because $\mathbf{E}_4$ does not happen, $\mathcal{B}$ never sends pre_b before P1' or C1' is guaranteed to be activated. If P1' is activated, Normal′ happens, if C1' is activated, P2' must have been activated previously, and thus Refund′ happens. Otherwise, recall that Alice never sends pre_a in the abort phase. Without pre_a and pre_b, P1' and C2' cannot be activated, and as A2' can not be activated either as shown before, the honest miner will include Alice's transaction, pre'_a to P2', once they mine a block. Thus, P2' will again be activated in polynomial time.

  Then, Alice will send _ to C1' when τ' Bitcoin time has passed since P2' is activated. Again, the honest miner will include Alice's transaction, _ to C1', once they mine a block. Thus, Refund′ will happen in polynomial time.

So far, we have shown one of Normal and Refund will happen and one of Normal′ and Refund′ will happen. To prove the lemma statement, it suffices to show that Refund and Normal′ never happen simultaneously. For the sake of reaching a contradiction, suppose Refund and Normal′ happen. Event Normal′ happens implies P1' is activated. Notice that P1' can be activated only when $pre'_b = pre_a$ is given. Ignoring the negligible probability that $\mathcal{B}$ can find pre_a by itself, Alice

must enter the execution phase and send pre_a to P1 at Ethereum time T_0. On the other hand, event Refund happens implies either (P2 + C1) are activated or Bob withdraws his deposit from Rapidash. Because Alice enters the execution phase, both Rapidash and Rapidash' must also enter the execution, and thus we exclude that Bob withdraws his deposit from Rapidash. Thus, event Refund happens only when (P2 + C1) are activated. Moreover, because Alice enters the execution phase, Bob must have sent pre_c to P1. Therefore, to activate P2, Bob has to send pre_b before C1 is guaranteed to be activated, which implies event $\mathbf{E}_2$ happens.

□

Lemma 6.5 (Blatant non-rationality of $\overline{\mathcal{R}}$ for Bob-miner coalition)**.** *Suppose that the parameter constraints in Section 6.1 hold and that the coalition $\mathcal{B}$ consists of Bob and miners controlling no more than α fraction of the mining power where $\alpha \in [0, 1 - 1/\mathsf{poly}(\lambda)]$. Given any PPT strategy $S_{\mathcal{B}} \in \overline{\mathcal{R}}$ for some (externally incentivized) coalition $\mathcal{B}$, there is a PPT strategy $\widehat{S}_{\mathcal{B}}$ such that*

$$\mathsf{util}^{\mathcal{B}}(\widehat{S}_{\mathcal{B}}, HS_{-\mathcal{B}}) > \mathsf{util}^{\mathcal{B}}(S_{\mathcal{B}}, HS_{-\mathcal{B}}).$$

Proof. Suppose the coalition $\mathcal{B}$ adopts a strategy in which $\mathbf{E}_1, \mathbf{E}_2, \mathbf{E}_3$ or $\mathbf{E}_4$ happens with non-negligible probability. We can construct a new strategy for $\mathcal{B}$ with strictly better expected utility. Specifically, consider a modified PPT strategy denoted $\widehat{S}_{\mathcal{B}}$: whenever by the original strategy, the first of $\mathbf{E}_1, \mathbf{E}_2, \mathbf{E}_3$ or $\mathbf{E}_4$ is about to happen, $\mathcal{B}$ simply stops sending any messages (including the message that is about to trigger $\mathbf{E}_1, \mathbf{E}_2, \mathbf{E}_3$ or $\mathbf{E}_4$) to the contract from that moment on.

Now, notice that by definition, when $\mathbf{E}_1$ or $\mathbf{E}_3$ happens, it must due to a message sent by $\mathcal{B}$. According to the protocol, honest Alice always enters the execution or the abort phase no later than Ethereum time T_0. Because Bitcoin time T_1' is later than Ethereum time T_0 by the choice of the parameters, when $\mathbf{E}_4$ happens, it must also due to a message sent by $\mathcal{B}$. Next, if Rapidash' has not been guaranteed to enter the execution phase, A1' cannot be guaranteed to be activated. Similarly, if Rapidash has not been guaranteed to enter the execution phase, B1 cannot be guaranteed to be activated. Thus, if anyone in $\mathcal{B}$ sends pre_c to (B2, P1 or C2) and sends pre_b to (P2, C2, A2' or C2') before Rapidash and Rapidash' both are guaranteed to enter the execution phase, one of $\mathbf{E}_1$ and $\mathbf{E}_3$ must happen. Consequently, if $\mathbf{E}_1$ and $\mathbf{E}_3$ do not happen while $\mathbf{E}_2$ happens, it must due to a message sent by $\mathcal{B}$. Thus, as long as $\mathcal{B}$ stops sending any messages before the first of $\mathbf{E}_1, \mathbf{E}_2, \mathbf{E}_3$ or $\mathbf{E}_4$ is about to happen, none of $\mathbf{E}_1, \mathbf{E}_2, \mathbf{E}_3$ and $\mathbf{E}_4$ can happen in the future. By Lemma 6.4, in polynomial time, one of (Normal+Normal′), (Normal+Refund′) and (Refund+Refund′) must happen. By direct calculation, we have the following table. Among (Normal + Normal′), (Normal + Refund′) and (Refund + Refund′), $\mathcal{B}$'s utility is at least $\$\mathsf{BV}(-\ddot{\mathrm{E}}x)$. In other words, we have $\mathsf{util}^{\mathcal{B}}(\widehat{S}_{\mathcal{B}}, HS_{-\mathcal{B}}) \geq \$\mathsf{BV}(-\ddot{\mathrm{E}}x)$. Thus, to show $\mathsf{util}^{\mathcal{B}}(\widehat{S}_{\mathcal{B}}, HS_{-\mathcal{B}}) \geq \mathsf{util}^{\mathcal{B}}(S_{\mathcal{B}}, HS_{-\mathcal{B}})$, we only need to show $\mathsf{util}^{\mathcal{B}}(S_{\mathcal{B}}, HS_{-\mathcal{B}}) < \$\mathsf{BV}(-\ddot{\mathrm{E}}x)$.

	Normal	Refund	Bomb
Normal′	$-\ddot{\mathrm{E}}x + \ddot{\mathrm{B}}x'$	$\ddot{\mathrm{B}}x'$	$-\ddot{\mathrm{E}}x - \ddot{\mathrm{E}}c_b + \ddot{\mathrm{B}}x'$
Refund′	$-\ddot{\mathrm{E}}x$	0	$-\ddot{\mathrm{E}}x - \ddot{\mathrm{E}}c_b$
Bomb′	$-\ddot{\mathrm{E}}x - \ddot{\mathrm{B}}c_b'$	$-\ddot{\mathrm{B}}c_b'$	$-\ddot{\mathrm{E}}x - \ddot{\mathrm{E}}c_b - \ddot{\mathrm{B}}c_b'$

Table 3: Bob's net profit under all possible cases.

We consider four cases, depending on whether $\mathbf{E}_1, \mathbf{E}_2, \mathbf{E}_3$ or $\mathbf{E}_4$ happens first in the original strategy $S_{\mathcal{B}}$.

Event $\mathbf{E}_1$ happens first. Now, consider some strategy $S_1 \in \overline{\mathcal{R}}$ with non-negligible probability that $\mathbf{E}_1$ happens. When $\mathbf{E}_1$ happens, because Bob is not guaranteed to withdraw his deposit from

RAPIDASH, the $1-\alpha$ fraction of honest miners would send the deposit to RAPIDASH and pre_c to B2 (and potentially C2 as well if pre_a and pre_b are available), if they are chosen to mine a block. By Lemma 6.3, if B2 or C2 is activated by an honest miner, the utility of $\mathcal{B}$ is at most $\$\mathsf{BV}(\text{₿}x' - \text{Ɇ}x - \text{Ɇ}c_b) + \$E$. On the other hand, if neither B2 nor C2 is activated by an honest miner, the utility of $\mathcal{B}$ is at most $\$\mathsf{BV}(\text{₿}x') + \$E$. When $\mathbf{E}_1$ happens, the probability that B2 or C2 is activated by an honest miner is at least $1-\alpha$. Thus, the utility of $\mathcal{B}$ is at most

$$(1-\alpha)(\$\mathsf{BV}(\text{₿}x' - \text{Ɇ}x - \text{Ɇ}c_b) + \$E) + \alpha(\$\mathsf{BV}(\text{₿}x') + \$E) < \$\mathsf{BV}(-\text{Ɇ}x),$$

where the inequality arises from the fact that $\$\mathsf{BV}(\text{Ɇ}c_b) > \frac{\$\mathsf{BV}(\text{₿}x' + \alpha\text{Ɇ}x) + \$E}{1-\alpha}$.

Event $\mathbf{E}_2$ happens first. Now, consider some strategy $S_2 \in \overline{\mathcal{R}}$ with non-negligible probability that $\mathbf{E}_2$ happens. When $\mathbf{E}_2$ happens, because both RAPIDASH and RAPIDASH' enter the execution phase before Ethereum time T and Bob sends pre_c before Ethereum time T_0, Alice enters the execution phase. In this case, Alice always sends pre_a to P1 at Ethereum time T_0. However, Bob also sends pre_b before C1 is guaranteed to be activated. Thus, the $1-\alpha$ fraction of honest miners would send (pre_a, pre_b, pre_c) to C2 (and potentially send pre_c to B2), if they are chosen to mine a block. By Lemma 6.3, if C2 or B2 is activated by an honest miner, the utility of $\mathcal{B}$ is at most $\$\mathsf{BV}(\text{₿}x' - \text{Ɇ}x - \text{Ɇ}c_b) + \$E$. On the other hand, if neither C2 nor B2 is activated by an honest miner, the utility of $\mathcal{B}$ is at most $\$\mathsf{BV}(\text{₿}x') + \$E$. When $\mathbf{E}_2$ happens, the probability that C2 or B2 is activated by an honest miner is at least $1-\alpha$. Thus, by the same calculation as the previous case, the utility of $\mathcal{B}$ is strictly less than $\$\mathsf{BV}(-\text{Ɇ}x)$.

Event $\mathbf{E}_3$ happens first. Now, consider some strategy $S_3 \in \overline{\mathcal{R}}$ with non-negligible probability that $\mathbf{E}_3$ happens. When $\mathbf{E}_3$ happens, because Bob is not guaranteed to withdraw his deposit from RAPIDASH', the $1-\alpha$ fraction of honest miners would send the deposit to RAPIDASH' and pre_b to A2' (and to C2' if pre'_a is available as well), if they are chosen to mine a block. By Lemma 6.3, if A2' or C2' is activated by an honest miner, the utility of $\mathcal{B}$ is at most $\$\mathsf{BV}(-\text{₿}c'_b) + \$E$. On the other hand, if neither A2' nor C2' is activated by an honest miner, the utility of $\mathcal{B}$ is at most $\$\mathsf{BV}(\text{₿}x') + \$E$. When $\mathbf{E}_3$ happens, the probability that A2' or C2' is activated by an honest miner is at least $1-\alpha$. Thus, the utility of $\mathcal{B}$ is at most

$$(1-\alpha)(\$\mathsf{BV}(-\text{₿}c'_b) + \$E) + \alpha(\$\mathsf{BV}(\text{₿}x') + \$E) < \$\mathsf{BV}(-\text{Ɇ}x),$$

where the inequality arises from the fact that $\$\mathsf{BV}(\text{₿}c'_b) > \frac{\$\mathsf{BV}(\text{Ɇ}x + \alpha\text{₿}x') + \$E}{1-\alpha}$.

Event $\mathbf{E}_4$ happens first. Now, consider some strategy $S_4 \in \overline{\mathcal{R}}$ with non-negligible probability that $\mathbf{E}_4$ happens. When $\mathbf{E}_4$ happens, because Alice enters the abort phase and Bob does not send _ to P2' before Bitcoin time T'_1, Alice will send pre'_a as soon as A1' is activated. However, Bob also sends pre_b before P1' or C1' is guaranteed to be activated. If Bob sends pre_b before A1' is activated, the $1-\alpha$ fraction of honest miners would send pre_b to A2', if they are chosen to mine a block. If Bob sends pre_b after A1' is activated, the $1-\alpha$ fraction of honest miners would send (pre'_a, pre_b) to C2', if they are chosen to mine a block. By Lemma 6.3, if A2' or C2' is activated by an honest miner, the utility of $\mathcal{B}$ is at most $\$\mathsf{BV}(-\text{₿}c'_b) + \$E$. On the other hand, if neither A2' nor C2' is activated by an honest miner, the utility of $\mathcal{B}$ is at most $\$\mathsf{BV}(\text{₿}x') + \$E$. When $\mathbf{E}_4$ happens, the probability that A2' or C2' is activated by an honest miner is at least $1-\alpha$. Thus, by the same calculation as the previous case, the utility of $\mathcal{B}$ is strictly less than $\$\mathsf{BV}(-\text{Ɇ}x)$.

Finally, notice that the above analysis holds even if $\mathcal{B}$ may post a new contract on the fly during the protocol execution, since all other players are honest and will not deposit money into the new contract. □

Lemma 6.6 (Against Externally Incentivized Bob-Miner Coalition)**.** *Suppose that the hash function $H(\cdot)$ is a one-way function. Let $\mathcal{C}$ be a group consisting of Alice and possibly any subset of the miners, and let $\mathcal{B}$ be a disjoint coalition consisting of Bob and at most α fraction of the mining power where $\alpha \in [0, 1 - 1/\mathsf{poly}(\lambda)]$. Suppose that $\mathcal{C}$ does not have external incentives but $\mathcal{B}$ may have up to $\$E$ amount of external incentives. Let $S_{\mathcal{B}}$ be an arbitrary PPT strategy of $\mathcal{B}$ that is not in $\overline{\mathcal{R}}$. Then, there exists a negligible function $\mathsf{negl}(\cdot)$ such that except with negligible probability $\mathsf{negl}(\lambda)$, it holds that*

$$\mathsf{util}^{\mathcal{C}}(HS_{\mathcal{C}}, S_{\mathcal{B}}, HS_{\mathcal{D}}) \geq 0,$$

where $\mathcal{D}$ denotes everyone else not in $\mathcal{C} \cup \mathcal{B}$.

Proof. By Lemma 6.5, any strategy that makes one of $\mathbf{E}_1, \mathbf{E}_2, \mathbf{E}_3, \mathbf{E}_4$ happen is blatantly non-rational. By direct calculation, we have the following table. By Lemma 6.4, if none of $\mathbf{E}_1, \mathbf{E}_2, \mathbf{E}_3, \mathbf{E}_4$

	Normal	Refund	Bomb
Normal′	$\ddot{\mathbb{E}}x - \ddot{\mathbb{B}}x'$	$-\ddot{\mathbb{B}}x'$	$-\ddot{\mathbb{E}}c_a - \ddot{\mathbb{B}}x'$
Refund′	$\ddot{\mathbb{E}}x$	0	$-\ddot{\mathbb{E}}c_a$
Bomb′	$\ddot{\mathbb{E}}x - \ddot{\mathbb{B}}x' - \ddot{\mathbb{B}}c'_a$	$-\ddot{\mathbb{B}}x' - \ddot{\mathbb{B}}c'_a$	$-\ddot{\mathbb{E}}c_a - \ddot{\mathbb{B}}x' - \ddot{\mathbb{B}}c'_a$

Table 4: Alice's net profit under all possible cases.

happens, then one of (Normal + Normal′), (Normal + Refund′) and (Refund + Refund′) must happen. Because $\$\mathsf{AV}(\ddot{\mathbb{E}}x - \ddot{\mathbb{B}}x') > 0$, except with some negligible probability, for all three possible cases Alice's utility is non-negative.

□

6.2.4 Against Externally Incentivized Alice-Miner Coalition

Lemma 6.7. *Let $\mathcal{A}$ be the coalition consisting of Alice and miners controlling no more than α fraction of the mining power where $\alpha \in [0, 1 - 1/\mathsf{poly}(\lambda)]$. Suppose Bob and at least $1 - \alpha$ fraction of mining power are honest. For any PPT strategy by $\mathcal{A}$, except with negligible probability, as long as none of $\mathbf{E}_5, \mathbf{E}_6, \mathbf{E}_7, \mathbf{E}_8$ happens, then, one and only one of the following statement holds.*

1. Normal *and* Normal′ *happen in polynomial time.*
2. Refund *and* Normal′ *happen in polynomial time.*
3. Refund *and* Refund′ *happen in polynomial time.*

Proof. First, we are going to show that one of Normal and Refund will happen. There are two cases.

- *Case 1: Bob enters the execution phase.* In this case, Rapidash and Rapidash' must enter the execution phase, and Bob already sent pre_c to P1 before Ethereum time T_0. There are two subcases.
  - *Subcase 1: Alice sends pre_a to P1 before Ethereum time T_1.* In this case, Bob never sends any transaction containing pre_b. Without pre_b, C2 can never be activated. As Bob is in the execution phase, B1 was activated and thus B2 can never be activated. As Alice sent pre_a to P1 and Bob sent pre_c to P1, either P1 is activated in polynomial time, or P2 is activated instead. If P1 is activated, Normal happens. If P2 is activated, Bob sends an empty message to C1, and thus C1 is activated in polynomial time and Refund happens.

