Ponyta: 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 for 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 users as strategic players, and assume honest miners. However, recent works revealed that the fairness of commonly deployed fair exchange protocols can be completely broken 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).

We provide the first formal treatment of side-contract-resilient fair exchange. We propose a new fair exchange protocol called Ponyta, and we prove that the protocol is incentive compatible in the presence of user-miner collusion. In particular, we show that Ponyta satisfies a coalition-resistant Nash equilibrium absent external incentives. Further, even when there exist arbitrary but bounded external incentives, Ponyta still protects honest players and ensure that they cannot be harmed. Last but not the least, our game-theoretic formulations also help lay the theoretical groundwork for studying side-contract-resilient fair exchange protocols. Finally, we present practical instantiations of Ponyta in Bitcoin and Ethereum with minimal overhead in terms of costs for the users involved in the fair exchange, thus showcasing instantiability of Ponyta with a wide range of cryptocurrencies.

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, BDM, Fuc, BKb, 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, BDM, Fuc, BKb, 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, BDM, Fuc, BKb], payment channels [PD, DW15, GM, MMSH, MBB+, 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+17a, GKM+22, BKa]. 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, Bon, MMS+, MHM18a, JSZ+21, Ham]. As a concrete example, a Hash Timelock Contract (HTLC) is one of a commonly employed mechanism 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 of $\$v$ coins. 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, MHM18a, JSZ+21, Ham]. Such attacks can be instantiated in various ways [TYME21, WHF19, HZ20, MHM18a, 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].

Tsabary et al. [TYME21] also made a pioneering attempt to try to overcome such bribery attacks. They proposed a new fair exchange mechanism called a *Mutual-Assured Destruction Hash Timelock Contract (MAD-HTLC)*. Just like the HTLC contract, in MAD-HTLC, Bob deposits $\$v$ into the contract upfront. The contract allows Alice to redeem the $\$v$ coins by revealing the secret pre_a that Bob wants to learn, and whose hash is hard-wired in the contract. If Alice does not redeem the coins by time T, Bob can claim back his deposit by revealing another secret pre_b whose hash is also hard-wired in the contract. Importantly, MAD-HTLC adds the following clever rule (henceforth called the *bomb*): anyone (including the miner of the block) who present both pre_a and pre_b can claim the $\$v$ coins for themselves. MAD-HTLC deters Bob from cheating in the following way: if a cheating Bob posts pre_b and attempts to claim his $\$v$ even after Alice has revealed pre_a, then any miner, who now has knowledge of both pre_a and pre_b, can preempt

Bob's transaction by triggering the bomb and claiming the $\$v$ coins itself. This effectively thwarts Bob's attempt. Although MAD-HTLC does seem to resist the known bribery mechanisms, it opens up some possible new attacks. For example, as soon as Alice publishes pre_a, if a Bob-miner coalition happens to mine the next block, they should claim the $\$v$ back by posting (pre_a, pre_b), and split off the gain among themselves. 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].

Therefore, the following natural and fundamental question remains open:

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.

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:

Ponyta: a side-contract resilient fair exchange protocol. We propose a new mechanism called Ponyta[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 probability proportional to the miner's stake. We prove that Ponyta achieves the following notions of game-theoretic fairness (informally stated, assuming a Proof-of-Work blockchain):

- **Cooperative strategy proofness.** We prove that absent any external incentives, any coalition of players (that does not simultaneously contain Alice and Bob[2]) 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, the honest behavior is a coalition-resistant Nash equilibrium.
- **Safe participation against external incentives.** 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. 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 to participate as long as they believe that the present protocol has only bounded externalities.

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

We now explain our idea in a nutshell. We are inspired by MAD-HTLC's elegant idea of using a bomb [TYME21]. The problem with MAD-HTLC is that triggering the bomb actually benefits a Bob-miner coalition. In our work, we make Alice and Bob put in extra collateral into the smart contract. If no player cheats, both players can get their collateral back at the end of the protocol. However, should Bob cheat causing the bomb to be triggered, not only can the miner triggering the

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

[2] If Alice and Bob were in the same coalition, then they would not need to do the fair exchange.

bomb obtain some payment, but also part of the collateral will be *burnt* and made unrecoverable. In this way, we make sure that triggering the bomb actually hurts an Alice-miner or a Bob-miner coalition. Even when the coalition may have arbitrary but bounded external incentives, we can adjust the collateral amount with respect to the maximum external incentives, such that triggering the bomb would always harm the coalition.

Definitional and conceptual contributions. Our work helps to lay 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. Equivalently, we are assuming 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, our modeling approach also captures 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 unenforceable in the real world).

Our notion of cooperative-strategy-proofness (or equivalently, coalition-resistant Nash) was proposed and adopted in a line of recent works [PS17a, CGL+18, WAS22, CS21]. However, to the best of our knowledge, no prior work in this space has formulated game-theoretic notions that capture a protocol's resilience in the presence of arbitrary (but bounded) external incentives. Partly, it seems almost impossible at first sight to provide any game-theoretic guarantee when players may have arbitrary external incentives. For example, Alice can be promised some incentive if she posted her secret pre_a just one second later but still well before the timeout. In this case a rational Alice will want to deviate from the honest protocol, and the honest protocol is no longer an equilibrium. Instead of asking for an equilibrium notion, as mentioned above, we opt for a safe-participation-type guarantee in the presence of externally incentivized, rational players whose incentives are bounded.

Finally, our definitional approach models players and coalitions as interactive Turing Machines who can send and receive a special type of variables 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. We first describe Ponyta assuming an existence of generic smart contracts. We then give an instantiation of Ponyta that is compatible with the scripting language of Bitcoin. To do so, we introduce two novel transaction-level tricks which we call *conditional timelock redeem* and *conditional burn*. The former ensures that coins from an address are redeemable only if T time has passed since the redeeming of coins from another address. The latter allows miners to redeem a portion of the coins from a target address while burning the rest under some conditions. These techniques may be of independent interest and lend to other applications. We also instantiate Ponyta in Solidity [Eth22], Ethereum's smart contract language, and deploy it on the Rinkeby testnet [rin22]. We show that the cost overhead for either user in terms of *gas cost* is insignificant compared to MAD-HTLC.

1.2 Additional Related Work

We now review additional related work besides those already mentioned. Our work is also related to financially fair protocols [BK14, MB17, BZ17, CGJ+17b, KB16, KMS+16], where a common theme is using 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, Ponyta allows honest miners to extract value should any cheating behavior take place.

2 Technical Roadmap

2.1 Problem Statement and Assumptions

Suppose that Alice has a secret s and Bob wants to buy the secret from Alice at a price of $\$v$. We would like to design a fair exchange protocol to accomplish this. 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. 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, we will formalize the notion of transactions and a smart contract execution model.

2.2 Strawman and Prior Approaches

For simplicity, in the technical roadmap, we will assume the existence of general smart contracts, although later, we will indeed show how to instantiate our protocol atop Bitcoin which supports only a restricted language for contracts.

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.

Let $T = 0$ be the moment when Bob deposited the initial $\$v$ into the contract. The honest protocol is simple: Alice sends s to the smart contract at $T = 0$, and Bob does nothing. We always want the miner's honest strategy to be consistent with the underlying consensus protocol, that is, the miner should include all outstanding transactions in the block.

Somewhat surprisingly, even this very simple protocol satisfies a coalition-resilient game theoretic notion. Consider any coalition consisting of either Alice or Bob as well as an arbitrary subset of the miners: the coalition should have no incentive to deviate from the honest protocol, if its goal is to maximize its own utility[3]. Such a notion is often referred to as *cooperative-strategy-proofness* (or *CSP fairness*). In particular, it implies that the *honest strategy is a coalition-resistant Nash equilibrium*.

This approach, 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. A line of work used hash timelock contracts (HTLCs) [atob, atoa, Her18, CGGN, Max] aimed at achieving dropout resilience, but at the price of losing CSP fairness. A standard HTLC contract 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.

The above contract allows Bob to recover its deposit should Alice drop 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 we make $T_1 \cdot \$f$ 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.

[3]For the time being, we assume that there is no external incentives, and we defer the discussion of this topic to Section 6.

MAD-HTLC. Tsabary et al. [TYME21] suggest a new contract called MAD-HTLC in an attempt to remove the undesirable 100%-miner-colluding equilibrium in the coalition-forming meta-game. As before, 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, MAD-HTLC does not have provable security in the presence of miner-user coalitions, as the authors acknowledge themselves [TYME21]. Their game-theoretical formulation considers only individual deviations. In fact, the following is straightforward to observe:

Observation: MAD-HTLC is NOT incentive compatible in the presence of *binding side contracts*.

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. As the authors of MAD-HTLC [TYME21] acknowledge themselves, MAD-HTLC provides no guarantee when Bob is a miner himself — however, in reality, since creating pseudonyms or public keys is cheap, it is difficult for Alice to check whether Bob is a miner.

2.3 Our Approach

In our Ponyta contract, during the preparation phase, Alice and Bob each deposits $\$c_a$ and $\$c_b + \$v$ into the contract respectively, where $\$c_a > \$v$ and $\$c_b > 2 \cdot \$v$. We refer to $\$c_a$ and $\$c_b$ as Alice and Bob's collateral, respectively. Note that Bob needs to deposit his collateral $\$c_b$ on top of the promised payment amount $\$v$. Our Ponyta contract works as follows where we use pre_a as the secret Alice wants to sell:

Ponyta contract

Payment:

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

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

P2: Time T_1 or greater: on receive pre_b from Bob such that $H(pre_b) = h_b$, send $\$v$ to Bob.

Collateral:

C1: At least T_2 after either P1 or P2 is activated: on receiving _ from anyone, send $\$c_a$ to Alice and send $\$c_b$ 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 $\$v$ to player P. All remaining coins are burnt.

In the above, assuming that the coalition consists of any constant fraction of the miners (e.g., even 99%), we can set the timeout values T_1 and T_2 to be appropriate constants as we discuss in more detail in Section 5.1.

There are two types of activation points in the contract: P1 and P2 are used to redistribute Bob's promised payment $\$v$, and C1 and C2 are used to redistribute the collateral part, that is $\$c_a$ and $\$c_b$. The activation points P1 and P2 are mutually exclusive i.e., only one can be successfully activated, and at most once. Similarly, C1 and C2 are mutually exclusive, too.

Intuition. Informally, the Ponyta contract provides game theoretic fairness due to the following intuition. Suppose that Alice posted pre_a. In this case, a Bob-miner coalition would not want pre_b to be revealed. If so, some other miners may be able to activate C2 before C1 is activated. In this case, the coalition would lose its collateral $\$c_b > 2 \cdot \$v$. Another strategy is for the coalition to claim both P2 and C2 and get $2 \cdot \$v$ back from the contract. Had they acted honestly, however, they could get $\$c_b > 2 \cdot \$v$ back, which is strictly better.

CSP-fairness absent external incentives. In Section 5, we shall prove that our Ponyta protocol satisfies CSP-fairness when players do not have external incentives. Informally, this means that an Alice-miner coalition or a Bob-miner coalition cannot increase its utility by deviating from the honest protocol, as long as the coalition does not control 100% of the mining power, and moreover, the parameters $\$c_a$, $\$c_b$, and T_2 are appropriately set. In other words, the honest protocol is a coalition-resistant Nash equilibrium.

Safe participation against external incentives. The potential existence of *arbitrary* external incentives significantly complicates mechanism design. At first sight, it seems almost impossible to provide any game-theoretic guarantee when there are arbitrary external incentives. For example, someone may offer to pay Alice \$1 if she posts her pre_a just one second later, but still well before T_1. In this case, Alice is incentivized to deviate from honest protocol even when others are playing honestly, and thus the honest behavior is no longer an equilibrium. Rather than asking for an equilibrium notion requiring that no one wants to deviate, we go for a notion of "safe participation" as explained below. We want to guarantee that even when some players may have arbitrary but bounded external incentives, as long as they are *rational* (where rationality is formally defined in Section 4), they cannot do any harm to honest players without external incentives. This ensures that even if Alice is an ordinary, simple-minded, non-strategic player, she can always feel safe to participate, as long as she believes that the current protocol has bounded externality to the rest of the world. In Section 4, we formally define this notion and call it *safe participation* against external incentives.

In Section 6, we present a variant of the Ponyta protocol, with the collateral amounts $\$c_a$ and $\$c_b$ adjusted w.r.t. the maximum external incentive, and we prove that this new variant satisfies

safe participation against arbitrary but bounded external incentives.

Ponyta disincentivizes 100% coalitions. To prove that our protocol satisfies CSP-fairness and safe participation against external incentives, 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. In the coalition forming meta-game, imagine that Bob posts a smart contract that promises to split off 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 $\$v$ itself by posting (pre_a, pre_b) — should Bob ever decide to cheat. To incentivize the miners to join, Bob needs to 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 detailed argument to Section 7.

In Section 7, we also show that this meta-game approach is not only a good complement to our formal foundations of side-contract-resilience, but also helpful for reasoning about the known bribery attacks [TYME21, WHF19, HZ20, MHM18a, JSZ+21] that pertain to the standard HTLC contract.

3 Model

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 block it mines.

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 the most general case, we view the start of protocol execution (i.e., $t = 0$), as the moment that the relevant smart contract is deployed. Sometimes, it may also be convenient to view the start of protocol execution as the moment when the parties involved have made their initial deposits into the contract. Later on when we define our protocols, we shall explicitly mention the start of protocol execution (i.e., $t = 0$). In our paper, an honest protocol is always a *simple* protocol that does not create additional smart contracts in the middle of the execution. Without loss of generality, we may assume that the honest protocol employs at most one contract — since if there are multiple, we can view the union of them as a single contract. Of course, strategic parties can create new smart contracts on the fly during the execution.

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 always 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 Utilities

We assume that the secret s is worth $\$v_a$ and $\$v_b$ to Alice and Bob, respectively. That is, Alice will lose utility $\$v_a$ if s is released to someone else, and Bob will gain $\$v_b$ if he learns s. We assume that $\$v_b > \$v > \$v_a$, such that Alice wants to sell the secret s 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 s. 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. Let $\$e_a$ and $\$e_b$ denote any possible external incentive for Alice and Bob, respectively — we assume that $\$e_a$ and $\$e_b$ may be a function of β and of the blockchain's state in general. Without loss of generality, we may assume that the *external incentive is non-negative* — if not, we can always offset the external incentive by its minimum possible value and this does not affect any player's behavior or change the game-theoretic analysis.

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

$$\$u_a = -\$d_a + \$r_a - \beta \cdot \$v_a + \$e_a(\beta, \ldots),$$

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

$$\$u_b = -\$d_b + \$r_b + \beta \cdot \$v_b + \$e_b(\beta, \ldots)$$

where ... any other variable that the external incentive functions may depend on, e.g., arbitrary on-chain states.

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. Let $\$e_m(\beta, \ldots) \geq 0$ be any external incentive for the miner, where the external incentive can depend on arbitrary variables such as the state of the blockchain. A miner's utility, denoted $\$u_m$, is defined as

$$\$u_m = -\$d_m + \$r_m + \$e_m(\beta, \ldots)$$

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

Throughout this paper, we assume that the total utility of all players from the protocol cannot exceed a polynomial function in the security parameter λ.

Regarding external incentives. In the earlier part of our paper where we prove CSP-fairness, we assume that parties do not have any external incentives. In the later part of our paper, we shall prove that our protocol protects honest individuals or groups even when the other players may have arbitrary but bounded 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 Convention for Writing Smart Contracts

For ease of exposition, we use a simplified notation for writing *meta-contracts*. A meta-contract is a platform-independent approach to express smart contract logic. Meta-contracts expressed in our notation can easily be instantiated using a general smart contract language such as Ethereum. By contrast, not every meta-contract expressible using our notational system can be instantiated atop Bitcoin, since Bitcoin's scripting language is not Turing-complete. However, for the contracts we propose in this paper, we will show that they can indeed be instantiated even atop Bitcoin.

A meta-contract will be expressed using the following style of notation:

A toy meta-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 meta-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.

To instantiate our meta-contracts 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.

We leave the concrete instantiation of our meta-contract to Section 9 and Section 10. In our meta-contract notation, we simply assume that there is an authenticated channel from each user to the smart contract.

4 Defining Incentive Compatibility

Henceforth, we use $\mathcal{C}$ to denote a coalition, and use $-\mathcal{C}$ to denote all parties of the protocol that are not part of the coalition. We use $HS_{\mathcal{C}}$ or $HS_{-\mathcal{C}}$ to denote the honest strategy executed by either the coalition $\mathcal{C}$ or its complement. Let $S_{\mathcal{C}}$ and $S'_{-\mathcal{C}}$ be the strategies of the coalition $\mathcal{C}$ and its complement. We use $\mathsf{util}^{\mathcal{C}}(S_{\mathcal{C}}, S'_{-\mathcal{C}})$ to denote the expected utility of $\mathcal{C}$ when the coalition $\mathcal{C}$ adopts the strategy $S_{\mathcal{C}}$ and the remaining parties adopt the strategy $S'_{-\mathcal{C}}$.

CSP fairness. We first define a game-theoretic fairness notion called cooperative strategy proofness (CSP fairness) — the same notion was formalize earlier in a recent line of 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 4.1 (CSP fairness)**.** We say that a fair exchange 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

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

where we use HS to mean the honest strategy.

Safe participation in the presence of external incentives. 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 $\mathcal{C}'$ can find 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. Therefore, if we insist that a rational $\mathcal{C}'$ must play the optimal strategy, it makes our resilience notion too weak. 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*.

Note that we can often show that some strategies are *blatantly non-rational* without having to find the optimal strategy. Indeed, later in Section 6, 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.

Definition 4.2 (Safe participation against external incentives)**.** We say that a protocol satisfies α-safe-participation against external incentives 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

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

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

Later in Section 6, we will define the strategy space $\mathcal{R}$ to be all PPT strategies except for some that are shown to be blatantly non-rational.

Since we assume without loss of generality that the external incentives are non-negative for any player, Definition 4.2 also ensures that even when the coalition $\mathcal{C}$ has external incentives, it cannot be harmed if it played honestly, as long as the total external incentives for $\mathcal{C}'$ are bounded.

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.

5 Warmup: Achieving CSP-Fairness Absent External Incentives

5.1 Construction

For ease of understanding, we first describe a warmup construction that satisfies CSP fairness absent external incentives. Later in Section 6, we describe a variant that satisfies both CSP fairness as well as safe participation in the presence of external incentives.

Below, we use pre_a to denote the secret that Alice wants to sell, and we assume that pre_a is sampled by Alice uniform at random from $\{0,1\}^\lambda$.

Ponyta contract

Parameters: $h_a, h_b, T_1, T_2, \$v, \$c_a, \$c_b$ such that $\$c_a > \$v$ and $\$c_b > 2 \cdot \$v$.

Preparation phase: Alice deposits $\$c_a$, Bob deposits $\$v + \$c_b$

Execution phase:

Payment:

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

P2: Time T_1 or greater: on receive pre_b from Bob such that $H(pre_b) = h_b$, send $\$v$ to Bob.

Collateral:

C1: At least T_2 after either P1 or P2 is activated: on receiving _ from anyone, send $\$c_a$ to Alice and send $\$c_b$ 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 $\$v$ to player P. All remaining coins are burnt.

As mentioned earlier, the activation points of the same type are mutually exclusive; that is, only one of the them can be activated, and at most once.

We now describe the honest protocol.

The Ponyta protocol. We assume that at the very beginning, Alice knows pre_a and Bob knows pre_b. During the preparation phase, Alice and Bob each makes a deposit of $\$c_a$ and $\$c_b + \$v$ into the Ponyta contract, respectively. Each party can freely withdraw its deposit as long as the other party has not deposited yet. However, once both parties have deposited, money redistribution is only possible through the activation points of the execution phase. The moment both parties have deposited into the contract, the protocol executoin starts, and we rename the current time t to be 0. The execution phase proceeds as follows — henceforth, we use the phrase "a player sends a message to an activation point" to mean that "the player posts a transaction containing the message and destined for the activation point":

- *Alice*: Alice sends pre_a to activation point P1 at $t = 0$. T_2 time after either P1 or P2 successfully completes, she posts an empty message _ to C1.
- *Bob*: If Alice failed to post a transaction containing a correct pre_a destined for P1 before time T_1, then Bob sends pre_b to P2 at time $t = T_1$. T_2 time after either P1 or P2 successfully completes, he sends an empty message _ to C1.

  If either P2 or C2 is successfully activated, Bob outputs the corresponding pre_a value included in the corresponding transaction. If C1 and P2 are successfully activated, Bob outputs $\perp$.
- *Miner*: The miner watches all transactions posted to P1, P2, and C2. If the miner has observed the correct values of both pre_a and pre_b from these posted transactions, then it sends (pre_a, pre_b) to C2. Further, any miner always includes all outstanding transactions in every block it mines. If there are multiple transactions posted to C2, the miner always places its own ahead of others (and thus invalidating the others).

Choice of timeouts. We now discuss the choice of the timeout values T_1 and T_2 in the Ponyta contract. For our game theoretic proofs (see Section 5.2) to hold, we can set $T_1 = 1$, and T_2 should

be set such that $\gamma^{T_2} < \frac{\$c_b - \$v}{\$c_b}$ where γ is the maximum percentage of mining power that can form a coalition with Bob. For example, suppose $\$c_b = 2\$v + \$\epsilon$ for some small $\$\epsilon$. Then, we need to ensure $\gamma^{T_2} \leq 1/2$. This means if $\gamma = 90\%$, we can set $T_2 = 7$. Asymptotically, for any $\gamma = O(1)$, T_2 is a constant. Increasing $\$c_b$ also helps to make T_2 smaller.

In practice, the network may be unstable (for either Alice or Bob), or there may be a spike in transaction volume causing congestion. In these cases, we can still set $T_1 = 1$, but Bob can choose to wait longer before posting pre_b. Alice and Bob can also coordinate through an offline channel to inform each other that they are going to post either pre_a or pre_b, to avoid the situation that both pre_a and pre_b are posted, leading to part of their collateral being burnt.

5.2 Proofs

Lemma 5.1. *Let $\mathcal{C}$ be any coalition that consists of Alice and an arbitrary subset of miners (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. Suppose everyone not in $\mathcal{C}$ plays the honest strategy. Then, the coalition $\mathcal{C}$ can play honestly, i.e., post pre_a to P1 at $t = 0$, and post _ to C1 when T_2 has passed since P1 or P2 is activated. In this case, $\mathcal{C}$ obtains utility $\$v - \$v_a$.

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 and deposit money into it[5] (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 $\mathcal{C}$ will not invoke the smart contract. There are two possibilities:

1. First, P1 is successfully activated at some point. Since C1 and C2 are mutually exclusive, and $\$c_a \geq \$v$, $\$v + \$c_a$ is the maximal amount that the coalition can redeem from the Ponyta contract. In this case, the honest Bob will output pre_a and thus the coalition $\mathcal{C}$'s utility is at most $\$v - \$v_a$, which is the same as following the honest strategy.

2. Second, P1 is never activated. In this case, it is impossible for the coalition to redeem any value from P1 or P2, and its utility is at most $\$c_a$. Since $\$v > \$v_a$ by our assumption, following the honest strategy maximizes the utility of $\mathcal{C}$.

□

Lemma 5.2. *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 $\gamma^{T_2} \leq \frac{\$c_b - \$v}{\$c_b}$, for any (even unbounded[6]) coalition strategy $S_{\mathcal{C}}$, it must be that*

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

[5] 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.

[6] Bob's coalition can be unbounded as long as the mining process is idealized as in our model.

Proof. Suppose everyone not in $\mathcal{C}$ plays the honest strategy. If the coalition follows the honest strategy as well, P1 and C1 will be activated, and Bob outputs pre_a. Thus, the utility of the coalition is $\$v_b - \$v$.

Now, consider the case where the coalition $\mathcal{C}$ deviates from the honest strategy. Just like in the proof of Lemma 5.1, we may assume that the coalition does not post any new contract during the protocol execution.

We may also assume that either P1 or P2 is successfully invoked, since otherwise, the coalition $\mathcal{C}$'s utility is at most $\$v_b - \$c_b$ by activating C2, which is no more than the honest case because $\$c_b > \$v$. There are two possibilities.

First, P1 is successfully activated. In this case, because $\$c_b > \$v$, the maximal value the coalition can get from the contract is to activate the C1 branch, and the utility of the coalition is at most $\$v_b - \$v$, which is the same as the honest case.

Second, P2 is activated. Let t^* be the time at which P2 is activated. There are two subcases. In the first subcase, the coalition also gets $\$v$ from C2 at time t^* or earlier. In this case, the coalition's utility is at most $\$v + \$v_b - \$c_b$, and since $\$c_b > 2 \cdot \$v$, this is less than the honest case. Henceforth, we may assume that the coalition has not got $\$v$ from C2 at time t^* or earlier. Since the honest Alice posts pre_a at $t = 0$, 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 $\$v_b - \$c_b$. 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

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

Recall that $\$c_b > 2 \cdot \$v$. Therefore, for $(\$v_b - \$c_b) \cdot (1 - p) + \$v_b \cdot p$ to exceed the honest utility $\$v_b - \$v$, it must be that $p > \frac{\$c_b - \$v}{\$c_b}$ which is not true by our assumption. We thus conclude that $\mathcal{C}$ cannot increase its utility through any deviation.

