BFastPay: A Routing-free Protocol for Fast Payment in Bitcoin Network

Xinyu Lei, Guan-Hua Tu, Tian Xie, Sihan Wang
Department of Computer Science and Engineering
Michigan State University
East Lansing, Michigan, USA
{leixinyu,ghtu,xietian1,wangsih3}@msu.edu

ABSTRACT

Bitcoin is the most popular cryptocurrency which supports payment services via the Bitcoin peer-to-peer network. However, Bitcoin suffers from a fundamental problem. In practice, a secure Bitcoin transaction requires the payee to wait for at least 6 block confirmations (one hour) to be validated. Such a long waiting time thwarts the wide deployment of the Bitcoin payment services because many usage scenarios require a much shorter waiting time. In this paper, we propose BFastPay to accelerate the Bitcoin payment validation. BFastPay employs a smart contract called BFPayArbitrator to host the payer's security deposit and fulfills the role of a trusted payment arbitrator which guarantees that a payee always receives the payment even if attacks occur. BFastPay is a routing-free solution that eliminates the requirement for payment routing in the traditional payment routing network (e.g., Lightning Network). The theoretical and experimental results show that BFastPay is able to significantly reduce the Bitcoin payment waiting time (e.g., from 60 mins to less than 1 second) with nearly no extra operation cost.

CCS CONCEPTS

• **Security and privacy → Distributed systems security**; **Security protocols**.

KEYWORDS

Blockchain; Bitcoin; Smart Contract; Fast Payment

ACM Reference Format:
Xinyu Lei, Guan-Hua Tu, Tian Xie, Sihan Wang. 2021. BFastPay: A Routing-free Protocol for Fast Payment in Bitcoin Network. In *Proceedings of the Eleventh ACM Conference on Data and Application Security and Privacy (CODASPY '21), April 26–28, 2021, Virtual Event, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3422337.3447823

1 INTRODUCTION

Bitcoin as a decentralized payment solution is increasingly gaining recognition and acceptance. For example, it has been accepted by many famous retailers and service providers such as Microsoft [10] and Samsung [15]. Nevertheless, Bitcoin suffers from a key problem. In practical applications, the payee needs to wait for 6 block confirmations (ave. 60 mins) for validating a Bitcoin transaction to defend against the potential double-spending attack launched by the payer. A shorter waiting time increases the risk of the double-spending attack in which the payer spends the same Bitcoin more than once and the payee loses the commodities/services without receiving the Bitcoin payment. The one-hour waiting time has seriously impeded the wide adoption of Bitcoin payment services because many businesses (e.g., vending machines) expect a much shorter waiting time. This problem is one of the most fundamental open problems of Bitcoin. In the past decade, researchers have devoted great efforts to address the problem, but no perfect solutions have been developed yet. We are thus motivated to seek for alternative approaches to address this critical problem.

In this paper, we aim to develop a new Bitcoin payment protocol that satisfies the following requirements. (1) **Mainly using Bitcoin.** The solution should enable users to use Bitcoin as the major payment cryptocurrency instead of requiring users to adopt other cryptocurrencies. (2) **Short waiting time.** The time required for the payee to validate a Bitcoin payment should be short while still defending against the double-spending attacks. (3) **Decentralization.** The solution should preserve Bitcoin's decentralization feature: no reliance on any centralized trusted third party is required. (4) **Low-cost.** The extra operation cost should be low.

The limitations of the prior art are discussed as follows. (1) **Non-Bitcoin-based.** One straightforward solution is to enforce users to use some other cryptocurrencies with faster transaction validation time. However, since Bitcoin has dominated in practical usage [1], it is desirable to develop solutions to support fast Bitcoin payment while keeping Bitcoin as the major payment currency. (2) **Prevention-based.** The solutions proposed in [20, 24, 30] deploy observers in the Bitcoin network to detect the conflicting Bitcoin transactions (i.e., multiple transactions that spent the same Bitcoin). The prevention-based solutions are not highly reliable since the conflicting Bitcoin transactions detection is probabilistic. (3) **Secure wallet-based.** Secure wallet-based solutions [22, 35] require users to trust the secure wallet, so they cannot ensure decentralization. (4) **Escrow-based.** The Lightning Network [31] and Duplex Micropayment Channels [21] are escrow-based solutions which support fast payment via payment routing network. Researchers have developed various solutions [29, 32, 34] to improve the routing performance, but the current escrow-based solutions still only support micropayments. Recently, a solution named Snappy is proposed in [26]. Snappy employs a majority-based governance model, which requires a group of payees (i.e., merchant) to serve as statekeepers to prevent double-spending attacks. It is hard to claim that

CODASPY '21, April 26–28, 2021, Virtual Event, USA

ACM ISBN 978-1-4503-8143-7/21/04…$15.00
https://doi.org/10.1145/3422337.3447823

Snappy preserves decentralization (as it is in Bitcoin) because the majority-based governance model requires coordination between a group of nodes. However, the concept of decentralization in Bitcoin can be achieved without any coordination between nodes (i.e., Bitcoin network is a decentralized self-sustainable system).