– *Subcase 2: Alice does not send pre_a to P1 before Ethereum time T_1.* In this case, Bob will send _ to A1'. As $\mathbf{E}_6$ does not happen, Alice never sends any transaction containing pre'_a before A1' is guaranteed to be activated, and thus A2' is never activated. Thus, A1' is activated in polynomial time and Bob sends pre_b to P2 as soon as A1' is activated. Because $\mathbf{E}_5$ does not happen, Alice never sends any transaction containing pre_a before P1 or C1 is activated. Thus, C2 can never be activated. As Bob is in the execution phase, B2 can never be activated. Thus, either P1 is activated (Normal happens), or C1 is activated (thus P2 has been activated before and Refund happens), or the honest miner will include Bob's transaction, pre_b to P2, once they mine a block. Then, when τ Ethereum time has passed since P2 is activated, Bob will send _ to C1 and thus Refund happens.

- *Case 2: Bob enters the abort phase.* In this case, Bob will send the withdrawal transaction to Rapidash, and _ to A1'. If Rapidash is still in the preparation phase, the honest miner will include Bob's withdrawal transactions once they mine a block. Thus, in polynomial time, either Rapidash enters the execution phase, or Bob successfully withdraws his deposit from Rapidash, which implies Refund happens.

  Henceforth, we assume Rapidash enters the execution phase in polynomial time. Note that as Bob is in the abort phase, Bob never sent pre_c and thus, up to negligible probability, B2 and C2 can not be activated. Because $\mathbf{E}_6$ does not happen, $\mathcal{A}$ never sends pre'_a before A1' is guaranteed to be activated. Thus, A1' is activated in polynomial time. Then, either Alice sends _ to P2 before Bitcoin time T'_1, or Bob sends pre_b to P2. Either way, either P1 is activated (thus Normal happens), or P2 is activated in polynomial time. When τ Ethereum time has passed since P2 is activated, Bob will send _ to C1. Thus, C1 will be activated, and so Refund happens.

Next, we are going to show that one of Normal′ and Refund′ will happen. There are two cases.

- *Case 1: Bob enters the execution phase.* First, note that because $\mathbf{E}_6$ does not happen, Alice never sends any transaction containing pre'_a before A1' is guaranteed to be activated, and as Bob sends pre_b only after A1' is activated, it follows that A2' is never activated.

  Suppose Alice sends pre_a to P1 before Ethereum time T_1. Then, Bob never sends pre_b, and so C2' can not be activated via (pre'_a, pre_b). Because $\mathbf{E}_7$ does not happen, C2' can not be activated via (pre_a, pre'_a). Thus, C2' is never activated. As Bob sends $pre'_b = pre_a$ to P1', either Normal′ happens, or P2' is activated. In the latter case, Bob sends _ to C1', and so Refund′ happens.

  Henceforth, we assume Alice does not send pre_a to P1 before Ethereum time T_1. As we have shown before, in this case either Normal happens, or Refund happens. If Normal happens, Bob sends pre_a to P1'. If Refund happens, Bob will send _ to P2' as soon as C1 is activated. As $\mathbf{E}_8$ does not happen, C2' can not be activated using pre'_a after Bitcoin time T'_1. As $\mathbf{E}_6$ does not happen, C2' can not be activated using pre'_a before Bitcoin time T'_1 either. Thus, C2' is never activated. Thus, in both cases (Normal and Refund), either P1' or P2' will be activated in polynomial time. If P1' is activated, Normal′ happens. If P2' is activated, Bob will send _ to C1'. The honest miners will include Bob's transaction, _ to C1', once they mine a block. Thus, C1' will be activated in polynomial time, so Refund′ happens.

- *Case 2: Bob enters the abort phase.* In this case, Bob will send the withdrawal transaction to Rapidash', _ to P2' and _ to A1'. If Rapidash' is still in the preparation phase, the

honest miner will include Bob's withdrawal transactions once they mine a block. Thus, in polynomial time, either RAPIDASH' enters the execution phase, or Bob successfully withdraws his deposit from RAPIDASH', which implies Refund' happens.

Henceforth, we assume RAPIDASH' enters the execution phase in polynomial time. Because $\mathbf{E}_6$ does not happen, $\mathcal{A}$ never sends pre'_a before A1' is guaranteed to be activated. As Bob's sends _ to A1', A1' will be activated in polynomial time. If Alice sends _ to P2 by Bitcoin time T'_1, Bob never sends pre_b, and thus C2' can be activated only via (pre_a, pre'_a) (excluded by $\mathbf{E}_7$). If Alice does not send _ to P2 by Bitcoin time T'_1, because $\mathbf{E}_8$ does not happen, Alice does not send any transaction containing pre'_a before P1' or C1' is guaranteed to be activated. Thus, C2' can not be activated via pre'_a. Thus, C2' is never activated. As Bob sends _ to P2', or Alice's transaction, either P1' or P2' will be activated in polynomial time. If P1' is activated, Normal' happens. If P2' is activated, Bob will send _ to C1'. Thus, C1' will be activated in polynomial time, so Refund' happens.

So far, we have shown one of Normal and Refund will happen and one of Normal' and Refund' will happen. To prove the lemma statement, it suffices to show that Normal and Refund' never happen simultaneously. For the sake of reaching a contradiction, suppose Normal and Refund' happen. Event Normal happens implies P1 is activated. If Bob enters the abort phase, Bob never sends pre_c to P1. Ignoring the negligible probability that $\mathcal{A}$ finds pre_c by itself and forges Bob's signature, Bob must enter the execution, which implies both RAPIDASH and RAPIDASH' enter the execution phase. On the other hand, event Refund' happens implies either (P2' + C1') are activated or Alice withdraws her deposit from RAPIDASH'. As we have shown, RAPIDASH' must enter the execution phase, so event Refund' happens implies (P2' + C1') are activated. In the execution phase, Bob will send _ to P2' only if C1 is activated, which is impossible given that P1 is activated. Thus, P2' can be activated only if Alice sends pre'_a to P2'. Because C1' can be activated when τ' Bitcoin time has passed since P2' is activated, Alice must send pre'_a to P2' before C1' is guaranteed to be activated. There are two subcases.

- *Subcase 1: pre_a is sent to P1 before Ethereum time T_1.* Because P2' must be activated after Bitcoin time T'_1 which is later than Ethereum time T_1, $\mathcal{A}$ must send pre_a to P1 and pre'_a to P2' before C1' is guaranteed to be activated. Thus, depends on whether A1' is guaranteed to be activated, either $\mathbf{E}_6$ or $\mathbf{E}_7$ happens.
- *Subcase 2: pre_a has not been sent to P1 before Ethereum time T_1.* Because Bob enters the execution phase, he will send pre_b to P2 as soon as A1' is activated. Thus, depending on whether A1' is guaranteed to be activated, either $\mathbf{E}_6$ or $\mathbf{E}_8$ happens.

Because we assume none of $\mathbf{E}_5, \mathbf{E}_6, \mathbf{E}_7, \mathbf{E}_8$ happens, either subcase leads to a contradiction.

□

Lemma 6.8 (Blatant non-rationality of $\overline{\mathcal{R}}$ for Alice-miner coalition)**.** *Suppose that the parameter constraints in Section 6.1 hold and that the coalition $\mathcal{A}$ consists of Alice and miners controlling no more than α fraction of the mining power where $\alpha \in [0, 1 - 1/\mathsf{poly}(\lambda)]$. Given any PPT strategy $S_{\mathcal{A}} \in \overline{\mathcal{R}}$ for some (externally incentivized) coalition $\mathcal{A}$, there is a PPT strategy $\widehat{S}_{\mathcal{A}}$ such that*

$$\mathsf{util}^{\mathcal{A}}(\widehat{S}_{\mathcal{A}}, HS_{-\mathcal{A}}) > \mathsf{util}^{\mathcal{A}}(S_{\mathcal{A}}, HS_{-\mathcal{A}}).$$

Proof. Suppose the coalition $\mathcal{A}$ adopts a strategy in which $\mathbf{E}_5, \mathbf{E}_6, \mathbf{E}_7$ or $\mathbf{E}_8$ happens with non-negligible probability. We can construct a new strategy for $\mathcal{A}$ with strictly better expected utility. Specifically, consider a modified PPT strategy denoted $\widehat{S}_{\mathcal{A}}$: whenever by the original strategy, the

first of $\mathbf{E}_5, \mathbf{E}_6, \mathbf{E}_7$ or $\mathbf{E}_8$ is about to happen, $\mathcal{A}$ simply stops sending any messages (including the message that is about to trigger $\mathbf{E}_5, \mathbf{E}_6, \mathbf{E}_7$ or $\mathbf{E}_8$) to the contract from that moment on.

Notice that by definition, when $\mathbf{E}_5, \mathbf{E}_6, \mathbf{E}_7$ or $\mathbf{E}_8$ happens, it must be caused by a message sent by $\mathcal{A}$. Thus, as long as $\mathcal{A}$ stops sending any messages before the first of $\mathbf{E}_5, \mathbf{E}_6, \mathbf{E}_7$ or $\mathbf{E}_8$ is about to happen, none of $\mathbf{E}_5, \mathbf{E}_6, \mathbf{E}_7$ and $\mathbf{E}_8$ can happen in the future. By Lemma 6.7, in polynomial time, one of (Normal + Normal'), (Refund + Normal') and (Refund + Refund') must happen. According to Table 4, among (Normal + Normal'), (Refund + Normal') and (Refund + Refund'), $\mathcal{A}$'s utility is at least $\$\mathsf{AV}(-฿x')$. In other words, we have $\mathsf{util}^{\mathcal{A}}(\widehat{S}_{\mathcal{A}}, HS_{-\mathcal{A}}) \geq \$\mathsf{AV}(-฿x')$. Thus, to show $\mathsf{util}^{\mathcal{A}}(\widehat{S}_{\mathcal{A}}, HS_{-\mathcal{A}}) \geq \mathsf{util}^{\mathcal{A}}(S_{\mathcal{A}}, HS_{-\mathcal{A}})$, we only need to show $\mathsf{util}^{\mathcal{A}}(S_{\mathcal{A}}, HS_{-\mathcal{A}}) < \$\mathsf{AV}(-฿x')$.

We consider four cases, depending on whether $\mathbf{E}_5, \mathbf{E}_6, \mathbf{E}_7$ or $\mathbf{E}_8$ happens first in the original strategy $S_{\mathcal{A}}$.

Event $\mathbf{E}_5$ happens first. Now, consider some strategy $S_5 \in \overline{\mathcal{R}}$ with non-negligible probability that $\mathbf{E}_5$ happens. When $\mathbf{E}_5$ happens, the honest Bob will send pre_b to P2 as soon as A1' is activated. Thus, when $\mathcal{A}$ sends a transaction containing pre_a, the $1-\alpha$ fraction of honest miners would send (pre_a, pre_b, pre_c) to C2, if they are chosen to mine a block. By Lemma 6.3, if C2 is activated by an honest miner, the utility of $\mathcal{A}$ is at most $\$\mathsf{AV}(-Ξc_a) + \$E$. On the other hand, if C2 is not activated by an honest miner, the utility of $\mathcal{A}$ is at most $\$\mathsf{AV}(Ξx) + \$E$. When $\mathbf{E}_5$ happens, the probability that C2 is activated by an honest miner is at least $1-\alpha$. Thus, the utility of $\mathcal{A}$ is at most

$$(1-\alpha)(\$\mathsf{AV}(-Ξc_a) + \$E) + \alpha(\$\mathsf{AV}(Ξx) + \$E) < \$\mathsf{AV}(-฿x'),$$

where the inequality arises from the fact that $\$\mathsf{AV}(Ξc_a) > \frac{\$\mathsf{AV}(฿x' + \alpha Ξx) + \$E}{1-\alpha}$.

Event $\mathbf{E}_6$ happens first. Now, consider some strategy $S_6 \in \overline{\mathcal{R}}$ with non-negligible probability that $\mathbf{E}_6$ happens. When $\mathbf{E}_6$ happens, the $1-\alpha$ fraction of honest miners would send pre'_a to A2' (or to C2' if additionally either pre'_b or pre_b is known), if they are chosen to mine a block. By Lemma 6.3, if either A2' or C2' is activated by an honest miner, the utility of $\mathcal{A}$ is at most $\$\mathsf{AV}(Ξx - ฿x' - ฿c'_a) + \$E$. On the other hand, if neither A2' nor C2' is activated by an honest miner, the utility of $\mathcal{A}$ is at most $\$\mathsf{AV}(Ξx) + \$E$. When $\mathbf{E}_6$ happens, the probability that A2' or C2' is activated by an honest miner is at least $1-\alpha$. Thus, the utility of $\mathcal{A}$ is at most

$$(1-\alpha)(\$\mathsf{AV}(Ξx - ฿x' - ฿c'_a) + \$E) + \alpha(\$\mathsf{AV}(Ξx) + \$E) < \$\mathsf{AV}(-฿x'),$$

where the inequality arises from the fact that $\$\mathsf{AV}(฿c'_a) > \frac{\$\mathsf{AV}(Ξx + \alpha ฿x') + \$E}{1-\alpha}$.

Event $\mathbf{E}_7$ happens first. Now, consider some strategy $S_7 \in \overline{\mathcal{R}}$ with non-negligible probability that $\mathbf{E}_7$ happens. When $\mathbf{E}_7$ happens, the $1-\alpha$ fraction of honest miners would send (pre_a, pre'_a) to C2', if they are chosen to mine a block. By Lemma 6.3, if C2' is activated by an honest miner, the utility of $\mathcal{A}$ is at most $\$\mathsf{AV}(Ξx - ฿x' - ฿c'_a) + \$E$. On the other hand, if C2' is not activated by an honest miner, the utility of $\mathcal{A}$ is at most $\$\mathsf{AV}(Ξx) + \$E$. When $\mathbf{E}_7$ happens, the probability that C2' is activated by an honest miner is at least $1-\alpha$. Thus, by the same calculation as the previous case, the utility of $\mathcal{A}$ is strictly less than $\$\mathsf{AV}(-฿x')$.

Event $\mathbf{E}_8$ happens first. Now, consider some strategy $S_8 \in \overline{\mathcal{R}}$ with non-negligible probability that $\mathbf{E}_8$ happens. If Bob enters the execution phase, because Alice does not send pre_a to P1 before Ethereum time T_1, Bob will send pre_b to P2 as soon as A1' is activated. On the other hand, if Bob enters the abort phase, because Alice does not send _ to P2 before Bitcoin time T'_1, Bob will send pre_b to P2 as soon as A1' is activated. In either case, Bob always sends pre_b to P2 as soon as A1' is activated.

Notice that if A1' is guaranteed to be activated at time Bitcoin time t^*, it must be activated no later than Bitcoin time $t^* + 1$. When $\mathbf{E}_8$ happens, A1' has been guaranteed to be activated. Thus, when $\mathbf{E}_8$ happens, pre'_a and pre_b are both publicly known. Then, the $1-\alpha$ fraction of honest miners would send (pre'_a, pre_b) to C2', if they are chosen to mine a block. By Lemma 6.3, if C2' is activated by an honest miner, the utility of $\mathcal{A}$ is at most $\$\mathsf{AV}(\text{Ð}x - \text{Ƀ}x' - \text{Ƀ}c'_a) + \$E$. On the other hand, if C2' is not activated by an honest miner, the utility of $\mathcal{A}$ is at most $\$\mathsf{AV}(\text{Ð}x) + \$E$. When $\mathbf{E}_8$ happens, the probability that C2' is activated by an honest miner is at least $1 - \alpha$. Thus, by the same calculation as the previous case, the utility of $\mathcal{A}$ is strictly less than $\$\mathsf{AV}(-\text{Ƀ}x')$.

Finally, notice that the above analysis holds even if $\mathcal{A}$ may post a new contract on the fly during the protocol execution, since all other players are honest and will not deposit money into the new contract.

□

Lemma 6.9 (Against Externally Incentivized Alice-Miner Coalition)**.** *Suppose that the hash function $H(\cdot)$ is a one-way function. Let $\mathcal{C}$ be a coalition consisting of Bob and possibly any subset of the miners, and let $\mathcal{A}$ be a disjoint coalition consisting of Alice and at most α fraction of the mining power where $\alpha \in [0, 1 - 1/\mathsf{poly}(\lambda)]$. Suppose that $\mathcal{C}$ does not have external incentives but $\mathcal{A}$ may have up to $\$E$ amount of external incentives. Let $S_{\mathcal{A}}$ be an arbitrary PPT strategy of $\mathcal{A}$ that is not in $\overline{\mathcal{R}}$. Then, there exists a negligible function $\mathsf{negl}(\cdot)$ such that except with negligible probability $\mathsf{negl}(\lambda)$, it holds that*

$$\mathsf{util}^{\mathcal{C}}(HS_{\mathcal{C}}, S_{\mathcal{A}}, HS_{\mathcal{D}}) \geq 0,$$

where $\mathcal{D}$ denotes everyone else not in $\mathcal{C} \cup \mathcal{A}$.

Proof. By Lemma 6.5, any strategy that makes one of $\mathbf{E}_5, \mathbf{E}_6, \mathbf{E}_7, \mathbf{E}_8$ happen is blatantly non-rational. By Lemma 6.7, if none of $\mathbf{E}_5, \mathbf{E}_6, \mathbf{E}_7, \mathbf{E}_8$ happen, the one of $(\mathsf{Normal} + \mathsf{Normal}')$, $(\mathsf{Refund} + \mathsf{Normal}')$, and $(\mathsf{Refund} + \mathsf{Refund}')$ must happen. According to Table 3, because $\$\mathsf{BV}(-\text{Ð}x + \text{Ƀ}x') > 0$, for all three possible cases, Bob's utility is never negative. □

6.2.5 Proof of Theorem 6.2

Now, we are ready to prove Theorem 6.2. We can divide into the following cases:

Case 1: Alice $\in \mathcal{C}$ and Bob $\in \mathcal{C}'$: covered by Lemma 6.6.