□

Theorem 5.3 (CSP fairness absent external incentives)**.** *Suppose that the hash function $H(\cdot)$ is a one-way function and that $\gamma^{T_2} \leq \frac{\$c_b - \$v}{\$c_b}$. Then, the* Ponyta *protocol satisfies γ-CSP-fairness.*

Proof. Lemmas 5.1 and 5.2 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. This is easy to see: since both Alice and Bob are honest, the coalition's utility is 0 unless it can 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 Ponyta is dropout resilient.

Theorem 5.4 (Dropout resilience)**.** Ponyta *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. 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. 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, an honest miner would include Alice's transaction and activate P1. Then, T_2 after P1 is activated, Bob would send an empty message to C2. As long as $1/\mathsf{poly}(\lambda)$ fraction of the mining power is honest, P1 and C1 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 and C1 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. □

6 Safe Participation in the Presence of External Incentives

In the previous section, we described a protocol that achieves CSP fairness when no one has any external incentives. In this section, we argue that a slight variant of the protocol, with an increased amount of collateral, can protect players who do not have external incentives from others who might have arbitrary but bounded external incentives. More concretely, we will prove that the new variant of Ponyta protocol satisfies "safe participation against external incentives" (see Definition 4.2).

6.1 Construction

Parameter Choices. We now increase the players' collateral $\$c_a$ and $\$c_b$ w.r.t. the maximum amount of external incentives. Henceforth, let $\alpha \in [0, 1 - \mathsf{poly}(\lambda)]$ denote the maximum fraction of mining power controlled by the set of externally incentived players, and let $\$E$ be the maximum amount of external incentive available. To ensure that our protocol satisfies γ-CSP-fairness and α-safe-participation against external incentives, we will set $\$c_a$ and $\$c_b$ such that the following relations are satisfied:

- $T_1 \geq 1$, $T_2 \geq 1$,
- $\$c_b > \max(\frac{\$v}{1-\gamma^{T_2}}, \frac{\$v_b+\$E}{1-\alpha})$, and
- $\$c_a > \max(\$v, \frac{-\$v_a+\$E}{1-\alpha})$.

The contract. Earlier in Section 5, we wanted to prove CSP fairness assuming no external incentives; in this case, without loss of generality, we defined time $t = 0$ to be the moment when both parties deposited into the contract. Now we want to prove safe participation against external incentives, we need to worry about the case when the external incentives encourage the players to deviate even before the deposits are made, and the effect of such deviation on the honest players. Thus, in our new contract and protocol, we make a couple of modifications to discourage misbehavior before the deposits are made. Specifically, in this section, we model the preparation phase of Ponyta protocol explicitly. The start of the execution (i.e. time 0) is defined to be the moment that Ponyta is posted on the blockchain.

Ponyta contract // Parameters: (h_a, h_b, T_1, T_2, \$$v$, \$$c_a$, \$$c_b$)

Preparation phase: Wait to receive \$$c_a$ deposit from Alice and \$$v + $\$$c_b$ deposit from Bob. On receive pre_a such that $H(pre_a) = h_a$ or on receive pre_b such that $H(pre_b) = h_b$ from anyone: record the event and do nothing. *If Alice and Bob have both deposited, and no pre_a or pre_b has been recorded, the preparation phase ends and the execution phase starts.*

A1: On receive _ from Alice, send \$$c_a$ to Alice. // *withdrawal transaction*

A2: On receive _ from anyone P: if a pre_a has been recorded and Alice has deposited \$$c_a$, send \$$c_a$ to P.

B1: On receive _ from Bob, send \$$v + $\$$c_b$ to Bob. // *withdrawal transaction*

B2: On receive _ from anyone P: if a pre_b has been recorded and Bob has deposited \$$v + $\$$c_b$, send \$$v + $\$$c_b$ to P.

Execution phase:

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

P2: Time T_1 or greater: on receive pre_b from Bob such that $H(pre_b) = h_b$, or on receiving _ from Alice, send \$$v$ to Bob.

C1: At least T_2 after either P1 or P2 is activated: on receiving _ from anyone, send \$$c_a$ to Alice and send \$$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 \$$v$ to player P. All remaining coins are burnt.

Conceptually, A2 and B2 discourage Alice and Bob to send any transaction containing pre_a and pre_b, respectively, before the Ponyta contract enters the execution phase. Specifically, if they posted pre_a or pre_b respectively before Ponyta enters the execution phase, then anyone can claim their entire deposit by sending _ to A2 or B2, respectively.

Ponyta protocol. We now describe our protocol.

- *Miner.* The miner's honest protocol is described below.
  - The miner watches all transactions posted to P1, P2, C1, C2, as well as all transactions posted to the preparation phase of Ponyta to see if they contain a valid pre_a or pre_b.
  - When Ponyta is still in the preparation phase, if the miner observes a valid pre_a, it immediately posts pre_a to the preparation phase, and it also posts _ to A2; if it ever sees a valid pre_b, it immediately posts pre_b to the preparation phase, and it also posts _ to B2.
  - As soon as the miner has observed both 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 first followed by others' transactions. Among its own transactions, posting pre_a or pre_b to the preparation phase of Ponyta should be ordered ahead of other transactions.
- *Alice and Bob.* Below, we define the honest protocol for Alice and Bob. Initially, Alice and Bob choose two absolute time T and T_1 such that $T_1 > T$. As mentioned, the start of the execution (i.e., time 0) is defined to be the moment when the Ponyta contract has been posted and takes effect.

Ponyta Protocol — Alice and Bob

Preparation Phase:

1. At $t = 0$, Alice and Bob deposit $\$c_a$ and $\$v + \$c_b$ into Ponyta, respectively.
2. At time T: if Ponyta has not entered the execution phase, go to the abort phase. Otherwise, if Ponyta entered the execution phase, go to the execution phase.

Execution Phase:

1. At time T, Alice sends pre_a to P1.
2. At time T_1, if Alice has not sent pre_a to P1, Bob sends pre_b to P2.
3. As soon as T_2 has passed since P1 or P2 is activated, Alice and Bob each post an empty message _ to C1.

Abort Phase:

1. If Alice or Bob has submitted a deposit transaction, then submit a withdrawal transaction, i.e., an empty message _ sent to A1 or B1, respectively.
2. As soon as Ponyta enters the execution phase,[a] Alice posts _ to P2, and Bob posts pre_b to P2.
3. As soon as T_2 has passed after P1 or P2 is activated, Alice and Bob each post an empty message _ to C1.

[a] Notice that Alice and Bob go to the abort phase if Ponyta enters the execution phase after time T.

CSP-Fairness and Dropout Resilience. Theorem 6.1 and Theorem 6.2 show that the above modified version of Ponyta still satisfies CSP fairness absent external incentives, as well as dropout resilience. The proofs only need minor modification from the arguments in Section 5.2. For completeness, we present the proofs formally in Appendix A.

Theorem 6.1 (CSP fairness absent external incentives)**.** *Suppose that the hash function $H(\cdot)$ is a one-way function and that $\gamma^{T_2} \leq \frac{\$c_b - \$v}{\$c_b}$. Then, the* Ponyta *protocol in Section 6.1 satisfies γ-CSP-fairness.*

Proof. Deferred to Appendix A. □

Theorem 6.2 (Dropout resilience)**.** Ponyta *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. Deferred to Appendix A. □

In the remainder of this section, we focus on proving that the new variant satisfies safe participation in the absence of external incentives.

6.2 Blatantly Non-Rational Strategies

In this section, we will define a family of strategies that lead to a non-negligible probability of triggering the "bomb" (i.e., activation points A2, B2 or C2) causing the users' collaterals to be

partially burnt. We argue that this family of strategies is blatantly non-rational. In other words, a rational Alice-miner coalition or Bob-miner coalition should never want to run any risk of triggering the bomb, assuming that the external incentives are limited w.r.t. the collateral at stake.

More formally, we define the a family of PPT strategies denoted $\overline{\mathcal{R}}$, and we will show given any PPT strategy $S \in \overline{\mathcal{R}}$, we can give a simple modification of S which makes the externally incentivized coalition better off — in this sense, the strategy space $\overline{\mathcal{R}}$ is blatantly non-rational. We say that C1 is "guaranteed to be activated" at some time t, iff either C1 was already activated before t, or a colluding miner has been chosen as the winning miner at time t, and it activates C1 in the new block it mines. We say Alice is *guaranteed to withdraw her deposit* at some time t, iff either Alice already withdrew her deposit before t, or all the following conditions hold.

- Ponyta is still in the preparation phase.
- A colluding miner has been chosen as the winning miner at time t.
- Alice and the colluding miner activates A1 or A2 in the new block it mines.

We define Bob is *guaranteed to withdraw his deposit* at some time t similarly, except that we replace "A1 or A2' with "B1 or B2" in the third bullet.

We say Ponyta is *guaranteed to enter the execution phase* at some time t, iff either Ponyta is already in the execution phase at time t, or all the following conditions hold.

- A colluding miner has been chosen as the winning miner at time t.
- Both parties already posted their deposits transactions.
- pre_a and pre_b are not yet recorded in Ponyta.
- The colluding miner make both deposits into Ponyta in the new block it mines.

Blatantly non-rational strategies. The family $\overline{\mathcal{R}}$ contains any PPT strategy that satisfies either of the following:

- The externally incentivized coalition $\mathcal{C}'$ is a Bob-miner coalition (including Bob alone), and moreover, with non-negligible probability, any of the following happens.

  $\mathbf{E}_1$: Before Ponyta is guaranteed to enter the execution phase and before Bob is guaranteed to withdraw his deposit, Bob posts the deposit transaction and anyone in $\mathcal{C}'$ posts a transaction containing pre_b to P2 or C2 or the preparation phase of Ponyta.

  $\mathbf{E}_2$: Ponyta enters the execution phase at time t such that $t \leq T$, and moreover, at time $t' \geq t$, someone in $\mathcal{C}'$ posts a transaction containing pre_b to P2 or C2 before C1 is guaranteed to be activated.

- The externally incentivized coalition $\mathcal{C}'$ is an Alice-miner coalition (including Alice alone), and moreover, with non-negligible probability, any of the following happens.

  $\mathbf{E}_3$: Before Ponyta is guaranteed to enter the execution phase and before Alice is guaranteed to withdraw her deposit, Alice posts the deposit transaction and anyone in $\mathcal{C}'$ posts a transaction containing pre_a to P1, C2 or the preparation phase of Ponyta.

  $\mathbf{E}_4$: Ponyta enters the execution phase at some time $t > T$, and moreover, at some time $t' \geq t$, anyone in $\mathcal{C}'$ posts a transaction containing pre_a to P1 or C2 before C1 is guaranteed to be activated.

$\mathbf{E}_5$: Suppose PONYTA enters the execution phase, and Alice does not post pre_a to P1 in $t \in [0, T_1)$. However, anyone in $\mathcal{C}'$ posts a transaction containing pre_a to P1 or C2 at T_1 or greater before C1 is guaranteed to be activated.

Lemma 6.3 (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 $\mathcal{C}'$ consists of Bob and miners controlling no more than $\alpha \leq 1 - 1/\mathsf{poly}(\lambda)$ fraction of the mining power. Given any strategy $S_{\mathcal{C}'} \in \overline{\mathcal{R}}$ for some (externally incentivized) coalition $\mathcal{C}'$, there is a strategy $\widehat{S}_{\mathcal{C}'}$ such that*

$$\mathsf{util}^{\mathcal{C}'}(\widehat{S}_{\mathcal{C}'}, HS_{-\mathcal{C}'}) \geq \mathsf{util}^{\mathcal{C}'}(S_{\mathcal{C}'}, HS_{-\mathcal{C}'})$$

Proof. Consider some strategy $S_{\mathcal{C}'} \in \overline{\mathcal{R}}$ in which $\mathbf{E}_1$ or $\mathbf{E}_2$ happens with non-negligible probability. We can construct a new strategy for $\mathcal{C}'$ with strictly better expected utility. Specifically, consider a modified PPT strategy denoted $\widehat{S}_{\mathcal{C}'}$: whenever by the original strategy, the first of $\mathbf{E}_1$ or $\mathbf{E}_2$ is about to happen, $\mathcal{C}'$ simply sends a withdrawal transaction to PONYTA and stops sending any messages (including the message that is about to trigger $\mathbf{E}_1$ or $\mathbf{E}_2$) to the contract from that moment on.

We consider two cases, depending on whether $\mathbf{E}_1$ or $\mathbf{E}_2$ happens first in the original strategy $S_{\mathcal{C}'}$.

Event $\mathbf{E}_1$ happens first. When $\mathbf{E}_1$ happens, because Bob is not guaranteed to withdraw his deposit from PONYTA, the $1 - \alpha$ fraction of honest miners would send pre_b to the preparation phase of PONYTA and activate B2 for themselves, if they are chosen to mine a block. Once B2 is activated by an honest miner, the coalition $\mathcal{C}'$ gets nothing and simply loses all the deposit from PONYTA. Moreover, Alice will send pre_a only when PONYTA enters the execution phase, and PONYTA never enters the execution phase if B2 is activated. Therefore, the utility of $\mathcal{C}'$ is at most $(-\$v - \$c_b + \$E)$ if the honest miner activates B2. On the other hand, if B2 is not activated by an honest miner, the utility of $\mathcal{C}'$ is at most $\$v_b + \$E$ — since $\$c_b > \$v$, the maximum is achieved if PONYTA enters the execution phase, P2 and C1 are activated, and pre_a is learned by Bob. When $\mathbf{E}_1$ happens, the probability that B2 is activated by an honest miner is at least $1 - \alpha$. Thus, the utility of $\mathcal{C}'$ is at most

$$(1 - \alpha)(-\$v - \$c_b + \$E) + \alpha(\$v_b + \$E) < 0,$$

where the inequality arises from the fact that $\$c_b > \frac{\alpha\$v_b + \$E}{1-\alpha} - \$v$.

The above shows that conditioned on $\mathbf{E}_1$ happening, the conditional expected utility of the coalition is negative. We now show that the modified strategy $\widehat{S}_{\mathcal{C}'}$ performs strictly better. There are two cases.

- Case 1: When $\mathbf{E}_1$ is about to happen, Bob already sent the deposit transaction to PONYTA, i.e., $\mathbf{E}_1$ is about to be triggered by sending pre_b. If $\mathcal{C}'$ now instead plays $\widehat{S}_{\mathcal{C}'}$, no one would send any transaction conotaining pre_b to P2, C2 or the preparation phase of PONYTA. Thus, B2 and C2 cannot be activated. If PONYTA does not enter the execution phase, Bob will send the withdrawal transaction to PONYTA before stopping sending anything. If PONYTA enters the execution phase and Alice goes to the execution phase, Alice will send pre_a to P1, and send _ to C1 when T_2 has passed after P1 is activated. If PONYTA enters the execution phase and Alice goes to the abort phase, Alice will send _ to P2, and send _ to C1 when T_2 has passed after P2 is activated. The $1 - \alpha$ fraction of honest miners would include those transactions, and it is not hard to check the utility of $\mathcal{C}'$ is at least zero no matter in which case.

- Case 2: When $\mathbf{E}_1$ is about to happen, Bob has not sent the deposit transaction to PONYTA, i.e., $\mathbf{E}_1$ is about to be triggered by sending the deposit transaction. Then, if $\mathcal{C}'$ now instead plays $\widehat{S}_{\mathcal{C}'}$, Bob would never send the deposit transaction, so the utility of $\mathcal{C}'$ is at least zero.

Event $\mathbf{E}_2$ happens first. When $\mathbf{E}_2$ happens, it must be that the honest Alice sends pre_a to P1 at time T. The $1-\alpha$ fraction of honest miners would activate C2 for themselves if they are chosen to mine a block. Thus, the probability that C2 is successfully activated by someone not in $\mathcal{C}'$ is at least $1-\alpha$. If C2 is activated by an honest miner, then the utility of $\mathcal{C}'$ is at most $\$v_b - \$c_b + \$E$, and the maximum is achieved if pre_a is learned by Bob and P2 is activated. On the other hand, if C2 is not activated by an honest miner, then the utility of $\mathcal{C}'$ cannot exceed $\$v_b + \$E$ assuming $\$c_b > \$v$, and the maximum is achieved if pre_a is learned by Bob and P2, C1 are activated. Therefore, conditioned on $\mathbf{E}_2$ happening, the utility of $\mathcal{C}'$ is at most

$$(1-\alpha)(\$v_b - \$c_b + \$E) + \alpha(\$v_b + \$E) < 0,$$

where the inequality arises from the fact that $\$c_b > \frac{\$v_b + \$E}{1-\alpha}$. Note that the above analysis holds even if $\mathcal{C}'$ 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.

The above shows that conditioned on $\mathbf{E}_2$ happening, the conditional expected utility of the coalition is negative. We now show that the modified strategy $\widehat{S}_{\mathcal{C}'}$ performs strictly better. When $\mathbf{E}_2$ is about to happen, Ponyta already entered the execution phase at time $t \le T$. Thus, Alice must enter the execution phase, and she will send pre_a to P1. Further, T_2 time after P1 is activated, she sends _ to C1. Conditioned on the traces in which $\mathbf{E}_2$ happens first when playing the original strategy $S_{\mathcal{C}'}$, if $\mathcal{C}'$ now plays $\widehat{S}_{\mathcal{C}'}$, no one in $\mathcal{C}'$ will post any transaction containing pre_b to P2 or C2, thus, P2 and C2 cannot be activated. The $1-\alpha$ fraction of honest miners would include Alice's transactions, so except the negligible probability, P1 and C1 will be activated in polynomial time. Thus, the utility of $\mathcal{C}'$ is $\$v_b - \$v > 0$.

□

Lemma 6.4 (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 $\mathcal{C}'$ consists of Alice and miners controlling at most $\alpha \le 1 - 1/\mathsf{poly}(\lambda)$ fraction of the mining power. Given any strategy $S_{\mathcal{C}'} \in \overline{\mathcal{R}}$ for some (externally incentivized) coalition $\mathcal{C}'$, there is a strategy $\widehat{S}_{\mathcal{C}'}$ such that*

$$\mathsf{util}^{\mathcal{C}'}(\widehat{S}_{\mathcal{C}'}, HS_{-\mathcal{C}'}) \ge \mathsf{util}^{\mathcal{C}'}(S_{\mathcal{C}'}, HS_{-\mathcal{C}'})$$

Proof. Consider some strategy $S_{\mathcal{C}'} \in \overline{\mathcal{R}}$ in which $\mathbf{E}_3$, $\mathbf{E}_4$ or $\mathbf{E}_5$ happens with non-negligible probability. We define a modified PPT strategy $\widehat{S}_{\mathcal{C}'}$ to be the following, whenever the first of $\mathbf{E}_3$, $\mathbf{E}_4$ or $\mathbf{E}_5$ is about to happen, send a withdrawal transaction to Ponyta, and stop sending any other messages including the message that is about to trigger $\mathbf{E}_3$, $\mathbf{E}_4$ or $\mathbf{E}_5$, from that moment on. We want to argue that the modified strategy $\widehat{S}_{\mathcal{C}'}$ performs strictly better for $\mathcal{C}'$. We consider three cases, depending on whether $\mathbf{E}_3$, $\mathbf{E}_4$ or $\mathbf{E}_5$ happens first in the original strategy $S_{\mathcal{C}'}$.

Event $\mathbf{E}_3$ happens first. In this case, the $1-\alpha$ fraction of honest miners would send pre_a to the preparation phase of Ponyta and activate A2 for themselves, if they are chosen to mine a block. Once A2 is activated by an honest miner, the coalition $\mathcal{C}'$ gets nothing and simply loses all the deposit from Ponyta. Moreover, pre_a is publicly learned. Therefore, the utility of $\mathcal{C}'$ is at most $(-\$v_a - \$c_a + \$E)$ if the honest miner activates A2. On the other hand, if A2 is not activated by an honest miner, the utility of $\mathcal{C}'$ is at most $(-\$v_a + \$E)$. When $\mathbf{E}_3$ happens, the probability that A2 is activated by an honest miner is at least $1-\alpha$. Thus, the utility of $\mathcal{C}'$ is at most

$$(1-\alpha)(-\$v_a - \$c_a + \$E) + \alpha(-\$v_a + \$E) < 0,$$

where the inequality arises from the fact that $\$c_a > \frac{-\$v_a + \$E}{1-\alpha}$.

The above shows that conditioned on $\mathbf{E}_3$ happening, the conditional expected utility of the coalition is negative. We now show that the modified strategy $\widehat{S}_{\mathcal{C}'}$ performs strictly better. There are two cases.

- Case 1: When $\mathbf{E}_3$ is about to happen, Alice already sent the deposit transaction to Ponyta, i.e., $\mathbf{E}_3$ is about to be triggered by sending pre_a. If $\mathcal{C}'$ now instead plays $\widehat{S}_{\mathcal{C}'}$, no one would send any transaction conotaining pre_a to P1, C2 or the preparation phase of Ponyta. Thus, A2, P1 and C2 cannot be activated. If Ponyta does not enter the execution phase, Alice will send the withdrawal transaction to Ponyta before becoming silent. If Ponyta enters the execution phase, Bob will send pre_b to P2, and send $\_$ to C1 when T_2 has passed after either P2 is activated. The $1-\alpha$ fraction of honest miners would include those transactions, thus, the utility of $\mathcal{C}'$ is at least zero except with negligible probability.
- Case 2: When $\mathbf{E}_3$ is about to happen, Alice has not sent the deposit transaction to Ponyta, i.e., $\mathbf{E}_3$ is about to be triggered by sending the deposit transaction. Then, if $\mathcal{C}'$ now instead plays $\widehat{S}_{\mathcal{C}'}$, Alice would never send the deposit transaction, so the utility of $\mathcal{C}'$ is at least zero.

Event $\mathbf{E}_4$ happens first. Because Ponyta enters the execution phase at some time $t > T$, Bob will go to the abort phase, and he sends pre_b to P2 at time t. Therefore, as soon as the event $\mathbf{E}_4$ happens, both pre_a and pre_b would be publicly known. In this case, the $1-\alpha$ fraction of honest miners would activate C2 for themselves if they are chosen to mine a block. Thus, the probability that C2 is successfully activated by an honest miner is at least $1-\alpha$. Notice that Alice's utility is deducted by $\$v_a$ if pre_a is learned by Bob. If C2 is activated by an honest miner, then the utility of $\mathcal{C}'$ is at most $-\$v_a - \$c_a + \$E$. On the other hand, if C2 is not activated by an honest miner, then the utility of $\mathcal{C}'$ cannot exceed $-\$v_a + \$E$ assuming $\$c_a > \$v$. Therefore, conditioned on $\mathbf{E}_4$ happening, the utility of $\mathcal{C}'$ is at most

$$(1-\alpha)(-\$v_a - \$c_a + \$E) + \alpha(-\$v_a + \$E) < 0,$$

where the inequality arises from the fact that $\$c_a > \frac{-\$v_a+\$E}{1-\alpha}$.

The above shows that conditioned on $\mathbf{E}_4$ happening, the conditional expected utility of the coalition is negative. We now show that the modified strategy $\widehat{S}_{\mathcal{C}'}$ performs strictly better. If $\mathcal{C}'$ now instead plays $\widehat{S}_{\mathcal{C}'}$, no one in $\mathcal{C}'$ would send pre_a to P1 or C2. Thus, P1 and C2 cannot be activated. In the abort phase, Bob will send pre_b to P2, and he will send $\_$ to C1 when T_2 has passed after P2 is activated. The $1-\alpha$ fraction of honest miners would include Bob's transactions, so except the negligible probability, P2 and C1 will be activated in polynomial time. Thus, the utility of $\mathcal{C}'$ is at least zero no matter in which case.

Event $\mathbf{E}_5$ happens first. Because Alice does not post pre_a to P1 in $t \in [0, T_1)$, Bob will post pre_b at time T_1. Therefore, as soon as the event $\mathbf{E}_5$ happens, both pre_a and pre_b would be publicly known. By using the same argument as the previous case, the utility of $\mathcal{C}'$ is at most

$$(1-\alpha)(-\$v_a - \$c_a + \$E) + \alpha(-\$v_a + \$E) < 0.$$

Note that the above analysis holds even if $\mathcal{C}'$ 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.

The above shows that conditioned on $\mathbf{E}_5$ happening, the conditional expected utility of the coalition is negative. We now show that the modified strategy $\widehat{S}_{\mathcal{C}'}$ performs strictly better. If $\mathcal{C}'$ now instead plays $\widehat{S}_{\mathcal{C}'}$, no one in $\mathcal{C}'$ would send pre_a to P1 or C2. Thus, P1 and C2 cannot be activated. Because Alice does not post pre_a to P1 in $t \in [0, T_1)$, Bob will send pre_b to P2 no matter

he is in the execution phase or the abort phase, and he will send _ to C1 when T_2 has passed after P2 is activated. The $1-\alpha$ fraction of honest miners would include Bob's transactions, so except the negligible probability, P2 and C1 will be activated in polynomial time. Thus, the utility of $\mathcal{C}'$ is at least zero no matter in which case.