To support fast payment in the Bitcoin network, we propose an *inter-blockchain escrow* approach (i.e., BFastPay) in this paper. BFastPay is a general approach that can be deployed on any programmable smart contract (PSC)-supported blockchain platform (e.g., Ethereum [7], EOSIO [5]). BFastPay is called an inter-blockchain (or cross-blockchain) escrow approach since the security deposit (i.e., collateral) is escrowed on another PSC-supported blockchain. BFastPay is designed based on two key insights. First, BFastPay employs a decentralized smart contract called BFPayArbitrator to host the payer's security deposit and fulfill the role of a trusted payment arbitrator which guarantees that a payee always receives the payment even if attacks occur. Note that BFastPay preserves the *decentralization* feature if the underlying PSC-supported blockchain employs a decentralized consensus algorithm. Second, BFastPay takes advantage of the fast consensus property of emerging PSC-supported blockchains (e.g., EOSIO blockchain only needs less than 1 second to validate a transaction [5]) to reduce the waiting time of the Bitcoin transaction. More specifically, BFastPay works as follows. At first, a payer escrows sufficient security deposit into BFPayArbitrator. While the payer submits a Bitcoin transaction to the Bitcoin network, (s)he simultaneously submits a Bitcoin fast payment request (BFPayReq) message to BFPayArbitrator. The BFPayReq message contains all information that BFPayArbitrator needs to make the arbitration if a payment dispute arises later. Once the BFPayReq transaction is successfully validated in the PSC-supported blockchain, the payee can deliver commodities/services to the payer. Hence, the waiting time is reduced to the time needed to validate a transaction on the PSC-supported blockchain. If there is a payment dispute later, BFastPay allows both parties to submit evidence to prove that they are the honest parties. If the payee successfully proves that (s)he does not receive Bitcoin payment, BFPayArbitrator pays the payee using the security deposit. Otherwise, the security deposit still belongs to the payer.

The major technical challenge is that it is hard for BFPayArbitrator to recognize the dishonest party in a payment arbitration because BFPayArbitrator cannot access Bitcoin blockchain (i.e., the inter-blockchain transaction validation is hard). In a payment arbitration, both the payer and the payee have incentives to upload fake evidence to BFPayArbitrator. The payer may submit fake evidence to cheat BFPayArbitrator that the Bitcoin has already paid, while the payee may also submit fake evidence to cheat BFPayArbitrator that no Bitcoin payment is received. However, BFPayArbitrator is unable to distinguish which party is dishonest without accessing the canonical Bitcoin blockchain to obtain the ground truth. To address this challenge, we develop a Bitcoin proof-of-work (PoW)-based payment arbitration mechanism for BFPayArbitrator to identify the dishonest party. The key idea is that our PoW-based arbitration mechanism enables the honest party (either the payer or the payee) to generate a valid proof from the Bitcoin subchain, whereas the dishonest party cannot. Hence, the Bitcoin miners *automatically and unconsciously* help the honest party to generate a valid proof to win the payment arbitration. The dishonest party has to defeat all Bitcoin miners in the mining race to win the payment arbitration, so it is hard for the dishonest party to win the payment arbitration.

In summary, this paper has three main contributions.

(1) We take the first step forward to study the inter-blockchain escrow protocol, BFastPay, which enables the Bitcoin blockchain to employ the fast consensus property of PSC-supported blockchains (which may adopt more advanced consensus techniques) to reduce its payment waiting time without requiring any modification on the current Bitcoin protocols.
(2) We develop a novel PoW-based mechanism for a smart contract BFPayArbitrator without accessing the Bitcoin blockchain to arbitrate whether a Bitcoin transaction has been successfully included into the Bitcoin blockchain or not. Therefore, BFPayArbitrator can ensure payment fairness.
(3) Our theoretical analysis and experimental results demonstrate that BFastPay achieves multiple design goals simultaneously. It is a secure, fast, decentralized, low-cost, and routing-free payment scheme.

The rest of the paper is organized as follows. Section 2 introduces some background knowledge. In Section 3, we introduce the adopted threat model and assumptions. Section 4 presents an overview of BFastPay. The arbitration mechanism used by BFastPay is illustrated in Section 5. Section 6 performs security analysis. Section 7 evaluates the operation cost of BFastPay. Section 8 compares BFastPay with the state-of-the-art solution. Section 9 reviews the related work. Some conclusions are given in Section 10.

2 PRELIMINARIES

This section introduces the preliminaries of the Bitcoin blockchain and programmable smart contract (PSC)-supported blockchains.

Bitcoin Blockchain. The Bitcoin blockchain [28] is a shared public ledger on which all Bitcoin transactions are recorded. Numerous Bitcoin transactions are put into a new block and appended to the blockchain in chronological order. When a block that contains a new Bitcoin transaction has been appended to the Bitcoin blockchain, this transaction has one block confirmation. When a subsequent block is appended to the blockchain, the number of block confirmation for this transaction is increased by one [4]. The current practice for accepting a secure Bitcoin transaction is: waiting for such transaction to have 6 block confirmations. ***Note that 6 block confirmations are based on an assumption that adversaries do not control more than 10% of the global hash power of the Bitcoin network and a double-spending probability of less than 0.1% is acceptable*** **[28].** Bitcoin network refers to the collection of nodes (e.g., miners, wallets) running the Bitcoin P2P protocol. The Bitcoin network uses a PoW-based method for reaching a consensus between different miners. The miner who can solve the hash-based PoW puzzle wins the right to produce a new block for the blockchain. A Bitcoin block header is 80 bytes containing information of (1) previous block hash field, (2) Merkle root field, (3) nonce field, etc. In the mining process, the miners try to find a nonce such that the mined block header meets the current Bitcoin network difficulty target, i.e., $\texttt{Hash(block header)} \leq \texttt{BTC\_diff\_target}$.

PSC-Supported Blockchains. Nowadays, blockchain technology is evolving beyond just supporting a cryptocurrency. Some emerging blockchains (e.g., Ethereum, EOSIO) support rich programmable

Table 1: The consensus mechanisms and transaction validation time of different PSC-supported blockchains.

Blockchain type	Blockchain	Consensus mechanism	Ave. tx validation time
Non-PSC-supported blockchain	Bitcoin	Hash-based proof-of-work (PoW) protocol	≈ 60 mins
PSC-supported blockchain	Ethereum	Modified "Greedy Heaviest Observed Subtree" (GHOST) protocol [7]	≈ 3 mins
	EOS	Asynchronous Byzantine Fault Tolerance (aBFT) protocol [5]	< 1 second
	Stellar [18]	Federated Byzantine Agreement (FBA) protocol [18]	< 5 seconds
	Cardano [25]	Ouroboros Proof-of-Stake (PoS) protocol [25]	< 5 mins
	NEO [11]	Delegated Byzantine Fault Tolerant (dBFT) protocol [11]	≈ 15 seconds

smart contract functionalities. A smart contract model typically consists of program code (run on the blockchain), a storage file (stored on the blockchain), and an account balance (recorded on the blockchain). A user can deploy a smart contract by posting a transaction to the blockchain. A user can send a message (via a transaction) to a smart contract to trigger its function execution. All content of the blockchain, smart contracts, and transactions is publicly visible. The smart contract can partially fulfill the role of a *trusted third party*. After auditing from involved users and validating on the PSC-supported blockchain, the smart contract code is immutable, and all code executes exactly according to how it was programmed. Hence, the smart contract can support the program-controlled fund transfer. The fund managed by the smart contract is represented in the form of *token*. For example, Ethereum blockchain uses ETH token and EOSIO blockchain uses EOS token.