Case 2: Bob $\in \mathcal{C}$ and Alice $\in \mathcal{C}'$: covered by Lemma 6.9.

Case 3: $\mathcal{C}$ is miner-only: It is straightforward to see that no matter how players outside $\mathcal{C}$ behave, as long as $\mathcal{C}$ behaves honestly, its utility is non-negative.

Case 4: Alice $\in \mathcal{C}$ and $\mathcal{C}'$ is miner-only: Notice that both Alice and Bob are assumed to be honest. In the protocol, Alice and Bob decide whether they will go to the abort phase according to whether Bob sends pre_c to P1. Thus, when Alice and Bob are both honest, both of them enter to the execution phase or both of them enter to the abort phase. There are two subcases.

- *Subcase 1: Both Alice and Bob enter to the execution phase.* Bob only sends pre_c to P1 when B1 has been activated. Ignoring the negligible probability that $\mathcal{C}'$ finds pre_c by itself, B2 can never be activated. Alice would send pre_a to P1 as soon as she enters the execution phase, and Bob would send $pre'_b = pre_a$ to P1' as soon as Alice sent pre_a to P1. Because Alice always sends pre_a to P1 before Ethereum time T_1, Bob never sends any transaction containing pre_b. Besides, Alice never sends any transaction containing pre'_a. Ignoring the

negligible probability that $\mathcal{C}'$ finds pre_b or pre'_a by itself, A2', C2 and C2' can never be activated. When Alice and Bob are in the execution phase, P2 can be activated only by Bob sending pre_b. As we have shown, Bob never sends any transaction containing pre_b, so P2 can never be activated. Moreover, when Alice and Bob are in the execution phase, P2' can be activated only by Bob sending _. However, Bob will only send _ to P2' if C1 is activated. Because P2 cannot be activated, C1 cannot be activated either. Thus, P2' cannot ever be activated. The $1-\alpha$ fraction of honest miner will include Alice's and Bob's transactions, so except the negligible probability, P1 and P1' will be activated in polynomial time. Thus, the honest $\mathcal{C}$ obtains non-negative utility.

- *Subcase 2: Both Alice and Bob go to the abort phase.* In this case, Bob never sends any transaction containing pre_c. In the abort phase Alice always sends _ to P2 at Bitcoin time T'_0, so Bob never sends any transaction containing pre_b. Similarly, Bob always sends _ to P2' at Bitcoin time T'_0, so Alice never sends any transaction containing pre'_a. Moreover, Alice never sends any transaction containing pre_a. Ignoring the negligible probability that $\mathcal{C}'$ finds $pre_a, pre_b, pre_c, pre'_a$ by itself, A2', B2, P1, C2, P1', C2' can never be activated.

  If Rapidash (Rapidash', resp.) has not entered the execution phase, Alice and Bob send the withdrawal transaction to Rapidash (Rapidash', resp.). The $1-\alpha$ fraction of honest miner will include Alice's and Bob's withdrawal transactions, so except the negligible probability, they can get their deposit back unless the contract enters the execution phase. Next, we analyze different cases depending which contract enters the execution phase.

  - Rapidash *enters the execution phase.* As we have shown, B2, P1, C2 cannot be activated. The $1-\alpha$ fraction of honest miner will include Alice's transaction, _ to P2, once they mine a block. Thus, P2 will be activated in polynomial time. When τ Ethereum time has passed since P2 is activated, Alice and Bob will send _ to C1. Again, the $1-\alpha$ fraction of honest miner will include Alice's and Bob's transactions, _ to C1, once they mine a block. Thus, C1 will be activated in polynomial time.
  - Rapidash' *enters the execution phase.* As we have shown, A2', P1', C2' cannot be activated. The $1-\alpha$ fraction of honest miner will include Bob's transaction, _ to P2', once they mine a block. Thus, P2' will be activated in polynomial time. When τ' Bitcoin time has passed since P2' is activated, Alice and Bob will send _ to C1'. Again, the $1-\alpha$ fraction of honest miner will include Alice's and Bob's transactions, _ to C1', once they mine a block. Thus, C1' will be activated in polynomial time.

In all cases, $\mathcal{C}$'s utility is non-negative except with negligible probability.

Case 5: Bob $\in \mathcal{C}$ and $\mathcal{C}'$ is miner-only: The argument is the same as Case 4.

6.3 Proof for Dropout Resilience

Theorem 6.10 (Dropout resilience of atomic swap)**.** *Suppose that $H(\cdot)$ is a one-way function and that all players are PPT machines. Our atomic swap protocol is dropout resilient. In other words, suppose at least $1/\mathsf{poly}(\lambda)$ fraction of the mining power is honest on either chain; if either Alice or Bob plays honestly but drops out before the end of the protocol, then with $1-\mathsf{negl}(\lambda)$ probability, the other party's utility must be non-negative.*

Proof. Throughout the proof, for any $X \in \{pre_a, pre_b, pre_c, pre'_a\}$, we ignore the negligible probability that the miners can find the preimage X by itself if Alice and Bob have never sent X before.

We first analyze the cases where Alice drops out. Notice that if any of $\mathbf{E}_5, \mathbf{E}_6, \mathbf{E}_7, \mathbf{E}_8$ is about to happen, it must be that Alice deviates from the protocol and is about to send a transaction that does not follow from the honest protocol. Because Alice is honest, no matter when does Alice drop out, none of $\mathbf{E}_5, \mathbf{E}_6, \mathbf{E}_7, \mathbf{E}_8$ would happen. By Lemma 6.7, one and only one of $(\mathsf{Normal} + \mathsf{Normal}')$, $(\mathsf{Refund} + \mathsf{Normal}')$ and $(\mathsf{Refund} + \mathsf{Refund}')$ will happen. According to Table 3, because $\$\mathsf{BV}(₿x' - Ɇx) > 0$, for any of the three cases, Bob's utility is non-negative.

Next, we analyze the cases where Bob drops out. Notice that if any of $\mathbf{E}_1, \mathbf{E}_2, \mathbf{E}_3, \mathbf{E}_4$ is about to happen, it must be that Bob deviates from the protocol and is about to send a transaction that does not follow from the honest protocol. Because Bob is honest, no matter when does Bob drop out, none of $\mathbf{E}_1, \mathbf{E}_2, \mathbf{E}_3, \mathbf{E}_4$ would happen. By Lemma 6.4, one and only one of $(\mathsf{Normal} + \mathsf{Normal}')$, $(\mathsf{Normal} + \mathsf{Refund}')$ and $(\mathsf{Refund} + \mathsf{Refund}')$ will happen. According to Table 4, because $\$\mathsf{AV}(Ɇx - ₿x') > 0$, for any of the three cases, Alice's utility is non-negative. □

7 Rapidash Disincentivizes a 100% Coalition

So far, to prove our coalition-resistant fairness notions, we assumed that the coalition wields strictly less than 100% of the mining power. Take the knowledge-coin protocol for example: if Bob can solicit a coalition of 100% of the mining power, then its best strategy is to wait for Alice to post pre_a, and then activate P2 and C1. In this way, Bob and the coalition effectively learns the secret pre_a for free.

7.1 The Meta-Game of Coalition Formation

We argue that Rapidash *provides strong disincentives for such a 100% coalition to form*, even in a world where one can post bribery contracts [Bon16, JSZ+21, MHM18, WHF19] or other smart contracts in an attempt to openly solicit everyone. To see this, we can view the process of soliciting coalition members as a meta-game. Let us first consider the knowledge-coin protocol. Imagine that Bob openly invites miners to join, e.g., through some smart contract. If the coalition wins, the additional gain of at most $\$v$ (relative to the honest case) will be split off among all miners proportional to their mining power, e.g., distributed to each miner who mined a block that starves P1 and C2.

In this meta-game, each miner has two choices, to join or not to join the coalition. It is not hard to see that everyone joining is not an equilibrium of this meta-game. Specifically, if every other miner joined, then I would be strictly better off *not* joining the coalition. If I do not join, the moment P2 is activated, I have a T_2 lead in time to mine a block in which I can redeem $\$\epsilon$ from the C2 branch. In particular, we may assume that every miner in the coalition commits to starving C2 in every block they mine, e.g., by placing a collateral that it will honor its commitment — if not, then the coalition will not be stable since a coalition member will be incentivized to defect from the coalition and claim C2 itself. This means that if I mine a block during the T_2 window after the activation of P2, I can claim $\$\epsilon$ from C2 for myself. Now, suppose I have γ fraction of mining power, and suppose that $T_2 > 1$. The probability that I mine a block in a window of T_2 length is $1 - (1-\gamma)^{T_2}$. Therefore, if I do not join the coalition, my expected gain would be $\$\epsilon \cdot (1 - (1-\gamma)^{T_2})$. If I join the coalition, my expected gain is $\gamma \cdot \$v$. Thus, as long as $\$\epsilon \cdot (1 - (1-\gamma)^{T_2}) > \gamma \cdot \$v$, my best strategy is to not join the coalition. This means that if everyone else joins the coalition, my best strategy is not to join. In other words, a 100% coalition is not an equilibrium of the coalition-forming meta-game. For example, if we choose $T_2 = 2$, then it suffices to choose $\$\epsilon > \$v \cdot \frac{1}{2-\gamma}$.

The above argument is for the knowledge-coin exchange protocol. For our atomic swap protocol, essentially the same meta-game analysis would apply.

7.2 Comparison with Prior Approaches

Using this coalition formation meta-game perspective, we can give a (hopefully clearer) re-exposition of some incentive attacks described in prior works [MHM18, Bon16, WHF19]. In particular, the earlier work of MAD-HTLC [TYME21] is motivated by the fact that the standard HTLC contract (see Section 2.2), has some coalition formation meta-games in which a 100% coalition is the equilibrium.

Meta-games for HTLC. Bob can post a bribery contract soliciting participation of miners: if Alice's redeeming transaction is censored until Bob claims the $\$v$ back through pre_b, then, Bob will equally re-distribute $\$(v-\epsilon)$ to every miner that helped to mine a block that starved Alice's transaction where $\$\epsilon$ is a small amount Bob keeps for himself. Suppose the transaction fees are 0, then every miner's best strategy is to join the coalition, and thus a 100% coalition is an equilibrium of the meta-game.

If Alice is offering a transaction fee of $\$f$ for her transaction, and assuming that $\$f < \$v/T_1$. Then, Bob can offer $\$(v-\epsilon)/T_1$ to everyone who helps to censor Alice's transaction until Bob could claim the $\$v$ back for himself. In this case, every miner's best strategy is to take the bribe which also effectively leads to a 100% coalition.

Tsabary et al. [TYME21] also describe the following meta-game. Suppose that the mining power of the smallest miner is $\gamma_{\min}$. Bob can offer to pay $\$f'$ to whoever mines the block immediately after T_1 that helps him claim his $\$v$ back. Let $\$f$ be the fee offered by Alice. Suppose that $\$f' \cdot \gamma_{\min} > \$f$, then everyone joining is also an equilibrium of this meta-game (assuming non-myopic miners), effectively leading to a 100% coalition. Roughly speaking, this is because if I give up on my immediate gain $\$f$ right now, there is at least $\gamma_{\min}$ probability that I will be the miner who mines the first block after T_1 which allows me to claim the richer reward $\$f'$ instead.

MAD-HTLC. The result of MAD-HTLC [TYME21] can be viewed as follows: by allowing the miner to claim $\$v$ itself through (pre_a, pre_b), it removes the undesirable 100%-coalition equilibrium in the coalition formation meta-game — the design of Rapidash is inspired by this elegant idea. Unfortunately, the design of MAD-HTLC incentivizes coalitions (with binding side contracts) to deviate in the protocol game itself. As mentioned earlier in Section 2.2, Bob colluding with a miner should always deviate: if it happens to be the miner when Alice posts pre_a, the coalition should always starve Alice's transaction and claim the $\$v$ itself by posting (pre_a, pre_b).

8 Bitcoin Instantiation

In this section we describe how Rapidash can be instantiated in Bitcoin with its limited scripting features.

8.1 Notation and Background

As described earlier, with general smart contracts, users send coins to contracts, the contracts then hold the coins until some logic is triggered to pay part to all of the coins to one or more user(s). Instead, Bitcoin uses an *Unspent Transaction Output* (UTXO) model, where coins are stored in addresses denoted by $Adr \in \{0,1\}^\lambda$ and addresses are spendable (i.e., used as input to a transaction) *exactly once*. Transactions can be posted on the blockchain to transfer coins from a

set of input addresses to a set of output addresses, and any remaining amount of coin is collected by the miner of the block as transaction fee.

More precisely, in Bitcoin transactions are generated by the transaction function tx. A transaction tx_A, denoted

$$tx_A := tx\left(\begin{matrix}[(Adr_1, \Phi_1, \$v_1), \ldots, (Adr_n, \Phi_n, \$v_n)],\\ [(Adr'_1, \Phi'_1, \$v'_1), \ldots, (Adr'_m, \Phi'_m, \$v'_m)]\end{matrix}\right),$$

charges v_i coins from each input address Adr_i for $i \in [n]$, and pays v'_i coins to each output address Adr'_j where $j \in [m]$. It must be guaranteed that $\sum_{i\in[n]} \$v_i \geq \sum_{j\in[m]} \$v'_j$. The difference $\$f = \sum_{i\in[n]} \$v_i - \sum_{j\in[m]} \$v'_j$ is offered as the transaction fee to the miner who includes the transaction in his block.

An address in Bitcoin is typically associated with a *script* $\Phi : \{0,1\}^\lambda \to \{0,1\}$ which states what conditions need to be satisfied for the coins to be spent from the address. In contrast to smart contracts that can verify arbitrary conditions for coins to be transacted, the scripting language of Bitcoin has limited expressiveness. A transaction is considered authorized if it is attached with witnesses $[x_1, \ldots, x_n]$ such that $\Phi_i(x_i) = 1$ (publicly computable) for all $i \in [n]$. A transaction is confirmed if it is included in the blockchain.

Thus, for Bitcoin, the logic of Rapidash must be encoded in scripts of addresses where the scripts are of limited expressiveness and the addresses are spendable exactly once. As we show, our Rapidash instantiation only requires some of the most standard scripts used currently in Bitcoin.

We largely rely on the following scripts: (1) computation of hash function[7] $H : \{0,1\}^* \to \{0,1\}^\kappa$, (2) transaction timestamp verification wrt. current height of the blockchain denoted by _NOW[8] and (3) digital signature verification. The signature scheme consists of the key generation algorithm $\mathsf{KGen}(1^\lambda)$ generating the signing key sk and the verification key pk, the signing algorithm $\mathsf{Sign}(\mathsf{sk}, m)$ generating a signature σ on the message m using sk, and the verification algorithm $\mathsf{Vf}(\mathsf{pk}, m, \sigma)$ validating the signature wrt. pk. [9] We say an address Adr (associated script Φ) is controlled by a user if the user knows a witness x s.t. $\Phi(x) = 1$.

8.2 Instantiating Rapidash Single Instance

We provide the list of all transactions in Table 5, the scripts associated with all addresses in Figure 1, and the relationship between the transactions, addresses, and scripts is depicted in Figure 2. Basically, Bob uses the transaction tx_{stp} to put his deposit $\$v + \$c_b$ into the address Adr_{stp}. The script on the address Adr_{stp} allows three ways to spend the deposit:

1. Use pre_a to pay $\$v$ amount to an address Adr_1^A owned by Alice, and $\$c_b$ to an address Adr_1^B owned by Bob.

2. After a timeout T_1 since the address Adr_{stp} comes into existence, use pre_b to pay the entire deposit amount $\$v + \$c_b$ to the address Adr_{P2}, which is associated with the script Φ_{P2}. Φ_{P2} says that either (1) T_2 time has passed after the address came into existence, in which case the $\$v + \$c_b$ coins in Adr_{P2} can be paid to Bob's address Adr_2^B, or (2) the pair (pre_a, pre_b) is

[7] $\kappa = 160$ in Bitcoin when using the opcode OP_HASH160.

[8] Instantiated using the opcode OP_CHECKSEQUENCEVERIFY in Bitcoin checking if the height of the blockchain has increased beyond some threshold after the script first appeared on the blockchain. It can also be instantiated with opcode OP_CHECKLOCKTIMEVERIFY in Bitcoin that checks if the current height of the blockchain is beyond a threshold.

[9] The signature scheme can be instantiated with either Schnorr or ECDSA in Bitcoin. ECDSA signatures are verified using the opcode OP_CHECKSIG and Schnorr signatures via the taproot fork.

Table 5: RAPIDASH's transactions in Bitcoin. Here Φ^B is the script that requires the signature under Bob's public key while Φ^A is the script that requires the signature under Alice's public key.