□

6.3 Proofs of Safe Participation Against External Incentives

Lemma 6.5 (Against externally incentivized Bob-miner coalition)**.** *Suppose that $H(\cdot)$ is a one-way function. Let $\mathcal{C}$ be a coalition consisting of Alice and possibly any subset of the miners, and let $\mathcal{C}'$ be a disjoint coalition consisting of Bob and at most $\alpha \leq 1 - 1/\mathsf{poly}(\lambda)$ fraction of the mining power. Suppose $\mathcal{C}$ does not have external incentives but $\mathcal{C}'$ may have up to $\$E$ amount of external incentives. Let $S_{\mathcal{C}'}$ be an arbitrary (possibly unbounded) strategy of $\mathcal{C}'$ that is not in the family $\overline{\mathcal{R}}$. Then, there exists a negligible function $\mathsf{negl}(\cdot)$ s.t.*

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

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

Proof. There are two cases.

Case 1: Alice goes to the abort phase. If Alice goes to the abort phase, she never sends any transaction containing pre_a. Ignoring the negligible probability that $\mathcal{C}'$ guess pre_a, A2, P1 and C2 cannot be activated. In the abort phase, Alice will send the withdrawal transaction to A1. Moreover, if Ponyta enters the execution phase, Alice will send _ to P2, and send _ to C1 when T_2 has passed after P2 is activated. The $1-\alpha$ fraction of the mining power will include Alice's transactions, so except the negligible probability, Alice can get all her deposit back in polynomial time. Thus, utility of the $\mathcal{C}$ is at least zero.

Case 2: Alice goes to the execution phase. In this case, Ponyta must enter the execution phase. Because $\mathbf{E}_1$ and $\mathbf{E}_2$ do not happen, $\mathcal{C}'$ will not post pre_b to P2 or C2 unless C1 is guaranteed to be activated. Because C1 is guaranteed to be activated only if P1 or P2 is activated, P2 and C2 cannot be activated. In the execution phase, Alice will send pre_a to P1, and send _ to C1 when T_2 has passed after P1 is activated. The $1-\alpha$ fraction of the mining power will include Alice's transactions, so except the negligible probability, P1 and C1 will be activated in polynomial time, in which case the utility of the honest $\mathcal{C}$ is at least $\$v - \$v_a > 0$.

□

Lemma 6.6 (Against externally incentivized Alice-miner coalition)**.** *Suppose that $H(\cdot)$ is a one-way function. Let $\mathcal{C}$ be a coalition consisting of Bob and possibly any subset of the miners, and let $\mathcal{C}'$ be a disjoint coalition consisting of Alice and at most $\alpha \leq 1 - 1/\mathsf{poly}(\lambda)$ fraction of the mining power. Suppose that $\mathcal{C}$ does not have external incentives but $\mathcal{C}'$ may have up to $\$E$ amount of external incentives. Let $S_{\mathcal{C}'}$ be an arbitrary PPT rational strategy of $\mathcal{C}'$ that is not in $\overline{\mathcal{R}}$. Then, there exists a negligible function $\mathsf{negl}(\cdot)$ such that*

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

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

Proof. There are three cases.

Case 1: Ponyta does not enter the execution phase before time T. In this case, Bob will go to the abort phase. Because $\mathbf{E}_3$ and $\mathbf{E}_4$ do not happen, anyone in $\mathcal{C}'$ would not send any transaction containing pre_a to P1 or C2 before C1 is guaranteed to be activated. However, C1 is guaranteed to be activated implies either P1 or P2 has been activated. Thus, P1 and C2 cannot be activated. If Ponyta does not enter the execution phase at all, Bob will send the withdrawal transaction to Ponyta. If Ponyta enters the execution phase after time T, Bob will send pre_b to P2, and send $\_$ to C1 when T_2 has passed after P2 is activated. Because $1-\alpha$ fraction of the miners will include Bob's transactions, except the negligible probability, the utility of $\mathcal{C}$ is zero.

Case 2: Ponyta enters the execution phase before time T and Alice's coalition $\mathcal{C}'$ posts a transaction containing pre_a to P1 in $t \in [0, T_1)$. If $\mathcal{C}$ follows the honest strategy, it would not post any pre_b to P2. Ignoring the negligible probability event that someone in $\mathcal{C}'$ can compute pre_b, C2 cannot be activated. Because $1-\alpha \geq 1/\mathsf{poly}(\lambda)$, except with negligible probability, C1 and one of (P1 and P2) must be activated in polynomially time. If it turns out P1 is activated, the utility of $\mathcal{C}$ is $\$v_b - \$v > 0$. If it turns out P2 is activated, the utility of $\mathcal{C}$ is $\$v_b > 0$.

Case 3: Ponyta enters the execution phase before time T and Alice's coalition $\mathcal{C}'$ does not post a transaction containing pre_a to P1 in $t \in [0, T_1)$. The honest $\mathcal{C}$ posts pre_b to P2 at time T_1, and posts an empty message $\_$ to C1 when T_2 has elapsed since P2 is activated. Because $\mathbf{E}_3$ and $\mathbf{E}_5$ do not happen, $\mathcal{C}'$ will not post pre_a to P1 until C1 is guaranteed to be activated. However, C1 is guaranteed to be activated implies either P1 or P2 has been activated. Thus, P1 and C2 cannot be activated. Moreover, $1-\alpha \geq 1/\mathsf{poly}(\lambda)$, it must be that both P2 and C1 are activated in polynomial time except with negligible probability. As long as both P2 and C1 are activated, $\mathcal{C}$'s utility is non-negative since Bob gets all of its deposit $\$c_b + \$v$ back. □

Theorem 6.7 (Safe participation against external incentives)**.** *Suppose that $H(\cdot)$ is a one-way function and suppose the parameters $\$c_a, \$c_b, \alpha$ satisfy the relations specified in Section 6.1. Then, the* Ponyta *protocol satisfies α-safe-participation against external incentives w.r.t. to the strategy space $\mathcal{R}$ that consists of all PPT strategies not in $\overline{\mathcal{R}}$.*

Proof. We can divide into the following cases:

Case 1: Alice $\in \mathcal{C}$ and Bob $\in \mathcal{C}'$: covered by Lemma 6.5.

Case 2: Bob $\in \mathcal{C}$ and Alice $\in \mathcal{C}'$: covered by Lemma 6.6.

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: Suppose $\mathcal{C}$ acts honestly and posts pre_a to P1 at $t = T$. Since Bob is assumed to be honest, he will not post pre_b. Ignoring the negligible probability event that someone in $\mathcal{C}'$ can compute pre_b, P2 and C2 cannot be activated. Observing that $1-\alpha > 1/\mathsf{poly}(\lambda)$, it must be that both P1 and C1 are confirmed in polynomial time except with negligible probability. As long as P1 and C1 are both confirmed, the honest $\mathcal{C}$ obtains non-negative utility.

Case 5: Bob $\in \mathcal{C}$ and $\mathcal{C}'$ is miner-only: Because both Alice and Bob are assumed to be honest, by using the same argument as in the previous case, it must be that both P1 and C1 are confirmed in polynomial time except with negligible probability. As long as P1 and C1 are both confirmed, the honest $\mathcal{C}$ obtains non-negative utility. □

Corollary 6.8. *Suppose that $H(\cdot)$ is a one-way function. Further, suppose that $\$c_b > \max(\$v, \frac{\$v_b+\$E}{1-\alpha}, \frac{\$v}{1-\gamma^{T_2}})$, $\$c_a > \max(\$v, \frac{-\$v_a+\$E}{1-\alpha})$. Then, our* Ponyta *protocol satisfies γ-CSP-fairness, α-safe-participation against external incentives, as well as dropout resilience.*

Proof. Straightforward by combining Theorem 6.1, Theorem 6.7, and Theorem 6.2. □

7 Ponyta 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. 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 Ponyta *provides strong disincentives for such a 100% coalition to form*, even in a world where one can post bribery contracts [Bon16, JSZ+19, MHM18b, 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. 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 $\$v$ 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 $\$v$ 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 $1 - (1-\gamma)^{T_2} > \gamma$. Therefore, if I do not join the coalition, my expected gain would be strictly greater than $\$v \cdot \gamma$ which is the expected gain had I joined 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.

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 [MHM18a, Bon, 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 Ponyta 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 Application: Atomic Coin Swap

Atomic swap allows two parties to exchange cryptocurrencies in the same chain or accross two different chains. In this section, we describe how to realize atomic swap by using Ponyta, and we show that the protocol satisfies CSP fairness and safe participation against external incentives.

We will describe our protocol for a cross-chain scenario since cross-chain swap is more general than over the same chain. We take Bitcoin and Ethereum as an example. We use $\text{₿}x'$ to denote x' units of Bitcoin, and use $\text{Ξ}x$ to denote x units of Ether. We use the notation $\text{₿}x' + \text{Ξ}x$ to denote x' units of Bitcoin and x units of Ether. In general, our atomic swap protocol can be used on any two chains where the mining process picks block proposers at random proportional either to their mining power or stake.

8.1 Model and Utility

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, $\$\mathsf{AV}(\text{Ξ}x + \text{₿}x') = \$v_a \cdot x + \$v'_a \cdot x'$ 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 $\text{₿}x'$ with Bob for $\text{Ξ}x$, and vice versa.

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

Utility. Let $\text{₿}d'_a, \text{Ð}d_a \geq 0$ be the cryptocurrencies that Alice or an Alice-miner coalition deposit into the smart contracts. Let $\text{₿}r'_a, \text{Ð}r_a \geq 0$ 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:

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

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:

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

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

where $\text{₿}r'_b, \text{Ð}r_b \geq 0$, denote the payment the Bob-miner coalition or Bob receives during the protocol, $\text{₿}d'_b, \text{Ð}d_b \geq 0$ 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 $\text{₿}r'_m, \text{Ð}r_m, \text{₿}d'_m, \text{Ð}d_m, \$e_m(\ldots) \geq 0$ are similarly defined but for the miner-only coalition.

Like before, we assume that the total utility of all players from the protocol cannot exceed a polynomial function in the security parameter λ.

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'.

Execution model. Since the cross-chain swap application involves two different chains, we need to make a few changes to the execution model. Recall that earlier in Section 3, users can post messages or smart contracts at the beginning of each time step t, and then a miner will be chosen to mine the block of the current time step t. The block mined can contain messages posted at any time less than or equal to t.

Now, we may imagine that users can post messages or at any (wall-clock) time. At any wallclock time t when there is a mining event for either the Bitcoin or Ethereum chain, a miner is chosen, and the newly mined block may contain any message posted at any time less than or equal to t.

In this section, without loss of generality, we may assume that our honest protocol involves one contract on each of the two chains — if there are multiple contracts on the same chain, we can always merge them into a single one. The moment both contracts are posted to their respective blockchains and take effect, we define the current time to be 0, and this marks the start of the protocol execution. The deposit phase in which users deposit money into the two contracts is considered part of the execution.

8.2 Construction

Notations. To realize a cross-chain atomic swap, we use two Ponyta contract instances, denoted Ponyta and Ponyta', for the Ethereum and Bitcoin chains, respectively. The idea is that Alice

would act first and redeem Ethers from Ponyta by revealing pre_a, and pre_a is exactly the secret Bob needs to redeem Bitcoins from Ponyta'.

In Section 5.1, we specify the parameters for a Ponyta contract to be a 7-tuple: (h_a, h_b, T_1, T_2, $\$v$, $\$c_a$, $\$c_b$). Since we need two contracts now, we denote the parameters for Ponyta as (h_a, h_b, T_1, τ, $\ddot{\text{E}}x$, $\ddot{\text{E}}c_a$, $\ddot{\text{E}}c_b$), and the parameters for Ponyta' as ($h'_b, h'_a, T'_1, \tau', \ddot{\text{B}}x', \ddot{\text{B}}c'_b, \ddot{\text{B}}c'_a$). Here, T_1 and T'_1 denote the Ethereum time and Bitcoin time, respectively. By contrast, τ and τ' are the number of time intervals measured in terms of the number of Ethereum and Bitcoin blocks, respectively.

Before deploying the contracts, Alice samples pre_a and pre'_a from $\{0,1\}^\lambda$, and Bob samples pre_b from $\{0,1\}^\lambda$ uniformly at random. Further, we will let $pre'_b = pre_a$. They then post the Ponyta and Ponyta' contracts on the Ethereum and Bitcoin blockchains, respectively. At this moment, the execution begins and we may rename the current time to be 0. During the protocol execution, Alice and Bob will have a T' window measured in Bitcoin time to deposit Bitcoin into the Ponyta' contract first, and then they will have a T window measured in Ethereum time to deposit Ether into the Ponyta contract. Note that one difference from the earlier Section 5 is that here, we explicit consider the preparation phase (in which the parties make deposits) as part of the protocol execution since we now have two contracts, and the order of the deposits matters.

Parameter choices. The parameters must satisfy the following constaints for the protocol to satisfy CSP-fairness against coalitions with at most γ fraction of mining power.

Parameter Constraints for Atomic Swap

Constraints for Ponyta (on Ethereum):

- $h_a = H(pre_a)$ and $h_b = H(pre_b)$.
- $T_1 > T$.
- $\ddot{\text{E}}c_a > \ddot{\text{E}}x$ and $\ddot{\text{E}}c_b > \ddot{\text{E}}x$.

Constraints for Ponyta' (on Bitcoin):

- $h'_b = H(pre_a)$ and $h'_a = H(pre'_a)$.[a]
- Bitcoin time T' < Ethereum time T, i.e., the Bitcoin block of length T' is mined before the Ethereum block of length[b] T.
- 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.
- $\ddot{\text{B}}c'_a > \ddot{\text{B}}x'$ and $\ddot{\text{B}}c'_b > \ddot{\text{B}}x'$.

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

- $\tau \geq 1$, $\tau' \geq 1$.
- $\gamma^\tau \leq \frac{\$\mathsf{AV}(\ddot{\text{E}}c_a - \ddot{\text{B}}x')}{\$\mathsf{AV}(\ddot{\text{E}}c_a)}$, $\gamma^{\tau'} \leq \frac{\$\mathsf{AV}(\ddot{\text{B}}c'_a - \ddot{\text{B}}x')}{\$\mathsf{AV}(\ddot{\text{B}}c'_a)}$, and $\gamma^\tau \leq \frac{\$\mathsf{BV}(\ddot{\text{E}}c_b - \ddot{\text{E}}x)}{\$\mathsf{BV}(\ddot{\text{E}}c_b)}$.

[a] Recall that $h'_b = h_a$ and Alice holds both pre_a and pre'_a initially.

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

Intuitively, the collateral values must be sufficiently large such that a coalition involving either Alice or Bob would never want to trigger the "bombs" in the protocol that cause their collateral to

be (partially) burnt. The timeouts τ and τ' must be sufficiently large w.r.t. γ, such that a coalition involving Alice or Bob would never want to take any gamble that would risk getting their collateral (partially) burnt. Finally, the choices of T_1 and T_1' account for the fact that we want the users to deposit into Ponyta' first before depositing into Ponyta; but the execution order would be the opposite, i.e., Ponyta is executed first and and then Ponyta'.

The contracts. We now describe the two contract instances for realizing cross-chain swap. Below, any time (or time interval) in Ponyta is measured by Ethereum blocks, and any time (or time interval) in Ponyta' is measured by Bitcoin blocks.

Ponyta contract (on Ethereum) // Parameters: (h_a, h_b, T_1, τ, Ðx, Ðc_a, Ðc_b)

Preparation phase: Wait to receive Ðc_a deposit from Alice and Ðx + Ðc_b deposit from Bob. On receive pre_a such that $H(pre_a) = h_a$ or on receive pre_b such that $H(pre_b) = h_b$ from anyone: record the event and do nothing. *If Alice and Bob have both deposited, and no pre_a or pre_b has been recorded, the preparation phase ends and the execution phase starts.*

A1: On receive _ from Alice, send Ðc_a to Alice.

A2: On receive _ from anyone P: if a pre_a has been recorded and Alice has deposited Ðc_a, send Ðc_a to P.

B1: On receive _ from Bob, send Ðx + Ðc_b to Bob.

B2: On receive _ from anyone P: if a pre_b has been recorded and Bob has deposited Ðx + Ðc_b, send Ðx + Ðc_b to P.

Execution phase:

P1: On receive pre_a from Alice such that $H(pre_a) = h_a$, send Ðx to Alice.

P2: Time T_1 or greater: on receive pre_b from Bob such that $H(pre_b) = h_b$, or on receiving _ from Alice, send Ðx to Bob.

C1: At least τ after either P1 or P2 is activated: on receiving _ from anyone, send Ðc_a to Alice and send Ðc_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 Ðx to player P. All remaining coins are burnt.

Ponyta' contract (on Bitcoin) // Parameters: (h_b', h_a', T_1', τ', Ƀx', Ƀc_b', Ƀc_a')

Preparation phase: Wait to receive Ƀx' + Ƀc_a' deposit from Alice and Ƀc_b' deposit from Bob. On receive pre_a' such that $H(pre_a') = h_a'$ from anyone, record the event. If Alice and Bob have both deposited and pre_a' has not been recorded, the preparation phase ends and the execution phase starts.

A1': On receive _ from Alice, send Ƀx' + Ƀc_a' to Alice.

A2': On receive _ from anyone P: if a pre_a' has been recorded and Alice has deposited Ƀx' + Ƀc_a', send Ƀx' + Ƀc_a' to P.

B1': On receive _ from Bob, send Ƀc_b' to Bob.

Execution phase:

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

P2': Time T'_1 or greater: on receiving pre'_a from Alice such that $H(pre'_a) = h'_a$ or on receiving _ from Bob, send $\ddot{\mathrm{B}}x'$ to Alice.

C1': At least τ' after either P1' or P2' is activated: on receiving _ from anyone, send $\ddot{\mathrm{B}}c'_a$ to Alice and send $\ddot{\mathrm{B}}c'_b$ to Bob.

C2': On receiving (pre'_b, pre'_a) from anyone P such that $H(pre'_b) = h'_b$ and $H(pre'_a) = h'_a$, send $\ddot{\mathrm{B}}x'$ to player P. All remaining coins are burnt.

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

In the above, allowing Alice to invoke P1' by sending _ and allowing Bob to invoke P2' by sending _ are needed for dropout resilience — we will discuss these subtleties in more detail after describe the full coin swap protocol. Additionally, we let Alice post an empty message _ to invoke P2. Interestingly, unlike P1' and P2', this is not exactly needed for dropout resilience — as we explain later, it provides stronger resilience in the case of an externally incentivized Bob-miner coalition.

Atomic swap protocol. We now describe our coin swap protocol.

- *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), as well as all transactions posted to the preparation phase of PONYTA and PONYTA', to see if they contain a valid $pre_a = pre'_b$, pre_b, and pre'_a.
  - When PONYTA is still in the preparation phase, if the miner observes a valid pre_a, it immediately posts pre_a to the preparation phase of PONYTA, and it also posts _ to A2; if it ever sees a valid pre_b, it immediately posts pre_b to the preparation phase of PONYTA, and it also posts _ to B2. Similarly, when PONYTA' is still in the preparation phase, if the miner observes a valid pre'_a, it immediately posts pre'_a to the preparation phase of PONYTA', and it posts _ to A2'.
  - 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 both 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. Among its own transactions, it always includes the pre_a, pre_b, or pre'_a posted to the perparation phases of PONYTA or PONYTA' ahead of others.
- *Alice and Bob.* Below, we define the honest protocol for Alice and Bob. As mentioned, the start of the execution (i.e., time 0) is defined to be the moment both contracts have been posted and take effect.

Atomic Swap Protocol — Alice and Bob

Preparation Phase:

1. At $t = 0$, Alice and Bob deposit $\ddot{\mathrm{B}}x' + \ddot{\mathrm{B}}c'_a$ and $\ddot{\mathrm{B}}c'_b$ into PONYTA', respectively.
2. At Bitcoin time T': if PONYTA' has not entered the execution phase, go to the abort phase;

else, Alice and Bob deposit Ξc_a and Ξx + Ξc_b into PONYTA, respectively.

3. At Ethereum time T: if PONYTA has not entered the execution phase, then go to the abort phase; else go to the execution phase.

Execution Phase:

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

Abort Phase:

1. For either PONYTA or PONYTA': if Alice (or Bob, resp.) has submitted a deposit transaction, then submit a corresponding withdrawal transaction, i.e., an empty message _ posted to A1 and A1' (or B1 and B1', resp.).
2. As soon as PONYTA' enters the execution phase, Alice posts pre'_a to P2'. Once P2' has been activated, she posts _ to P2.
3. As soon as PONYTA enters the execution phase, Bob posts pre_b to P2. Once P2 has been activated or Bob has withdrawn his deposit from PONYTA or Bob entered the abort phase prior to sending its deposit transaction to PONYTA, Bob posts _ to P2';
4. If τ' *Bitcoin time* has passed since P1' or P2' is activated, send _ to C1'; similarly, if τ *Ethereum time* has passed since P1 or P2 is activated, send _ to C1.

As mentioned, the coin swap protocol consists of two contract instances, one running on each blockchain. Once the players have deposited money into the contracts, Alice will redeem Ether from the PONYTA contract first by posting pre_a to P1. This in turn allows Bob to redeem Bitcoin from PONYTA' by posting $pre'_b = pre_a$ to P1'.

Some subtleties. Observe that users deposit into PONYTA' first before PONYTA, but the execution order is reversed, i.e., PONYTA is executed first before PONYTA'. This is to prevent the attack where Alice redeems the Ethers from PONYTA first, but does not deposit into PONYTA' which prevents Bob from redeeming Bitcoins from PONYTA'.

In comparison with our contract in Section 5.1, there are a couple differences. First, we now additionally allow Alice to trigger P1' by posting an empty message _, and allow Bob to trigger P2' by posting _. These modifications are needed for dropout resilience. Specifically, the latter modification is needed in the following situation: if Alice aborts prior to posting her pre_a, we want to make sure that Bob can activate P2' (after activating P2 or withdrawing his deposit from PONYTA), so that he can get his deposit back from PONYTA'. The former modification is needed in the following situation. Suppose Alice has posted pre_a to P1 but Bob aborts. In this case, Alice should be able to trigger either P1' or P2'. Since Alice has already posted pre_a, it is not safe for her to post pre'_a to P2' since making both pre_a and pre'_a public can cause the bomb to be triggered.

Therefore, we want Alice to be able to trigger P1' by posting $\_$ after P1 is activated, such that she can eventually get her deposit back from Ponyta'.

Besides the above changes to P1' and P2', we also allow Alice to invoke P2 by posting an empty message $\_$. This helps in the following scenario in the presence of an externally incentivized Bob-miner coalition. Imagine that Bob fails to make his deposit early enough such that Alice enters the aborting phase which means that she will post pre'_a to P2' (this will happen if Ponyta' enters the execution phase). Now, imagine that Bob manages to deposit into Ponyta before Alice can withdraw her deposit from Ponyta. In this case, the external incentives may encourage Bob to delay posting pre_b to P2 which will cause Alice's deposit in Ponyta to be held up longer. Allowing Alice to invoke P2 with an empty message after P2' has been activated provides resilience to such a delaying strategy. At a more technical level, this also makes our proof easier since we need not argue that it is blatantly non-rational for Bob to not post pre_b in such a scenario.

Last but not the least, in the atomic swap protocol, the preparation phase is a little more sophisticated. In particular, we introduce several activation points to discourage the parties from posting pre_a, pre_b, or pre'_a too early, before the respective contracts enter the execution phase. This will be needed later in our proof of safe participation in the presence of external incentives.

8.3 Proof of CSP-Fairness Absent External Incentives

Lemma 8.1. *Let $\mathcal{C}$ be any coalition that consists of Alice and an arbitrary subset of miners controlling at most γ fraction of mining power. Then, as long as $\gamma^\tau \leq \frac{\$\mathsf{AV}(\text{Ð}c_a - \text{Ð}x')}{\$\mathsf{AV}(\text{Ð}c_a)}$ and $\gamma^{\tau'} \leq \frac{\$\mathsf{AV}(\text{Ð}c'_a - \text{Ð}x')}{\$\mathsf{AV}(\text{Ð}c'_a)}$, 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. If $\mathcal{C}$ follows the honest strategy, P1, C1, P1', and C1' will be activated, and the utility of $\mathcal{C}$ is $\$\mathsf{AV}(\text{Ð}x - \text{Ð}x') > 0$. Now, we analyze the cases if $\mathcal{C}$ deviates from the honest strategy, and show that the utility never exceeds the honest case. 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 $\mathcal{C}$ will not invoke the smart contract. Further, we may assume that $\mathcal{C}$ cannot gain anything from the activation point B2, since the honest Bob will not post pre_b unless Ponyta enters the execution phase, and the probability that $\mathcal{C}$ guesses the correct pre_b is negligible.

In the following, we do the case analysis for each phase.

Bob goes to the abort phase at Bitcoin time T'. In this case, Alice fails to make her deposit into Ponyta' by Bitcoin time T'. Further, Bob never deposits anything into Ponyta. Henceforth we may assume that $\mathcal{C}$ can withdraw whatever it deposits into Ponyta (i.e., $\mathcal{C}$ neither gains nor loses from Ponyta), since otherwise its utility will only be smaller. Suppose Ponyta' enters the execution phase. Since $\text{Ð}c'_a > \text{Ð}x'$, $\text{Ð}x' + \text{Ð}c'_a$ is the optimal amount that $\mathcal{C}$ can get from Ponyta' (from P2' and C1'), which is the same amount that Alice deposits into Ponyta'. On the other hand, if Ponyta' does not enter the execution phase, then Alice cannot gain from Ponyta'. No matter in which case, the utility of $\mathcal{C}$ is at most 0.