As shown in Table 1, *different blockchains exploit different consensus mechanisms, resulting in the different time needed to validate a transaction.* Compared with Bitcoin, many recently developed PSC-supported blockchains have a shorter transaction validation time. For instance, as analyzed in [12], Ethereum needs about 12 confirmations (about 3 mins) to achieve a similar degree of security as 6 confirmations (about 60 mins) on Bitcoin blockchain.

3 THREAT MODEL AND ASSUMPTIONS

Threat Model. The security threat mainly comes from the payer or the payee. (1) **Payer.** The payer may double-spend the Bitcoin and hence no Bitcoin payment will be received by the payee. Additionally, the payer attempts to win the payment arbitration to avoid releasing the security deposit to the payee. If the payer successfully double-spends the Bitcoin and wins the arbitration, then the payee loses the commodities/services without receiving any payment. This attack is a double-spending attack. (2) **Payee.** The payee is considered as a *semi-benign* entity. The payee may still raise a dispute and hope to win the arbitration even if (s)he has received the Bitcoin payment. If the payee wins the arbitration, then (s)he can receive payments twice. This attack is a double-payment attack. We do not consider the risk that the payee refuses to deliver commodities/services to the payer even if the payer correctly finishes the payment phase required by BFastPay because such risk exists in any payment method. Therefore, handling such risk is out of the scope of this paper. We note that this problem has been studied in [23].

Assumptions. We have the following assumptions. (1) Both the payer and payee are rational and they would defend for their own interests (e.g., the payer/payee would take actions to thwart any double-payment/double-spending attempt). (2) Both the payer and the payee can control a portion of the global Bitcoin hash power to launch attacks. However, either payer or payee cannot control more than 50% of the global hash power for Bitcoin mining. We assume that the remaining hash power is controlled by honest miners, who work together to extend the longest Bitcoin blockchain as stipulated by the Bitcoin protocol. (3) The smart contract platforms adopted by BFastPay are secure. We admit that some existing smart contract platforms have security vulnerabilities. However, developers have devoted great efforts to fix the known vulnerabilities and bring them into real-world applications. (4) Both the payer and payee have fairly reliable Internet connections during BFastPay service.

4 BFASTPAY OVERVIEW

In this section, we first briefly describe BFastPay flowchart and then clarify how to use the security deposit in BFastPay.

4.1 BFastPay Flowchart

Figure 1 depicts BFastPay flowchart, which consists of two phases: the payment phase and the arbitration phase.

Payment Phase. There are three steps (steps 1-3 in Figure 1) in this phase.

(1) Before using the BFastPay service, the payer agent should first escrow sufficient security deposit to BFPayArbitrator which is deployed on a PSC-supported blockchain. The security deposit is added to BFPayArbitrator via a transaction sending from the payer's PSC-supported blockchain account address `Payer_addr` to BFPayArbitrator. Then, the payer agent can send a Bitcoin transaction to the payee.
(2) While the payer agent broadcasts the Bitcoin transaction to the Bitcoin network, the payer agent simultaneously submits a BFPayReq message to BFPayArbitrator. The BFPayReq message consists of (1) the Bitcoin payment information, (2) the Bitcoin blockchain information, (3) the PSC-supported blockchain information, and (4) the transaction amount. Table 2 summaries all of the information carried in the BFPayReq message. Note that the payer needs to refer to the public sources (e.g., [3]) to get the real-time conversion rate between bitcoin and the PSC-supported blockchain token to compute the amount of token with a matching value. BFastPay uses the conversion rate when the Bitcoin transaction occurs. Future conversion rate fluctuations do not affect the fairness of the transaction.

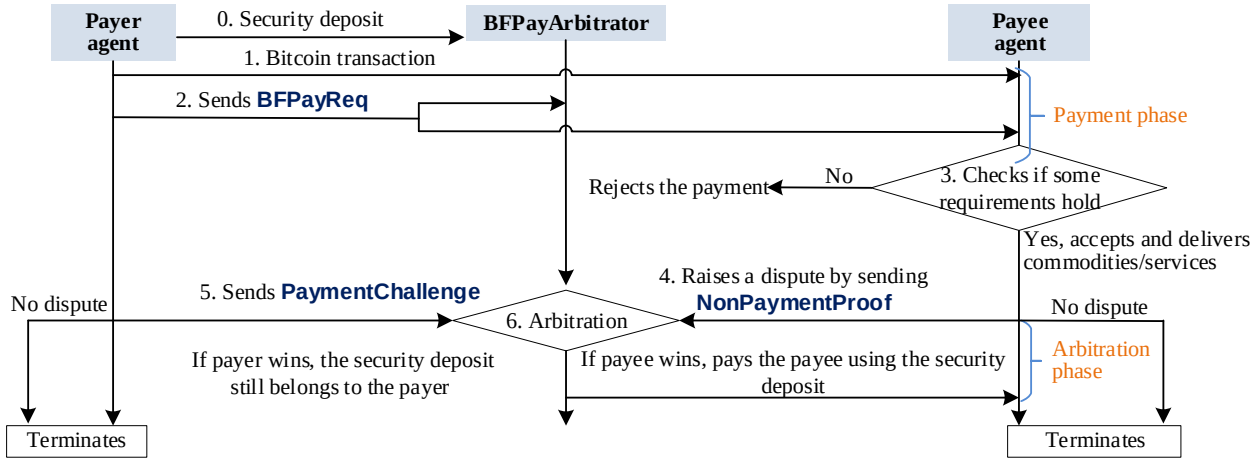

Figure 1: The flowchart of BFastPay.