	Description
tx_{stp}	$tx\begin{pmatrix}[(Adr_0^B, \Phi^B, \$v + \$c_b)],\\ [(Adr_{\mathsf{stp}}, \Phi_{\mathsf{stp}}, \$v + \$c_b)]\end{pmatrix}$
tx_{P1}	$tx\begin{pmatrix}[(Adr_{\mathsf{stp}}, \Phi_{\mathsf{stp}}, \$v + \$c_b)],\\ [(Adr_1^A, \Phi^A, \$v), (Adr_1^B, \Phi^B, \$c_b)]\end{pmatrix}$
tx_{P2}	$tx\begin{pmatrix}[(Adr_{\mathsf{stp}}, \Phi_{\mathsf{stp}}, \$v + \$c_b),\\ [(Adr_{\mathrm{P2}}, \Phi_{\mathrm{P2}}, \$v + \$c_b)]\end{pmatrix}$
tx_{C1}	$tx\begin{pmatrix}[(Adr_{\mathrm{P2}}, \Phi_{\mathrm{P2}}, \$v + \$c_b)\\ [(Adr_2^B, \Phi^B, \$v + \$c_b)]\end{pmatrix}$
tx_{C2}	$tx\begin{pmatrix}[(Adr_{\mathsf{stp}}, \Phi_{\mathsf{stp}}, \$v + \$c_b)],\\ [(Adr_{\mathsf{burn}}, \Phi_{\mathsf{burn}}, \$v + \$c_b - \$\epsilon)]\end{pmatrix}$
$tx_{\mathrm{C2}}^{\mathrm{P2}}$	$tx\begin{pmatrix}[(Adr_{\mathrm{P2}}, \Phi_{\mathrm{P2}}, \$v + \$c_b)\\ [(Adr_{\mathsf{burn}}, \Phi_{\mathsf{burn}}, \$v + \$c_b - \$\epsilon)]\end{pmatrix}$

revealed, in which case $\$v + \$c_b - \$\epsilon$ coins can be paid to some burn address Adr_{burn} whose private key is known to nobody, and the remaining $\$\epsilon$ is paid as fee to the miner who mines the block.

3. Use the pair (pre_a, pre_b) to pay $\$v + \$c_b - \$\epsilon$ amount to some burn address Adr_{burn} whose private key is known to nobody, the remaining $\$\epsilon$ is paid as fee to the miner who mines the block.

To make sure that Alice and Bob cannot unilaterally spend from the address Adr_{stp}, and Adr_{P2}, their associated scripts require *signatures from both Alice and Bob* to spend from these addresses. Note also that the transactions tx_{P1}, tx_{P2}, and tx_{C2} needed to spend from Adr_{stp} via activation points P1, P2, or C2 are signed with *different* public keys of Alice and Bob for each activation point, i.e., $(\mathsf{pk}_a, \mathsf{pk}_b)$, $(\mathsf{pk}'_a, \mathsf{pk}'_b)$, and $(\mathsf{pk}''_a, \mathsf{pk}''_b)$ respectively. This makes sure that each transaction can invoke only the intended activation point. Similarly for transactions tx_{C2} and $tx_{\mathrm{C2}}^{\mathrm{P2}}$ spending from Adr_{P2}.

Protocol flow. Before setting up RAPIDASH on the blockchain, Alice and Bob agree on the setup transaction tx_{stp}. The transaction must be signed by Bob to take effect. However, before signing tx_{stp}, Alice and Bob agree on and sign all redeeming transactions, i.e., tx_{P1}, tx_{P2}, tx_{C1},$tx_{\mathrm{C2}}^{\mathrm{P2}}$, and tx_{C2}. Alice and Bob now broadcast all these transactions (not including tx_{stp}) and both of their signatures — notice that they cannot be published on the Bitcoin blockchain yet because the addresses they depend on, Adr_{stp} or Adr_{P2}, are not ready yet.

At this moment, Bob reveals his signatures on tx_{stp}. Once tx_{stp} is published on the Bitcoin blockchain, the *execution phase* starts. During the execution phase, either Alice reveals pre_a and publishes transaction tx_{P1} (along with signatures on the transaction), or Bob reveals pre_b and publishes transaction tx_{P2} (along with signatures on the transaction) after T_1 time has passed since the confirmation of tx_{stp}. In the honest run of the protocol, if tx_{P1} is confirmed, Bob gets back his collateral immediately. If not, Bob can redeem the collateral after waiting for time $T_1 + T_2$ using tx_{P2} and tx_{C1}. In the event of misbehavior leading to both pre_a and pre_b being revealed, any miner

$\Phi_{\mathsf{stp}}(tx, pre_a, pre_b, \sigma_a, \sigma_b)$

P1: **if** $(H(pre_a) = h_a) \wedge (\mathsf{Vf}(\mathsf{pk}_a, tx, \sigma_a) = 1) \wedge (\mathsf{Vf}(\mathsf{pk}_b, tx, \sigma_b) = 1)$
then return 1

P2: **if** $(\_\text{NOW} > T_1) \wedge (H(pre_b) = h_b) \wedge (\mathsf{Vf}(\mathsf{pk}'_a, tx, \sigma_a) = 1) \wedge$
$(\mathsf{Vf}(\mathsf{pk}'_b, tx, \sigma_b) = 1)$ **then return** 1

C2: **if** $(\mathsf{Vf}(\mathsf{pk}''_a, tx, \sigma_a) = 1) \wedge (\mathsf{Vf}(\mathsf{pk}''_b, tx, \sigma_b) = 1) \wedge (H(pre_a) = h_a) \wedge$
$(H(pre_b) = h_b)$ **then return** 1

// Values $h_a, h_b, \mathsf{pk}_a, \mathsf{pk}_b, T_1, \mathsf{pk}'_a, \mathsf{pk}'_b, \mathsf{pk}''_a, \mathsf{pk}''_b$ are hardwired

$\Phi_{\text{P2}}(tx, pre_a, pre_b, \sigma_a, \sigma_b)$

C1: **if** $(\_\text{NOW} > T_2) \wedge (\mathsf{Vf}(\mathsf{pk}_a, tx, \sigma_a) = 1) \wedge (\mathsf{Vf}(\mathsf{pk}_b, tx, \sigma_b) = 1)$
then return 1

C2: **if** $(\mathsf{Vf}(\mathsf{pk}'_a, tx, \sigma_a) = 1) \wedge (\mathsf{Vf}(\mathsf{pk}'_b, tx, \sigma_b) = 1) \wedge (H(pre_a) = h_a)$
$\wedge (H(pre_b) = h_b)$ **then return** 1

// Values $T_2, h_a, h_b, \mathsf{pk}_a, \mathsf{pk}_b, \mathsf{pk}'_a, \mathsf{pk}'_b$ are hardwired

Figure 1: The description of scripts Φ_{stp} and Φ_{P2}. Here tx is the transaction spending from the script. Keys $\mathsf{pk}_a, \mathsf{pk}'_a$ and pk''_a belong to Alice, $\mathsf{pk}_b, \mathsf{pk}'_b$ and pk''_b belong to Bob.

in the system can immediately spend from the C2 branch of either Adr_{stp}, or Adr_{P2}, and burn all coins except $\$\epsilon$ coins as transaction fee for itself.

8.3 Instantiating Atomic Swap

In this section we show how we can instantiate both Rapidash and Rapidash' in Bitcoin's scripting language for the atomic swap protocol from Section 5 and Section 6.1. The techniques for the instantiations follow quite closely to the techniques from above.

8.3.1 Instantiating Rapidash from Section 5

We have minor differences compared to the single instance instantiation.

Transactions. We describe below the different transactions needed for our Rapidash instantiation.

- We now have an additional payment redeem transaction, $tx_{\text{P2}}^{\text{empty}}$ (see Table 6) apart from tx_{P1} and tx_{P2} that redeem from the payment address Adr_{stp}. We have the transaction $tx_{\text{P2}}^{\text{empty}}$ that redeems $\$x + \$c_b$ coins to an auxiliary address Adr_{P2}. The description of Φ_{stp} is given below in Figure 3. We set the transaction $tx_{\text{P2}}^{\text{empty}}$ to redeem the coins from the (E2) branch. This transaction will correspond to the empty message call to the Rapidash contract in activation point P2. The script Φ_{stp} has a modification in the C2 branch, where we require either (pre_a, pre_b, pre_c) along with the signatures of Alice and Bob. Similarly the script Φ_{P2} of the auxiliary addresses is modified in its C2 branch.
- In addition to the collateral redeem transaction tx_{C1} we have the transaction $tx_{\text{C1}}^{\text{empty}}$ which redeems the coins to Bob from the auxiliary address generated by $tx_{\text{P2}}^{\text{empty}}$. We have modified

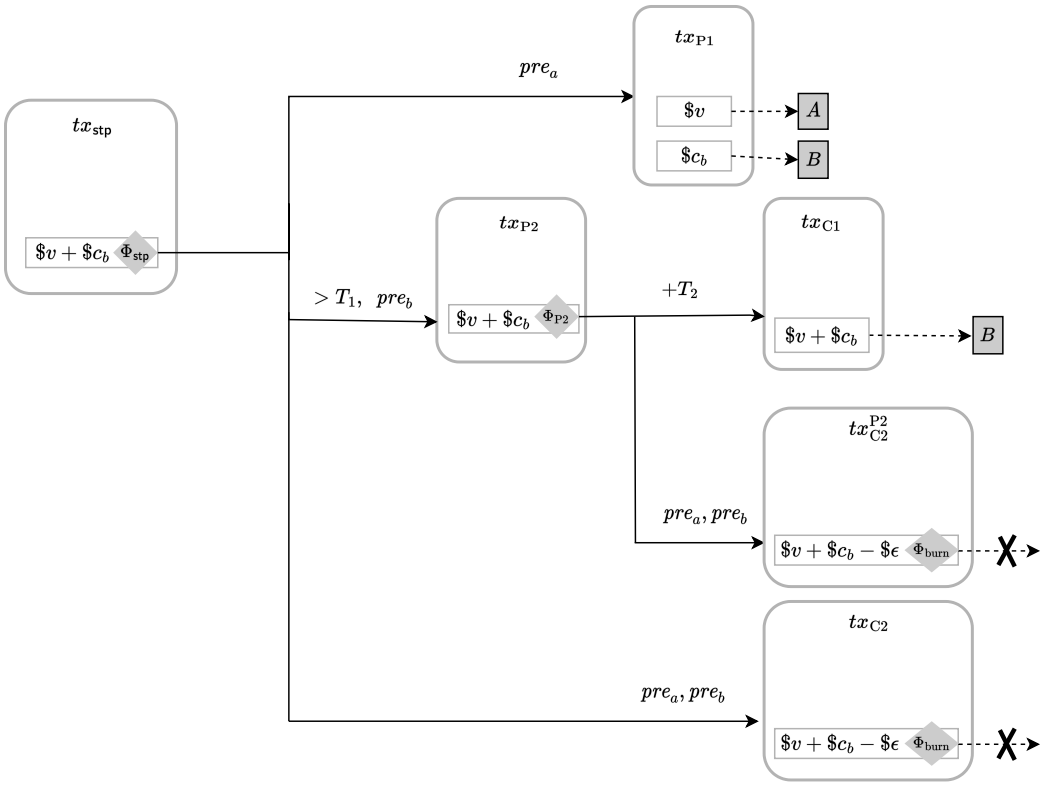

Figure 2: The transaction flow of RAPIDASH in Bitcoin absent external incentives. Rounded boxes denote transactions, rectangles within are outputs of the transaction. Incoming arrows denote transaction inputs, outgoing arrows denote how an output can be spent by a transaction at the end of the arrow. Solid lines indicate the transaction output can be spent only if both users sign the spending transaction. Dashed arrows indicate that the output can be spent by one user (A for Alice, and B for Bob). The timelock (T_1 and T_2) associated with a transaction output is written over the corresponding outgoing arrow.

transactions $tx_{\mathrm{C2}}^{\mathrm{P2}}$ and tx_{C2} which can be redeemed only if pre_a, pre_b and pre_c are revealed, such that $H(pre_a) = h_a, H(pre_b) = h_b$, and $H(pre_c) = h_c$. We have an additional transaction $tx_{\mathrm{C2}}^{\mathrm{P2,empty}}$ that redeems the coins from the auxiliary address of $tx_{\mathrm{P2}}^{\mathrm{empty}}$ if pre_a, pre_b and pre_c are revealed. Unlike the single instance, here we replace T_2 with τ. A pictorial description of the transaction flow is described in Figure 4.

Protocol Flow. Alice and bob first agree on the setup transaction tx_{stp} and sign the redeeming transactions $tx_{\mathrm{P1}}, tx_{\mathrm{P2}}, tx_{\mathrm{C1}}, tx_{\mathrm{C2}}^{\mathrm{P2}}, tx_{\mathrm{C1}}^{\mathrm{empty}}, tx_{\mathrm{C2}}^{\mathrm{P2,empty}}$ and tx_{C2} and broadcast all these transactions and the respective signatures, like before. They sign the transaction $tx_{\mathrm{P2}}^{\mathrm{empty}}$ such that only Alice has both signatures and she keeps them privately for now. Finally, they sign the setup transaction tx_{stp} and publish it on the blockchain, starting the execution phase.

Whenever Alice wishes to activate P2 branch with an empty message call, she publishes the transaction $tx_{\mathrm{P2}}^{\mathrm{empty}}$ along with the valid signatures she has in her possession. If $tx_{\mathrm{P2}}^{\mathrm{empty}}$ is published on the blockchain, activation point C1 can be activated by $tx_{\mathrm{C1}}^{\mathrm{empty}}$ after a timeout of τ time units. The rest of the protocol proceeds exactly as the description of the swap protocol.

Table 6: Description of additional transactions in Bitcoin for RAPIDASH atomic swap with CSP fairness. Here Φ^B is a script that requires a signature from Bob's public key, respectively.

	Description
$tx_{\text{P2}}^{\text{empty}}$	$tx\begin{pmatrix}[(Adr_{\mathsf{stp}}, \Phi_{\mathsf{stp}}, \$x + \$c_b)],\\ [(Adr_{\text{P2}}, \Phi_{\text{P2}}, \$x + \$c_b)]\end{pmatrix}$
$tx_{\text{C1}}^{\text{empty}}$	$tx\begin{pmatrix}[(Adr_{\text{P2}}, \Phi_{\text{P2}}, \$x + \$c_b)],\\ [(Adr_2^B, \Phi^B, \$x + \$c_b)]\end{pmatrix}$
$tx_{\text{C2}}^{\text{P2,empty}}$	$tx\begin{pmatrix}[(Adr_{\text{P2}}, \Phi_{\text{P2}}, \$x + \$c_b)],\\ [(Adr_{\mathsf{burn}}, \Phi_{\mathsf{burn}}, \$x + \$c_b - \$\epsilon)]\end{pmatrix}$

$\Phi_{\mathsf{stp}}(tx, pre_a, pre_b, \sigma_a, \sigma_b)$

P1: **if** $(H(pre_a) = h_a) \wedge (H(pre_c) = h_c) \wedge$
$(\mathsf{Vf}(\mathsf{pk}_a, tx, \sigma_a) = 1) \ \wedge (\mathsf{Vf}(\mathsf{pk}_b, tx, \sigma_b) = 1)$
then return 1

P2: **if** $(\_\text{NOW} > T_1) \wedge (H(pre_b) = h_b)$
$\wedge (\mathsf{Vf}(\mathsf{pk}'_a, tx, \sigma_a) = 1) \wedge (\mathsf{Vf}(\mathsf{pk}'_b, tx, \sigma_b) = 1)$
then return 1

E2: **if** $(\_\text{NOW} > T_1) \wedge (\mathsf{Vf}(\mathsf{pk}^3_a, tx, \sigma_b) = 1) \wedge (\mathsf{Vf}(\mathsf{pk}^3_b, tx, \sigma_b) = 1)$
then return 1

C2: **if** $(\mathsf{Vf}(\mathsf{pk}''_a, tx, \sigma_a) = 1) \wedge (\mathsf{Vf}(\mathsf{pk}''_b, tx, \sigma_b) = 1) \wedge$
$\Big((H(pre_a) = h_a) \wedge (H(pre_b) = h_b) \wedge (H(pre_c) = h_c)\Big)$
then return 1

// Values $h'_a, h_a, h_b, h_c, \mathsf{pk}_a, \mathsf{pk}_b, T_1, \mathsf{pk}'_a, \mathsf{pk}'_b, \mathsf{pk}''_a, \mathsf{pk}''_b, \mathsf{pk}^3_a, \mathsf{pk}^3_b$ are hardwired

$\Phi_{\text{P2}}(tx, pre'_a, pre_b, pre_a, pre_c, \sigma_a, \sigma_b)$

C1: **if** $(\_\text{NOW} > \tau) \wedge (\mathsf{Vf}(\mathsf{pk}_a, tx, \sigma_a) = 1) \wedge (\mathsf{Vf}(\mathsf{pk}_b, tx, \sigma_b) = 1)$
then return 1

C2: **if** $(\mathsf{Vf}(\mathsf{pk}'_a, tx, \sigma_a) = 1) \wedge (\mathsf{Vf}(\mathsf{pk}'_b, tx, \sigma_b) = 1) \wedge$
$\Big((H(pre_a) = h_a) \wedge (H(pre_b) = h_b) \wedge (H(pre_c) = h_c)\Big)$
then return 1

// Values $\tau, h_a, h_b, h_c, \mathsf{pk}_a, \mathsf{pk}_b, \mathsf{pk}'_a, \mathsf{pk}'_b$ are hardwired

Figure 3: The description of script Φ_{stp} and Φ_{P2} for atomic swap with CSP fairness. Here tx is the transaction spending from the script. Keys $(\mathsf{pk}_a, \mathsf{pk}'_a, \mathsf{pk}''_a, \mathsf{pk}^3_a)$ and $(\mathsf{pk}_b, \mathsf{pk}'_b, \mathsf{pk}''_b, \mathsf{pk}^3_b)$ belong to Alice and Bob, respectively.