Bob goes to the abort phase at Ethereum time T. In this case, because we require Bitcoin time T' is earlier than Ethereum time T, Ponyta' must enter the execution phase, and Bob

deposits $\ddot{\text{E}}x + \ddot{\text{E}}c_b$ into PONYTA. If PONYTA does not enter the execution phase, then, the optimal utility of $\mathcal{C}$ is 0 by activating P2' and C1'.

Below, we focus on the case when PONYTA' enters the execution phase. Since $\ddot{\text{E}}c_a > \ddot{\text{E}}x$, and $\ddot{\text{B}}c'_a > \ddot{\text{B}}x'$, the utility of $\mathcal{C}$ can exceed $\$\mathsf{AV}(\ddot{\text{E}}x - \ddot{\text{B}}x')$ only if both P1 and P2' are activated. In the following, we will show that, because $\mathcal{C}$ only controls at most γ fraction of mining power, its expected utility cannot exceed $\$\mathsf{AV}(\ddot{\text{E}}x - \ddot{\text{B}}x')$ if P1 is activated.

Because Bob goes to the abort phase at Ethereum time T, PONYTA must enter the execution phase at Ethereum time $t' \geq T$. Moreover, Bob posts pre_b to P2 at Ethereum time t' as part of the abort procedure. Now, P1 is activated at some Ethereum time $t^* \geq t'$. Thus, both pre_a and pre_b are publicly known after t^*. Thus, during Ethereum time $t = (t^*, t^* + \tau]$, any miner in $-\mathcal{C}$ will activate C2 if it mines a block. We say $\mathcal{C}$ loses the race if a non-colluding miner mines a new block during Ethereum time $(t^*, t^* + \tau]$. Otherwise, we say $\mathcal{C}$ wins the race. If $\mathcal{C}$ loses the race, it gets nothing from C1 or C2, so its utility is at most $\$\mathsf{AV}(\ddot{\text{E}}x - \ddot{\text{E}}c_a)$ (from P1, P2' and C1'). Else if $\mathcal{C}$ wins the race, then its utility is at most $\$\mathsf{AV}(\ddot{\text{E}}x)$, and the maximum utility can be obtained only by activating P1, C1, P2' and C1'. The probability p that $\mathcal{C}$ wins the race is upper bounded by $p \leq \gamma^\tau$. Therefore, the expected utility of $\mathcal{C}$ is upper bounded by

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

Since $p \leq \gamma^\tau \leq \frac{\$\mathsf{AV}(\ddot{\text{E}}c_a - \ddot{\text{B}}x')}{\$\mathsf{AV}(\ddot{\text{E}}c_a)}$, we have

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

Bob goes to the execution phase. In this case, both PONYTA and PONYTA' enter the execution phase. In the execution phase, $\mathcal{C}$ can behave in the following ways.

1. Suppose Alice sends pre_a to P1 before Ethereum time T_1. As we have shown , the utility of $\mathcal{C}$ exceeds $\$\mathsf{AV}(\ddot{\text{E}}x - \ddot{\text{B}}x')$ only if P1, C1, P2', and C1' are activated. Suppose that P2' is activated at Bitcoin time $t^* \geq T'_1$. Thus, pre'_a is publicly known after Bitcoin time t^*. Moreover, since Bitcoin time $T'_1 >$ Ethereum time T_1, when P2' is activated, pre_a is also publicly known. Thus, during Bitcoin time $(t^*, t^* + \tau']$, any miner in $-\mathcal{C}$ will activate C2' if it wins a block. We say $\mathcal{C}$ loses the race if a non-colluding miner mines a new block during Bitcoin time $(t^*, t^* + \tau']$. Otherwise, we say $\mathcal{C}$ wins the race. If $\mathcal{C}$ loses the race, it gets nothing from C1' or C2', and it gets at most $\$\mathsf{AV}(\ddot{\text{E}}c_a)$ from the union of C1 and C2 since $\ddot{\text{E}}c_a > \ddot{\text{E}}x$. Thus, its utility is at most $\$\mathsf{AV}(\ddot{\text{E}}x - \ddot{\text{B}}c'_a)$ (from P1, C1 and P2'). Else if $\mathcal{C}$ wins the race, then its utility is at most $\$\mathsf{AV}(\ddot{\text{E}}x)$ which can be achieved by activating P1, C1, P2' and C1'. The probability p that $\mathcal{C}$ wins the race is upper bounded by $p \leq \gamma^{\tau'}$. Therefore, the expected utility of $\mathcal{C}$ is upper bounded by

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

   Since $p \leq \gamma^{\tau'} \leq \frac{\$\mathsf{AV}(\ddot{\text{B}}c'_a - \ddot{\text{B}}x')}{\$\mathsf{AV}(\ddot{\text{B}}c'_a)}$, we have

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

2. Suppose Alice does not send pre_a to P1 before Ethereum time T_1. In this case, Bob will send pre_b to P2 at Ethereum time T_1. If P2 is activated, the utility of $\mathcal{C}$ is at most 0 (from C1, P2' and C1'), which is less than the honest case. On the other hand, suppose P1 is activated at

some Ethereum time $t^* \geq T_1$. Then, both pre_a and pre_b are publicly known after Ethereum time t^*. Using the same argument we used for the case "Bob goes to the abort phase at Ethereum time T", since $p \leq \gamma^\tau \leq \frac{\$\mathsf{AV}(\ddot{\mathbb{E}}c_a - \ddot{\mathbb{B}}x')}{\$\mathsf{AV}(\ddot{\mathbb{E}}c_a)}$, the utility of $\mathcal{C}$ is upperbounded by

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

□

Lemma 8.2. *Suppose that the hash function $H(\cdot)$ is a one-way function. 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 $\gamma^\tau \leq \frac{\$\mathsf{BV}(\ddot{\mathbb{E}}c_b - \ddot{\mathbb{E}}x)}{\$\mathsf{BV}(\ddot{\mathbb{E}}c_b)}$, for any PPT coalition strategy $S_{\mathcal{C}}$, it must be that there is a negligible function $\mathsf{negl}(\cdot)$ such that*

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

Proof. If $\mathcal{C}$ follows the honest strategy, P1, C1, P1', and C1' will be activated, and the utility of $\mathcal{C}$ is $\$\mathsf{BV}(\ddot{\mathbb{B}}x' - \ddot{\mathbb{E}}x) > 0$. Now, we analyze the cases if $\mathcal{C}$ deviates from the honest strategy, and show that the utility never exceeds the honest case. 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 $\mathcal{C}$ will not invoke the smart contract.

Alice aborts. In this case, Alice never sends pre_a to P1. We may assume that $\mathcal{C}$ cannot gain anything from A2 or A2', since the honest Alice never sends pre_a and she does not send pre'_a before Ponyta' enters the execution phase. Since Alice never posts pre_a, except with negligible probability, P1' is never activated. Since $\ddot{\mathbb{B}}c'_b > \ddot{\mathbb{B}}x'$, the coalition $\mathcal{C}$ can get at most $\ddot{\mathbb{B}}c'_b$ from Ponyta' which is the same as its deposit.

We now analyze the Ponyta contract. There are two cases:

- Ponyta never entered the execution phase. In this case, the coalition $\mathcal{C}$ cannot gain anything from Ponyta.
- Ponyta entered the execution phase. Since $\ddot{\mathbb{E}}c_b > \ddot{\mathbb{E}}x$, the coalition $\mathcal{C}$ can get at most $\ddot{\mathbb{E}}(c_b + x)$ from Ponyta which is the same its deposit.

In both cases, $\mathcal{C}$'s utility is at most 0.

Alice goes to the execution phase. In this case, both Ponyta and Ponyta' enter the execution phase. Recall that if $\mathcal{C}$ plays honestly, P1, C1, P1', and C1' will be activated and its utility is $\$\mathsf{BV}(\ddot{\mathbb{B}}x' - \ddot{\mathbb{E}}x)$. Because $\ddot{\mathbb{E}}c_b > \ddot{\mathbb{E}}x$ and $\ddot{\mathbb{B}}c'_b > \ddot{\mathbb{B}}x'$, the utility of $\mathcal{C}$ exceeds $\$\mathsf{BV}(\ddot{\mathbb{B}}x' - \ddot{\mathbb{E}}x)$ only if P2 is activated.

Because Alice is honest, she always sends pre_a to P1 at Ethereum time T. Now, suppose P2 is activated at t^*. Notice that P2 can be activated only after Ethereum time $T_1 > T$, at which pre_a is already publicly known. Notice that the activation of P2 implies pre_b is publicly known. Thus, during Ethereum time $(t^*, t^* + \tau]$, any miner in $-\mathcal{C}$ will activate C2 if it mines a block. We say $\mathcal{C}$ loses the race if a non-colluding miner mines a new block during Ethereum time $(t^*, t^* + \tau]$. Otherwise, we say $\mathcal{C}$ wins the race. If $\mathcal{C}$ loses the race, it gets nothing from C1 or C2, and since $\ddot{\mathbb{B}}c'_b > \ddot{\mathbb{B}}x'$, $\mathcal{C}$ can get at most $\$\mathsf{BV}(\ddot{\mathbb{B}}c'_b)$ from the union of C1' and C2'. Thus its utility is at most $\$\mathsf{BV}(\ddot{\mathbb{B}}x' - \ddot{\mathbb{E}}c_b)$ which is achieved by activating from P2, P1' and C1'. Else if $\mathcal{C}$ wins the race, then its utility is at most $\$\mathsf{BV}(\ddot{\mathbb{B}}x')$ which is achieved by invoking P2, C1, P1' and C1'. The probability

p that $\mathcal{C}$ wins the race is upper bounded by $p \leq \gamma^\tau$. Therefore, the expected utility of $\mathcal{C}$ is upper bounded by

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

Since $p \leq \gamma^\tau \leq \frac{\$\mathsf{BV}(\ddot{\mathrm{E}}c_b - \ddot{\mathrm{E}}x)}{\$\mathsf{BV}(\ddot{\mathrm{E}}c_b)}$, we have

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

□

Theorem 8.3 (CSP fairness absent external incentives for coin swap)**.** *Suppose that the hash function $H(\cdot)$ is a one-way function. Moreover, suppose $\gamma^\tau \leq \frac{\$\mathsf{AV}(\ddot{\mathrm{E}}c_a - \ddot{\mathrm{B}}x')}{\$\mathsf{AV}(\ddot{\mathrm{E}}c_a)}$, $\gamma^{\tau'} \leq \frac{\$\mathsf{AV}(\ddot{\mathrm{B}}c'_a - \ddot{\mathrm{B}}x')}{\$\mathsf{AV}(\ddot{\mathrm{B}}c'_a)}$, and $\gamma^\tau \leq \frac{\$\mathsf{BV}(\ddot{\mathrm{E}}c_b - \ddot{\mathrm{E}}x)}{\$\mathsf{BV}(\ddot{\mathrm{E}}c_b)}$. Then, the atomic swap protocol satisfies γ-CSP-fairness.*

Proof. Lemma 8.1 and Lemma 8.2 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.

Henceforth, we consider the miner-only coalition, so Alice and Bob always follow the protocol. In the protocol, Alice and Bob decide whether they will go to the abort phase at Bitcoin time T' and at Ethereum time T, i.e., they make this decision using on-chain state when the blockchains have reached lengths T' and T, respectively. Therefore, both Alice and Bob go to the execution phase or both of them go to the abort phase. Now, a miner-only coalition can get positive utility only if at least one of the following is activated: A2, B2, A2', C2 or C2'. Since the honest Alice and Bob never post pre_a, pre_b, pre'_a before the corresponding contract enters the execution phase, except with negligible probability, the miner-only coalition cannot profit from A2, B2, or A2'. We now show that the miner-only coalition cannot profit from C2 or C2' except with negligible probability. If both Alice and Bob go to the execution phase, they would never post any transaction containing pre_b or pre'_a. Since $H(\cdot)$ is one-way and the coalition is PPT, C2 and C2' are never activated except with negligible probability. On the other hand, if both Alice and Bob go to the abort phase, they would never post any transaction containing $pre_a = pre'_b$. Since $H(\cdot)$ is one-way and the coalition is PPT, C2 and C2' are never activated except with negligible probability. To sum up, except with negligible probability, the utility of the coalition is at most 0, which is the same as the honest case. □

8.4 Achieving Safe Participation Against External Incentives

8.4.1 Parameter Choices

To achieve safe participation against external incentives, our construction is still the same as the one in Section 8.2. However, we now increase the players' collateral $\ddot{\mathrm{E}}c_a$, $\ddot{\mathrm{B}}c'_a$, $\ddot{\mathrm{E}}c_b$ and $\ddot{\mathrm{B}}c'_b$ w.r.t. the maximum amount of external incentives. Henceforth, let $\alpha \in [0, 1 - \mathsf{poly}(\lambda)]$ denote the maximum fraction of mining power controlled by the set of externally incentived players, and let $\$E$ be the maximum amount of external incentive available. We will set $\ddot{\mathrm{E}}c_a$, $\ddot{\mathrm{B}}c'_a$, $\ddot{\mathrm{E}}c_b$ and $\ddot{\mathrm{B}}c'_b$ such that the following relations are satisfied:

- $\$\mathsf{AV}(\ddot{\mathrm{E}}c_a) > \frac{\$\mathsf{AV}(\ddot{\mathrm{E}}x + \ddot{\mathrm{B}}x') + \$E}{1-\alpha}$,
- $\$\mathsf{AV}(\ddot{\mathrm{B}}c'_a) > \frac{\$\mathsf{AV}(\ddot{\mathrm{E}}x + \ddot{\mathrm{B}}x') + \$E}{1-\alpha}$, and

- $\$\mathsf{BV}(\ddot{\mathrm{E}}c_b) > \frac{\$\mathsf{BV}(\alpha\ddot{\mathrm{E}}x+\ddot{\mathrm{B}}x')+\$E}{1-\alpha}$.

8.4.2 Non-Rational Strategies

Terminologies. We define the 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 $\mathcal{C}$ (the case for Bob-miner coalition 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 Alice is *guaranteed to withdraw her deposit from* Ponyta at some time t, iff either Alice already withdrew her deposit from Ponyta before t, or all the following conditions hold.

- Ponyta is still in the preparation phase.
- A colluding miner has been chosen as the winning miner at time t.
- Alice and the colluding miner activates A1 or A2 in the new block it mines.

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

We say Ponyta is *guaranteed to enter the execution phase* at some time t, iff either Ponyta is already in the execution phase at time t, or all the following conditions hold.

- A colluding miner has been chosen as the winning miner at time t.
- Both parties already posted their deposits transactions.
- pre_a and pre_b are not yet recorded in Ponyta.
- The colluding miner make both deposits into Ponyta in the new block it mines.

We define Ponyta' is *guaranteed to enter the execution phase* at some time t similarly, except that we require pre'_a *is not recorded in* Ponyta' in the third bullet.

Non-rational strategies. The family $\overline{\mathcal{R}}$ contains any strategy that satisfies either of the following:

- The externally incentivized coalition $\mathcal{C}'$ is a Bob-miner coalition (including Bob alone), and moreover, with non-negligible probability, any of the following happens:

  $\mathbf{E}_1$: Before Ponyta is guaranteed to enter the execution phase and before Bob is guaranteed to withdraw his deposit from Ponyta, Bob posts the deposit transaction to Ponyta and anyone in $\mathcal{C}'$ posts a transaction containing pre_b to P2 or C2 or the preparation phase of Ponyta.

  $\mathbf{E}_2$: Ponyta enters the execution phase at Ethereum time t such that $t \leq T$, and moreover, at some Ethereum time $t' \geq t$, anyone in $\mathcal{C}'$ posts a transaction containing pre_b to P2 or C2 before C1 is guaranteed to be activated.

- The externally incentivized coalition $\mathcal{C}'$ is an Alice-miner coalition (including Alice alone), and moreover, with non-negligible probability, any of the following happens.

- $\mathbf{E}_3$: Before C1' is guaranteed to be activated and before Alice is guaranteed to withdraw her deposit from Ponyta', the following happens: Alice has posted her deposit transaction to Ponyta', and someone in $\mathcal{C}'$ has posted 1) a transaction containing pre_a to P1, C2, or C2', and 2) a transaction containing pre'_a to P2' or C2'.
- $\mathbf{E}_4$: Before Ponyta is guaranteed to enter the execution phase and before Alice is guaranteed to withdraw her deposit from Ponyta, Alice posts the deposit transaction to Ponyta and anyone in $\mathcal{C}'$ posts a transaction containing pre_a to P1, C2, or C2' or the preparation phase of Ponyta.
- $\mathbf{E}_5$: Ponyta enters the execution phase at some Ethereum time $t > T$, and moreover, at some time $t' \geq t$, someone in $\mathcal{C}'$ posts a transaction containing pre_a to P1, C2, or C2' before C1 is guaranteed to be activated.
- $\mathbf{E}_6$: Suppose Ponyta enters the execution phase, and Alice does not send pre_a to P1 until Ethereum time T_1. However, Alice sends a transaction containing pre_a to P1 or C2 at Ethereum time T_1 or greater before C1 is guaranteed to be executed.

Lemma 8.4. *Suppose that Bob is honest and at least* $1/\mathsf{poly}(\lambda)$ *fraction of the mining power is honest. Then, except with negligible probability, if* $\mathbf{E}_4$, $\mathbf{E}_5$, *and* $\mathbf{E}_6$ *does not happen, the following hold.*

1. *If* Ponyta *enters the execution phase, then, one of P1 and P2 will be activated, and one of C1 and C2 will be activated in polynomial time.*
2. *If* Ponyta' *enters the execution phase, then, one of P1' and P2' will be activated, and one of C1' and C2' will be activated in polynomial time.*

Further, if Alice is honest and at least $1/\mathsf{poly}(\lambda)$ *fraction of the mining power is honest, then, except with negligible probability, as long as* $\mathbf{E}_1$ *and* $\mathbf{E}_2$ *do not happened, the above also hold.*

Proof. Note that the honest Alice or Bob will post _ to C1 sufficiently long after P1 or P2 is activated, and similarly post _ to C1' sufficiently long after P1' or P2' is activated. It suffices to prove that except with negligible probability, if Ponyta enters the execution phase, then P1 or P2 will be activated in polynomial time; similarly, if Ponyta' enters the execution phase, then P1' or P2' will be activated in polynomial time. Below, we analyze the cases when Alice or Bob is honest one by one. Recall that if Ponyta enters the execution phase, then Ponyta' must have entered the execution phase.

Alice is honest. If Alice goes to the execution phase, she will send pre_a to P1 at Ethereum time T, and she also send _ to P1' as soon as P1 has been activated. By our assumption, in this case, Bob will not post pre_b to P2 until C1 is guaranteed to be activated, and thus P2 cannot be activated. Otherwise, if Alice goes to the abort phase, as soon as Ponyta' enters the execution phase, Alice sends pre'_a to P2', and she also sends _ to P2 as soon as P2' has been activated. In either case, as long as $1/\mathsf{poly}(\lambda)$ fraction of the mining power is honest, except with negligible probability, at least one of (P1 and P2) and one of (P1' and P2') will be activated if the corresponding contract enters the execution phase.

Bob is honest. First, suppose Bob enters the execution phase. If Alice posts pre_a to P1 by Ethereum time T_1, then Bob posts $pre'_b = pre_a$ to P1'. Otherwise, if Alice does not post pre_a to P1 by Ethereum time T_1, Bob will post pre_b to P2, and as soon as P2 is activated, he posts _ to P2' — by our assumption that $\mathbf{E}_4$ and $\mathbf{E}_6$ do not happen, in this case Alice does not post pre_a to P1 before C1 is guaranteed to be activated, and thus P1 cannot be activated.

Next, suppose Bob goes to the abort phase. In this case, if Ponyta does not enter the execution phase, Bob will post $\_$ to P2'. If Ponyta enters the execution phase, Bob will post pre_b to P2, and as soon as P2 is activated, he posts $\_$ to P2'. Also, by our assumption that $\mathbf{E}_4$ and $\mathbf{E}_5$ do not happen, Alice will not post pre_a to P1 before C1 is guaranteed to be activated and thus P1 cannot be activated. Therefore, we conclude that (P1 or P2) and (P1' or P2') will be activated if the corresponding contract enters the execution phase. □

Claim 8.5. *Suppose that the coalition $\mathcal{C}'$ consists of Bob and miners controlling no more than $\alpha \leq 1 - 1/\mathsf{poly}(\lambda)$ fraction of the mining power, and everyone else including Alice is honest. Then, except with negligible probability, if Alice goes to the abort phase, the following claims hold.*

- *Alice's utility is at least zero.*
- *Suppose that $\mathcal{C}'$ never posts pre_b or Bob never posts his deposit transaction to* Ponyta*, and he sends a withdrawal transactions to* Ponyta *or* Ponyta' *as long as he has sent a corresponding deposit transaction. Then, Bob's utility is at least zero.*

Proof. If Alice goes to the abort phase, she would never send pre_a to any activation point. Thus, ignoring the negligible probability that $\mathcal{C}'$ can guess $pre_a = pre'_b$, P1, C2, P1' and C2' are never activated. Notice that because Alice is honest, it is impossible that Ponyta enters the execution phase, but Ponyta' does not. Thus, there are only three possible cases.

- Case 1: Suppose neither Ponyta' and Ponyta enters the execution phase. In the abort phase, for either Ponyta or Ponyta', if Alice has submitted a deposit transaction, then she will submit a corresponding withdrawal transaction. The $1 - \alpha$ fraction of honest miners would include the withdrawal transactions if they are chosen to mine a block. Further, the honest aborting Alice never posts pre_a or pre'_a. Thus, except with negligible probability, Alice can get her deposits back in polynomial time.

  Similarly, suppose that Bob has sent a withdrawal transaction to Ponyta or Ponyta' as long as he has sent a corresponding deposit transaction, and moreover, either $\mathcal{C}'$ has not sent pre_b or Bob never sent any deposit transaction into Ponyta, then except with negligible probability, he can get back all of his deposits in polynomial time.

- Case 2: Suppose Ponyta' enters the execution phase, but Ponyta does not. Due to the same reasoning as Case 1, Alice and Bob can get their deposit back from Ponyta except with negligible probability.

  Then, as we showed above, P1' and C2' are never activated except with negligible probability. Due to Lemma 8.4, P2' and C1' will be activated in polynomial time except with negligible probability. Therefore, Alice's and Bob's utilities are both at least zero except with negligible probability.

- Case 3: Suppose both Ponyta' and Ponyta enter the execution phase. As we showed above, P1, C2, P1' and C2' are never activated except with negligible probability. Due to Lemma 8.4, P2, C1, P2' and C1' will be activated in polynomial time except with negligible probability. Therefore, Alice's and Bob's utilities are both at least zero except with negligible probability.

□

Lemma 8.6 (Blatant non-rationality of $\overline{\mathcal{R}}$ for Bob-miner coalition)**.** *Suppose that the parameter constraints in Section 8.4.1 hold and that the coalition $\mathcal{C}'$ consists of Bob and miners controlling no more than α fraction of the mining power. Given any PPT strategy $S_{\mathcal{C}'} \in \overline{\mathcal{R}}$ for some (externally incentivized) coalition $\mathcal{C}'$, there is a PPT strategy $\widehat{S}_{\mathcal{C}'}$ such that*

$$\mathsf{util}^{\mathcal{C}'}(\widehat{S}_{\mathcal{C}'}, HS_{-\mathcal{C}'}) \geq \mathsf{util}^{\mathcal{C}'}(S_{\mathcal{C}'}, HS_{-\mathcal{C}'}).$$

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

We consider two cases, depending on whether $\mathbf{E}_1$ or $\mathbf{E}_2$ happens first in the original strategy $S_{\mathcal{C}'}$.

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 Ponyta, the $1-\alpha$ fraction of honest miners would send pre_b to the preparation phase of Ponyta and activate B2 for themselves, if they are chosen to mine a block. Once B2 is activated by an honest miner, the coalition $\mathcal{C}'$ gets nothing and simply loses all the deposit from Ponyta. Thus, the utility of $\mathcal{C}'$ is at most $\$\mathsf{BV}(\text{₿}x' - \text{Ɇ}x - \text{Ɇ}c_b) + \$E$, where the maximum is achieved when Ponyta' enters the execution phase, P1' and C1' are activated, and we assume $\$\mathsf{BV}(\text{₿}c'_b) > \$\mathsf{BV}(\text{₿}x')$. On the other hand, if B2 is not activated by an honest miner, the utility of $\mathcal{C}'$ is at most $\$\mathsf{BV}(\text{₿}x') + \$E$, where the maximum is achieved when Ponyta' and Ponyta both go the the execution phase, P2, C1, P1' and C1' are activated, and we assume $\$\mathsf{BV}(\text{Ɇ}c_b) > \$\mathsf{BV}(\text{Ɇ}x)$ and $\$\mathsf{BV}(\text{₿}c'_b) > \$\mathsf{BV}(\text{₿}x')$. When $\mathbf{E}_1$ happens, the probability that B2 is activated by an honest miner is at least $1-\alpha$. Thus, the utility of $\mathcal{C}'$ 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 $\$\mathsf{BV}(\text{Ɇ}c_b) > \frac{\$\mathsf{BV}(\text{₿}x' + \alpha\text{Ɇ}x) + \$E}{1-\alpha}$.