(3) The payee agent receives the BFPayReq message and checks if the following two requirements hold. (1) Each field of the BFPayReq message is correct[1]. (2) The payer's available security deposit should not be less than the escrowed Bitcoin transaction amount[2]. If the payee agent finds any field in the BFPayReq message is incorrect or the security deposit is insufficient, the payee rejects the Bitcoin transaction. Otherwise, the payee waits for BFPayReq to be validated by the PSC-supported blockchain before accepting the Bitcoin transaction and delivering commodities/services to the payer. The waiting time is thus reduced to be the time needed to validate a transaction on the PSC-supported blockchain.

Arbitration Phase. If the payee agent successfully receives the Bitcoin payment later and does not raise a dispute during a pre-defined arbitration time window, then the BFastPay service life cycle finishes and terminates. Otherwise, BFastPay allows the payee agent to raise a dispute by sending NonPaymentProof to BFPayArbitrator. Then, the BFastPay arbitration phase is activated. ***Note that there are no attacks in the vast majority of real-world Bitcoin payment cases, so the arbitration phase is rarely activated.*** There are three steps (steps 4-6 in Figure 1) in the arbitration phase.

(1) To raise a dispute, the payee agent needs to generate NonPaymentProof and submits it to BFPayArbitrator for arbitration within a pre-defined arbitration time window.

(2) The payer agent examines BFPayArbitrator to check if there is a NonPaymentProof message received by BFPayArbitrator during the arbitration time window. If the payer finds that NonPaymentProof message is received, BFPayArbitrator allows the payer to generate PaymentChallenge and to send it to BFPayArbitrator within a pre-defined rebuttal time window.

(3) BFPayArbitrator arbitrates the dispute based on the received NonPaymentProof and PaymentChallenge. If the payee wins the dispute, BFPayArbitrator pays the payee using the payer's security deposit. Otherwise, the security deposit still belongs to the payer. After the arbitration process, the BFastPay service finishes and terminates.

The detailed arbitration mechanism is further described in Section 5. Note that no manual operations are needed in using BFastPay. The payer agent and payee agent (e.g., smartphone) run BFastPay software to automatically finish the required operations. The software modules of BFastPay are introduced in Section 7.1.

Table 2: The information in BFPayReq message.

Element type	Field name	Description
Payment info	`BTC_TxID`	The Bitcoin transaction ID
	`BTC_Tx_time`	The Bitcoin transaction time
Blockchain info	`BTC_diff_target`	Current Bitcoin network difficulty target
	`Block_hash`	The hash of the latest Bitcoin block header that does not include the escrowed Bitcoin tx
PSC-supported blockchain info	`Payer_addr`	The payer's PSC-supported blockchain account address
	`Payee_addr`	The payee's PSC-supported blockchain account address
Transaction amount	`Token_amount`	The amount of token needs to transfer to the payee if a double-spending attack occurs

4.2 Security Deposit Clarification

The security deposit in BFPayArbitrator has two states: free and frozen. Consider that the payer has a security deposit worth S_1 dollars and a BFastPay Bitcoin payment has a transaction amount

[1]To check the correctness of BFPayReq message, the payee can refer to the trust sources and make a field-to-filed comparison. More specifically, the payment information (`BTC_TxID`, `BTC_Tx_time`) and blockchain information (`BTC_diff_target`, `Block_hash`) is public on the Bitcoin Blockchain. The correct PSC-supported blockchain information (`Payer_addr`, `Payee_addr`) is also known by the payee. In addition, the transaction amount (`Token_amount`) can be computed by the payee because the real-time conversion rate is publicly accessible by the payee.

[2]The adopted check mechanism is exactly the same as the mechanism used in the cash-based payments: after the payer gives cash to the payee, the payee must check and ensure that (1) it is not the fake cash and (2) the amount of cash is sufficient before accepting it.

worth S_2 dollars, where $S_1 > S_2$. Then, during the service life cycle of BFastPay, BFPayArbitrator freezes S_2 dollars. The remaining $(S_1 - S_2)$ dollars are free. After the termination of a BFastPay fast payment service, if the frozen security deposit does not release to the payee, then it will be free again. The free deposit can be used for concurrent or future BFastPay fast payment services. Therefore, if both parties are honest (the vast majority of cases), the payer can enjoy a one-time deposit and permanent BFastPay fast payment services. *Note that the payer can withdraw free security deposit to his own account at any time.*

5 BFASTPAY ARBITRATION

In this section, we describe (1) how to design NonPaymentProof and PaymentChallenge, (2) how to check the validity of NonPaymentProof and PaymentChallenge, and (3) the detailed PoW-based arbitration mechanism used in BFPayArbitrator.

5.1 NonPaymentProof Design and Validation

In an arbitration, to convince BFPayArbitrator that the escrowed Bitcoin transaction is not included in the Bitcoin blockchain, the payee needs to submit NonPaymentProof to BFPayArbitrator. NonPaymentProof contains a number of linked block headers (named *block header proof*). The Block header proof here is used to prove that sufficient PoW has been done to extend the Bitcoin blockchain, in which the escrowed Bitcoin transaction is not included. Figure 2 illustrates the structure of the block header proof. The number of the linked block headers in the block header proof is defined as the length of NonPaymentProof (denoted as n_1).

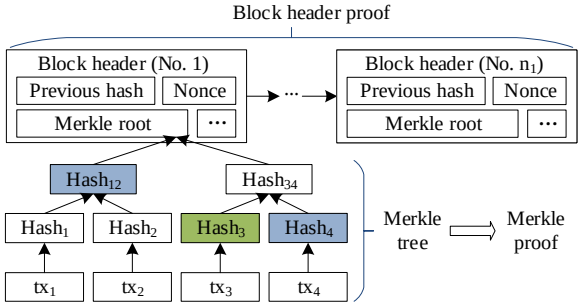

Figure 2: The structures of the block header proof and the Merkle proof.