8.3.2 Instantiating Rapidash' from Section 5

We describe all the transactions, addresses and scripts needed in the RAPIDASH' instantiation for the atomic swap case. Notice that the roles of Alice and Bob are reversed compared to RAPIDASH above. Specifically, in RAPIDASH', Bob can use pre'_b to retrieve the coins from the payment address,

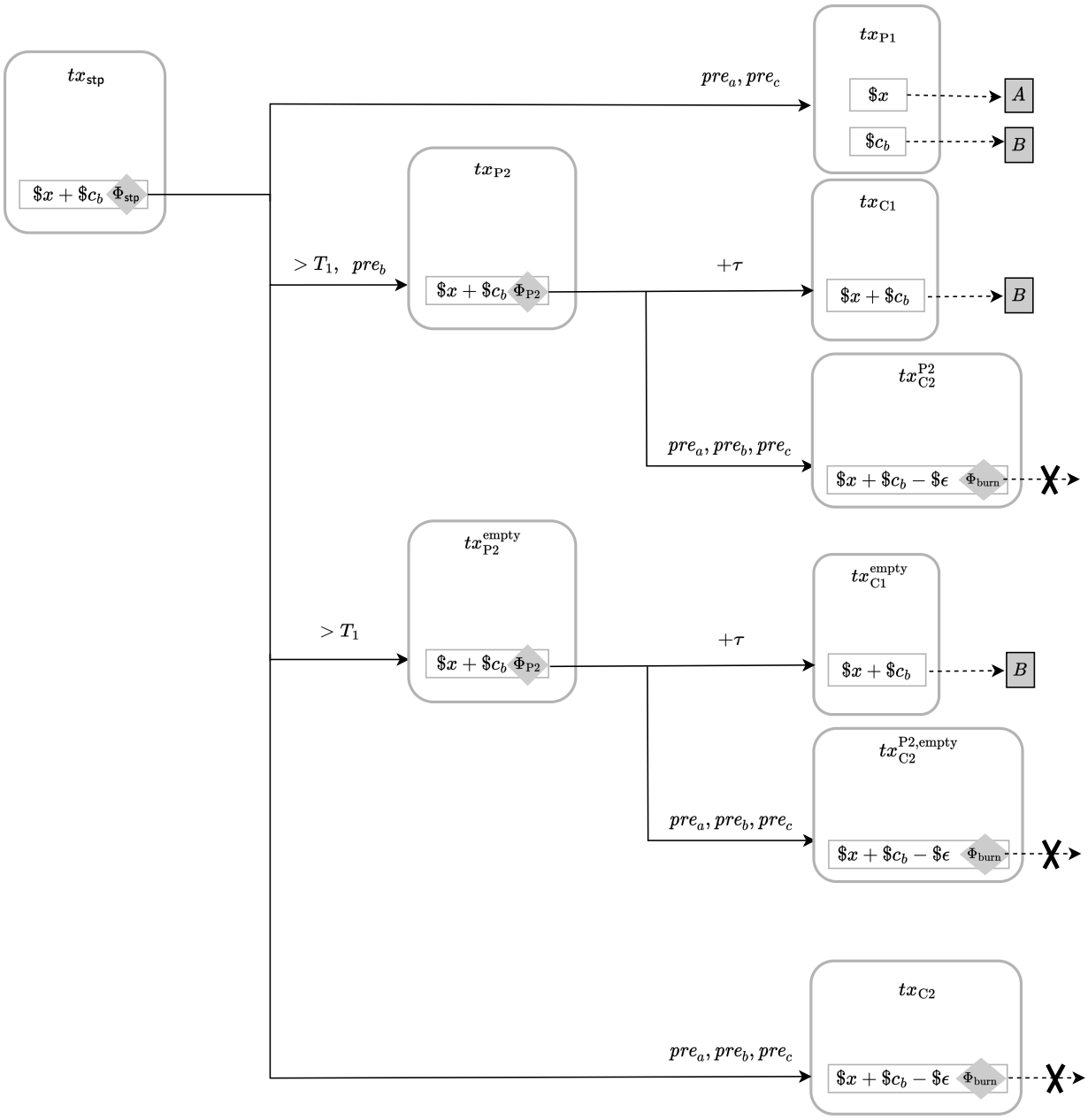

Figure 4: The transaction flow of RAPIDASH in Bitcoin for atomic swap with CSP fairness. Rounded boxes denote transactions, rectangles within are outputs of the transaction. Incoming arrows denote transaction inputs, outgoing arrows denote how an output can be spent by a transaction at the end of the arrow. Solid lines indicate the transaction output can be spent only if both users sign the spending transaction. Dashed arrows indicate that the output can be spent by one user (A for Alice, and B for Bob).

while Alice can use pre'_a after a timeout of T'_1 to retrieve the coins. The main difference between this instantiation and the RAPIDASH instantiation above is that in the execution phase both the payment address activation points P1′ and P2′ can be activated by empty message calls. We also have modified collateral redeem transactions that redeem the coins from the C2′ branch of the Φ'_{stp}.

Transactions. We describe below the different transactions needed for our RAPIDASH' instantia-

Table 7: Description of additional transaction in Bitcoin for RAPIDASH' atomic swap with CSP fariness. Here Φ^A and Φ^B are scripts that require a signature from Alice's and Bob's public key, respectively.

	Description
$tx_{\mathrm{P1}'}^{\mathrm{empty}}$	$tx\left(\begin{array}{c}[(Adr'_{\mathsf{stp}}, \Phi'_{\mathsf{stp}}, \$x' + \$c'_a + \$c'_b)], \\ [(Adr_1^A, \Phi^A, \$c'_a), (Adr_1^B, \Phi^B, \$x' + \$c'_b)]\end{array}\right)$

$\Phi'_{\mathsf{stp}}(tx, pre'_a, pre'_b, pre_b, \sigma_a, \sigma_b)$

P1': **if** $(H(pre'_b) = h'_b) \wedge (\mathsf{Vf}(\mathsf{pk}_a, tx, \sigma_a) = 1)$
$\wedge (\mathsf{Vf}(\mathsf{pk}_b, tx, \sigma_b) = 1)$ **then return** 1

P2': **if** $(\_\mathrm{NOW} > T'_1) \wedge (H(pre'_a) = h'_a)$
$\wedge (\mathsf{Vf}(\mathsf{pk}'_a, tx, \sigma_a) = 1) \wedge (\mathsf{Vf}(\mathsf{pk}'_b, tx, \sigma_b) = 1)$
thenreturn 1

E1': **if** $(\mathsf{Vf}(\mathsf{pk}''_a, tx, \sigma_a) = 1) \wedge (\mathsf{Vf}(\mathsf{pk}''_b, tx, \sigma_b) = 1)$
then return 1

E2': **if** $(\_\mathrm{NOW} > T'_1) \wedge (\mathsf{Vf}(\mathsf{pk}^3_a, tx, \sigma_b) = 1) \wedge$
$(\mathsf{Vf}(\mathsf{pk}^3_b, tx, \sigma_b) = 1)$ **then return** 1

C2': **if** $(\mathsf{Vf}(\mathsf{pk}^4_a, tx, \sigma_a) = 1) \wedge (\mathsf{Vf}(\mathsf{pk}^4_b, tx, \sigma_b) = 1) \wedge$
$\Big(\Big((H(pre'_a) = h'_a) \wedge (H(pre_b) = h_b)\Big) \vee \Big((H(pre'_a) = h'_a) \wedge (H(pre'_b) = h'_b)\Big)\Big)$
then return 1

// Values $h'_a, h'_b, h_b, \mathsf{pk}_a, \mathsf{pk}_b, T_1, \mathsf{pk}'_a, \mathsf{pk}'_b, \mathsf{pk}''_a, \mathsf{pk}''_b, \mathsf{pk}^3_a, \mathsf{pk}^3_b, \mathsf{pk}^4_a, \mathsf{pk}^4_b$ are hardwired

$\Phi_{\mathrm{P2}'}(tx, pre'_a, pre_b, pre'_b, \sigma_a, \sigma_b)$

C1': **if** $(\_\mathrm{NOW} > \tau') \wedge (\mathsf{Vf}(\mathsf{pk}_a, tx, \sigma_a) = 1) \wedge (\mathsf{Vf}(\mathsf{pk}_b, tx, \sigma_b) = 1)$
then return 1

C2': **if** $(\mathsf{Vf}(\mathsf{pk}^4_a, tx, \sigma_a) = 1) \wedge (\mathsf{Vf}(\mathsf{pk}^4_b, tx, \sigma_b) = 1) \wedge$
$\Big(\Big((H(pre'_a) = h'_a) \wedge (H(pre_b) = h_b)\Big) \vee \Big((H(pre'_a) = h'_a) \wedge (H(pre'_b) = h'_b)\Big)\Big)$
then return 1

// Values $\tau', h'_a, h_b, h'_b, \mathsf{pk}_a, \mathsf{pk}_b, \mathsf{pk}'_a, \mathsf{pk}'_b$ are hardwired

Figure 5: The description of script Φ'_{stp} for RAPIDASH' in atomic swap with CSP fairness.

tion. We have the same set of transactions that are analog of the RAPIDASH instantiation, except for one additional transaction $tx_{\mathrm{P1}'}^{\mathrm{empty}}$ (see Table 7). The transaction redeems the coins from the payment address Adr'_{stp} using the (E1') branch of Φ'_{stp}. The description of Φ'_{stp} is given below in Figure 5 with Alice and Bob's roles being reversed in RAPIDASH'. This transaction will correspond to the empty message call to RAPIDASH' activation point P1'. The script Φ'_{stp} (and correspondingly $\Phi_{\mathrm{P2}'}$) has a modification in the C2' branch, where we require either (pre'_a, pre'_b) or (pre'_a, pre_b) along with the signatures of Alice and Bob. We have the corresponding redeeming transactions as $tx_{\mathrm{C2}'}^{\mathrm{P2}'}$, $tx_{\mathrm{C2}'}^{\mathrm{P2}',\mathrm{empty}}$ and $tx_{\mathrm{C2}'}$ similar to RAPIDASH. A pictorial description of the transaction flow for

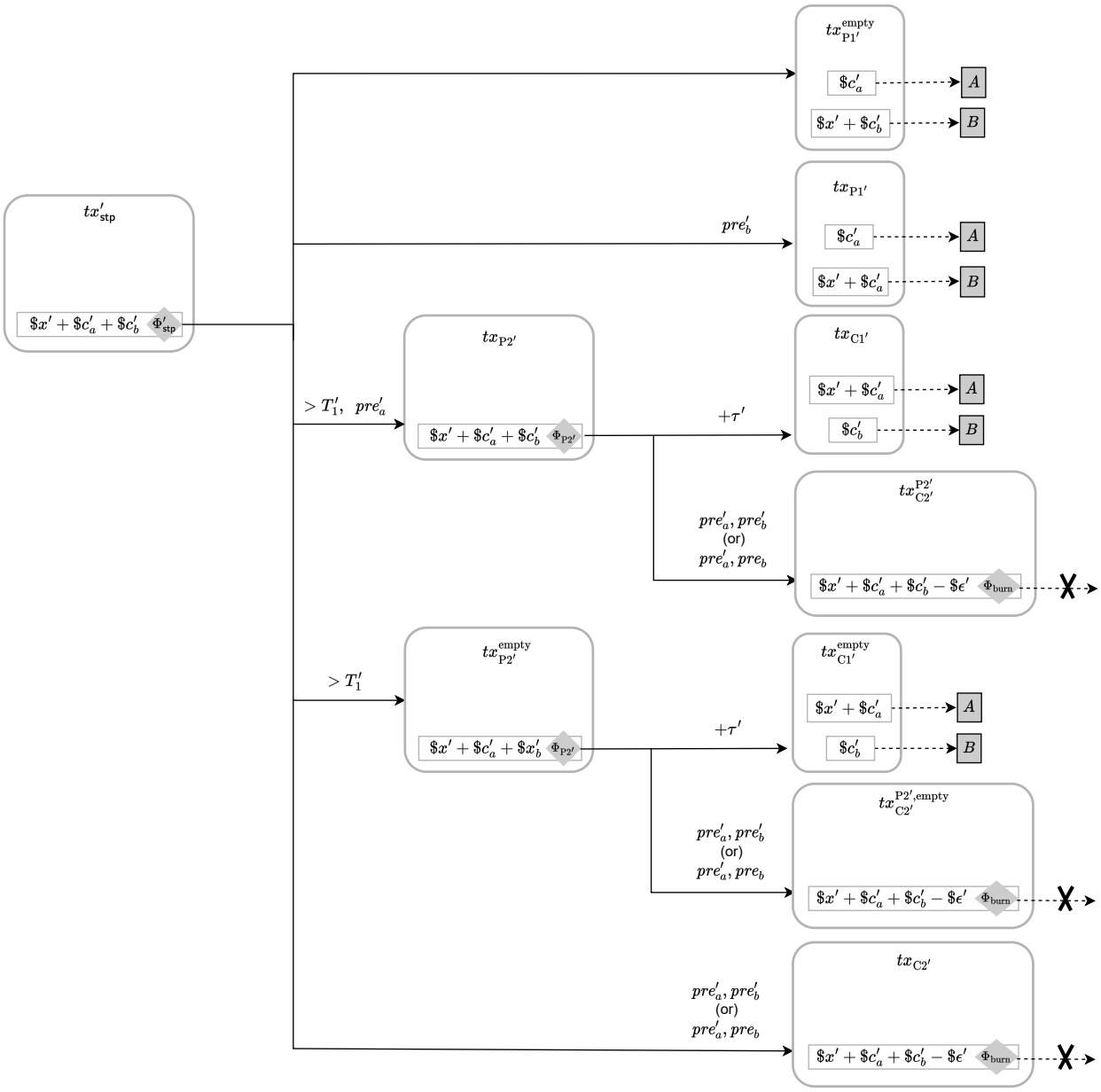

Figure 6: The transaction flow of RAPIDASH' in Bitcoin for atomic swap with CSP fairness. Rounded boxes denote transactions, and rectangles within are outputs of the transaction. Incoming arrows denote transaction inputs, outgoing arrows denote how an output can be spent by a transaction at the end of the arrow. Solid lines indicate the transaction output can be spent only if both users sign the spending transaction. Dashed arrows indicate that the output can be spent by one user (A for Alice, and B for Bob). The timelock (T'_1 and τ') associated with a transaction output is written over the corresponding outgoing arrow.

payment and collateral redeem is given in Figure 6.

Protocol Flow. Alice and Bob, first agree on the setup transaction tx'_{stp} and sign the redeeming transactions. They broadcast all these transactions and the respective signatures, like before. However, this time Alice and Bob sign the transaction $tx^{\text{empty}}_{\text{P1}'}$ such that only Alice has both

signatures. She does not broadcast the signatures and keeps them private. Similarly, Alice and Bob sign the transaction $tx^{\text{empty}}_{\text{P2}'}$ such that only Bob has both signatures. He keeps them private and does not broadcast them. Notice that none of the transactions can be published on the blockchain yet as the setup transaction is not yet published. Finally, they sign the setup transaction tx'_{stp} and publish it on the blockchain, thus starting the execution phase.

Whenever Alice wishes to activate P1′ in Rapidash' with an empty message, she publishes the transaction $tx^{\text{empty}}_{\text{P1}'}$ along with the valid signatures she has in her possession. Similarly, whenever Bob wishes to activate P2′ in Rapidash' with an empty message, he publishes the transaction $tx^{\text{empty}}_{\text{P2}'}$ along with the valid signatures he has in his possession. If $tx^{\text{empty}}_{\text{P2}'}$ is published on the blockchain, activation point C1′ can be activated by $tx^{\text{empty}}_{\text{C1}'}$ after a timeout of τ' time units. Rest of the flow follows exactly the description of the atomic swap protocol.

8.3.3 Instantiating Rapidash from Section 6.1

We have minor differences compared to the Rapidash instantiation from Section 8.3.1 including the fact that Alice additionally locks collateral of c_a. Another point of difference is the setup transaction tx_{stp} additionally transfers a small amount $\$\eta$ coins (for example $\$\eta = 1$ Satoshi) to an output address Adr_{B1}. This new address will aid us in realizing the activation points B1 and B2.

Transactions. We describe below the different transactions needed for our Rapidash instantiation. In addition to the above transaction, we have two new transactions tx_{B1} and tx_{B2}, corresponding to the activation points B1 and B2, respectively. Transaction tx_{B1} redeems $\$\eta$ coins from the address Adr_{B1} provided a timeout of T_1 has passed since the setup transaction was published on the blockchain. The other transaction tx_{B2} redeems the $\$\eta$ coins from Adr_{B1}, as well as the coins from the setup address Adr_{stp} provided pre_c is released. The transaction burns $(\$x + \$c_a + \$c_b + \$\eta - \$\epsilon)$ coins leaving behind $\$\epsilon$ coins as transaction fees. The description of the script Φ_{B1} associated with Adr_{B1} is given below in Figure 7.

A pictorial description of the transaction flow is described in Figure 8.

Table 8: Description of additional transactions in Bitcoin for Rapidash atomic swap with bounded maximin fairness. Here Φ^A and Φ^B are scripts that require a signature from Alice's public key and Bob's public key, respectively. Other transactions are the same as in prior Rapidash instantiations.

	Description
tx_{stp}	$tx\begin{pmatrix}[(Adr_0^A, \Phi^A, \$c_a)(Adr_0^B, \Phi^B, \$x + \$c_b + \$\eta)],\\ [(Adr_{\text{B1}}, \Phi_{\text{B1}}, \$\eta)(Adr_{\mathsf{stp}}, \Phi_{\mathsf{stp}}, \$v + \$c_a + \$c_b)]\end{pmatrix}$
tx_{B1}	$tx\begin{pmatrix}[(Adr_{\text{B1}}, \Phi_{\text{B1}}, \$\eta)],\\ [(Adr_B, \Phi^B, \$\eta)]\end{pmatrix}$
tx_{B2}	$tx\begin{pmatrix}[(Adr_{\text{B1}}, \Phi_{\text{B1}}, \$\eta), (Adr_{\mathsf{stp}}, \Phi_{\mathsf{stp}}, \$x + \$c_a + \$c_b)],\\ [(Adr_{\mathsf{burn}}, \Phi_{\mathsf{burn}}, \$x + \$c_a + \$c_b + \$\eta - \$\epsilon)]\end{pmatrix}$

Protocol Flow. Alice and bob first agree on the setup transaction tx_{stp} and sign all other redeeming transactions. They ensure that the signatures on the transaction tx_{B1} are only with Bob, and the signatures on $tx^{\text{empty}}_{\text{P2}}$ are only with Alice. Alice and Bob keep these signatures privately and broadcast all other transactions and signatures into the network. By posting the setup transaction on the blockchain, we enter the execution phase.