The above shows that conditioned on $\mathbf{E}_1$ happening, the conditional expected utility of the coalition is less than $\$\mathsf{BV}(-\text{Ɇ}x)$. We now show that the modified strategy $\widehat{S}_{\mathcal{C}'}$ performs strictly better. There are two cases.

- At some time t, $\mathcal{C}'$ for the first time wants to send a deposit transaction to Ponyta, and at some time $t' \leq t$, $\mathcal{C}$ sent pre_b to P2 or C2. Then, if $\mathcal{C}'$ plays $\widehat{S}_{\mathcal{C}'}$, Bob would never send the deposit transaction to Ponyta, because it stops sending any messages the moment $\mathbf{E}_1$ is about to happen. In this case, Alice will go to the abort phase. According to Claim 8.5, because $\mathcal{C}'$ sends the withdrawal transactions to Ponyta' and Ponyta, the utility of $\mathcal{C}'$ is at least zero.

- At some time t, $\mathcal{C}'$ for the first time wants to send pre_b to P2 or C2 or the preparation phase of Ponyta; and at $t' \leq t$, $\mathcal{C}'$ sent its deposit transaction to Ponyta. Now, suppose $\mathcal{C}'$ plays $\widehat{S}_{\mathcal{C}'}$ instead, so $\mathcal{C}'$ would never send a transaction containing pre_b to P2 or C2 or the preparation phase of Ponyta. Thus, P2 and C2 are never activated. If Alice goes to the abort phase, because $\mathcal{C}'$ sends the withdrawal transactions to Ponyta' and Ponyta, the utility of $\mathcal{C}'$ is at least zero by Claim 8.5. Otherwise, if Alice goes to the execution phase, Alice will send pre_a to P1, and τ Ethereum time has passed since P1 is activated, she will send _ to C1. Because the $1-\alpha$ fraction of honest miners would include Alice's transactions if they are chosen to

mine a block, P1 and C1 will be activated in polynomial time except with negligible probability. Because Alice goes to the execution phase, she would never post pre'_a to any activation point. Thus, ignoring the negligible probability that $\mathcal{C}'$ can guess pre'_a, C2' would never be activated. Now, Alice sends _ to P1' as soon as P1 has been activated, and τ' Bitcoin time has passed after P1' or P2' is activated, Alice sends _ to C1'. Moreover, since the $1-\alpha$ fraction of honest mining power would include Alice's transactions whenever they are chosen to mine a block, C1' and one of (P1' and P2') will be activated in polynomial time except with negligible probability. If it turns out P1, C1, P1' and C1' are activated, the utility of $\mathcal{C}'$ is $\$\mathsf{BV}(\text{₿}x' - \text{Ð}x) > 0$. Otherwise, if it turns out P1, C1, P2' and C1' are activated, the utility of $\mathcal{C}'$ is $\$\mathsf{BV}(-\text{Ð}x)$.

No matter in which case, once $\mathcal{C}'$ plays $\widehat{S}_{\mathcal{C}'}$, its utility is at least $\$\mathsf{BV}(-\text{Ð}x)$, which is more than the case if $\mathcal{C}'$ plays S_1.

Event $\mathbf{E}_2$ happens first. Consider some strategy $S_2 \in \overline{\mathcal{R}}$ with non-negligible probability that $\mathbf{E}_2$ happens (before $\mathbf{E}_1$ happens). When $\mathbf{E}_2$ happens, it must be that the honest Alice posts pre_a to P1 at Ethereum time T. Further, the $1-\alpha$ fraction of honest miners would activate C2 for themselves if they are chosen to mine a block. Thus, the probability that C2 is successfully activated by someone not in $\mathcal{C}'$ is at least $1-\alpha$. If C2 is activated by an honest miner, then the utility of $\mathcal{C}'$ is at most $\$\mathsf{BV}(\text{₿}x' - \text{Ð}c_b) + \$E$ (from P2, P1' and C1'), assuming $\$\mathsf{BV}(\text{₿}c'_b) > \$\mathsf{BV}(\text{₿}x')$. On the other hand, if C2 is not activated by an honest miner, then the utility of $\mathcal{C}'$ cannot exceed $\$\mathsf{BV}(\text{₿}x') + \$E$ (from P2, C1, P1' and C1') assuming that $\$\mathsf{BV}(\text{Ð}c_b) > \$\mathsf{BV}(\text{Ð}x)$ and $\$\mathsf{BV}(\text{₿}c'_b) > \$\mathsf{BV}(\text{₿}x')$. Therefore, conditioned on $\mathbf{E}_2$ happening and Alice entering the execution phase, the utility of $\mathcal{C}'$ is at most

$$(1-\alpha)(\$\mathsf{BV}(\text{₿}x' - \text{Ð}c_b) + \$E) + \alpha(\$\mathsf{BV}(\text{₿}x') + \$E) < 0,$$

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

The above shows that conditioned on $\mathbf{E}_2$ happening, the conditional expected utility of the coalition is negative. We now show that the modified strategy $\widehat{S}_{\mathcal{C}'}$ performs strictly better. Because Ponyta enters the execution phase at Ethereum time t such that $t \leq T$, it must be that Alice goes to the execution phase. In this case, she never posts pre'_a. Because $\mathcal{C}'$ is PPT, ignoring the negligible probability that $\mathcal{C}'$ can guess pre'_a, P2' and C2' are never activated. If $\mathcal{C}'$ plays $\widehat{S}_{\mathcal{C}'}$, C2 is never activated, since $\mathcal{C}'$ would never send pre_b to C2. In the execution phase, Alice will send pre_a to P1, and she sends _ to P1' as soon as P1 has been activated. When enough time has passed, Alice sends _ to C1 and C1'. Because the $1-\alpha$ fraction of honest miners would include Alice's transactions if they are chosen to mine a block, C1, P1', C1' and one of (P1 and P2) will be activated in polynomial time. If it turns out P1, C1, P1' and C1' are activated, the utility of $\mathcal{C}'$ is $\$\mathsf{BV}(\text{₿}x' - \text{Ð}x) > 0$. Otherwise, if it turns out P2, C1, P1' and C1' are activated, the utility of $\mathcal{C}'$ is $\$\mathsf{BV}(\text{₿}x') > 0$.

Finally, notice that the above analysis holds even if $\mathcal{C}'$ 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. □

To facilitate our proofs, we define an additional event $\mathbf{E}_7$ to be the following:

- $\mathbf{E}_7$: before Ponyta' is guaranteed to enter the execution phase and before Alice is guaranteed to withdraw her deposit from Ponyta', Alice posts her deposit transaction to Ponyta', and someone in $\mathcal{C}'$ posts a transaction containing pre'_a to P2' or C2' or the preparation phase of Ponyta.

Claim 8.7. *Suppose the coalition $\mathcal{C}'$ consists of Alice and miners controlling no more than $\alpha \leq 1-1/\mathsf{poly}(\lambda)$ fraction of the mining power, and everyone else including Bob is honest. Then, except with negligible probability, the following claims hold.*

- *Bob's utility is at least zero.*
- *Suppose events $\mathbf{E}_3$, $\mathbf{E}_4$, $\mathbf{E}_5$, $\mathbf{E}_6$ and $\mathbf{E}_7$ never happen, and Alice sends a withdrawal transactions to* Ponyta *or* Ponyta' *as long as she has sent a corresponding deposit transaction. Then, the utility of $\mathcal{C}'$ is at least $\$\mathsf{AV}(-฿x')$.*

Proof. We have the following facts:

- As long as $\mathbf{E}_7$ does not happen, A2' cannot be activated.
- As long as $\mathbf{E}_4$ does not happen, A2 cannot be activated.
- As long as $\mathbf{E}_3$ does not happen, C2' cannot be activated.
- If Ponyta never enters the execution phase, then the honest Bob will never post pre_b. Therefore, ignoring the negligible probability event that $\mathcal{C}'$ can guess pre_b, B2 cannot be activated.

If Ponyta (or Ponyta' resp.) does not enter the execution phase, since A2 and B2 (or A2', resp.) are never activated, the honest Bob can get all of his deposit back from Ponyta (or Ponyta' resp.), by posting a corresponding withdrawal transaction.

By Lemma 8.4 and since C2' cannot be activated, if Ponyta' enters the execution phase, (P1' or P2') and C1' must be activated — in this case, Bob cannot lose money from Ponyta', and Alice loses at most $฿x'$ from Ponyta'.

Therefore, if Ponyta does not enter the execution phase, then no matter whether Ponyta' enters the execution phase or not, j;w Bob has non-negative utility and Alice loses at most $฿x'$. Below, we show that if Ponyta enters the execution phase (in which case Ponyta' must also enter the execution phase), it must be that Bob has non-negative utility and Alice loses at most $฿x'$, too. We may divide into the following two cases.

Bob goes to the abort phase. In this case, Ponyta must enter the execution phase after Ethereum time T. Since $\mathbf{E}_4$ and $\mathbf{E}_5$ do not happen, we have that if Ponyta enters the execution phase (which must happen after Ethereum time T), P1 and C2 cannot be activated. By Lemma 8.4, in this case, P2 and C1 must be activated, and both Alice and Bob will not lose money from Ponyta.

Bob enters the execution phase. If Alice sent pre_a to P1 before Ethereum time T_1, then the honest Bob would never post pre_b. Thus, ignoring the negligible probability that $\mathcal{C}'$ can guess pre_b, P2 and C2 cannot be activated. Since $\mathbf{E}_3$ does not happen and Bitcoin time $T_1' >$ Ethereum time T_1, no one in $\mathcal{C}'$ would send pre_a' to P2' before C1' is guaranteed to be activated. Therefore, P2' cannot be activated. By Lemma 8.4 and since P2, C2, P2' and C2' cannot be activated, P1, C1, P1' and C1' must be activated. In this case, both Alice and Bob has non-negative utility.

Next, suppose Alice does not send pre_a to P1 before Ethereum time T_1. Because $\mathbf{E}_4$ and $\mathbf{E}_6$ do not happen, Alice would not send pre_a to P1 or C2 before C1 is guaranteed to be activated. Thus, P1 and C2 cannot be activated. By Lemma 8.4 and the fact that C2' also cannot be activated, it must be that P2, C1, C1' and (P1' or P2') is activated. Therefore, Bob has non-negative utility and the utility of $\mathcal{C}'$ is at least $\$\mathsf{AV}(-฿x')$.

□

Claim 8.8. *Suppose that the parameter constraints in Section 8.4.1 hold and that the coalition $\mathcal{C}'$ consists of Alice and miners controlling no more than $\alpha \leq 1 - 1/\mathsf{poly}(\lambda)$ fraction of the mining power. Then, if any event among $\mathbf{E}_3$, $\mathbf{E}_4$, $\mathbf{E}_5$, $\mathbf{E}_6$ and $\mathbf{E}_7$ happens, the utility of $\mathcal{C}'$ must be less than $\$\mathsf{AV}(-\text{₿}x')$.*

Proof. We consider five cases, depending on whether $\mathbf{E}_3$, $\mathbf{E}_4$, $\mathbf{E}_5$, $\mathbf{E}_6$ or $\mathbf{E}_7$ happens first. If two events $\mathbf{E}_i$ and $\mathbf{E}_j$ happen simultaneously where $i < j$, we consider it as $\mathbf{E}_i$ happens first.

Event $\mathbf{E}_3$ happens first. There are two possible cases.

- Case 1: When $\mathbf{E}_3$ happens, PONYTA' already entered the execution phase or is guaranteed to enter the execution phase, and moreover, C1' is not guaranteed to be activated yet. In this case, the $1-\alpha$ fraction of honest miners would activate C2' for themselves if they are chosen to mine a block. Thus, the probability that C2' is successfully activated by someone not in $\mathcal{C}'$ is at least $1-\alpha$. If C2' is activated by an honest miner, then the utility of $\mathcal{C}'$ is at most $\$\mathsf{AV}(\text{Ɇ}x - \text{₿}c'_a) + \$E$ (from P1, C1 and P2') assuming $\$\mathsf{AV}(\text{Ɇ}c_a) > \$\mathsf{AV}(\text{Ɇ}x)$. On the other hand, if C2' is not activated by an honest miner, then the utility of $\mathcal{C}'$ cannot exceed $\$\mathsf{AV}(\text{Ɇ}x) + \$E$ (from P1, C1, P2' and C1') assuming that $\$\mathsf{AV}(\text{Ɇ}c_a) > \$\mathsf{AV}(\text{Ɇ}x)$ and $\$\mathsf{AV}(\text{₿}c'_a) > \$\mathsf{AV}(\text{₿}x')$. Therefore, conditioned on $\mathbf{E}_3$ happens and Alice goes to the execution phase, the utility of $\mathcal{C}'$ is at most

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

  where the inequality arises from the fact that $\$\mathsf{AV}(\text{₿}c'_a) > \frac{\$\mathsf{AV}(\text{Ɇ}x+\text{₿}x')+\$E}{1-\alpha}$. Note that the above analysis holds even if $\mathcal{C}'$ 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.

- Case 2: When $\mathbf{E}_3$ happens, PONYTA' is still in the preparation phase and not guaranteed to enter the execution phase, and moreover, Alice is not guaranteed to withdraw her deposit from PONYTA' yet. Then, the $1-\alpha$ fraction of honest miners would send pre'_a to the preparation phase of PONYTA' and activate A2' for themselves, if they are chosen to mine a block. Once A2' is activated by an honest miner, the coalition $\mathcal{C}'$ gets nothing and simply loses all the deposit from PONYTA'. Moreover, because pre'_a has been recorded, PONYTA' never enters the execution phase, and Bob would never send the deposit transaction to PONYTA. Therefore, the utility of $\mathcal{C}'$ is at most $\$\mathsf{AV}(-\text{₿}x' - \text{₿}c'_a) + \$E$ if the honest miner activates A2'. On the other hand, if A2' is not activated by an honest miner, the utility of $\mathcal{C}'$ is at most $\$\mathsf{AV}(\text{Ɇ}x) + \$E$ (from P1, C1, P2' and C1'), where we assume both PONYTA' and PONYTA enter the execution phase and we assume that $\$\mathsf{AV}(\text{Ɇ}c_a) > \$\mathsf{AV}(\text{Ɇ}x)$ and $\$\mathsf{AV}(\text{₿}c'_a) > \$\mathsf{AV}(\text{₿}x')$. When $\mathbf{E}_3$ happens, the probability that A2' is activated by an honest miner is at least $1-\alpha$. Thus, the utility of $\mathcal{C}'$ is at most

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

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

Event $\mathbf{E}_4$ happens first. When $\mathbf{E}_4$ happens, because Alice is not guaranteed to withdraw her deposit from PONYTA, the $1-\alpha$ fraction of honest miners would send pre_a to the preparation phase of PONYTA and activate A2 for themselves, if they are chosen to mine a block. Once A2 is activated by an honest miner, the coalition $\mathcal{C}'$ gets nothing and simply loses all the deposit from PONYTA. Since $\$\mathsf{AV}(\text{₿}c'_a) > \$\mathsf{AV}(\text{₿}x')$, Alice cannot gain any Bitcoin from PONYTA' no matter

whether it enters the execution phase or not. Therefore, the utility of $\mathcal{C}'$ is at most $\$\mathsf{AV}(-\ddot{\mathrm{E}}c_a)+\$E$ if an honest miner activates A2. On the other hand, if A2 is not activated by an honest miner, the utility of $\mathcal{C}'$ is at most $\$\mathsf{AV}(\ddot{\mathrm{E}}x)+\$E$, and the maximum utility can be achieved by activating P1, C1, P2' and C1' since $\$\mathsf{AV}(\ddot{\mathrm{E}}c_a) > \$\mathsf{AV}(\ddot{\mathrm{E}}x)$ and $\$\mathsf{AV}(\ddot{\mathrm{B}}c'_a) > \$\mathsf{AV}(\ddot{\mathrm{B}}x')$. When $\mathbf{E}_2$ happens, the probability that A2 is activated by an honest miner is at least $1-\alpha$. Thus, the utility of $\mathcal{C}'$ is at most

$$(1-\alpha)(\$\mathsf{AV}(-\ddot{\mathrm{E}}c_a)+\$E)+\alpha(\$\mathsf{AV}(\ddot{\mathrm{E}}x)+\$E) < \$\mathsf{AV}(-\ddot{\mathrm{B}}x'),$$

where the inequality arises from the fact that $\$\mathsf{AV}(\ddot{\mathrm{E}}c_a) > \frac{\$\mathsf{AV}(\alpha\ddot{\mathrm{E}}x+\ddot{\mathrm{B}}x')+\$E}{1-\alpha}$.

Event $\mathbf{E}_5$ happens first. When $\mathbf{E}_5$ happens, because PONYTA enters the execution phase after Ethereum time T, Bob goes to the abort phase, and he will post pre_b as soon as PONYTA enters the execution phase. Thus, when $\mathbf{E}_5$ happens, pre_a and pre_b are publicly known. Thus, the $1-\alpha$ fraction of honest miners would activate C2 for themselves if they are chosen to mine a block. Thus, the probability that C2 is successfully activated by someone not in $\mathcal{C}'$ is at least $1-\alpha$. If C2 is activated by an honest miner, then the utility of $\mathcal{C}'$ is at most $\$\mathsf{AV}(\ddot{\mathrm{E}}x-\ddot{\mathrm{E}}c_a)+\$E$ (from P1, P2' and C1') assuming $\$\mathsf{AV}(\ddot{\mathrm{B}}c'_a) > \$\mathsf{AV}(\ddot{\mathrm{B}}x')$. On the other hand, if C2 is not activated by an honest miner, then the utility of $\mathcal{C}'$ cannot exceed $\$\mathsf{AV}(\ddot{\mathrm{E}}x)+\$E$ (from P1, C1, P2' and C1') assuming that $\$\mathsf{AV}(\ddot{\mathrm{E}}c_a) > \$\mathsf{AV}(\ddot{\mathrm{E}}x)$ and $\$\mathsf{AV}(\ddot{\mathrm{B}}c'_a) > \$\mathsf{AV}(\ddot{\mathrm{B}}x')$. Therefore, conditioned on $\mathbf{E}_5$ happens, the utility of $\mathcal{C}'$ is at most

$$(1-\alpha)(\$\mathsf{AV}(\ddot{\mathrm{E}}x-\ddot{\mathrm{E}}c_a)+\$E)+\alpha(\$\mathsf{AV}(\ddot{\mathrm{E}}x)+\$E) < \$\mathsf{AV}(-\ddot{\mathrm{B}}x'),$$

where the inequality arises from the fact that $\$\mathsf{AV}(\ddot{\mathrm{E}}c_a) > \frac{\$\mathsf{AV}(\ddot{\mathrm{E}}x+\ddot{\mathrm{B}}x')+\$E}{1-\alpha}$.

Event $\mathbf{E}_6$ happens first. Because Alice does not send pre_a to P1 before Ethereum time T_1, Bob will send pre_b to P2 at Ethereum time T_1. When $\mathbf{E}_6$ happens, pre_a and pre_b are publicly known. Then, using the same argument as in the previous case, we have conditioned on $\mathbf{E}_6$ happens, the utility of $\mathcal{C}'$ is at most

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

Event $\mathbf{E}_7$ happens first. When $\mathbf{E}_7$ happens, the $1-\alpha$ fraction of honest miners would send pre'_a to the preparation phase of PONYTA' and activate A2' for themselves, if they are chosen to mine a block. Once A2' is activated by an honest miner, the coalition $\mathcal{C}'$ gets nothing and simply loses all the deposit from PONYTA'. Moreover, because pre'_a has been recorded, PONYTA' never enters the execution phase, and Bob would never send the deposit transaction to PONYTA. Therefore, the utility of $\mathcal{C}'$ is at most $\$\mathsf{AV}(-\ddot{\mathrm{B}}x'-\ddot{\mathrm{B}}c'_a)+\$E$ if the honest miner activates A2'. On the other hand, if A2' is not activated by an honest miner, the utility of $\mathcal{C}'$ is at most $\$\mathsf{AV}(\ddot{\mathrm{E}}x)+\$E$ (from P1, C1, P2' and C1'), where we assume both PONYTA' and PONYTA enter the execution phase and we assume that $\$\mathsf{AV}(\ddot{\mathrm{E}}c_a) > \$\mathsf{AV}(\ddot{\mathrm{E}}x)$ and $\$\mathsf{AV}(\ddot{\mathrm{B}}c'_a) > \$\mathsf{AV}(\ddot{\mathrm{B}}x')$. When $\mathbf{E}_1$ happens, the probability that A2' is activated by an honest miner is at least $1-\alpha$. Thus, the utility of $\mathcal{C}'$ is at most

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

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

□

Lemma 8.9 (Blatant non-rationality of $\overline{\mathcal{R}}$ for Alice-miner coalition)**.** *Suppose that the parameter constraints in Section 8.4.1 hold and that the coalition $\mathcal{C}'$ consists of Alice and miners controlling no*

more than α fraction of the mining power. Given any PPT strategy $S_{\mathcal{C}'} \in \overline{\mathcal{R}}$ for some (externally incentivized) coalition $\mathcal{C}'$, there is a PPT strategy $\widehat{S}_{\mathcal{C}'}$ such that

$$\mathsf{util}^{\mathcal{C}'}(\widehat{S}_{\mathcal{C}'}, HS_{-\mathcal{C}'}) \geq \mathsf{util}^{\mathcal{C}'}(S_{\mathcal{C}'}, HS_{-\mathcal{C}'})$$

Proof. Recall that the non-rational strategy space $\overline{\mathcal{R}}$ includes PPT strategies in which $\mathbf{E}_3$, $\mathbf{E}_4$, $\mathbf{E}_5$, or $\mathbf{E}_6$ happens with non-negligible probability. To facilitate our proof, we expand the non-rational strategy $\overline{\mathcal{R}}$ to additionally include PPT strategies where $\mathbf{E}_7$ happens with non-negligible probability — let $\overline{\mathcal{R}}'$ be the expanded strategy space. It suffices to prove that given any PPT strategy in expanded strategy space $\overline{\mathcal{R}}'$, we can construct a new PPT strategy which does strictly better in expectation. Let S be a PPT strategy in $\overline{\mathcal{R}}'$. We now consider a modified strategy denoted $\widehat{S}$: whenever by the original strategy, any of $\mathbf{E}_3$, $\mathbf{E}_4$, $\mathbf{E}_5$, $\mathbf{E}_6$ and $\mathbf{E}_7$ is about to happen, $\mathcal{C}'$ simply sends the withdrawal transactions to Ponyta' and Ponyta, and stops sending any messages to the contract from that moment on, including the message that was going to trigger the events $\mathbf{E}_3$, $\mathbf{E}_4$, $\mathbf{E}_5$, $\mathbf{E}_6$ or $\mathbf{E}_7$.

Conditioned on the execution traces in which $\mathbf{E}_3$, $\mathbf{E}_4$, $\mathbf{E}_5$, $\mathbf{E}_6$ and $\mathbf{E}_7$ do not happen in the original strategy S, the new strategy $\widehat{S}$ results in the same utility for $\mathcal{C}'$ as the original strategy S. Thus, it suffices to show that conditioned on the execution traces in which $\mathbf{E}_3$, $\mathbf{E}_4$, $\mathbf{E}_5$, $\mathbf{E}_6$ and $\mathbf{E}_7$ would have happened with the original strategy S (but does not happen now with $\widehat{S}$), the new strategy $\widehat{S}$ results in strictly better utility for $\mathcal{C}'$ than S.

Claim 8.8 shows that conditioned on any of $\mathbf{E}_3$, $\mathbf{E}_4$, $\mathbf{E}_5$, $\mathbf{E}_6$ and $\mathbf{E}_7$ happening, the expected utility of the coalition adopting the original strategy S is less than $\$\mathsf{AV}(-₿x')$. We argue that the new strategy $\widehat{S}$ gives the coalition $\mathcal{C}'$ a higher expected utility than S. Notice that when $\mathcal{C}'$ plays $\widehat{S}$, none of $\mathbf{E}_3$, $\mathbf{E}_4$, $\mathbf{E}_5$, $\mathbf{E}_6$ and $\mathbf{E}_7$ will happen. Moreover, notice that the honest Bob either goes to the abort phase or the execution phase. Thus, by Claim 8.7, the utility of $\mathcal{C}'$ is at least $\$\mathsf{AV}(-₿x')$. □

8.4.3 Proofs of Safe Participation Against External Incentives

In the remainder of this section, we prove that the same coin swap protocol, with the parameters specified in Section 8.4.1, satisfies safe participation against external incentives.