A valid NonPaymentProof should satisfy the following four requirements (R1)-(R4).

- (**R1**) The block headers meet the current Bitcoin difficulty target: Hash(block header No. i) $\leq$ BTC_diff_target, for $i = 1, \cdots, n_1$.
- (**R2**) The block headers indeed form a linked blockchain: Hash(block header No. i) = the previous hash field in block header No. $i+1$, for $i = 1, \cdots, n_1 - 1$.
- (**R3**) The block header chain indeed extends from the latest Bitcoin block when the escrowed Bitcoin transaction occurs: the previous hash field in the first block header equals Block_hash.
- (**R4**) The escrowed Bitcoin transaction is not included in the first block header in NonPaymentProof.

On receipt of NonPaymentProof, BFPayArbitrator can check whether the requirements (R1)-(R3) are satisfied or not, but it cannot check whether the fourth requirement (R4) is satisfied or not (it is hard to prove non-inclusion of a transaction. For example, the classic *Merkle proof* [28] can prove inclusion but not non-inclusion). To prevent the payee from submitting NonPaymentProof which does not satisfy the requirement (R4), BFastPay allows the payer to reveal such cheating behavior of the payee by submitting PaymentChallenge (by using the Merkle proof) to prove that the escrowed Bitcoin transaction is included in the first block header of NonPaymentProof. If there is no such PaymentChallenge received and the requirements (R1)-(R3) are satisfied, then NonPaymentProof is treated to be valid. Otherwise, it is invalid.

5.2 PaymentChallenge Design and Validation

On receipt of NonPaymentProof, BFPayArbitrator checks if one of the requirements from (R1)-(R3) is not satisfied. If yes, BFPayArbitrator directly lets the payer win the arbitration without any requirements for PaymentChallenge. If no, the payer needs to submit PaymentChallenge to BFPayArbitrator. There are two cases as described below.

Case 1: NonPaymentProof Satisfies Requirement (R4). In this case, PaymentChallenge consists of two components: the block header proof and the Merkle proof. (1) *Block Header Proof.* The block header proof in PaymentChallenge is used to prove that sufficient PoW has been done to extend the Bitcoin blockchain. Figure 2 depicts the structure of a block header proof. The number of the linked block headers in the block header proof is defined as the length of PaymentChallenge (denoted by n_2). A valid block header proof should satisfy the requirements (R1)-(R3) as stated in Section 5.1. (2) *Merkle Proof.* The Merkle proof is used to prove that the escrowed Bitcoin transaction is indeed included in the first block header of the block header proof. The Merkle proof is generated from a Merkle tree, as shown in Figure 2. The Merkle tree has a number of leaf nodes at the bottom of the tree containing hashes of each transaction in a Bitcoin block. An intermediate node in the tree is the hash of its two children. Finally, a single root node (called a Merkle root) can be obtained. How to generate and validate the Merkle proof can be illustrated by the following example. As shown in Figure 2, in order to validate the inclusion of $\mathtt{tx}_3$, the payer only needs to provide a Merkle proof, which consists of two parts: (1) the Merkle branch [$\mathtt{Hash}_4$, $\mathtt{Hash}_{12}$], and (2) the branch position [right, left]. To validate the Merkle proof, BFPayArbitrator computes

$$\mathtt{Hash}_3 = \mathtt{Hash}(\mathtt{tx}_3),$$
$$\mathtt{Hash}_{34} = \mathtt{Hash}(\mathtt{Hash}_3 || \mathtt{Hash}_4), (\mathtt{Hash}_4 \text{ on } \mathtt{right}),$$
$$\mathtt{Hash}_{1234} = \mathtt{Hash}(\mathtt{Hash}_{12} || \mathtt{hash}_{34}), (\mathtt{Hash}_{12} \text{ on } \mathtt{left}),$$

where || represents concatenation. If $\mathtt{Hash}_{1234}$ equals the Merkle root of the first block header, then $\mathtt{tx}_3$ is indeed included in the block header. If the Merkle proof in PaymentChallenge can successfully prove the inclusion of the escrowed Bitcoin transaction, then it is valid. Otherwise, the Merkle proof is considered to be invalid. Note that PaymentChallenge is valid if both its block header proof and Merkle proof are valid. Otherwise, it is invalid.

Case 2: NonPaymentProof Does not Satisfy Requirement (R4). In this case, PaymentChallenge only contains the Merkle proof, which is submitted to BFPayArbitrator to prove that NonPaymentProof does not satisfy requirement (R4). In other words, the Merkle proof is used to prove that the escrowed Bitcoin transaction is indeed included in the first block header of NonPaymentProof. The inclusion-proof process is exactly the same as described above. If the Merkle proof successfully proves that NonPaymentProof does not satisfy the requirement (R4), then PaymentChallenge is valid. Otherwise, it is invalid.

5.3 PoW-based Arbitration Mechanism

We next introduce the arbitration window and rebuttal time window settings. Then, we describe the PoW-based arbitration mechanism. Last, the strategy which can ensure the honest party to win is illustrated.

Arbitration and Rebuttal Time Window Settings. Let T_c denote the elapsed time since the Bitcoin transaction is broadcast to the Bitcoin network. The arbitration time window is set to be $[T_c - \epsilon, T_c]$ and the rebuttal time window is set to be $[T_c, T_c + \epsilon]$. We set ϵ to be 5 mins in BFastPay. To simplify our theoretical analysis later, we treat that both the payee and the payer submit their evidence (i.e., NonPaymentProof and PaymentChallenge) for arbitration at the same time point T_c. The adjustable parameter T_c is also called the mining *competition time period*.

Arbitration Mechanism. The key mechanism in the payment arbitration is: ***If both NonPaymentProof and PaymentChallenge are valid, the winner is the party who submits a block header proof that carries more PoW.*** The precise and complete logic of the arbitration mechanism is illustrated as follows.