Whenever Bob wishes to activate B1 branch of Rapidash, he publishes tx_{B1} along with the corresponding signatures on the blockchain. Notice that since tx_{B1} and tx_{B2} spend from the address

$\Phi_{\text{B1}}(tx, pre_c, \sigma_a, \sigma_b)$

B1: **if** $(\_\text{NOW} > T_1) \wedge (\mathsf{Vf}(\mathsf{pk}_a, tx, \sigma_a) = 1) \ \wedge (\mathsf{Vf}(\mathsf{pk}_b, tx, \sigma_b) = 1)$
then return 1

B2: **if** $(H(pre_c) = h_c) \wedge (\mathsf{Vf}(\mathsf{pk}'_a, tx, \sigma_a) = 1) \ \wedge (\mathsf{Vf}(\mathsf{pk}'_b, tx, \sigma_b) = 1)$
then return 1

// Values $h_c, T_1, \mathsf{pk}_a, \mathsf{pk}_b, \mathsf{pk}'_a, \mathsf{pk}'_b$ are hardwired

$\Phi_{\mathsf{stp}}(tx, pre_a, pre_b, \sigma_a, \sigma_b)$

B2: **if** $(H(pre_c) = h_c) \wedge (\mathsf{Vf}(\mathsf{pk}_a, tx, \sigma_a) = 1) \ \wedge (\mathsf{Vf}(\mathsf{pk}_b, tx, \sigma_b) = 1)$
then return 1

P1: **if** $(H(pre_a) = h_a) \wedge (H(pre_c) = h_c) \wedge$
$(\mathsf{Vf}(\mathsf{pk}'_a, tx, \sigma_a) = 1) \ \wedge (\mathsf{Vf}(\mathsf{pk}'_b, tx, \sigma_b) = 1)$
then return 1

P2: **if** $(\_\text{NOW} > T_1) \wedge (H(pre_b) = h_b)$
$\wedge (\mathsf{Vf}(\mathsf{pk}''_a, tx, \sigma_a) = 1) \wedge (\mathsf{Vf}(\mathsf{pk}''_b, tx, \sigma_b) = 1)$
then return 1

E2: **if** $(\_\text{NOW} > T_1) \wedge (\mathsf{Vf}(\mathsf{pk}^3_a, tx, \sigma_b) = 1) \wedge (\mathsf{Vf}(\mathsf{pk}^3_b, tx, \sigma_b) = 1)$
then return 1

C2: **if** $(\mathsf{Vf}(\mathsf{pk}^4_a, tx, \sigma_a) = 1) \wedge (\mathsf{Vf}(\mathsf{pk}^4_b, tx, \sigma_b) = 1) \wedge$
$\Big((H(pre_a) = h_a) \wedge (H(pre_b) = h_b) \wedge (H(pre_c) = h_c)\Big)$
then return 1

// Values $h'_a, h_a, h_b, h_c, \mathsf{pk}_a, \mathsf{pk}_b, T_1 \mathsf{pk}'_a, \mathsf{pk}'_b, \mathsf{pk}''_a, \mathsf{pk}''_b$
// $\mathsf{pk}^4_a, \mathsf{pk}^4_b, \mathsf{pk}^5_a, \mathsf{pk}^5_b$ are hardwired

Figure 7: The description of script Φ_{stp} and Φ_{B1} for an atomic swap with bounded maximin fairness. Here tx is the transaction spending from the script. Keys $(\mathsf{pk}_a, \mathsf{pk}'_a, \mathsf{pk}''_a, \mathsf{pk}^3_a, \mathsf{pk}^4_a, \mathsf{pk}^5_a)$ and $(\mathsf{pk}_b, \mathsf{pk}'_b, \mathsf{pk}''_b, \mathsf{pk}^3_b, \mathsf{pk}^4_b, \mathsf{pk}^5_b)$ belong to Alice and Bob, respectively. The script Φ_{P2} is the same as in Figure 3.

Adr_{B1}, they are mutually exclusive, meaning only one of them can be posted on the blockchain. Notice that any of the other branches can be activated if and only if B1 was activated (or in other words B2 was not activated with tx_{B2}). The rest of the protocol proceeds exactly as the description of the swap protocol and previously described in Section 8.3.1.

8.3.4 Instantiating Rapidash' from Section 6.1

Similar to the case of instantiations for CSP fairness, the roles of Alice and Bob are reversed. The only difference between the instantiation here and the one for Rapidash' with CSP fairness from Section 8.3.2 is the addition of two transactions that activate A1′ and A2′ branches of Rapidash'. Similar to the Rapidash instantiation with bounded maximin fairness, we have an additional address $Adr_{\text{A1}'}$ that is created by the setup transaction to facilitate the activation points A1′ and A2′.

Transactions. We describe below the two additional transactions $tx_{\text{A1}'}$ and $tx_{\text{A2}'}$ that activate A1′ and A2′, respectively. Transaction $tx_{\text{A1}'}$ spends the $\$\eta$ coins from the address $Adr_{\text{A1}'}$ after a

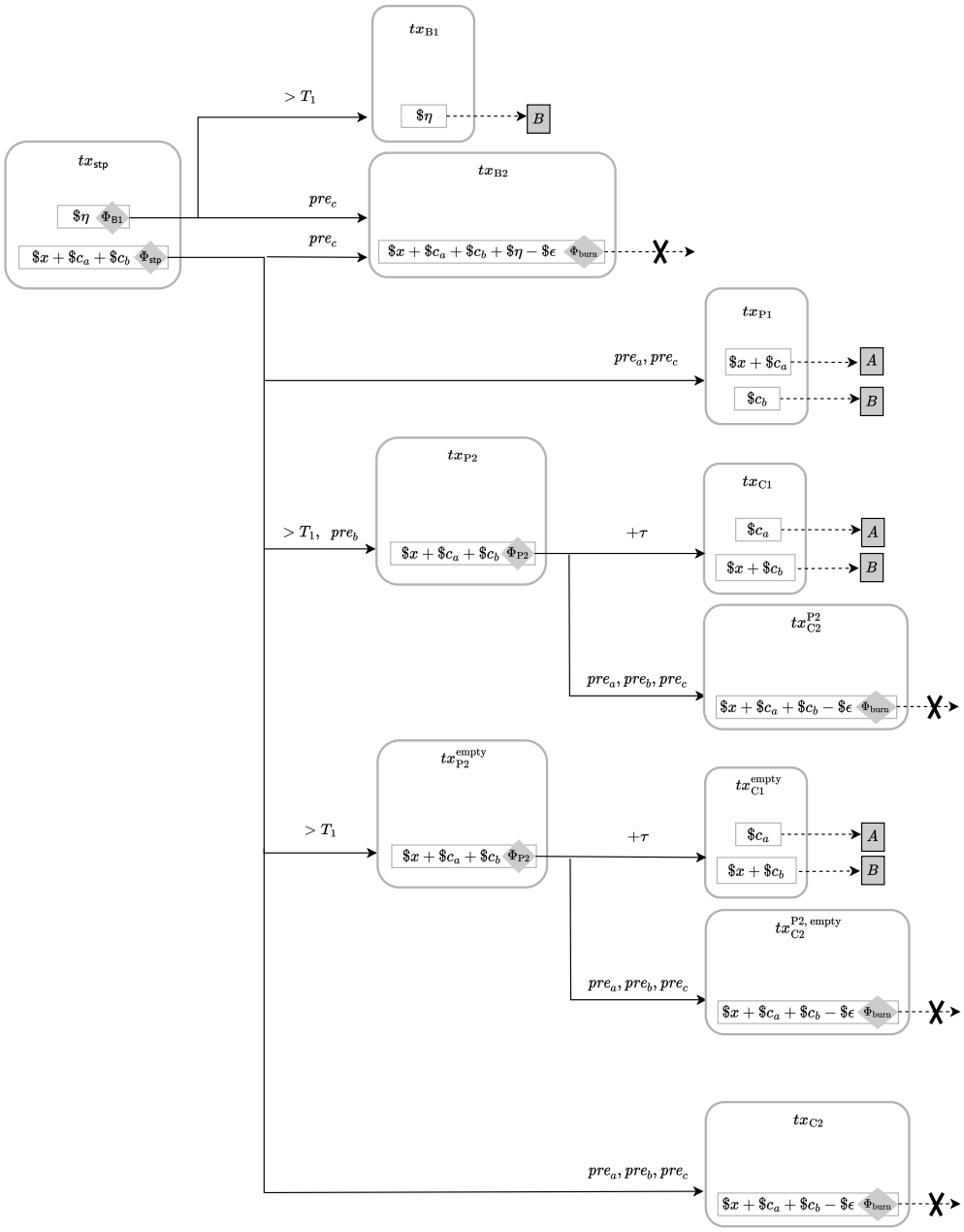

Figure 8: The transaction flow of RAPIDASH in Bitcoin for an atomic swap with bounded maximin fairness. Rounded boxes denote transactions, and rectangles within are outputs of the transaction. Incoming arrows denote transaction inputs, outgoing arrows denote how an output can be spent by a transaction at the end of the arrow. Solid lines indicate the transaction output can be spent only if both users sign the spending transaction. Dashed arrows indicate that the output can be spent by one user (A for Alice, and B for Bob).

timeout of T_1' to an address of Alice. Transaction $tx_{\text{A2}'}$ redeems the coins from both $Adr_{\text{A1}'}$ and Adr'_{stp} and burns all but ϵ' coins. This transaction requires either pre'_a or pre_b to be revealed along with the signatures of Alice and Bob. The transactions are described in Table 9 and the script $\Phi_{\text{A1}'}$ and the modified script Φ'_{stp} are described in Figure 9. A pictorial description of the transaction flow is described in Figure 10.

Table 9: Description of additional transaction in Bitcoin for RAPIDASH' atomic swap with bounded maximin fairness. Here Φ^A is the script that requires a signature from Alice's public key.

	Description
tx'_{stp}	$tx\begin{pmatrix}[(Adr_0^A, \Phi^A, \$x' + \$c_a + \$\eta)(Adr_0^B, \Phi^B, \$c_b)],\\ [(Adr_{\mathrm{A1}'}, \Phi_{\mathrm{A1}'}, \$\eta)(Adr'_{\mathsf{stp}}, \Phi'_{\mathsf{stp}}, \$x' + \$c'_a + \$c'_b)]\end{pmatrix}$
$tx_{\mathrm{A1}'}$	$tx\begin{pmatrix}[(Adr_{\mathrm{A1}'}, \Phi_{\mathrm{A1}'}, \$\eta)],\\ [(Adr_A, \Phi^A, \$\eta)]\end{pmatrix}$
$tx_{\mathrm{A2}'}$	$tx\begin{pmatrix}[(Adr_{\mathrm{A1}'}, \Phi_{\mathrm{A1}'}, \$\eta), (Adr'_{\mathsf{stp}}, \Phi'_{\mathsf{stp}}, \$x' + \$c'_a + \$c'_b)],\\ [(Adr_{\mathsf{burn}}, \Phi_{\mathsf{burn}}, \$x' + \$c'_a + \$c'_b + \$\eta - \$\epsilon')]\end{pmatrix}$

$\Phi_{\mathrm{A1}'}(tx, pre'_a, pre_b, \sigma_a, \sigma_b)$

A1': **if** $(\_\mathrm{NOW} > T'_1) \wedge (\mathsf{Vf}(\mathsf{pk}_a, tx, \sigma_a) = 1) \ \wedge (\mathsf{Vf}(\mathsf{pk}_b, tx, \sigma_b) = 1)$
then return 1

A2': **if** $((H(pre'_a) = h'_a) \vee (H(pre_b) = h_b)) \wedge$
$(\mathsf{Vf}(\mathsf{pk}'_a, tx, \sigma_a) = 1) \ \wedge (\mathsf{Vf}(\mathsf{pk}'_b, tx, \sigma_b) = 1)$
then return 1

// Values $h'_a, h_b, T'_1, \mathsf{pk}_a, \mathsf{pk}_b, \mathsf{pk}'_a, \mathsf{pk}'_b$ are hardwired

$\Phi'_{\mathsf{stp}}(tx, pre'_a, pre'_b, pre_b, \sigma_a, \sigma_b)$

A2': **if** $((H(pre'_a) = h'_a) \vee (H(pre_b) = h_b)) \wedge$
$(\mathsf{Vf}(\mathsf{pk}_a, tx, \sigma_a) = 1) \ \wedge (\mathsf{Vf}(\mathsf{pk}_b, tx, \sigma_b) = 1)$
then return 1

P1': **if** $(H(pre'_b) = h'_b) \wedge (\mathsf{Vf}(\mathsf{pk}'_a, tx, \sigma_a) = 1)$
$\wedge (\mathsf{Vf}(\mathsf{pk}'_b, tx, \sigma_b) = 1)$ **then return** 1

P2': **if** $(\_\mathrm{NOW} > T'_1) \wedge (H(pre'_a) = h'_a)$
$\wedge (\mathsf{Vf}(\mathsf{pk}''_a, tx, \sigma_a) = 1) \wedge (\mathsf{Vf}(\mathsf{pk}''_b, tx, \sigma_b) = 1)$
then return 1

E1': **if** $(\mathsf{Vf}(\mathsf{pk}^3_a, tx, \sigma_a) = 1) \wedge (\mathsf{Vf}(\mathsf{pk}^3_b, tx, \sigma_b) = 1)$
then return 1

E2': **if** $(\_\mathrm{NOW} > T'_1) \wedge (\mathsf{Vf}(\mathsf{pk}^4_a, tx, \sigma_b) = 1) \wedge$
$(\mathsf{Vf}(\mathsf{pk}^4_b, tx, \sigma_b) = 1)$ **then return** 1

C2': **if** $(\mathsf{Vf}(\mathsf{pk}^5_a, tx, \sigma_a) = 1) \wedge (\mathsf{Vf}(\mathsf{pk}^5_b, tx, \sigma_b) = 1) \ \wedge$
$\Big(\big((H(pre'_a) = h'_a) \wedge (H(pre_b) = h_b)\big) \vee \big((H(pre'_a) = h'_a) \wedge (H(pre'_b) = h'_b)\big)\Big)$
then return 1

// Values $h'_a, h'_b, h_b, \mathsf{pk}_a, \mathsf{pk}_b, T_1, \mathsf{pk}'_a, \mathsf{pk}'_b, \mathsf{pk}''_a, \mathsf{pk}''_b, \mathsf{pk}^3_a, \mathsf{pk}^3_b, \mathsf{pk}^4_a, \mathsf{pk}^4_b, \mathsf{pk}^5_a, \mathsf{pk}^5_b$ are hardwired

Figure 9: The description of scripts $\Phi_{\mathrm{A1}'}$ and Φ'_{stp} for RAPIDASH' in atomic swap with bounded maximin fairness.

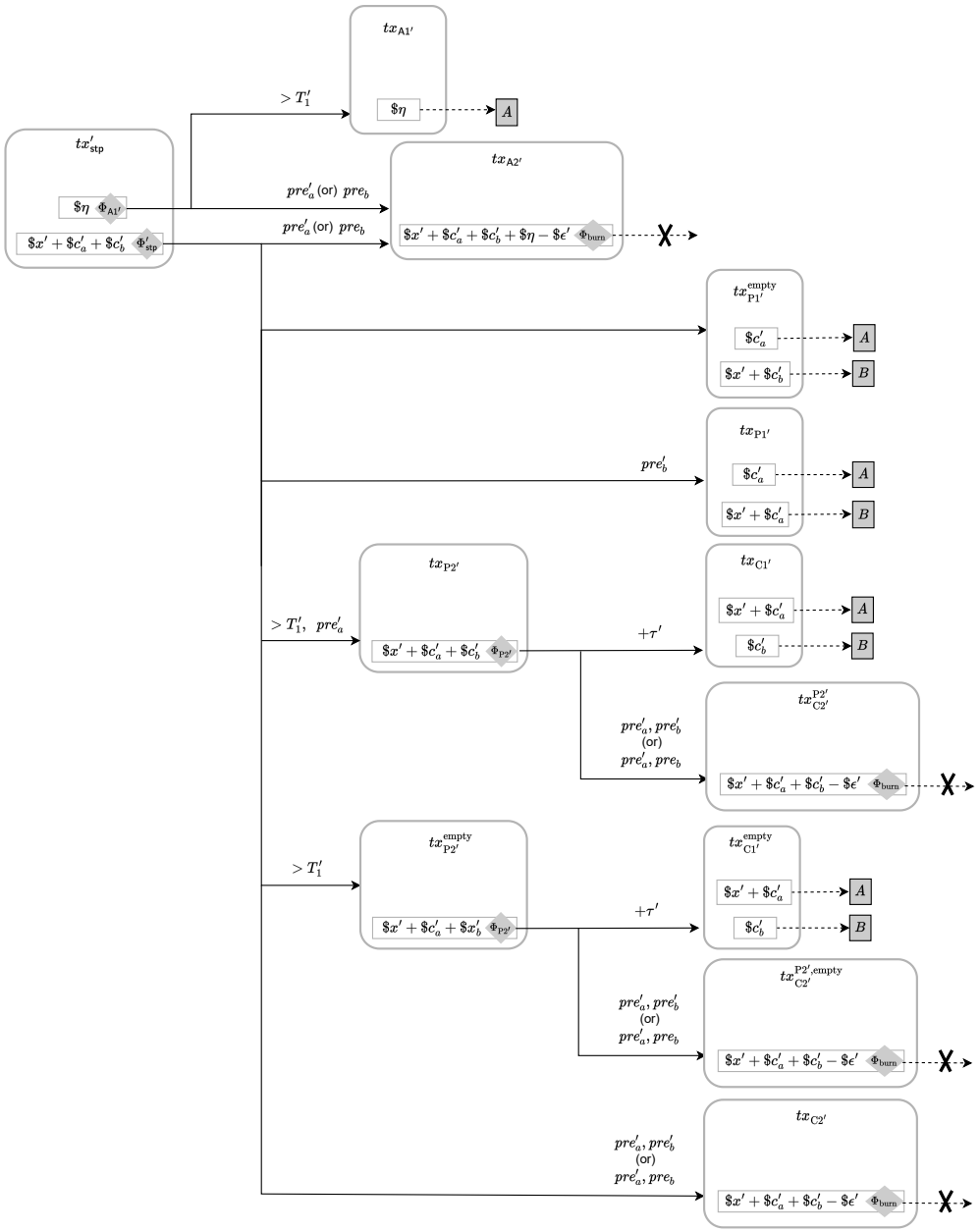

Figure 10: The transaction flow of RAPIDASH' in Bitcoin for an atomic swap with bounded maximin fairness. Rounded boxes denote transactions, and rectangles within are outputs of the transaction. Incoming arrows denote transaction inputs, outgoing arrows denote how an output can be spent by a transaction at the end of the arrow. Solid lines indicate the transaction output can be spent only if both users sign the spending transaction. Dashed arrows indicate that the output can be spent by one user (A for Alice, and B for Bob). The timelock (T_1' and τ') associated with a transaction output is written over the corresponding outgoing arrow.