Lemma 8.10 (Against externally incentivized Bob-miner coalition)**.** *Suppose that $H(\cdot)$ is a one-way function. Let $\mathcal{C}$ be a coalition consisting of Alice and possibly any subset of the miners, and let $\mathcal{C}'$ be a disjoint coalition consisting of Bob and at most α fraction of the mining power. Suppose that $\mathcal{C}$ does not have external incentives but $\mathcal{C}'$ may have up to $\$E$ amount of external incentives. Let $S_{\mathcal{C}'}$ be an arbitrary PPT strategy of $\mathcal{C}'$ that is not in $\overline{\mathcal{R}}$. Then, there exists a negligible function $\mathsf{negl}(\cdot)$ such that*

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

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

Proof. Depending on how the coalition deviates from the protocol, Alice must make one of the following decisions.

1. Alice goes to the abort phase at Bitcoin time T'.
2. Alice goes to the abort phase at Ethereum time T.
3. Alice goes to the execution phase.

In the following, we analyze each case.

Alice goes to the abort phase at Bitcoin time T'. In this case, Alice already sent deposit transaction to Ponyta'. Further, Alice never sends deposit transaction to Ponyta, so Ponyta never goes to the execution phase.

If Alice withdraws her deposit from Ponyta' before Ponyta' goes to the execution phase, Alice's utility is 0. On the other hand, if Ponyta' goes to the execution phase, Alice can get her deposit back only through the P-type and C-type activation points. Because Alice never posts any transaction containing pre_a, and $\mathcal{C}'$ is PPT, P1' and C2' are never invoked except with negligible probability. Because any miner in $\mathcal{C} \cup \mathcal{D}$ and honest Alice always follow the steps in the abort phase, P2' and C1' would be activated in polynomial time. Thus, the utility of $\mathcal{C}$ is non-negative except with negligible probability.

Alice goes to the abort phase at Ethereum time T. In this case, Ponyta' goes to the execution phase, and Alice sends the deposit transaction to Ponyta. Notice that since Alice goes to the abort phase, she never posts any transaction containing pre_a. Because $\mathcal{C}'$ is PPT, P1' and C2' are never activated except with negligible probability. In the abort phase, Alice posts pre'_a to P2' and sufficiently long after P2' is activated, she posts $\_$ to C1'. Since the honest mining power is at least $1 - \alpha \geq 1/\mathsf{poly}(\lambda)$, except with negligible probability, P2' and C1' would be activated in polynomial time.

Suppose Alice withdraws her deposit from Ponyta before Ponyta goes to the execution phase. Then, the utility of $\mathcal{C}$ is non-negative up to a negligible probability. In the following of the proof, we focus on the case that Ponyta goes to the execution phase before Alice withdraws her deposit. Because Alice never posts any transaction containing pre_a and $\mathcal{C}'$ is PPT, P1 and C2 are never activated except with negligible probability. Recall also that as soon as P2' is activated, Alice posts $\_$ to P2, and sufficiently long after P2 is activated, Alice posts $\_$ to C1. Since $1 - \alpha \geq 1/\mathsf{poly}(\lambda)$, P2 and C1 will be activated in polynomial time except with negligible probability. As long as P2, C1, P2' and C1' are activated, the utility of $\mathcal{C}$ is non-negative.

Alice goes to the execution phase. In this case, both Ponyta and Ponyta' enter the execution phase. Since the coalition $\mathcal{C}'$ does not adopt any strategy in $\overline{\mathcal{R}}$, $\mathcal{C}'$ will not post pre_b to P2 or C2, unless C1 is guaranteed to be activated (see events $\mathbf{E}_1$ and $\mathbf{E}_2$). Since Alice is honest and posts pre_a to P1 and sufficiently long after P1 is activated she posts $\_$ to C1, recall also that there is at least $1/\mathsf{poly}(\lambda)$ fraction of honest mining power, P1 and C1 will be activated except with negligible probability.

Note that Alice will not post pre'_a to P2' if she enters the execution phase. Since $\mathcal{C}'$ is PPT and $H(\cdot)$ is a one-way function, it holds that P2' or C2' will never be activated except with negligible probability. Further, Alice posts $\_$ to P1' after P1 is activated, and sufficiently long after P1' is activated, Alice posts to C1'. Since there is at least $1/\mathsf{poly}(\lambda)$ fraction of honest mining power, P1' and C1' will be activated except with negligible probability.

In summary, we have shown that P1, C1, P1', C1' will be activated except with negligible probability. Consequently, $\mathcal{C}$'s utility is at least $\$\mathsf{AV}(\text{Ð}x - \text{Ƀ}x') > 0$.

□

Lemma 8.11 (Against externally incentivized Alice-miner coalition)**.** *Suppose that $H(\cdot)$ is a one-way function. Let $\mathcal{C}$ be a coalition consisting of Bob and possibly any subset of the miners, and let $\mathcal{C}'$ be a disjoint coalition consisting of Alice and at most α fraction of the mining power. Suppose that $\mathcal{C}$ does not have external incentives but $\mathcal{C}'$ may have up to $\$E$ amount of external incentives. Let $S_{\mathcal{C}'}$ be an arbitrary PPT strategy of $\mathcal{C}'$ that is not in $\overline{\mathcal{R}}$. Then, there exists a negligible function*

$\mathsf{negl}(\cdot)$ *such that*

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

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

Proof. Depending on how the coalition deviates from the protocol, Bob must make one of the following decisions.

1. Bob goes to the abort phase at Bitcoin time T'.
2. Bob goes to the abort phase at Ethereum time T.
3. Bob goes to the execution phase.

In the following, we analyze each case.

Bob goes to the abort phase at Bitcoin time T'. In this case, Bob already sent the deposit transaction to Ponyta'. Further, Bob never sends the deposit transaction to Ponyta, so Ponyta will never enter the execution phase.

If Bob withdraws from Ponyta' before Ponyta' enters the execution phase, the utility of $\mathcal{C}$ is 0. On the other hand, if Ponyta' goes to the execution phase, Bob can get his deposit back only through the P-type or C-type activation points. Since $\mathcal{C}'$ does not adopt a strategy in $\overline{\mathcal{R}}$, except with negligible probability, $\mathcal{C}'$ does not post a transaction containing pre_a and pre'_a at the same time, unless C1' is guaranteed to be activated (event $\mathbf{E}_3$). Thus, C2' would never be activated. Since Bob did not send a deposit transaction to Ponyta, he will post _ to P2' as part of its abort procedure, and he will post _ to C1' after either P1' or P2' is activated. Since $1 - \alpha \geq 1/\mathsf{poly}(\lambda)$, either (P1' and C1') or (P2' and C1') will be activated in polynomial time except with negligible probability. In either case, the utility of $\mathcal{C}$ is non-negative.

Bob goes to the abort phase at Ethereum time T. In this case, Ponyta' enters the execution phase, and Bob also already send the deposit transaction to Ponyta. Suppose Bob withdraws from Ponyta before Ponyta enters the execution phase. Using a similar argument as in the previous case, we can argue that either (P1' and C1') or (P2' and C1') are activated except with negligible probability, and in either case, the utility of $\mathcal{C}$ is non-negative.

We now focus on the case where Ponyta enters the execution phase, so Bob can get his deposit back only through the P-type and the C-type activation points. Because Bob goes to the abort phase, Ponyta must enter the execution phase after Ethereum time T. Since $\mathcal{C}'$ does not adopt a strategy in $\overline{\mathcal{R}}$, except with negligible probability, $\mathcal{C}'$ will never post any transaction containing pre_a to C2 either before or after Ponyta enters the execution phase, and before C1 is guaranteed to be activated (see events $\mathbf{E}_4$ and $\mathbf{E}_5$). Moreover, because Ponyta' enters the execution phase, Alice must post the deposit transaction to Ponyta'. Since $\mathcal{C}'$ does not adopt a strategy in $\overline{\mathcal{R}}$, except with negligible probability, $\mathcal{C}'$ will never post a transaction containing both pre_a and pre'_a to C2', unless C1' is guaranteed to be activated (see event $\mathbf{E}_3$). Combining the arguments above, we conclude C2 and C2' will never be activated.

In the abort phase, Bob posts pre_b to P2 as soon as Ponyta enters the execution phase. Recall also that $1 - \alpha \geq 1/\mathsf{poly}(\lambda)$, therefore, P2 and C1 will be activated in polynomial time except with negligible probability. Once P2 has been activated, Bob posts _ to P2' as part of its abort procedure, and sufficiently long after P1' or P2' is activated, he posts _ to C1'. Therefore, except with negligible probability, either (P1' and C1') or (P2' and C1') is activated. In either case, the utility of $\mathcal{C}$ is non-negative.

Bob goes to the execution phase. In this case, both PONYTA and PONYTA' enter the execution phase. Notice that if Bob goes to the execution phase, PONYTA must enter the execution phase before Ethereum time T. There are two subcases.

1. **Alice posts pre_a to P1 before Ethereum time T_1.** Since $\mathcal{C}$ follows the honest strategy, it would not post any transaction containing pre_b. Since $\mathcal{C}'$ is PPT, P2 and C2 are never activated except with negligible probability. Further, since $1-\alpha \geq 1/\mathsf{poly}(\lambda)$ and the honest Bob will always post $\_$ to C1 sufficiently long after P1 is activated, except with negligible probability, both P1 and C1 are activated in polynomially bounded time.

   In this case, Alice must have posted the deposit transaction to PONYTA'. Since $\mathcal{C}'$ does not adopt any strategy in $\overline{\mathcal{R}}$, the following holds except with negligible probability:

   (a) $\mathcal{C}'$ would not send (pre_a, pre'_a) to C2' before C1' is guaranteed to be activated (see event $\mathbf{E}_3$). Hence, C2' would never be activated.

   (b) Since C1' cannot be guaranteed to be activated at Ethereum time T_1 (which is before Bitcoin time T'_1), and Alice posts pre_a to P1 at Ethereum time T_1, $\mathcal{C}'$ cannot post pre'_a before C1' is guaranteed to be activated (see event $\mathbf{E}_3$). Note also that the honest Bob will not post pre_b to P2 or $\_$ to P2'. Thus, P2' will never be activated.

   On the other hand, the honest Bob will post $pre'_b = pre_a$ to P1' and will post $\_$ to C1' sufficiently long after P1' is activated. Therefore, except with negligible probability, P1' and C1' will be activated. In summary, P1, C1, P1', C1' are activated except with negligible probability, and this means that the utility of $\mathcal{C}$ is non-negative.

2. **Alice does not post pre_a to P1 before Ethereum time T_1.** In this case, Bob sends pre_b to P2 at Ethereum time T_1, he posts $\_$ to C1 sufficiently long after P2 is activated, and moreover, Bob would never send any transaction to P1'. Since $\mathcal{C}'$ adopts a PPT strategy that is not in $\overline{\mathcal{R}}$, except with negligible probability, Alice will not post pre_a to P1 before C1 is guaranteed to be activated (see event $\mathbf{E}_6$). Therefore, except with negligible probability, P2 and C1 are activated in polynomial time.

   Once P2 is activated, the honest Bob sends $\_$ to P2', and sufficiently long after either P1' or P2' is activated, Bob sends $\_$ to C1'. Since $\mathcal{C}'$ adopts a strategy that is not in $\overline{\mathcal{R}}$, except with negligible probability, $\mathcal{C}'$ would not send (pre_a, pre'_a) to C2' before C1' is guaranteed to be activated (see event $\mathbf{E}_3$). Thus, C2' would never be activated. Therefore, except with negligible probability, it must be that either (P1' and C1') or (P2' and C1') are activated. In either case, the utility of $\mathcal{C}$ is non-negative.

□

Theorem 8.12 (Safe participation against external incentives)**.** *Suppose that $H(\cdot)$ is a one-way function and suppose the parameters $\ddot{\mathbb{E}}c_a, \ddot{\mathbb{B}}c'_a, \ddot{\mathbb{E}}c_b, \ddot{\mathbb{B}}c'_b, \alpha$ satisfy the relations specified in Section 8.4.1. Then, the atomic swap protocol satisfies α-safe-participation against external incentives.*

Proof. We can divide into the following cases:

Case 1: Alice $\in \mathcal{C}$ and Bob $\in \mathcal{C}'$: covered by Lemma 8.10.

Case 2: Bob $\in \mathcal{C}$ and Alice $\in \mathcal{C}'$: covered by Lemma 8.11.

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 at Bitcoin time T' and at Ethereum time T, i.e., they make this decision using on-chain state when the blockchains have reached lengths T' and T, respectively. Therefore, both Alice and Bob go to the execution phase or both of them go to the abort phase.

Suppose both Alice and Bob go to the execution phase. Alice would send pre_a to P1 at Ethereum time T, and Bob would send $pre'_b = pre_a$ to P1' at Ethereum time T_1. When τ Ethereum time has passed after P1 is activated, both parties send _ to C1; similarly, when τ' Bitcoin time has passed after P1' is activated, both parties send _ to C1'. Besides, Alice and Bob would never post any transaction containing pre_b or pre'_a. Because $\mathcal{C}'$ is PPT, except with negligible probability event that someone in $\mathcal{C}'$ can compute pre_b or pre'_a, P2, C2, P2' and C2' cannot be activated. The $1-\alpha$ fraction of honest miner will include Alice's and Bob's transactions, so except the negligible probability, P1, C1, P1' and C1' will be activated in polynomial time. Thus, the honest $\mathcal{C}$ obtains non-negative utility.

Suppose both Alice and Bob go to the abort phase. Claim 8.5 guarantees that the honest $\mathcal{C}$ obtains non-negative utility.

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

8.5 Proof of Dropout Resilience

Theorem 8.13 (Dropout resilience of atomic swap)**.** *Our atomic swap protocol 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. If Alice drops out, Bob's utility must be at least zero guaranteed by Claim 8.7.

In the following, we focus on the case where Bob drops out. If Alice goes to the abort phase, Alice's utility is at least zero by Claim 8.5. Henceforth, we assume Alice goes to the execution phase. In this case, Alice sends pre_a to P1 at Ethereum time T, and she sends _ to P1' as soon as P1 has been activated. When τ Ethereum time has passed after P1 is activated, Alice sends _ to C1; similarly, when τ' Bitcoin time has passed after P1' is activated, Alice sends _ to C1. In the execution phase, Alice never sends pre'_a to any activation point. Because Alice sends pre_a to P1 at Ethereum time T, Bob never sends pre_b to any activation point. Ignoring the negligible probability that pre'_a or pre_b is guessed, P2, C2, P2' and C2' are never activated. Because the $1/\mathsf{poly}(\lambda)$ fraction of the miner would include Alice's transactions, P1, C1, P1' and C1' will be activated in polynomial time. Thus, except the negligible probability, Alice's utility is $\$\mathsf{AV}(\text{Ð}x - \text{Ð}x') > 0$.

□

9 Bitcoin Instantiation

In this section we describe how Ponyta contract from Section 5 that is CSP-Fair absent external incentives, can be instantiated in Bitcoin using its limited script support. We describe our instantiation for Ponyta with safe participation in Appendix B and for atomic swaps in Appendix C. We start by introducing notation for payments in Bitcoin.

9.1 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 Ponyta 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 Ponyta 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. $\Phi(x) = 1$.

9.2 Bitcoin Instantiation of Ponyta With CSP-Fairness Absent External Incentives

We first describe the tools we require for our Ponyta instantiation with CSP-Fairness absent external incentives from Section 5.

[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 1: PONYTA's transactions in Bitcoin absent external incentives.

	Description
tx_{stp}	$tx\left(\begin{matrix}[(Adr_0^A, \Phi_0^A, \$c_a), (Adr_0^B, \Phi_0^B, \$v + \$c_b)], \\ [(Adr_{\mathsf{pay}}, \Phi_{\mathsf{pay}}, \$v), (Adr_{\mathsf{col}}, \Phi_{\mathsf{col}}, \$c_a + \$c_b)]\end{matrix}\right)$
tx_{P1}	$tx\left(\begin{matrix}[(Adr_{\mathsf{pay}}, \Phi_{\mathsf{pay}}, \$v)], \\ [(Adr_1^A, \Phi_1^A, \$v - \$\epsilon), (Adr_{\mathrm{P1}}^{\mathsf{ax}}, \Phi_{\mathrm{P1}}^{\mathsf{ax}}, \$\epsilon)]\end{matrix}\right)$
tx_{P2}	$tx\left(\begin{matrix}[(Adr_{\mathsf{pay}}, \Phi_{\mathsf{pay}}, \$v)], \\ [(Adr_1^B, \Phi_1^B, \$v - \$\epsilon), (Adr_{\mathrm{P2}}^{\mathsf{ax}}, \Phi_{\mathrm{P2}}^{\mathsf{ax}}, \$\epsilon)]\end{matrix}\right)$
tx_{C1}^1	$tx\left(\begin{matrix}[(Adr_{\mathsf{col}}, \Phi_{\mathsf{col}}, \$c_a + \$c_b), (Adr_{\mathrm{P1}}^{\mathsf{ax}}, \Phi_{\mathrm{P1}}^{\mathsf{ax}}, \$\epsilon)], \\ [(Adr_2^A, \Phi_2^A, \$c_a + \$\epsilon), (Adr_2^B, \Phi_2^B, \$c_b)]\end{matrix}\right)$
tx_{C1}^2	$tx\left(\begin{matrix}[(Adr_{\mathsf{col}}, \Phi_{\mathsf{col}}, \$c_a + \$c_b), (Adr_{\mathrm{P2}}^{\mathsf{ax}}, \Phi_{\mathrm{P2}}^{\mathsf{ax}}, \$\epsilon)], \\ [(Adr_3^A, \Phi_3^A, \$c_a), (Adr_3^B, \Phi_3^B, \$c_b + \$\epsilon)]\end{matrix}\right)$
tx_{C2}	$tx\left(\begin{matrix}[(Adr_{\mathsf{col}}, \Phi_{\mathsf{col}}, \$c_a + \$c_b)], \\ [(Adr_{\mathsf{burn}}, \Phi_{\mathsf{burn}}, \$c_a + \$c_b - \$v)]\end{matrix}\right)$

9.2.1 Addresses, Scripts, and Transactions

We now describe all the transactions, addresses, and scripts needed in our instantiation.

Setup. In PONYTA's Bitcoin instantiation, the players place their deposits into two addresses, the payment address Adr_{pay} and the collateral address Adr_{col}. During the preparation phase, Alice and Bob prepare the setup transaction tx_{stp} that moves their coins into the payment address Adr_{pay} and the collateral address Adr_{col}.

Payment redeem. The payment redeem transactions tx_{P1} and tx_{P2} (see Figure 2) redeem from the payment address Adr_{pay}, with $\$v - \$\epsilon$ coins paid to Alice's or Bob's address, respectively, and $\$\epsilon$ coins paid to an auxiliary address which will later be needed for implementing a conditional timelock redeem. The script Φ_{pay} associated with the address Adr_{pay} provides two ways to redeem (see Figure 1), corresponding to the activation points P1 and P2 in our earlier meta-contract. The two redeem transactions tx_{P1} and tx_{P2} redeem from each of these branches, respectively.

Collateral redeem. The collateral redeem transactions denoted tx_{C1}^1, tx_{C1}^2 and tx_{C2} (see Table 1) redeem coins from the collateral address Adr_{col}. The script Φ_{col} associated with the collateral address Adr_{col} provides two ways of redeeming, corresponding to C1 and C2 in our earlier meta-contract, respectively. The transaction tx_{C2} redeems from C2 branch, paying $\$c_a + \$c_b - \$v$ coins to a burn-address Adr_{burn}, and the remaining to the miner who mines the block.

The transactions tx_{C1}^1 and tx_{C1}^2 redeem from the C1 branch, depending on which payment transaction tx_{P1} or tx_{P2} respectively, was activated earlier.

To implement the timeout condition associated with the C1 branch, tx_{C1}^1 and tx_{C1}^2 also each refer to an auxiliary input address, $Adr_{\mathrm{P1}}^{\mathsf{ax}}$ and $Adr_{\mathrm{P2}}^{\mathsf{ax}}$, respectively. These two auxiliary addresses are outputs of payment transactions tx_{P1} and tx_{P2} respectively. Addresses $Adr_{\mathrm{P1}}^{\mathsf{ax}}$ and $Adr_{\mathrm{P2}}^{\mathsf{ax}}$ have scripts $\Phi_{\mathrm{P1}}^{\mathsf{ax}}$ and $\Phi_{\mathrm{P2}}^{\mathsf{ax}}$, respectively that enforce the timeouts.

We provide the list of all transactions in Table 1, the scripts associated with all addresses in Figure 1, and the relationship between the transactions, addresses, and scripts is depicted in Figure 2.

To make sure that Alice and Bob cannot unilaterally spend from the payment address Adr_{pay},

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

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

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

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

C1: **if** $(\mathsf{Vf}(\mathsf{pk}_a, tx, \sigma_a) = 1) \wedge (\mathsf{Vf}(\mathsf{pk}_b, tx, \sigma_b) = 1)$
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 $h_a, h_b, \mathsf{pk}_a, \mathsf{pk}_b, \mathsf{pk}'_a, \mathsf{pk}'_b$ are hardwired

$\Phi^{\mathsf{ax}}_{\mathrm{P1}}(tx, \sigma_a, \sigma_b)$

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

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

$\Phi^{\mathsf{ax}}_{\mathrm{P2}}(tx, \sigma_a, \sigma_b)$

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

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

Figure 1: The description of scripts $\Phi_{\mathsf{pay}}, \Phi_{\mathsf{col}}, \Phi^{\mathsf{ax}}_{\mathrm{P1}}$ and $\Phi^{\mathsf{ax}}_{\mathrm{P2}}$. Here tx is the transaction spending from the script. Keys pk_a and pk'_a belong to Alice, pk_b and pk'_b belong to Bob. In Φ_{pay} (and Φ_{col}) either activation point P1 (C1) or activation point P2 (C2) must be satisfied for the script to return 1.

the collateral address Adr_{col}, and the auxiliary addresses $Adr^{\mathsf{ax}}_{\mathrm{P1}}$ and $Adr^{\mathsf{ax}}_{\mathrm{P2}}$, their associated scripts require *signatures from both Alice and Bob* to spend from these addresses. Note also that the transactions tx_{P1} and tx_{P2} needed to spend from P1 and P2 must be signed with different public keys of Alice and Bob, i.e., $\mathsf{pk}_a, \mathsf{pk}_b$, and pk'_a, pk'_b, respectively. This ensures that Bob cannot invoke P1 with tx_{P2} which specifies Bob, rather than Alice, as the recipient address. In summary, assuming security of the signature scheme, no other transaction is able to spend from the addresses Adr_{pay}, Adr_{col}, and the auxiliary addresses $Adr^{\mathsf{ax}}_{\mathrm{P1}}$ and $Adr^{\mathsf{ax}}_{\mathrm{P2}}$, besides those that they are connected to through a solid line in Figure 2.

9.2.2 Conditional Timelock Redeem and Conditional Burning

We highlight the ideas needed to realize conditional timelock redeem and conditional burning.

Conditional Timelock Redeem. For enforcing a timelock on the collateral coins, general smart contracts let the contracts check if T_2 time has passed since the payment coins were redeemed. In the absence of general smart contracts, we have to rely on the one-time spendability of Bitcoin addresses. To do this, we rely on the auxiliary address $Adr^{\mathsf{ax}}_{\mathrm{P1}}$ ($Adr^{\mathsf{ax}}_{\mathrm{P2}}$) to ensure that the coins in

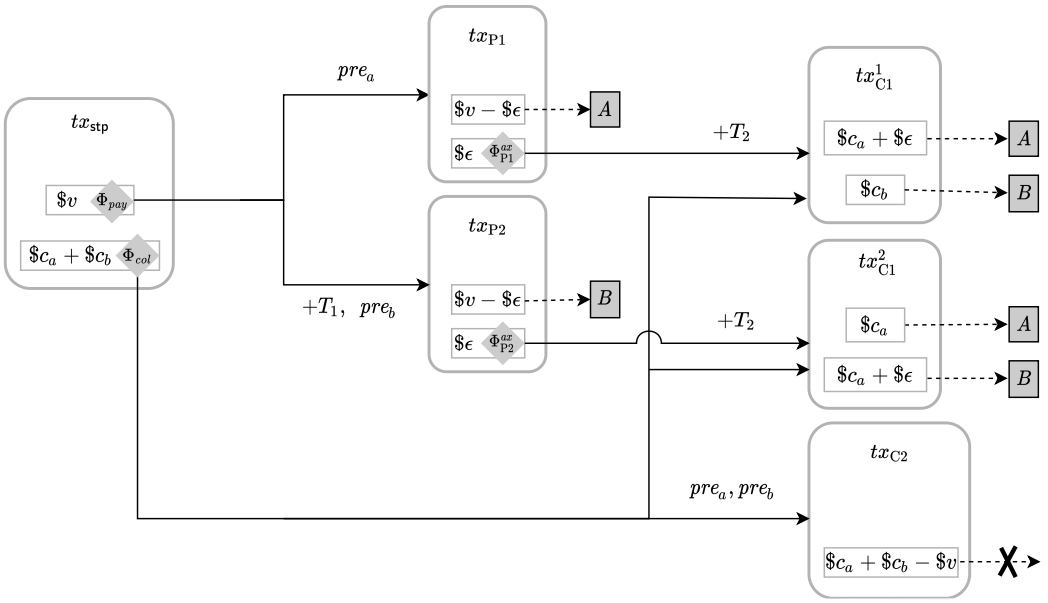

Figure 2: The transaction flow of PONYTA in Bitcoin absent external incentives. Rounded boxes denote transactions, rectangles within are outputs of the transaction. Incoming arrows denote transaction inputs, outgoing arrows denote how an output can be spent by a transaction at the end of the arrow. Solid 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.

the collateral address Adr_{col} are timelocked for time T_2 after the coins from the payment address Adr_{pay} are redeemed by Alice (Bob). Note that $Adr^{\mathsf{ax}}_{\text{P1}}$ ($Adr^{\mathsf{ax}}_{\text{P2}}$) is an address created when the payment address is redeemed via tx_{P1} by Alice (tx_{P2} by Bob). $Adr^{\mathsf{ax}}_{\text{P1}}$ is set such that its associated script only allows the coins from the address to be redeemed, if (1) both Alice and Bob sign and (2) time T_2 has passed since $Adr^{\mathsf{ax}}_{\text{P1}}$ was created on the blockchain (similar for $Adr^{\mathsf{ax}}_{\text{P2}}$). By allowing only transactions tx^1_{C1} and tx^2_{C1} to spend from branch C1 of Adr_{col} and requesting that they *also* spend from $Adr^{\mathsf{ax}}_{\text{P1}}$ or $Adr^{\mathsf{ax}}_{\text{P2}}$, we design a mechanism such that for branch C1 the collateral coins in Adr_{col} can be redeemed only if the coins from the auxiliary address $Adr^{\mathsf{ax}}_{\text{P1}}$ or $Adr^{\mathsf{ax}}_{\text{P2}}$ are redeemed *simultaneously*. This effectively enforces a timelock of T_2 (after tx_{P1} or tx_{P2} is published on the blockchain) on the redeeming of the collateral coins by Alice and Bob. We refer to this technique of enforcing a conditional timelock on the collateral coins through simultaneous spending of a timelocked auxiliary address as *conditional timelock redeem*. The addresses $Adr^{\mathsf{ax}}_{\text{P1}}$ and $Adr^{\mathsf{ax}}_{\text{P2}}$ only hold a very small amount of value $\$\epsilon$, their only role is to enable the above mechanism. The auxiliary address scripts $\Phi^{\mathsf{ax}}_{\text{P1}}$ and $\Phi^{\mathsf{ax}}_{\text{P2}}$ are described in Figure 1.