- Case 1 (NonPaymentProof does not satisfy the requirements (R1)-(R3)): the payer wins.
- Case 2 (NonPaymentProof satisfies the requirements (R1)-(R3) but no PaymentChallenge is received): the payee wins.
- Case 3 (NonPaymentProof satisfies the requirements (R1)-(R3) and PaymentChallenge with only Merkle proof is received): if PaymentChallenge is valid, then the payer wins. Otherwise, the payee wins.
- Case 4 (NonPaymentProof satisfies the requirements (R1)-(R3) and PaymentChallenge with both the block header proof and the Merkle proof is received): if PaymentChallenge is invalid, then the payee wins. Otherwise, the winner is the party who submits a block header proof that carries more PoW. In the case of a tie, BFPayArbitrator lets the payee win. Mathematically, recall that the block header lengths of NonPaymentProof and PaymentChalleng are denoted as n_1 and n_2, respectively. A simplified description is: if $n_1 \geq n_2$, then the payee wins; otherwise, the payer wins.

Strategy of the Honest Party. To ensure the honest party to win the arbitration, BFastPay leverages the following strategy. There are two cases.

- Case 1 (the escrowed Bitcoin transaction is not included in the Bitcoin blockchain): in this case, the payee is the honest party who can directly truncate a segment of the block header chain from the Bitcoin blockchain to generate a long valid NonPaymentProof (the truncated block header chain segment starts with the block header in which the previous hash field equals `Block_hash` (see Table 2) and ends with the latest block header). If the payer adopts the same strategy as the payee, the payer cannot get a valid PaymentChallenge simply because the escrowed Bitcoin transaction is not included in the Bitcoin blockchain so that a valid Merkle proof of inclusion cannot be generated by the payer. Therefore, the dishonest party (i.e., the payer) has to fabricate PaymentChallenge.
- Case 2 (the escrowed Bitcoin transaction is included in the Bitcoin blockchain): in this case, the payer is the honest party who can directly truncate a segment of block header chain from the Bitcoin blockchain to generate a long valid PaymentChallenge, whereas the dishonest party (i.e., the payee) has to fabricate NonPaymentProof.

Why the Honest Party can Win? The honest miners always work to extend the Bitcoin blockchain from which the honest party can truncate a segment of block header chain to generate a long valid NonPaymentProof (in case the payee is honest) or a long valid PaymentChallenge (in case the payer is honest). Hence, by extending the Bitcoin blockchain, the honest miners actually help the honest party to generate what is desirable (i.e., either NonPaymentProof or PaymentChallenge). Recall the above PoW-based arbitration rule, the dishonest party has to defeat the honest miners in the block generation (i.e., mining) race in order to win. In a nutshell, the reason why the honest party can win the arbitration is: ***Because honest miners (with a large portion of hash power) always help the honest party, the dishonest party (with a small portion of hash power) is hard to win the Bitcoin block generating race (i.e., mining race) in the long run.*** Note that the miners help the honest party *unconsciously and automatically*. The coordination with honest miners is never required. The detailed security analysis is presented in Section 6.

Handling Transaction Delay. The above design is based on the fact that the escrowed Bitcoin transaction sets a sufficient transaction fee to ensure that it is mined in the very first block after it is broadcasted to the Bitcoin network. For a Bitcoin transactions with a low transaction fee, it may wait for several blocks to be included in the Bitcoin blockchain, resulting in the transaction delay issue. Theoretically, there is no transaction delay issue if the transaction fee is set to be sufficient high because the mining priority of miners is determined by the transaction fee.

The following method can be used to handle the transaction delay issue. Consider that the transaction fee is too low and the escrowed Bitcoin transaction is mined after n' blocks since it is broadcasted. If BFPayArbitrator receives NonPaymentProof from the payee, then the payer can send a PaymentChallenge in the same way. The only difference is that the length of PaymentChallenge is counted as $n_1 - n'$, which means that the first n' blocks are not counted in the PoW. As mentioned above, the miners automatically help the payer to extend the length of PaymentChallenge, so the payer can still win the arbitration in the long run. In this case, BFastPay just needs to increase the parameter T_c to ensure the payer to win.

6 SECURITY ANALYSIS

In this section, we first analyze how BFastPay defends the hash-based double-spending attack and double-payment attack. Then, we analyze how BFastPay defends some other possible attacks.

Zero-Sum Game. An attack in BFastPay is a zero-sum game between the payer and the payee. In a double-spending or a double-payment attack, the gain or loss of one party is exactly balanced by the loss or gain of the other party. Therefore, the payer and the payee have a conflict of interests and no collusion attacks are possible. When one party attempts to launch an attack, the other party will try to prevent the attack to defend for his/her own interests.

Mathematical Notations. The payer-controlled and the payee-controlled hash power are represented as αH and βH, respectively, where H is the global hash power for mining Bitcoin. The remaining hash power (denoted as γH) is controlled by the honest miners. It follows that $\alpha + \beta + \gamma = 1$.

6.1 Defending Double-spending Attack

In the double-spending attack, the adversary is the payer who has successfully double spent the Bitcoin and tries to win the payment arbitration. The payee is the defender, who attempts to win the arbitration and receive the payment from BFPayArbitrator.

Payer (Adversary). To maximize the probability of winning the arbitration, the payer needs to submit a PaymentChallenge as long as possible. The canonical Bitcoin blockchain extended by the honest miners cannot be used to generate a valid PaymentChallenge since the escrowed Bitcoin transaction is not included. This means that the payer has to fabricate a valid PaymentChallenge by investing his/her controlled hash power. Therefore, PaymentChallenge is generated with hash power αH.

Payee (Defender). To maximize the probability of winning the arbitration, the payee needs to submit a valid NonPaymentProof as long as possible. Note that the honest miners will extend the canonical Bitcoin blockchain and a valid NonPaymentProof can be directly generated from the canonical Bitcoin blockchain. Therefore, the payee does not need to generate NonPaymentProof by only relying on his/her controlled hash power. To get a longer NonPaymentProof at the arbitration time, the payee should invest his/her controlled hash power to work together with the honest miners to extend the canonical Bitcoin blockchain. Therefore, NonPaymentProof is generated with hash power $(\beta + \gamma)H = (1 - \alpha)H$.