Protocol Flow. We proceed exactly as the RAPIDASH' instantiation from Section 8.3.2, except for the two additional transactions in $tx_{A1'}$ and $tx_{A2'}$. Alice and Bob sign the two transactions prior to signing the setup transaction. Alice has the signatures on $tx_{A1'}$ that she keeps privately,

Table 10: Ethereum gas cost comparison.

Contract	Deploy (Gas)	Redeem path	Gas
HTLC	380,159	Alice redeem	35,851
		Bob redeem	34,932
MAD-HTLC	581,002	Optimistic case, Alice and Bob	102,505
		Refund, Bob	104,611
		Deposit bomb, Miner	61,008
		Collateral bomb, Miner	46,063
He-HTLC	894,346	Optimistic case, Alice and Bob	72,723
		Refund, Bob	123,337
		Collateral bomb, Miner	70,327
Rapidash	934,958	Optimistic case (P1), Alice and Bob	73,246
		Refund (P2 + C1), Bob	123,543
		Bomb (C2), Miner	70,327

while $tx_{\mathrm{A2'}}$ and the signatures on this transaction are broadcast to the network like in Section 8.3.2. After the setup transaction tx'_{stp} is published on the blockchain and the execution phase begins, whenever A1′ is to be activated, Alice publishes $tx_{\mathrm{A1'}}$ and the signatures on the blockchain. To activate A2′, transaction $tx_{\mathrm{A2'}}$ along with the corresponding signatures, and either pre'_a or pre_b are published on the blockchain. Notice that as required, only one of these two transactions can be posted allowing us to realize that A1′ and A2′ are mutually exclusive. The rest of the protocol proceeds as the description for the atomic swap and is similar to the one from Section 8.3.2.

9 Ethereum Instantiation

We implemented the knowledge-coin exchange, the CSP-fair atomic swap and the bounded maximin fair atomic swap Rapidash protocols from Section 4, Section 5, and Section 6.1 in Solidity, the smart contract language used in Ethereum which supports general smart contracts. We deployed each Rapidash protocol on Goerli, an Ethereum's testnet.

In Ethereum, the price of a transaction depends on its *gas* usage, which describes the cost of each operation performed by the transaction in units specific to Ethereum implementation. First, we evaluate our knowledge-coin exchange protocol, and compare its costs to those of HTLC, MAD-HTLC, and He-HTLC. Then, we evaluate our two atomic swap protocols.

9.1 Comparison of Rapidash's Knowledge-Coin Exchange to HTLC, MAD-HTLC, and He-HTLC

Our implementation of the knowledge-coin exchange Rapidash contract consists of a single contract which includes the initialization as well as all redeem paths. In Table 10, we compare the gas costs of various operations of the knowledge-coin exchange Rapidash contract with those of MAD-HTLC and HTLC. First, note that the gas cost for the initial deployment of each contract far outweighs those of the redeem transactions. As expected, Rapidash incurs deployment costs that are a bit higher than those of MAD-HTLC, as Rapidash contains slightly more code (80 LoC in Rapidash vs. 72 in MAD-HTLC). Also as expected, costs of Rapidash, are very similar to those of He-HTLC, which is concurrent to Rapidash. However, note that the total redeem cost in the

Table 11: CSP-fair atomic swap, gas cost.

Contract	Deploy (Gas)	Redeem path	Gas
Rapidash	1,097,177	Normal path (P1), Alice	52,279
		Normal path (P1), Bob	56,681
		Refund path (P2 + C1), Bob	123,631
		Bomb path (C2), Miner	42,266
Rapidash'	1,514,861	Input, Alice	50,465
		Input, Bob	55,817
		Withdraw, Alice	38,228
		Withdraw, Bob	35,911
		Optimistic case (P1'), Alice	54,904
		Optimistic case (P1'), Bob	58,656
		Refund (P2' + C1'), Alice	118,379
		Refund (P2' + C1'), Bob	114,647
		Bomb (C2'), Miner	53,431

optimistic case (when both Alice and Bob behave honestly) in Rapidash is actually lower than that of MAD-HTLC – 73,246 in Rapidash vs. 102,505 in MAD-HTLC, as the latter consists of Alice obtaining the deposit, and Bob retrieving the collateral.

9.2 Evaluation of Rapidash's Atomic Swap Protocols

For each atomic swap protocol, our implementation in Ethereum consists of two contracts, one for Rapidash, and one for Rapidash'. Each of these includes the initialization as well as all redeem paths. Note that only one of these contracts will be deployed on Ethereum, the other will be deployed on another blockchain [10].

In Table 11, we specify the gas costs of various operations of our CSP-fair atomic swap protocol. In Table 12, we show the gas costs of various operations of our atomic swap protocol which satisfies bounded maximin fairness. As expected, gas costs of the CSP-fair version are lower than those of the bounded maximin fairness version, as the former has slightly less code than the latter (252 LoC for the CSP-fair atomic swap, 340 LoC for the atomic swap which satisfies the bounded maximin fairness property).

References

[AHS22] Sepideh Avizheh, Preston Haffey, and Reihaneh Safavi-Naini. Privacy-preserving fairswap: Fairness and privacy interplay. *Proc. Priv. Enhancing Technol.*, 2022.

[Aso98] N. Asokan. *Fairness in Electronic Commerce*. PhD thesis, 1998.

[ASW97] N. Asokan, Matthias Schunter, and Michael Waidner. Optimistic protocols for fair exchange. In *ACM CCS*, 1997.

[atoa] Submarine swap in lightning network. https://wiki.ion.radar.tech/tech/research/submarine-swap.

[10] Users can choose which contract to employ on Ethereum, and which on another blockchain

Table 12: Bounded maximin fair atomic swap, gas cost.

Contract	Deploy (Gas)	Redeem path	Gas
RAPIDASH	1,617,281	Input, Alice	48,355
		Input, Bob	50,583
		Withdraw, Alice	35,956
		Withdraw, Bob	38,271
		Optimistic case (P1), Alice	35,405
		Optimistic case (B1 + P1), Bob	88,399
		Refund (P2 + C1), Alice	114,662
		Refund (B1 + P2 + C1), Bob	147,567
		Early Bomb (B2), Miner	50,295
		Bomb (C2), Miner	57,152
RAPIDASH'	1,514,861	Input, Alice	50,650
		Input, Bob	48,333
		Withdraw, Alice	38,251
		Withdraw, Bob	35,912
		Optimistic case (P1'), Alice	54,925
		Optimistic case (P1'), Bob	58,679
		Refund (A1' + P2' + C1'), Alice	149,634
		Refund (A1' + P2' + C1'), Bob	145,894
		Early bomb (A2'), Miner	49,956
		Bomb (C2'), Miner	53,475

[atob] What is atomic swap and how to implement it. https://www.axiomadev.com/blog/what-is-atomic-swap-and-how-to-implement-it/.

[BBSU12] Simon Barber, Xavier Boyen, Elaine Shi, and Ersin Uzun. Bitter to better – how to make bitcoin a better currency. In *Financial Cryptography and Data Security (FC)*, 2012.

[BDM16] Wacław Banasik, Stefan Dziembowski, and Daniel Malinowski. Efficient zero-knowledge contingent payments in cryptocurrencies without scripts. In *ESORICS*, 2016.

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

[BK] Sergiu Bursuc and Steve Kremer. Contingent payments on a public ledger: Models and reductions for automated verification. In *ESORICS 2019*.

[BK14] Iddo Bentov and Ranjit Kumaresan. How to Use Bitcoin to Design Fair Protocols. In *CRYPTO*, 2014.

[Bon16] Joseph Bonneau. Why buy when you can rent? In *FC*, 2016.

[BZ17] Massimo Bartoletti and Roberto Zunino. Constant-deposit multiparty lotteries on bitcoin. In *Financial Cryptography and Data Security*, 2017.

[CCWS21] Kai-Min Chung, T-H. Hubert Chan, Ting Wen, and Elaine Shi. Game-theoretic fairness meets multi-party protocols: The case of leader election. In *CRYPTO*, 2021.

[CGGN] Matteo Campanelli, Rosario Gennaro, Steven Goldfeder, and Luca Nizzardo. Zero-knowledge contingent payments revisited: Attacks and payments for services. In *ACM CCS 2017*.

[CGJ+17] Arka Rai Choudhuri, Matthew Green, Abhishek Jain, Gabriel Kaptchuk, and Ian Miers. Fairness in an unfair world: Fair multiparty computation from public bulletin boards. In *ACM CCS*, 2017.

[CGL+18] Kai-Min Chung, Yue Guo, Wei-Kai Lin, Rafael Pass, and Elaine Shi. Game theoretic notions of fairness in multi-party coin toss. In *TCC*, volume 11239, pages 563–596, 2018.

[CS21] Hao Chung and Elaine Shi. Foundations of transaction fee mechanism design, November 2021. arXiv:2111.03151. URL: https://arxiv.org/pdf/2111.03151.pdf.

[DEF18] Stefan Dziembowski, Lisa Eckey, and Sebastian Faust. Fairswap: How to fairly exchange digital goods. In *ACM CCS*, 2018.

[DEFM19] Stefan Dziembowski, Lisa Eckey, Sebastian Faust, and Daniel Malinowski. Perun: Virtual payment hubs over cryptocurrencies. In *IEEE Symposium on Security and Privacy*, 2019.

[DFH18] Stefan Dziembowski, Sebastian Faust, and Kristina Hostáková. General state channel networks. In *ACM CCS*, CCS '18, page 949–966, 2018.

[DW15] Christian Decker and Roger Wattenhofer. A fast and scalable payment network with bitcoin duplex micropayment channels. In *Stabilization, Safety, and Security of Distributed Systems*, 2015.

[EFS20] Lisa Eckey, Sebastian Faust, and Benjamin Schlosser. Optiswap: Fast optimistic fair exchange. In *ASIA CCS*, 2020.

[Eth22] Ethereum. The Solidity contract-oriented programming language, 2022. URL: https://github.com/ethereum/solidity.

[Fuc] Georg Fuchsbauer. Wi is not enough: Zero-knowledge contingent (service) payments revisited. In *ACM CCS 2019*.

[GKL15] Juan A. Garay, Aggelos Kiayias, and Nikos Leonardos. The bitcoin backbone protocol: Analysis and applications. In *Eurocrypt*, 2015.

[GKM+22] Vipul Goyal, Abhiram Kothapalli, Elisaweta Masserova, Bryan Parno, and Yifan Song. Storing and retrieving secrets on a blockchain. In *PKC*, 2022.

[GM] Matthew Green and Ian Miers. Bolt: Anonymous payment channels for decentralized currencies. In *ACM CCS 2017*.

[goe22] Goerli testnet, 2022. URL: https://goerli.net.

[Ham] Matthew Hammond. Blockchain interoperability series: Atomic swaps. https://medium.com/@mchammond/atomic-swaps-eebd0fa8110d.

[Her18] Maurice Herlihy. Atomic cross-chain swaps. In *PODC*, 2018.

[HZ20] Jona Harris and Aviv Zohar. Flood & loot: A systemic attack on the lightning network. In *AFT*, 2020.

[JMM14] Danushka Jayasinghe, Konstantinos Markantonakis, and Keith Mayes. Optimistic fair-exchange with anonymity for bitcoin users. In *International Conference on e-Business Engineering*, 2014.

[JSZ+21] Aljosha Judmayer, Nicholas Stifter, Alexei Zamyatin, Itay Tsabary, Ittay Eyal, Peter Gazi, Sarah Meiklejohn, and Edgar Weippl. Pay to win: Cheap, crowdfundable, cross-chain algorithmic incentive manipulation attacks on pow cryptocurrencies. In *FC WTSC*, 2021.

[KB16] Ranjit Kumaresan and Iddo Bentov. Amortizing secure computation with penalties. In *CCS*, 2016.

[KMS+16] Ahmed Kosba, Andrew Miller, Elaine Shi, Zikai Wen, and Charalampos Papamanthou. Hawk: The blockchain model of cryptography and privacy-preserving smart contracts. In *S&P*, 2016.

[Max] Gregory Maxwell. The first successful zero-knowledge contingent payment. https://bitcoincore.org/en/2016/02/26/zero-knowledge-contingent-payments-announcement/.

[MB17] Andrew Miller and Iddo Bentov. Zero-collateral lotteries in bitcoin and ethereum. In *EuroS&P Workshops*, 2017.

[MBB+19] Andrew Miller, Iddo Bentov, Surya Bakshi, Ranjit Kumaresan, and Patrick McCorry. Sprites and state channels: Payment networks that go faster than lightning. In *FC*, 2019.

[MD19] Mahdi H. Miraz and David C. Donald. Atomic cross-chain swaps: Development, trajectory and potential of non-monetary digital token swap facilities. In *AETiC*, 2019.

[MES16] Malte Möser, Ittay Eyal, and Emin Gün Sirer. Bitcoin covenants. In *FC Workshops*, 2016.

[MHM18] Patrick McCorry, Alexander Hicks, and Sarah Meiklejohn. Smart contracts for bribing miners. In *FC Workshops*, 2018.

[Mic03] Silvio Micali. Simple and fast optimistic protocols for fair electronic exchange. In *PODC*, 2003.

[MMA] Patrick McCorry, Malte Möser, and Syed Taha Ali. Why preventing a cryptocurrency exchange heist isn't good enough. In *Security Protocols Workshop 2018*.

[MMS+] Giulio Malavolta, Pedro Moreno-Sanchez, Clara Schneidewind, Aniket Kate, and Matteo Maffei. Anonymous multi-hop locks for blockchain scalability and interoperability. In *NDSS 2019*.

[MMSH16] Patrick Mccorry, Malte Möser, Siamak F. Shahandasti, and Feng Hao. Towards bitcoin payment networks. In *Australasian Conference on Information Security and Privacy*, 2016.

[PD] Joseph Poon and Thaddeus Dryja. The bitcoin lightning network: Scalable off-chain instant payments. https://lightning.network/lightning-network-paper.pdf.

[PG99] Henning Pagnia and Felix C. Gartner. On the impossibility of fair exchange without a trusted third party. Technical report, 1999.

[PS17a] Rafael Pass and Elaine Shi. Fruitchains: A fair blockchain. In *PODC*, 2017.

[PS17b] Rafael Pass and Elaine Shi. Rethinking large-scale consensus. In *CSF*, 2017.

[PSS17] Rafael Pass, Lior Seeman, and Abhi Shelat. Analysis of the blockchain protocol in asynchronous networks. In *Eurocrypt*, 2017.

[TYME21] Itay Tsabary, Matan Yechieli, Alex Manuskin, and Ittay Eyal. MAD-HTLC: because HTLC is crazy-cheap to attack. In *S&P*, 2021.

[vdM19] Ron van der Meyden. On the specification and verification of atomic swap smart contracts. In *IEEE ICBC*, 2019.

[WAS22] Ke Wu, Gilad Asharov, and Elaine Shi. A complete characterization of game-theoretically fair, multi-party coin toss. In *Eurocrypt*, 2022.

[WHF19] Fredrik Winzer, Benjamin Herd, and Sebastian Faust. Temporary censorship attacks in the presence of rational miners. In *IEEE European Symposium on Security and Privacy Workshops*, 2019.

[WSZN] Sarisht Wadhwa, Jannis Stoeter, Fan Zhang, and Kartik Nayak. He-htlc: Revisiting incentives in htlc. Cryptology ePrint Archive, Paper 2022/546.

[ZHL+19] A Zamyatin, D Harz, J Lind, P Panayiotou, A Gervais, and W Knottenbelt. Xclaim: trustless, interoperable, cryptocurrency-backed assets. In *S& P*, 2019.