Conditional burning. The transaction tx_{C2} achieves conditional burning. tx_{C2} transfers $\$c_a + \$c_b - \$v$ coins from Adr_{col} to a burn-address Adr_{burn} (with associated script Φ_{burn} not controlled by anyone[10]), thus leaving $\$v$ coins as transaction fee to any miner mining the transaction into his block. This transaction is set to be valid only if (1) both Alice and Bob have signed it, and (2) the values pre_a and pre_b are attached.

[10]in Bitcoin setting the script Φ_{burn} to be the opcode OP_RETURN makes the coins sent to this address to be unspendable

9.2.3 Protocol Flow of our Ponyta Instantiation

Before setting up PONYTA on the blockchain, Alice and Bob agree on the setup transaction tx_{stp}. The transaction must be signed by both Alice and Bob to take effect. However, before signing tx_{stp}, Alice and Bob agree on and sign all redeeming transactions, including tx_{P1}, tx_{P2}, tx^1_{C1}, tx^2_{C2}, 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_{pay} and Adr_{col}, are not ready yet.

At this moment, Alice and Bob both reveal their 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 publishing tx_{stp}. In the honest run of the protocol, after either of the above redeem paths are published on the blockchain, Alice and Bob can redeem the collateral after waiting for time T_2 using either tx^1_{C1} (if Alice redeemed the payment) or tx^2_{C1} (if Bob redeemed the payment). If one of the users misbehave, and try to activate both redeem paths, for instance, a strategic Alice reveals pre_a (along with tx_{P1}) when Bob has already revealed pre_b and tx_{P2}, any miner in the system can immediately spend from the C2 branch of ϕ_{col}, and burn the collateral of Alice and Bob while redeeming $\$v$ coins as transaction fee for himself. Note that this is possible, because before setting up PONYTA Alice and Bob had broadcast all redeem transactions and signatures, including the transaction tx_{C2} and their signatures on the transaction. As both pre_a and pre_b are revealed, the miner has enough information to authorize the transaction tx_{C2} on the blockchain, and thus publish the transaction, the signatures (from Alice and Bob), and the values pre_a and pre_b, in *his own* block. He obtains $\$v$ coins as fee from the transaction while $\$c_a + \$c_b - \$v$ are burnt.

9.2.4 Estimated Transaction Costs

Standard Bitcoin transactions let addresses of Alice and Bob to be *pay-to-public key-hash* scripts, and the scripts Φ_{pay} and Φ_{col} can be initiated using *pay-to-script-hash* scripts as done in [TYME21]. We can optimize the size of the scripts by using shared public keys between Alice and Bob instead of separate keys. For example, instead of requiring signatures wrt. pk_a and pk_b for the payment redeem tx_{P1}, we require a single signature wrt. pk_{ab} which is a shared public key between Alice and Bob. This means that users need to jointly generate a single signature instead of separately generating 2 signatures under independent public keys. With this, we estimate the size of our payment address script to be 121 bytes and our collateral address script to be 115 bytes. Excluding the standard Bitcoin transaction overhead like version number, transaction id, number of inputs, outputs etc, the size of the setup transaction tx_{stp} is 320 bytes, our payment redeem transactions (both tx_{P1} and tx_{P2}) are 290 bytes each when using ECDSA signatures. Our collateral redeem transactions tx^1_{C1} and tx^2_{C1} each are of size 366 bytes, while tx_{C2} is of size 228 bytes, again excluding the standard overheads.

10 Ethereum Instantiation

We implemented PONYTA (from Section 5.1) in Solidity, the smart contract language used in Ethereum which supports general smart contracts. Note that it is easy to extend the Solidity contracts to the PONYTA instantiations from Sections 6 and 8.2 with simple additional calls to the contract for withdrawal of coins.

Table 2: 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	Dep. Alice	60,239
		Dep. Bob	62,345
		Dep. Miner	61,008
		Col. Bob	42,266
		Col. Miner	46,063
Ponyta	759,859	Dep. Alice	67,748
		Dep. Bob	69,785
		Col. Alice/Bob	53,698
		Col. Miner	51,936

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. In Table 2, we compare the gas costs of various operations of Ponyta with those of MAD-HTLC and HTLC. The gas cost for the initial deployment of the contract far outweighs those of the redeem transactions. As expected, Ponyta incurs costs that are a bit higher than those of MAD-HTLC, as Ponyta contains slightly more code (103 LoC in Ponyta vs. 72 in MAD-HTLC).

Our implementation in Ethereum is straight-forward and consists of a single contract which includes the initialization as well as all redeem paths.

We deployed Ponyta on Rinkeby, Ethereum's testnet. We posted transactions redeeming the payment through paths P1 and P2 (see Section 5.1), as well as the transaction redeeming the collateral through path C1, and the transaction which sends a payout to a user and causes coin burning through path C2 (Section 5.1).

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.

[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.

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

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

[BKa] Iddo Bentov and Ranjit Kumaresan. How to use bitcoin to design fair protocols. In *CRYPTO*.

[BKb] 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.

[Bon] Joseph Bonneau. Why buy when you can rent? - bribery attacks on bitcoin-style consensus. In *Financial Cryptography Workshops 2016*.

[Bon16] Joseph Bonneau. Why buy when you can rent? In *International Conference on Financial Cryptography and Data Security*, pages 19–26. Springer, 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*. Springer-Verlag, 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+17a] 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.

[CGJ+17b] 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 *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, CCS '17, page 719–728, New York, NY, USA, 2017. Association for Computing Machinery.

[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*.

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

[Her18] Maurice Herlihy. Atomic cross-chain swaps. In *Proceedings of the 2018 ACM Symposium on Principles of Distributed Computing*, PODC '18, page 245–254, New York, NY, USA, 2018. Association for Computing Machinery.

[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 *2014 IEEE 11th International Conference on e-Business Engineering*, pages 44–51, 2014.

[JSZ+19] Aljosha Judmayer, Nicholas Stifter, Alexei Zamyatin, Itay Tsabary, Ittay Eyal, Peter Gazi, Sarah Meiklejohn, and Edgar R Weippl. Pay-to-win: Incentive attacks on proof-of-work cryptocurrencies. *IACR Cryptol. ePrint Arch.*, 2019:775, 2019.

[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 *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, CCS '16, page 418–429, New York, NY, USA, 2016. Association for Computing Machinery.

[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 *2016 IEEE Symposium on Security and Privacy (SP)*, pages 839–858, 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+] Andrew Miller, Iddo Bentov, Surya Bakshi, Ranjit Kumaresan, and Patrick McCorry. Sprites and state channels: Payment networks that go faster than lightning. In *Financial Cryptography 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 *Annals of Emerging Technologies in Computing (AETiC)*, 2019.

[MES16] Malte Möser, Ittay Eyal, and Emin Gün Sirer. Bitcoin covenants. In *Financial Cryptography Workshops*, volume 9604 of *Lecture Notes in Computer Science*, pages 126–141. Springer, 2016.

[MHM18a] Patrick McCorry, Alexander Hicks, and Sarah Meiklejohn. Smart contracts for bribing miners. In *Financial Cryptography Workshops*, 2018.

[MHM18b] Patrick McCorry, Alexander Hicks, and Sarah Meiklejohn. Smart contracts for bribing miners. In *International Conference on Financial Cryptography and Data Security*, pages 3–18. Springer, 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*.

[MMSH] Patrick Mccorry, Malte Möser, Siamak F. Shahandasti, and Feng Hao. Towards bitcoin payment networks. In *Proceedings, Part I, of the 21st Australasian Conference on Information Security and Privacy - Volume 9722, 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*, pages 115–129. IEEE Computer Society, 2017.

[PSS17] Rafael Pass, Lior Seeman, and Abhi Shelat. Analysis of the blockchain protocol in asynchronous networks. In *Advances in Cryptology - EUROCRYPT 2017 - 36th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Paris, France, April 30 - May 4, 2017, Proceedings, Part II*, pages 643–673, 2017.

[rin22] Rinkeby testnet, 2022. URL: https://www.rinkeby.io/.

[TYME21] Itay Tsabary, Matan Yechieli, Alex Manuskin, and Ittay Eyal. MAD-HTLC: because HTLC is crazy-cheap to attack. In *IEEE Symposium on Security and Privacy*, pages 1230–1248. IEEE, 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 *2019 IEEE European Symposium on Security and Privacy Workshops (EuroS PW)*, pages 357–366, 2019.

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

A Deferred Proofs

In this section, we provide the proofs of Theorem 6.1 and Theorem 6.2.

A.1 Proof of Theorem 6.1

Before proving Theorem 6.1, we introduce two useful lemmas.

Lemma A.1. *Let $\mathcal{C}$ be any coalition that consists of Alice and an arbitrary subset of miners (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. Suppose everyone not in $\mathcal{C}$ plays the honest strategy. Then, the coalition $\mathcal{C}$ can play honestly, i.e., deposit $\$c_a$ into Ponyta at $t = 0$, send pre_a to P1 at $t = T$, and post _ to C1 when T_2 has passed since P1 or P2 is activated. In this case, $\mathcal{C}$ obtains utility $\$v - \$v_a$.

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 and deposit money into it — 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:

1. First, P1 is successfully activated at some point. Since C1 and C2 are mutually exclusive, and $\$c_a \geq \$v$, $\$v + \$c_a$ is the maximal amount that the coalition can redeem from the PONYTA contract. In this case, the honest Bob will output pre_a and thus the coalition $\mathcal{C}$'s utility is at most $\$v - \$v_a$, which is the same as following the honest strategy.

2. Second, P1 is never activated. In this case, it is impossible for the coalition to redeem any value from P1 or P2, and its utility is at most zero. Since $\$v > \$v_a$ by our assumption, following the honest strategy maximizes the utility of $\mathcal{C}$.

□

Lemma A.2. *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 $\gamma^{T_2} \leq \frac{\$c_b - \$v}{\$c_b}$, for any (even unbounded[11]) 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. Suppose everyone not in $\mathcal{C}$ plays the honest strategy. If the coalition follows the honest strategy as well, P1 and C1 will be activated, and Bob will output pre_a. Thus, the utility of the coalition is $\$v_b - \$v$.

Now, consider the case where the coalition $\mathcal{C}$ deviates from the honest strategy. Just like in the proof of Lemma A.1, we may assume that the coalition does not post any new contract during the protocol execution.

Because $\$c_b > \$v$, if Bob does not learn pre_a, the utility of $\mathcal{C}$ is at most zero where Bob simply gets all his deposit back from PONYTA. Notice that Alice posts pre_a only when she goes to the execution phase. Thus, if Alice goes to the abort phase, the utility of $\mathcal{C}$ is at most zero.

Henceforth, we assume Alice goes to the execution phase. There are two possibilities. First, P1 is successfully activated. In this case, because $\$c_b > \$v$, the maximal value the coalition can get from the contract is to activate the C1 branch, and the utility of the coalition is at most $\$v_b - \$v$, which is the same as the honest case.

Second, P2 is activated. Let t^* be the time at which P2 is activated. There are two subcases. In the first subcase, the coalition also gets $\$v$ from C2 at time t^* or earlier. In this case, the coalition's utility is at most $\$v + \$v_b - \$c_b$, and since $\$c_b > 2 \cdot \$v$, this is less than the honest case. Henceforth, we may assume that the coalition has not got $\$v$ from C2 at time t^* or earlier. Since the honest Alice posts pre_a at $t = T$ and P2 can only be activated after time 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 $\$v_b - \$c_b$. 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

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

Recall that $\$c_b > 2 \cdot \$v$. Therefore, for $(\$v_b - \$c_b) \cdot (1 - p) + \$v_b \cdot p$ to exceed the honest utility $\$v_b - \$v$, it must be that $p > \frac{\$c_b - \$v}{\$c_b}$ which is not true by our assumption. We thus conclude that $\mathcal{C}$ cannot increase its utility through any deviation. □

[11] Bob's coalition can be unbounded as long as the mining process is idealized as in our model.

Proof of Theorem 6.1. Lemma A.1 and Lemma A.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. This is easy to see: since both Alice and Bob are honest, the coalition's utility is 0 unless it can find pre_b on its own — the probability of this happening is negligibly small due to the one-wayness of the hash function $H(\cdot)$.

A.2 Proof of Theorem 6.2

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

- Case 1: Alice drops out before sending the deposit transaction. In this case, Ponyta never enters the execution phase. Bob will go to the abort phase, and he will send the withdrawal transaction to Ponyta. Because at least $1/\mathsf{poly}(\lambda)$ fraction of the mining power is honest, except the negligible probability, Bob can get all his deposit back in polynomial time.
- Case 2: Alice already sent the deposit transaction but drops out before posting a transaction containing pre_a. Because at least $1/\mathsf{poly}(\lambda)$ 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.
- Case 3: Alice drops out after she already posted a transaction containing pre_a. In this case, an honest miner would include Alice's transaction and activate P1. Then, T_2 after P1 is activated, Bob would send _ to C2. As long as at least $1/\mathsf{poly}(\lambda)$ fraction of the mining power is honest, P1 and C1 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. There are three possible cases.

- Case 1: Bob drops out before sending the deposit transaction. In this case, Ponyta never enters the execution phase. Alice will go to the abort phase, and she will send the withdrawal transaction to Ponyta. Because at least $1/\mathsf{poly}(\lambda)$ fraction of the mining power is honest, except the negligible probability, Alice can get all his deposit back in polynomial time.
- Case 2: Bob drops out after sending the deposit transaction. In this case, Alice always posts a transaction containing pre_a, and except with negligible probability, P1 and C1 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.

B Bitcoin Instantiation of Ponyta with Safe Participation in the Presence of Externally Incentivized Players

In this section we describe how Ponyta from Section 6 that guarantee safe participation in the presence of externally incentivized players, can be instantiated in Bitcoin. The main ideas behind our instantiation is the same as in Section 9.2 except we have a few additional transactions before the setup transaction to capture the preparation phases of Ponyta from Section 6.1 and to capture the empty message calls in the execution phase.

B.1 Transactions

We first highlight the additional transactions below.

Transactions for the Preparation Phase. Recall that the additional preparation phase in PONYTA contract from Section 6.1 lets the coins deposited by either Alice or Bob be withdrawn under some conditions, even before the PONYTA execution phase begins. To capture this in our Bitcoin instantiation, we let both Alice and Bob redirect their coins via additional transactions, before these coins are input to the setup transaction tx_{stp} where the payment and collateral addresses are created.

More specifically, in the PONYTA instantiation from Section 9.2, Alice and Bob from their addresses, transfer $\$c_a$ and $\$x + \$c_b$ coins respectively, via the setup transaction tx_{stp} to create the payment address Adr_{pay} and the collateral address Adr_{col}. However now, we have a new transaction tx_{prep} that transfers $\$c_a$ coins from Alice's address to an address Adr^A_{prp} and $\$x + \$c_b$ coins from Bob's address to an address Adr^B_{prp}, *before* the transaction tx_{stp} is generated. The description of the transaction is given in Table 3 and the scripts Φ^A_{prp} and Φ^B_{prp} associated with Adr^A_{prp} and Adr^B_{prp}, respectively, are described in Figure 3.

Table 3: Additional transaction in Bitcoin instantiation of PONYTA with safe participation. Here Φ_{pay} is the script described in Figure 3. The script $\Phi^{\mathsf{ax}}_{\mathrm{P2}}$ is the same as described in Figure 1.

	Description
tx_{prep}	$tx\left(\begin{array}{c}[(Adr^A_0, \Phi^A_0, \$c_a), (Adr^B_0, \Phi^B_0, \$x + \$c_b)],\\ [(Adr^A_{\mathsf{prp}}, \Phi^A_{\mathsf{prp}}, \$c_a), (Adr^B_{\mathsf{prp}}, \Phi^B_{\mathsf{prp}}, \$x + \$c_b)]\end{array}\right)$
$tx^{\mathrm{empty}}_{\mathrm{P2}}$	$tx\left(\begin{array}{c}[(Adr_{\mathsf{pay}}, \Phi_{\mathsf{pay}}, \$x)],\\ [(Adr^A_1, \Phi^A_1, \$x - \$\epsilon), (Adr^{\mathsf{ax}}_{\mathrm{P2}}, \Phi^{\mathsf{ax}}_{\mathrm{P2}}, \$\epsilon)]\end{array}\right)$
$tx^{2,\mathrm{empty}}_{\mathrm{C1}}$	$tx\left(\begin{array}{c}[(Adr_{\mathsf{col}}, \Phi_{\mathsf{col}}, \$c_a + \$c_b), (Adr^{\mathsf{ax}}_{\mathrm{P2}}, \Phi^{\mathsf{ax}}_{\mathrm{P2}}, \$\epsilon)],\\ [(Adr^A_3, \Phi^A_3, \$c_a), (Adr^B_3, \Phi^B_3, \$c_b + \$\epsilon)]\end{array}\right)$

Intuitively, the script Φ^A_{prp} is set such that the coins can be redeemed (1) by any user producing pre_a, (2) by Alice herself or, (3) by the PONYTA setup transaction tx_{stp} generated later. Similarly, the script Φ^B_{prp} is set such that the coins can be redeemed (1) by any user producing pre_b, (2) by Bob himself or, (3) by the PONYTA setup transaction tx_{stp} generated later. A pictorial description of the transaction flow is given in Figure 4. Here tx_{A1}, $tx^{\mathrm{empty}}_{\mathrm{A2}}$, tx_{B1} and $tx^{\mathrm{empty}}_{\mathrm{B2}}$ are some transactions that activate the branches A1, A2, B1 and B2, respectively of PONYTA. That is, if pre_a (pre_b) is known already, any user P can activate A1 (B1) that redeems the coins to herself. Alice (Bob) can withdraw the coins by activating A2 (B2).

If none of the above transactions are used to redeem the coins from Adr^A_{prp} and Adr^B_{prp}, then transaction tx_{stp} can be used to redeem the coins. Recall that once tx_{stp} is published, it creates the payment and the collateral addresses. This marks the end of the preparation phase and the start of the execution phase.

Setup Transaction. The setup transaction tx_{stp} like in Section 9.2 creates two addresses, the payment address Adr_{pay} and the collateral address Adr_{col}. But there is a slight change in the script associated with Adr_{pay}.

Payment Redeem Transactions. We now have one additional transaction $tx^{\mathrm{empty}}_{\mathrm{P2}}$ (see Table 3 for description) apart from the transactions tx_{P1} and tx_{P2} that are the same as in Section 9.2. The new transaction $tx^{\mathrm{empty}}_{\mathrm{P2}}$ redeems $\$x - \$\epsilon$ coins to Bob's address and $\$\epsilon$ coins are paid to an auxiliary

$\Phi^A_{\mathsf{prp}}(tx, pre_a, \sigma_a)$

A1: **if** $(H(pre_a) = h_a)$ **then return** 1
A2: **if** $(\mathsf{Vf}(\mathsf{pk}_a, tx, \sigma_a) = 1)$ **then return** 1
// Values h_a, pk_a are hardwired

$\Phi^B_{\mathsf{prp}}(tx, pre_b, \sigma_b)$

B1: **if** $(H(pre_b) = h_b)$ **then return** 1
B2: **if** $(\mathsf{Vf}(\mathsf{pk}_b, tx, \sigma_b) = 1)$ **then return** 1
// Values h_b, pk_b are hardwired

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

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** $(\_\mathrm{NOW} > T_1) \wedge (H(pre_b) = h_b) \wedge (\mathsf{Vf}(\mathsf{pk}'_a, tx, \sigma_a) = 1) \wedge (\mathsf{Vf}(\mathsf{pk}'_b, tx, \sigma_b) = 1)$
then return 1
E: **if** $(\_\mathrm{NOW} > T_1) \wedge (\mathsf{Vf}(\mathsf{pk}''_a, tx, \sigma_a) = 1)$ **then return** 1
// Values h_a, h_b, pk_a, pk_b, T_1, pk'_a, pk'_b, pk''_a are hardwired

Figure 3: The description of script Φ^A_{prp}, Φ^B_{prp} and Φ_{pay}. Here tx is the transaction spending from the script. The key pk_a belongs to Alice, and the key pk_b belong to Bob.

address for the purpose of conditional timelock redeem. The script Φ_{pay} associated with Adr_{pay} (see Figure 3) is a slight modification of the one from Figure 1. Concretely, the script Φ_{pay} now has an additional branch to redeem the coins. Intuitively, this additional branch (denoted by E) allows for coins to be redeemed from Adr_{pay} if after a timeout of T_1, Alice just signs the corresponding redeeming transaction. We set the transaction $tx^{\mathrm{empty}}_{\mathrm{P2}}$ to redeem the coins from this (E) branch. This will correspond to the empty message call to the PONYTA contract (from Section 6.1) in the activation point P2.

Collateral Redeem Transactions. The additional branch and therefore an additional transaction $tx^{\mathrm{empty}}_{\mathrm{P2}}$ that was introduced for redeeming the payment, also introduces an additional transaction to redeem the collateral coins from Adr_{col}. That is, in addition to transactions $tx^{1}_{\mathrm{C1}}, tx^{2}_{\mathrm{C1}}$ and tx_{C2} just as in the instantiation from Section 9.2, we have an additional transaction $tx^{2,\mathrm{empty}}_{\mathrm{C1}}$ (see Table 3 for description). The new transaction $tx^{2,\mathrm{empty}}_{\mathrm{C1}}$ redeems coins from the C1 activation point, depending on if the transaction $tx^{\mathrm{empty}}_{\mathrm{P2}}$ was activated earlier. The C1 branch timeout of τ for the transaction $tx^{2,\mathrm{empty}}_{\mathrm{C1}}$ is implemented via the auxiliary address created by $tx^{\mathrm{empty}}_{\mathrm{P2}}$ (conditional timelock redeem). The logic for conditional burning is unchanged from the instantiation from Section 9.2. A pictorial description of the transaction flow is given in Figure 5.