Attack Success Rate Analysis. The attack success rate is equivalent to the probability that the adversary successfully fabricates an alternative chain (mined with αH hash power) longer than the honest Bitcoin blockchain (mined with $(\beta + \gamma)H$ hash power) in the competition time period T_c. Because $\alpha < \beta + \gamma$, the probability decreases exponentially with an increasing T_c. We now analyze the probability that the adversary (with αH hash power) wins the mining race against the miners (with $(1 - \alpha)H$ hash power) in T_c time. The number of blocks expected to be mined by the Bitcoin miners in a certain time can be modeled as *Poisson distribution* [33]. Suppose that the honest miners produce one block per 10 mins on average, the probability of mining exactly k_1 blocks in T_c mins is given by

$$P_1(k_1, T_c) = e^{-\frac{T_c}{10}} \frac{(\frac{T_c}{10})^{k_1}}{k_1!}, \tag{1}$$

where $e = 2.71828\cdots$ is the base of the natural logarithm. Likewise, the probability of mining exactly k_2 blocks in T_c mins for the adversary is given by

$$P_2(k_2, T_c) = e^{-\frac{\alpha T_c}{10(1-\alpha)}} \frac{(\frac{\alpha T_c}{10(1-\alpha)})^{k_2}}{k_2!}. \tag{2}$$

The double-spending attack occurs if the adversary produces an alternative blockchain longer than the honest blockchain. Table 3 enumerates all the cases for double-spending attacks and their probability. By summing up all cases, the probability of double-spending attack P_{ds} can be computed by

$$P_{ds} = \sum_{k_1=0}^{+\infty} [P_1(k_1, T_c) \sum_{k_2=k_1+1}^{+\infty} P_2(k_2, T_c)], \tag{3}$$

where $P_1(k_1, T_c)$ and $P_2(k_2, T_c)$ are defined in Equation (1) and Equation (2), respectively.

Table 3: Double-spending attacks cases and their probabilities.

Case num.	Case description	Double-spending probability
0	honest miners mine 0 block; adversary mines ≥1 blocks	$P_1(0, T_c) \sum_{k_2=1}^{+\infty} P_2(k_2, T_c)$
1	honest miners mine 1 block; adversary mines ≥2 blocks	$P_1(1, T_c) \sum_{k_2=2}^{+\infty} P_2(k_2, T_c)$
2	honest miners mine 2 block; adversary mines ≥3 blocks	$P_1(2, T_c) \sum_{k_2=3}^{+\infty} P_2(k_2, T_c)$
...	...	...

Figure 3 plots the probability of double-spending P_{ds} as a function of T_c when $\alpha = \beta = 0.1$, $\gamma = 0.8$. It shows that P_{ds} is decreasing exponentially with an increasing T_c. Accordingly, the double-spending probability P_{ds} can be reduced to sufficiently small by increasing the parameter T_c in BFastPay. For example, if $T_c = 95$ mins, then it holds that $P_{ds} = 0.096\% < 0.1\%$.

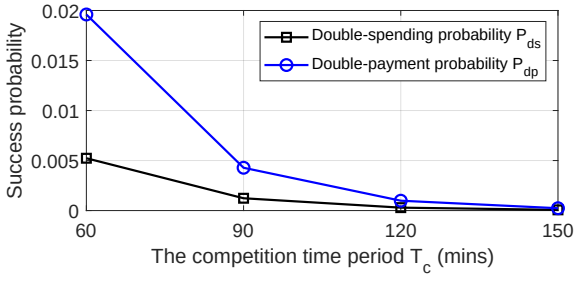

Figure 3: The success probability of double-spending/payment as a function of T_c ($\alpha = \beta = 0.1$, $\gamma = 0.8$).

6.2 Defending Double-payment Attack

In the double-payment attack, the payee is the adversary who has successfully received the Bitcoin payment and tries to receive a second payment from BFPayArbitrator. The payer is the defender, who wants to win the arbitration and prevent the payee from receiving a second payment from BFPayArbitrator. In the double-payment attack, NonPaymentProof is generated with hash power βH and PaymentChallenge is generated with hash power $(\alpha+\gamma)H = (1-\beta)H$. By the similar analysis as P_{ds}, the probability of double-payment attack P_{dp} is given by

$$P_{dp} = \sum_{k_1=0}^{+\infty} [P_1(k_1, T_c) \sum_{k_2=k_1}^{+\infty} P_2(k_2, T_c)], \tag{4}$$

where $P_1(k_1, T_c)$ and $P_2(k_2, T_c)$ are defined in Equation (1) and Equation (2), respectively. The sole difference in computing p_{dp} and p_{ds} is caused by the arbitration mechanism in dealing with the tie: if the lengths of NonPaymentProof and PaymentChallenge equal, then BFPayArbitrator lets the payee win. Figure 3 plots the probability of double-payment P_{dp} as a function of T_c. As expected, P_{dp} is decreasing exponentially with an increasing T_c. Thus, the double-payment probability P_{dp} can be reduced to sufficiently small by increasing the parameter T_c.

6.3 Defending Other Attacks

Fake BFPayReq Attack. In BFastPay, the payer may submit a fake BFPayReq message to BFPayArbitrator to launch a variety of attacks. For example, the payer can send BFPayReq with `Token_amount` = 0 to BFPayArbitrator, where `Token_amount` specifies the amount of token that should be paid to the payee if Bitcoin transaction is not included in the Bitcoin blockchain. This indicates that the payee will receive no payment if the Bitcoin is double spent by the payer. To resist the fake BFPayReq message attack, the payee must check the correctness of BFPayReq message. If BFPayReq message is correct, the payee can accept the Bitcoin payment and deliver commodities/services to the payer. Otherwise, the payee rejects the Bitcoin payment.