Supplementary Materials

A Proof of Theorem 6.1

Before proving Theorem 6.1, we give some useful lemmas.

Lemma A.1. *Suppose the parameters are set according to Section 6.1. Then, the following statements hold.*

- *Suppose the coalition $\mathcal{A}$ consists of Alice and an arbitrary $\gamma \in [0,1]$ fraction of the mining power. The utility of $\mathcal{A}$ can be more than the honest case, that is, $\$\mathsf{AV}(\text{Ð}x - \text{Ƀ}x')$, only if* Normal *and* Refund′ *both happen.*
- *Suppose the coalition $\mathcal{B}$ Bob and an arbitrary $\gamma \in [0,1]$ fraction of the mining power. The utility of $\mathcal{B}$ can be more than the honest case, that is, $\$\mathsf{BV}(\text{Ƀ}x' - \text{Ð}x)$, only if* Refund *and* Normal′ *both happen.*

Proof. Notice that if any of Normal, Refund and Bomb happens, no coin is left in Rapidash, so no one can get more coin from Rapidash anymore. Thus, consider all possible cases, including none of Normal, Refund and Bomb happens, we have the following table.

which is activated	net profit of Alice's coalition	net profit of Bob's coalition
none	$-\text{Ð}c_a$	$-\text{Ð}x - \text{Ð}c_b$
Normal	$\text{Ð}x$	$-\text{Ð}x$
Refund	0	0
Bomb	$\leq \text{Ð}\epsilon - \text{Ð}c_a$	$\leq \text{Ð}\epsilon - \text{Ð}x - \text{Ð}c_b$

Table 13: The net profit of Bob's coalition from Rapidash.

Similarly, if any of Normal′, Refund′ and Bomb′ happens, no coin is left in Rapidash', so no one can get more coin from Rapidash' anymore. Thus, consider all possible cases, including none of Normal′, Refund′ and Bomb′ happens, we have the following table.

which is activated	net profit of Alice's coalition	net profit of Bob's coalition
none	$-\text{Ƀ}x' - \text{Ƀ}c'_a$	$-\text{Ƀ}c'_b$
Normal′	$-\text{Ƀ}x'$	$\text{Ƀ}x'$
Refund′	0	0
Bomb′	$\leq \text{Ƀ}\epsilon' - \text{Ƀ}x' - \text{Ƀ}c'_a$	$\leq \text{Ƀ}\epsilon' - \text{Ƀ}c'_b$

Table 14: The net profit of Alice's coalition and Bob's coalition from Rapidash'.

Alice-miner coalition. Suppose the coalition $\mathcal{A}$ consists of Alice and an arbitrary $\gamma \in [0,1]$ fraction of the mining power. If $\mathcal{A}$ follows the protocol, Normal and Normal′ will happen, and the utility of $\mathcal{C}$ is $\$\mathsf{AV}(\text{Ð}x - \text{Ƀ}x') > 0$. When Normal and Refund′ both happen, $\mathcal{A}$'s utility is $\$\mathsf{AV}(\text{Ð}x)$. Now, we will show that that this is the only possible scenario for $\mathcal{A}$'s utility to exceed the honest case. For the sake of reaching a contradiction, suppose $\mathcal{A}$'s utility is strictly greater than $\$\mathsf{AV}(\text{Ð}x - \text{Ƀ}x')$ while one of Normal and Refund′ does not happen. There are two cases.

- **Case 1:** Normal **does not happen.** Because $\text{Ð}c_a > \text{Ð}\epsilon$, we have $\text{Ð}\epsilon - \text{Ð}c_a < 0$. Thus, if Normal does not happen, the net profit from Rapidash is at most 0. However, because

$\ddot{\mathrm{B}}x' > \ddot{\mathrm{B}}\epsilon'$, we have $\ddot{\mathrm{B}}\epsilon' - \ddot{\mathrm{B}}x' - \ddot{\mathrm{B}}c'_a < 0$. Thus, the net profit from RAPIDASH' is also at most 0. Consequently, the utility of $\mathcal{C}$ is at most zero, which is less than $\$\mathsf{AV}(\ddot{\mathrm{E}}x - \ddot{\mathrm{B}}x')$.

- **Case 2: Refund' does not happen.** Because $\ddot{\mathrm{B}}c'_a > \ddot{\mathrm{B}}\epsilon'$, we have $\ddot{\mathrm{B}}\epsilon' - \ddot{\mathrm{B}}x' - \ddot{\mathrm{B}}c'_a < -\ddot{\mathrm{B}}x'$. Thus, assuming Refund' does not happen, the net profit from RAPIDASH' is at most $-\ddot{\mathrm{B}}x'$. However, the net profit from RAPIDASH is at most $\ddot{\mathrm{E}}x$. Thus, the utility of $\mathcal{C}$ is at most $\$\mathsf{AV}(\ddot{\mathrm{E}}x - \ddot{\mathrm{B}}x')$, which is the same as the honest case.

Bob-miner coalition. Using a completely symmetric proof, we can show that the only way for a Bob-miner coalition's utility to exceed the honest case is when Refund and Normal' both happen. □

Lemma A.2 (Alice-miner coalition)**.** *Suppose that the hash function $H(\cdot)$ is a one-way function. Let $\mathcal{A}$ be any coalition that consists of Alice and γ fraction of mining power. Then, as long as $\gamma^{\tau'} \leq \frac{\ddot{\mathrm{B}}c'_a}{\ddot{\mathrm{B}}c'_a + \ddot{\mathrm{B}}x'}$, for any PPT coalition strategy $S_{\mathcal{A}}$, except with negligible probability, it must be*

$$\mathsf{util}^{\mathcal{A}}(S_{\mathcal{A}}, HS_{-\mathcal{A}}) \leq \mathsf{util}^{\mathcal{A}}(HS_{\mathcal{A}}, HS_{-\mathcal{A}})$$

where $HS_{-\mathcal{A}}$ denotes the honest strategy for everyone not in $\mathcal{A}$.

Proof. Recall that the utility of $\mathcal{A}$ is $\$\mathsf{AV}(\ddot{\mathrm{E}}x - \ddot{\mathrm{B}}x') > 0$ under an honest execution. Now, suppose $\mathcal{A}$ may deviate from the protocol. We may assume that the coalition does not post any new smart contract on the fly and deposit money into it — if it did so, it cannot recover more than its deposit since any player not in $\mathcal{A}$ will not invoke the smart contract. We analyze the possible cases depending on which phase Bob enters.

Bob enters the abort phase. If RAPIDASH never enters the execution phase, the net profit of $\mathcal{A}$ from RAPIDASH is at most zero. Now, assume RAPIDASH enters the execution phase. When Bob enters the abort phase, he never sends any transaction containing pre_c. Ignoring the negligible probability that $\mathcal{A}$ finds pre_c by itself, B2, P1, and C2 can never be activated. Because Alice does not get any coin from B1, P2 or C1, the net profit of $\mathcal{A}$ from RAPIDASH is at most zero. On the other hand, because $\ddot{\mathrm{B}}x' > \ddot{\mathrm{B}}\epsilon'$, the net profit of $\mathcal{A}$ from RAPIDASH' is at most zero, no matter whether RAPIDASH' enters the execution phase or not.

To sum up, except with negligible probability, the utility of $\mathcal{A}$ is at most zero, which is less than the honest case.

Bob enters the execution phase. If Bob enters the execution phase, both RAPIDASH and RAPIDASH' must enter the execution phase. By Lemma A.1, the utility of $\mathcal{A}$ can exceed the honest case only when Normal and Refund' both happen, so we assume it is the case. Because RAPIDASH' enters the execution phase, for Refund' to happen, P2' must be activated. When Bob enters the execution phase, P2' can be activated only either 1) by Bob sending _ to P2' after C1 has been activated, or 2) by Alice sending pre'_a to P2'. Consider the first scenario. In this case, since C1 has been activated, Alice cannot get any money from RAPIDASH. However, from RAPIDASH', Alice can get at most zero. Thus, the utility of $\mathcal{A}$ is less than the honest case.

Now consider the second case. Suppose that P2' is activated at Bitcoin time $t^* \geq T'_1$, so pre'_a is publicly known after Bitcoin time t^*. By assumption, Normal happens, so P1 must be activated. In this case, $\mathcal{A}$ has to send pre_a to P1.

- *Case 1: $\mathcal{A}$ sends pre_a to P1 before Ethereum time T_1.* Since Ethereum time T_1 is earlier than Bitcoin time T'_1, pre_a and pre'_a are both publicly known at Bitcoin time t^*. Recall that

$pre_a = pre'_b$. Thus, during Bitcoin time $(t^*, t^* + \tau']$, any honest miner will activate C2' if it wins a block. We say $\mathcal{A}$ loses the race if a non-colluding miner mines a new block during Bitcoin time $(t^*, t^* + \tau']$. Otherwise, we say $\mathcal{A}$ wins the race. If $\mathcal{A}$ loses the race, it gets nothing from C1' or C2', and its utility is at most $\$\mathsf{AV}(\ddot{\mathrm{E}}x - \ddot{\mathrm{B}}x' - \ddot{\mathrm{B}}c'_a)$. Else if $\mathcal{A}$ wins the race, then its utility is at most $\$\mathsf{AV}(\ddot{\mathrm{E}}x)$. which can be achieved by activating P2', C1' and P1. The probability p that $\mathcal{A}$ wins the race is upper bounded by $p \leq \gamma^{\tau'}$. Therefore, the expected utility of $\mathcal{A}$ is upper bounded by

$$\$\mathsf{AV}((\ddot{\mathrm{E}}x - \ddot{\mathrm{B}}x' - \ddot{\mathrm{B}}c'_a) \cdot (1-p) + \ddot{\mathrm{E}}x \cdot p). \tag{1}$$

Since $p \leq \gamma^{\tau'} \leq \frac{\ddot{\mathrm{B}}c'_a}{\ddot{\mathrm{B}}c'_a + \ddot{\mathrm{B}}x'}$, we have

$$\$\mathsf{AV}((\ddot{\mathrm{E}}x - \ddot{\mathrm{B}}x' - \ddot{\mathrm{B}}c'_a) \cdot (1-p) + \ddot{\mathrm{E}}x \cdot p) < \$\mathsf{AV}(\ddot{\mathrm{E}}x - \ddot{\mathrm{B}}x').$$

- *Case 2: $\mathcal{A}$ does not send any transaction containing pre_a before Ethereum time T_1.* In this case, the honest Bob will send _ to A1' at Ethereum time T_1. If A1' has not been activated at Bitcoin time $t^* \geq T'_1$, then, during Bitcoin time $(t^*, t^* + \tau']$, any honest miner will activate A2' if it wins a block. On the other hand, if A1' has been activated at Bitcoin time $t^* \geq T'_1$, the honest Bob will send pre_b to P2 as soon as A1' is activated. Thus, at Bitcoin time $t^* \geq T'_1$, pre'_a and pre_b are both publicly known. Thus, during Bitcoin time $(t^*, t^* + \tau']$, any honest miner will activate C2' if it wins a block. By the same calculation as the previous case, since $p \leq \gamma^{\tau'} \leq \frac{\ddot{\mathrm{B}}c'_a}{\ddot{\mathrm{B}}c'_a + \ddot{\mathrm{B}}x'}$, we have $\$\mathsf{AV}((\ddot{\mathrm{B}}x' - \ddot{\mathrm{B}}c'_a + \ddot{\mathrm{E}}x) \cdot (1-p) + \ddot{\mathrm{E}}x \cdot p) < \$\mathsf{AV}(\ddot{\mathrm{E}}x - \ddot{\mathrm{B}}x')$.

□

Lemma A.3 (Bob-miner coalition)**.** *Suppose that the hash function $H(\cdot)$ is a one-way function. Let $\mathcal{B}$ be any coalition that consists of Bob and a subset of miners controlling at most γ fraction of mining power. Then, as long as $\gamma^{\tau} \leq \frac{\ddot{\mathrm{E}}c_b}{\ddot{\mathrm{E}}c_b + \ddot{\mathrm{E}}x}$, for any PPT coalition strategy $S_{\mathcal{B}}$, except with negligible probability, it must be*

$$\mathsf{util}^{\mathcal{B}}(S_{\mathcal{B}}, HS_{-\mathcal{B}}) \leq \mathsf{util}^{\mathcal{B}}(HS_{\mathcal{B}}, HS_{-\mathcal{B}}).$$

Proof. Recall that the utility of $\mathcal{B}$ is $\$\mathsf{BV}(\ddot{\mathrm{B}}x' - \ddot{\mathrm{E}}x) > 0$ under an honest execution. Now, suppose $\mathcal{B}$ may deviate from the protocol. We may assume that the coalition does not post any new smart contract on the fly and deposit money into it —— if it did so, it cannot recover more than its deposit since any player not in $\mathcal{B}$ will not invoke the smart contract. We analyze the two possible cases depending on which phase Alice enters.

Alice enters the abort phase. If Rapidash' never enters the execution phase, the net profit of $\mathcal{B}$ from Rapidash' is at most zero. Now, assume Rapidash' enters the execution phase. When Alice enters the abort phase, she never sends any transaction containing $pre_a = pre'_b$. Ignoring the negligible that $\mathcal{B}$ finds pre'_b by itself, P1' can never be activated, which means Normal' never happens. According to Table 14, if Normal' does not happens, the net profit of $\mathcal{B}$ from Rapidash' is at most zero. On the other hand, because $\ddot{\mathrm{E}}x > \ddot{\mathrm{E}}\epsilon$, the net profit of $\mathcal{B}$ from Rapidash is at most zero, no matter Rapidash enters the execution phase or not.

To sum up, except with negligible probability, the utility of $\mathcal{B}$ is at most zero, which is less than the honest case.

Alice enters the execution phase. By Lemma A.1, the utility of $\mathcal{B}$ can be more than the honest case only if Refund and Normal' both happen, so we assume it is the case. Because Alice enters

the execution phase, both RAPIDASH and RAPIDASH' must enter the execution phase. In this case, Refund happens only if P2 is activated. When Alice enters the execution phase, she never sends _ to P2, so P2 must be activated by pre_b sent by Bob. Therefore, we may assume that P2 is activated at Ethereum time $t^* \geq T_1$, and pre_b is publicly known after Ethereum time t^*. If Alice enters the execution, Bob must have sent pre_c before Ethereum time T_0. Moreover, Alice sends pre_a to P1 at Ethereum time T_0 and $T_0 < T_1$. Therefore, pre_a, pre_b and pre_c are all publicly known at Ethereum time t^*. Thus, during Ethereum time $(t^*, t^* + \tau]$, any honest miner will activate C2 if it wins a block. We say $\mathcal{B}$ loses the race if a non-colluding miner mines a new block during Ethereum time $(t^*, t^* + \tau]$. Otherwise, we say $\mathcal{B}$ wins the race. If $\mathcal{B}$ loses the race, it gets nothing from C1 or C2, and its utility is at most $\$\mathsf{BV}(₿x' - Ξx - Ξc_b)$ which can be achieved if P1' is activated. Else if $\mathcal{B}$ wins the race, then its utility is at most $\$\mathsf{BV}(₿x')$ which can be achieved by activating P2, C1 and P1'. Since $p \leq \gamma^\tau \leq \frac{Ξc_b}{Ξc_b + Ξx}$, we have

$$\$\mathsf{BV}((₿x' - Ξx - Ξc_b) \cdot (1-p) + ₿x' \cdot p) < \$\mathsf{BV}(₿x' - Ξx).$$

□

Proof of Theorem 6.1 Now, we are ready to to prove Theorem 6.1. In Lemma A.2 and Lemma A.3, we show that the atomic swap protocol satisfies γ-CSP-fairness when the coalition consists of Alice or Bob, and possibly with some miners. Because we assume that Alice and Bob are not in the same coalition, it remains to show γ-CSP-fairness when the coalition $\mathcal{C}$ consists only of miners controlling at most γ fraction of the mining power.

Henceforth, we assume Alice and Bob are both honest. It is clear from the protocol that the honest Alice and honest Bob always make the same decision whether to enter the execution phase or abort phase.

Next, when $\mathcal{C}$ follows the protocol, its utility is always zero. Suppose $\mathcal{C}$ may deviate from the protocol. Notice that the utility of $\mathcal{C}$ can be positive only when A2', B2, C2 or C2' is activated. Because Bob only sends pre_c when B1 has been activated, ignoring the negligible probability that $\mathcal{C}$ find pre_c by itself, B2 can never be activated. In the following, we will show that A2', C2 and C2' are never activated except with negligible probability. There are two possible cases.

- *Case 1: both Alice and Bob enter the execution phase.* In this case, Alice always sends pre_a to P1, and she never sends any transaction containing pre'_a. Ignoring the negligible probability that $\mathcal{C}$ finds pre'_a by itself, C2' can never be activated, and A2' can only be activated by pre_b. Moreover, Alice always sends pre_a to P1 at latest at Ethereum time T_0, and thus Bob will not post any transaction containing pre_b. Ignoring the negligible probability that $\mathcal{C}$ finds pre_b by itself, A2' and C2 can never be activated. To sum up, except the negligible probability, the utility of $\mathcal{C}$ is at most zero, which is the same as the honest case.

- *Case 2: both Alice and Bob enter the abort phase.* In this case, Bob always sends _ to P2' and Alice always sends _ to P2. Thus, Bob never sends any transaction containing pre_b, and Alice never sends any transaction containing pre'_a. Ignoring the negligible probability that $\mathcal{C}$ finds pre_b or pre'_a by itself, A2', C2 and C2' cannot be activated by (pre'_a, pre_b). Thus, except with negligible probability, the utility of $\mathcal{C}$ is at most zero, which is the same as the honest case.