B.2 Protocol Flow

Alice and Bob, first sign the preparation transaction tx_{prep} as described above and publish the same on the blockchain. Alice or Bob can withdraw coins from Adr^A_{prp} and Adr^B_{prp} using $tx^{\mathrm{empty}}_{\mathrm{A2}}$ and $tx^{\mathrm{empty}}_{\mathrm{B2}}$, respectively, whenever they wish, during the preparation phase. Any party can withdraw

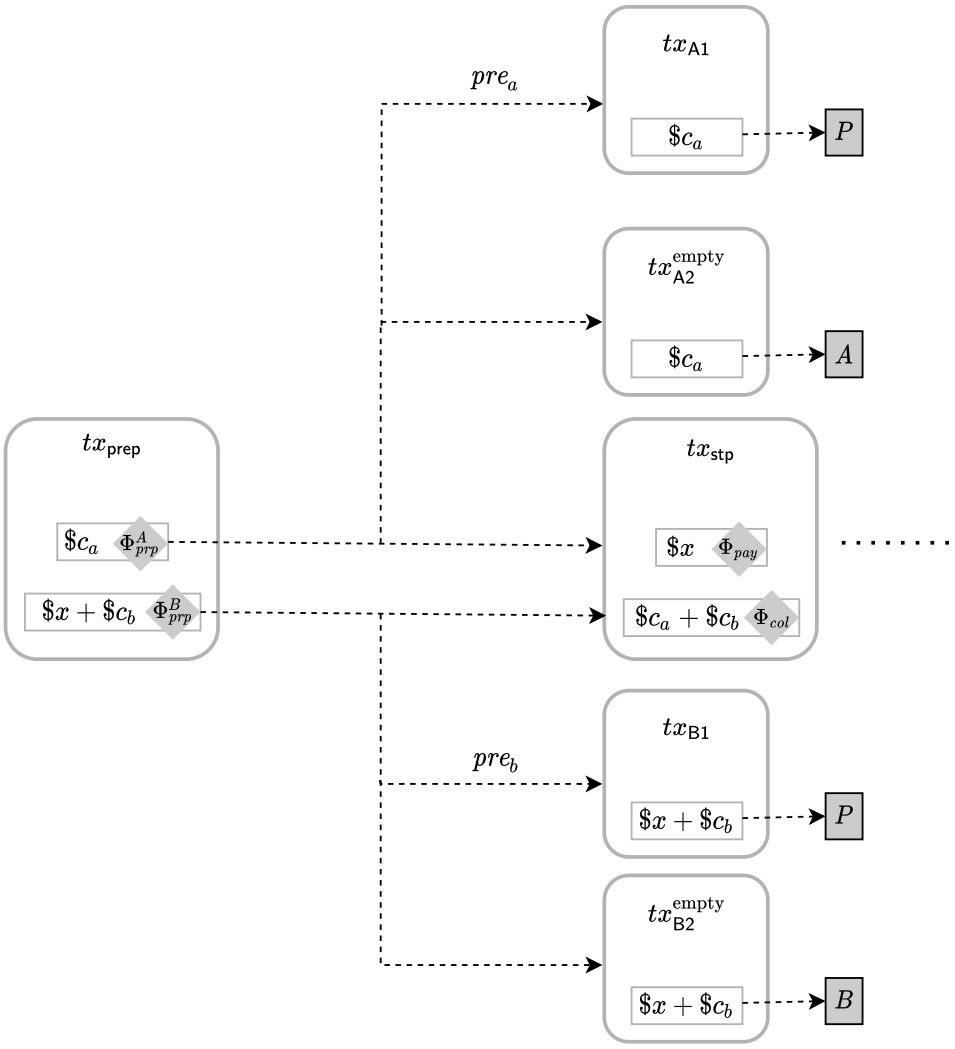

Figure 4: The transaction flow during the preparation phase of Ponyta in Bitcoin with safe participation. Dashed arrows indicate that the output can be spent by one user. We denote A for Alice, B for Bob, or P for some user. Transaction tx_{A1} (tx_{B1}) can be published by P if value pre_a (pre_b) is also published together. Transaction $tx^{\text{empty}}_{\text{A2}}$ ($tx^{\text{empty}}_{\text{B2}}$) can be published by A (B) along with their signature on the transaction.

the coins from Adr^A_{prp}, by publishing an empty transaction tx_{A1} and attaching pre_a to it. Similarly, any party can withdraw the coins from Adr^B_{prp}, by publishing an empty transaction tx_{B1} and attaching pre_b to it.

Alice and Bob, first sign the redeeming transactions $tx_{\text{P1}}, tx_{\text{P2}}, tx^1_{\text{C1}}, tx^2_{\text{C1}}, tx_{\text{C2}}$, and $tx^{2,\text{empty}}_{\text{C1}}$. They broadcast all these transactions and the respective signatures, like before. However, this time Alice signs the transaction $tx^{\text{empty}}_{\text{P2}}$, and keeps it private and not broadcast it. Finally, they sign the setup transaction tx_{stp} and publish it on the blockchain. Again, this is only possible if the coins in the addresses Adr^A_{prp} or Adr^B_{prp} of the preparation transaction tx_{prep} are unspent.

Whenever Alice wishes to activate P2 in Ponyta with an empty message, she publishes the transaction $tx^{\text{empty}}_{\text{P2}}$ along with the valid signatures she has in her possession. If this transaction is published on the blockchain, activation point C1 can be activated by $tx^{2,\text{empty}}_{\text{C1}}$ after a timeout of

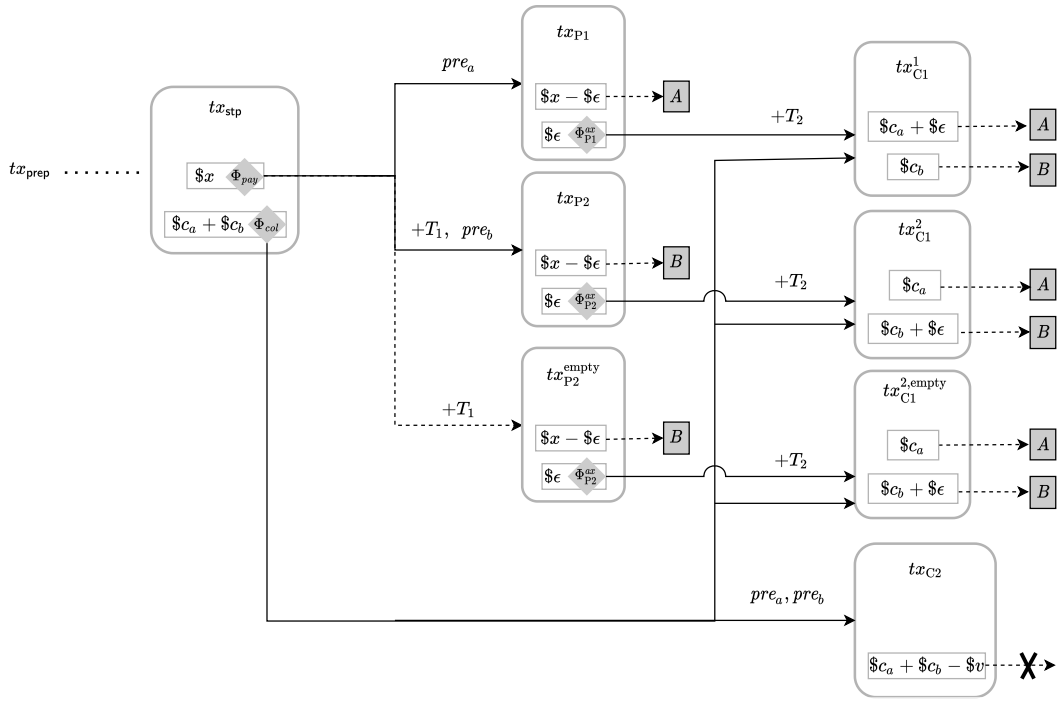

Figure 5: The transaction flow during the execution phase of Ponyta in Bitcoin with safe participation. The preparation phase from Figure 4 precedes the setup transaction. The dashed arrow for $tx^{\text{empty}}_{\text{P2}}$ indicates that Alice can herself publish the transaction without requiring Bob's signature on it.

T_2 time units. Conditional burning via tx_{C2} activates C2 if the parties misbehave similar to the instantiation from Section 9.

Estimated Transaction Costs. We estimate the size of the transaction tx_{prep} to be 320 bytes. The size of the scripts Φ^{A}_{prp} and Φ^{B}_{prp} are 56 bytes each. The size of tx_{A1}, and tx_{B1} are 121 bytes each and the size of $tx^{\text{empty}}_{\text{A2}}$ and $tx^{\text{empty}}_{\text{B2}}$ are 161 bytes each. The size of the setup transaction is 366 bytes which is an increase of 46 bytes compared to the Ponyta instantiation from Section 9.2. The sizes of tx_{P1} and tx_{P2} increase by 70 bytes approximately, due to the additional spending branch (E) in Φ_{pay}. Transaction $tx^{\text{empty}}_{\text{P2}}$ is approximately 335 bytes excluding the standard overheads. Transaction $tx^{2,\text{empty}}_{\text{C1}}$ has the same size as tx^{1}_{C1} (or tx^{2}_{C1}). As before our estimates exclude the standard overhead associated with transactions.

C Bitcoin Instantiation of Atomic Swap

In this section we describe how we can instantiate Ponyta and Ponyta' contracts from Section 8.2 in Bitcoin. The techniques for the instantiations follow quite closely to the techniques from above. Therefore we only highlight the key modifications needed. The Ponyta instantiation is the same as in Appendix B except that the timeout value T_2 is replaced with the value τ. The remainder of this section will focus on the Ponyta' instantiation.

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

pre'_a after a timeout of T'_1 to retrieve the coins. The main difference between this instantiation and the PONYTA instantiation above is that, in the preparation phase, we do not have the activation point B2′ anymore, and in the execution phase both the payment address activation points P1′ and P2′ can be activated by empty message calls.

C.1 Transactions

We describe the new transactions we require for PONYTA'.

Transactions for the Preparation Phase. We have the preparation transaction tx'_{prep} that transfers $\$x' + \$c'_a$ coins from Alice's address to an address Adr'_{prp}. The transaction is published on the blockchain. The description of the transaction is given in Table 4 and the script Φ'_{prp} associated with Adr'_{prp} is described in Figure 6.

Intuitively, the script Φ'_{prp} is set such that the coins can be redeemed (1) by any user producing pre'_a, (2) by Alice herself or, (3) by the PONYTA' setup transaction tx'_{stp}. A pictorial description of the transaction flow is given in Figure 7. Here $tx_{\mathrm{A1}'}$ and $tx^{\mathrm{empty}}_{\mathrm{A2}'}$ are some transactions that activate the branches A1′, and A2′, respectively. That is, if pre'_a is known already, any user P can activate A1′ and redeem the coins herself, while Alice can withdraw her coins by activating A2′. Note that activation point B1′ can be activated by Bob by simply not sending $\$c'_b$ coins as input to the setup transaction, but rather spending it himself with some other transaction.

If none of the above transactions are used to redeem the coins from Adr'_{prp} and if Bob does not activate B1′, then transaction tx'_{stp} (described below) can be used to redeem the coins from Adr'_{prp}, along with the $\$c'_b$ coins of Bob.

Table 4: Additional transactions in Bitcoin for PONYTA' atomic swap from Section 8.2.

	Description
tx'_{prep}	$tx\left(\begin{matrix}[(Adr_0^A, \Phi_0^A, \$x' + \$c'_a)],\\ [(Adr'_{\mathsf{prp}}, \Phi'_{\mathsf{prp}}, \$x' + \$c'_a)]\end{matrix}\right)$
$tx^{\mathrm{empty}}_{\mathrm{P1}'}$	$tx\left(\begin{matrix}[(Adr'_{\mathsf{pay}}, \Phi'_{\mathsf{pay}}, \$x')],\\ [(Adr_1^B, \Phi_1^B, \$x' - \$\epsilon), (Adr^{\mathsf{ax}}_{\mathrm{P1}'}, \Phi^{\mathsf{ax}}_{\mathrm{P1}'}, \$\epsilon)]\end{matrix}\right)$
$tx^{\mathrm{empty}}_{\mathrm{P2}'}$	$tx\left(\begin{matrix}[(Adr'_{\mathsf{pay}}, \Phi'_{\mathsf{pay}}, \$x')],\\ [(Adr_1^A, \Phi_1^A, \$x' - \$\epsilon), (Adr^{\mathsf{ax}}_{\mathrm{P2}'}, \Phi^{\mathsf{ax}}_{\mathrm{P2}'}, \$\epsilon)]\end{matrix}\right)$
$tx^{1,\mathrm{empty}}_{\mathrm{C1}'}$	$tx\left(\begin{matrix}[(Adr'_{\mathsf{col}}, \Phi'_{\mathsf{col}}, \$c'_a + \$c'_b), (Adr^{\mathsf{ax}}_{\mathrm{P1}'}, \Phi^{\mathsf{ax}}_{\mathrm{P1}'}, \$\epsilon)],\\ [(Adr_2^A, \Phi_2^A, \$c'_a + \$\epsilon), (Adr_2^B, \Phi_2^B, \$c'_b)]\end{matrix}\right)$
$tx^{2,\mathrm{empty}}_{\mathrm{C1}'}$	$tx\left(\begin{matrix}[(Adr'_{\mathsf{col}}, \Phi'_{\mathsf{col}}, \$c'_a + \$c'_b), (Adr^{\mathsf{ax}}_{\mathrm{P2}'}, \Phi^{\mathsf{ax}}_{\mathrm{P2}'}, \$\epsilon)],\\ [(Adr_3^A, \Phi_3^A, \$c'_a), (Adr_3^B, \Phi_3^B, \$c'_b + \$\epsilon)]\end{matrix}\right)$

Setup Transaction. The setup transaction tx'_{stp} analogous to tx_{stp} creates the two addresses Adr'_{pay} and Adr'_{col}.

Payment Redeem Transactions. We now have two additional payment redeem transactions, $tx^{\mathrm{empty}}_{\mathrm{P1}'}$ and $tx^{\mathrm{empty}}_{\mathrm{P2}'}$ (see Table 4) apart from $tx_{\mathrm{P1}'}$ and $tx_{\mathrm{P2}'}$ that redeem from the payment address Adr'_{pay}. We have the transaction $tx^{\mathrm{empty}}_{\mathrm{P1}'}$ that redeems $\$x' - \$\epsilon$ coins to Bob's address and $\$\epsilon$ coins are paid to an auxiliary address. The transaction $tx^{\mathrm{empty}}_{\mathrm{P2}'}$ redeems $\$x' - \$\epsilon$ coins to Alice's address and $\$\epsilon$ coins are paid to an auxiliary address. The description of Φ'_{pay} is given below in Figure 6 with Alice and Bob's roles being reversed in PONYTA'. We set the two transactions $tx^{\mathrm{empty}}_{\mathrm{P1}'}$ and

$\Phi'_{\mathsf{prp}}(tx, pre'_a, \sigma_a)$

A1': **if** $(H(pre'_a) = h'_a)$ **then return** 1

A2': **if** $(\mathsf{Vf}(\mathsf{pk}_a, tx, \sigma_a) = 1)$ **then return** 1

// Values h'_a, pk_a are hardwired

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

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

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

E2: **if** $(\mathsf{Vf}(\mathsf{pk}''_b, tx, \sigma_b) = 1)$ **then return** 1

// Values h_a, h_b, pk_a, pk_b, T_1, pk'_a, pk'_b, pk''_a, pk''_b are hardwired

Figure 6: The description of scripts Φ'_{prp} and Φ'_{pay}.

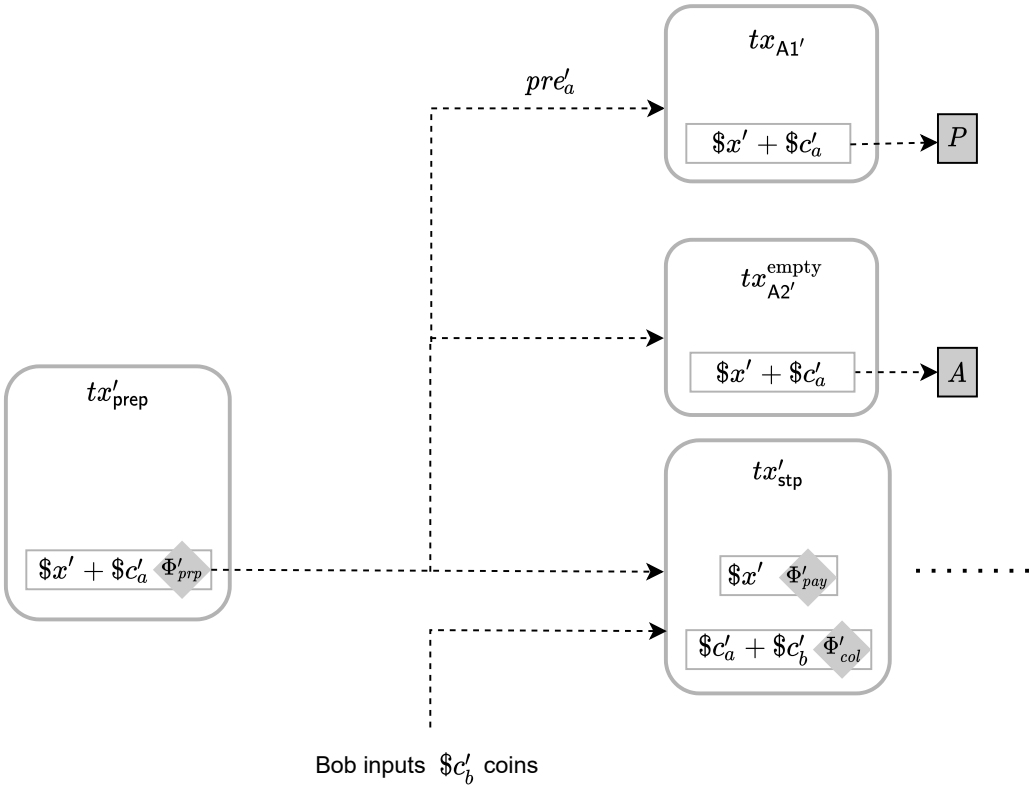

Figure 7: The transaction flow of preparation phase in Ponyta' in Bitcoin for atomic swap. We denote A for Alice, and P for some user. Transaction $tx_{\mathrm{A1'}}$ can be published by P if value pre'_a is also published together. Transaction $tx^{\mathrm{empty}}_{\mathrm{A2'}}$ ($tx^{\mathrm{empty}}_{\mathrm{B2}}$) can be published by A along with their signature on the transaction. If Bob does not input $\$c'_b$ coins to tx'_{stp}, it is considered that he activated B1′ branch of Ponyta'.

$tx^{\text{empty}}_{\text{P2}'}$ to redeem the coins from the (E1) and (E2) branches, respectively. These transactions will correspond to the empty message calls to the PONYTA' contract in activation points P1′ and P2′, respectively. The script Φ'_{col} is the same as Φ_{col} in PONYTA except we use the values pre'_a and pre'_b (instead of pre_a and pre_b). Similarly the scripts $\Phi^{\mathsf{ax}}_{\text{P1}'}$ and $\Phi^{\mathsf{ax}}_{\text{P2}'}$ (of the auxiliary addresses) are the same as $\Phi^{\mathsf{ax}}_{\text{P1}}$ and $\Phi^{\mathsf{ax}}_{\text{P2}}$, respectively, from Figure 1, except we replace the timeout value τ with τ'.

Collateral Redeem Transactions. In addition to transactions $tx^{1}_{\text{C1}'}, tx^{2}_{\text{C1}'}$ and $tx_{\text{C2}'}$, we have two additional transactions $tx^{1,\text{empty}}_{\text{C1}'}$ and $tx^{2,\text{empty}}_{\text{C1}'}$ (see Table 4). Both of these new transactions redeem coins from the C1′ activation point, depending on whether transaction $tx^{\text{empty}}_{\text{P1}'}$ or $tx^{\text{empty}}_{\text{P2}'}$ was activated earlier, respectively. Similar to the PONYTA instantiation, C1′ branch timeout of τ' for $tx^{1,\text{empty}}_{\text{C1}'}$ and $tx^{2,\text{empty}}_{\text{C1}'}$ is implemented via the auxiliary addresses created by $tx^{\text{empty}}_{\text{P1}'}$ and $tx^{\text{empty}}_{\text{P2}'}$. The logic for conditional burning and conditional timelock redeem are unchanged. A pictorial description of the transaction flow for payment and collateral redeem is given in Figure 8.

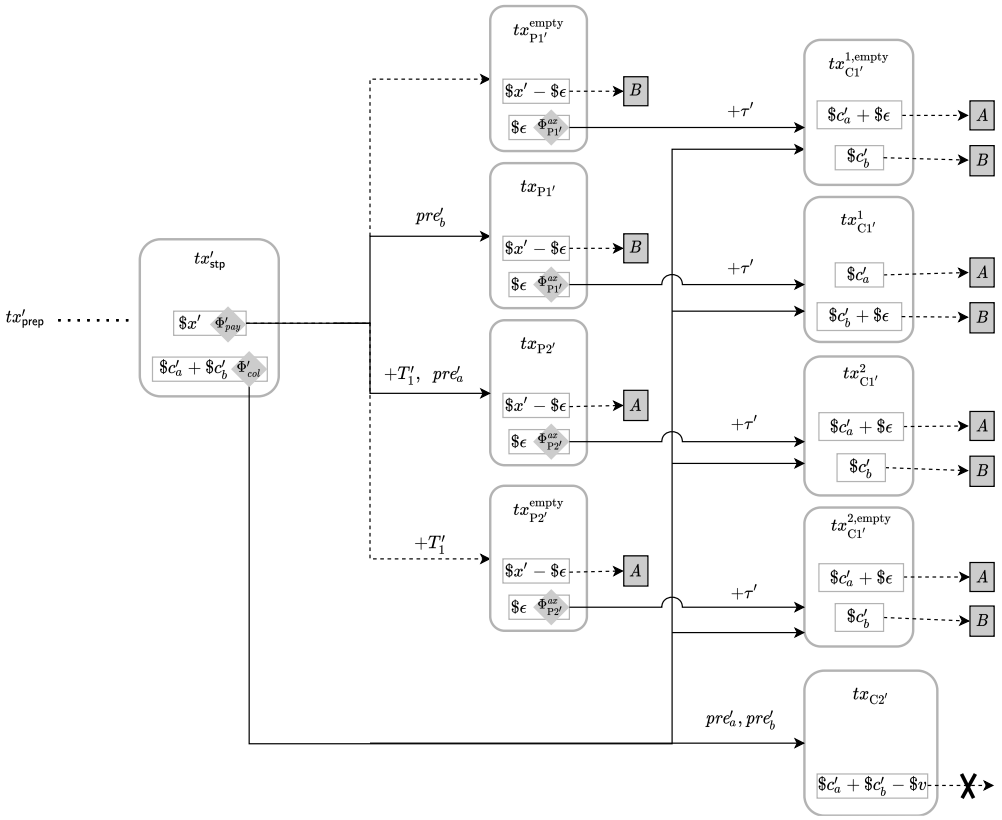

Figure 8: The transaction flow of PONYTA' in Bitcoin for atomic swap. The preparation phase from Figure 7 precedes the setup transaction. The dashed arrow for $tx^{\text{empty}}_{\text{P1}'}$ indicates Alice can herself publish the transaction without requiring Bob's signature on it and the dashed arrow for $tx^{\text{empty}}_{\text{P2}'}$ indicates that Bob can himself publish the transaction without requiring Alice's signature on it.

C.2 Protocol Flow

Alice publishes the preparation transaction tx'_{prep} on the blockchain and she can withdraw coins from Adr'_{prp} whenever she wishes to during the preparation phase using the transaction $tx^{\text{empty}}_{\text{A2}'}$.

Any party can withdraw the coins from Adr'_{prp}, by publishing an empty transaction $tx_{\mathrm{A1}'}$ and attaching pre'_a with it. Bob may activate B1′ by simply not sending as input the coins $\$c'_b$ into tx'_{stp} anymore.

Alice and Bob, first sign the redeeming transactions $tx_{\mathrm{P1}'}$, $tx_{\mathrm{P2}'}$, $tx^{1}_{\mathrm{C1}'}$, $tx^{2}_{\mathrm{C1}'}$, $tx_{\mathrm{C2}'}$, $tx^{1,\mathrm{empty}}_{\mathrm{C1}'}$, and $tx^{2,\mathrm{empty}}_{\mathrm{C1}'}$. They broadcast all these transactions and the respective signatures, like before. However, this time Alice signs the transaction $tx^{\mathrm{empty}}_{\mathrm{P1}'}$ and Bob signs the transaction $tx^{\mathrm{empty}}_{\mathrm{P2}'}$, and do not broadcast these transactions and keep them private with themselves for now. They sign the setup transaction tx'_{stp} and publish it on the blockchain.

Whenever Alice wishes to activate P1′ in Ponyta' with an empty message, she publishes the transaction $tx^{\mathrm{empty}}_{\mathrm{P1}'}$ along with the valid signature she has in her possession. Similarly, whenever Bob wishes to activate P2′ in Ponyta' with an empty message, he publishes the transaction $tx^{\mathrm{empty}}_{\mathrm{P2}'}$ along with the valid signature he has in his possession. If either of these transactions are published on the blockchain, activation point C1′ can be activated by either $tx^{1,\mathrm{empty}}_{\mathrm{C1}'}$ or $tx^{2,\mathrm{empty}}_{\mathrm{C1}'}$ after a timeout of τ' time units. Rest of the execution proceeds analogous to Ponyta from Appendix B.

Estimated Transaction Costs. In terms of cost, (1) the sizes of $tx_{\mathrm{P1}'}$ and $tx_{\mathrm{P2}'}$ are the same as tx_{P1} and tx_{P2} from Appendix B, (2) transactions $tx^{\mathrm{empty}}_{\mathrm{P1}'}$ and $tx^{\mathrm{empty}}_{\mathrm{P2}'}$ are the same size as $tx^{\mathrm{empty}}_{\mathrm{P2}}$ from Appendix B and (3) transactions $tx^{1,\mathrm{empty}}_{\mathrm{C1}'}$ and $tx^{2,\mathrm{empty}}_{\mathrm{C1}'}$ are of the same size as $tx^{1}_{\mathrm{C1}'}$ (or $tx^{2}_{\mathrm{C1}'}$) which remain unchanged from Appendix B.