Impersonation Attack. An adversary may impersonate either the payer or the payee to launch attacks. For example, the adversary can impersonate the payee to raise a dispute by sending a fake NonPaymentProof to BFPayArbitrator. This impersonation attack can be defended by the access control provided by BFPayArbitrator. BFPayArbitrator stores the account addresses of both payer and payee (see Table 2), so only NonPaymentProof sent from the payee's account address and PaymentChallenge sent from the payer's account address can be accepted by BFPayArbitrator.

Segment Replay Attack. In a segment replay attack, the adversary may replay a segment of Bitcoin blockchain to generate NonPaymentProof or PaymentChallenge. For example, in a payment dispute, the payee may truncate any segment from Bitcoin blockchain to generate a long NonPaymentProof and try to send it to BFPayArbitrator to win the payment arbitration. This attack cannot succeed because NonPaymentProof or PaymentChallenge generated by the above way cannot meet the requirement (R3), so they are invalid. The party who sends an invalid NonPaymentProof or an invalid PaymentChallenge will lose the arbitration immediately.

Pre-mining Attack. The adversary with less hash power is hard to win the arbitration in a long competition time period T_c, but the adversary may fabricate NonPaymentProof or PaymentChallenge ahead of time to make them long enough to win the later arbitration. To defend the pre-mining attacks, BFPayArbitrator requires valid NonPaymentProof and PaymentChallenge to extend from the latest Bitcoin block when the Bitcoin transaction occurs (via checking the requirement (R3)). The latest block hash will be updated whenever a new Bitcoin block is generated, so the block hash (at the time when the escrowed Bitcoin transaction occurs) cannot be known by the adversary in advance. Therefore, the adversary cannot pre-mine NonPaymentProof or PaymentChallenge to launch the attack.

7 EVALUATION OF COST

BFastPay is a general approach that can be deployed on any PSC-supported blockchain platform. We instantiate BFastPay on top of two popular PSC-supported blockchains (i.e., Ethereum and EOSIO) and then we evaluate their operation cost.

7.1 Implementation

BFastPay Modules. Figure 4 shows the modules in BFastPay prototype. The payer agent consists of the BFPayReqGen module and the PaymentChallengeGen module. The payee agent contains the BFPayReqCheck module and the NonPaymentProofGen module. All of the four modules connect to both the Bitcoin network and the Ethereum/EOSIO network to listen/access the needed information. The four modules work as follows. (1) The BFPayReqGen module generates the BFPayReq message and sends it to BFPayArbitrator whenever there are requests from the payer. All of the information in the BFPayReq message is publicly accessible. (2) The EscrowCheck module can automatically check if BFPayReq sent by the payer agent is correct or not by comparing it with the ground truth accessible from the public sources. (3) The NonPaymentProofGen module helps the honest payee to generate NonPaymentProof from the Bitcoin blockchain. (4) The PaymentChallengeGen module generates PaymentChallenge by truncating a segment of the block header chain from Bitcoin blockchain. Some implementation details are skipped due to the lack of space.

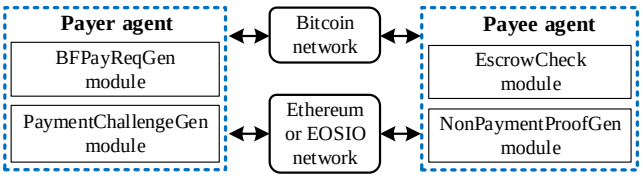

Figure 4: BFastPay modules.

Ethereum and EOSIO-based BFPayArbitrator. Ethereum-based BFPayArbitrator is implemented by Solidity [16, 17]. The browser-based compiler and IDE called Remix [13] is applied to develop BFPayArbitrator. We use the Ethereum test network Rinkeby [14] to test BFPayArbitrator. EOSIO-based BFPayArbitrator is developed by EOSIO C++ in the EOS Contract Development Toolkit (CDT) 1.3.1 [6].

Table 4: Cost for different operations in Ethereum-based BFastPay.

Operations	Pre-operations	Ethereum tx (BFPayReq)	Ethereum tx (NonPaymentProof)	Ethereum tx (PaymentChallenge[1])	Ethereum tx (PaymentChallenge[2])	Ethereum tx (send deposit)
Cost (gas)	2.46×10^6	2.78×10^5	1.04×10^5	1.32×10^5	8.08×10^4	6.60×10^4
Cost fee in ETH	4.9×10^{-3}	5.56×10^{-4}	2.08×10^{-4}	2.54×10^{-4}	1.62×10^{-4}	1.32×10^{-4}
Cost in $ (300$/ETH)	$1.47	$0.167	$0.0624	$0.0762	$0.0486	$0.0396
1: PaymentChallenge contains both the Merkle proof and the block header proof. 2: PaymentChallenge contains only the Merkle proof.						

Table 5: EOS token needed to stake for different operations in EOSIO-based BFastPay.

Operations	Pre-operations	EOSIO tx (BFPayReq)	EOSIO tx (NonPaymentProof)	EOSIO tx (PaymentChallenge[1])	EOSIO tx (PaymentChallenge[2])	EOSIO tx (send deposit)
EOS needed	3.6	0.5	1.1	1.6	0.8	0.1
1: PaymentChallenge contains both the Merkle proof and the block header proof. 2: PaymentChallenge contains only the Merkle proof.						

7.2 Evaluation

In this section, the experiment settings are first introduced. Then, we evaluate the operation cost of Ethereum-based BFastPay and EOSIO-based BFastPay.

Experiment Settings. In the experiments, the competition time T_c is set to be 95 mins (i.e., the security deposit will be frozen for 95 mins and then be free again). If $T_c = 95$ mins, then $P_{ds} = 0.096\% < 0.1\%$ according to Equation (3). That is, the double-spending probability in BFastPay is less than 0.1% in the presence of an adversary with 10% of the global hash power, Therefore, BFastPay achieves a comparable security level as the 6-confirmation-waiting approach against the double-spending attacks[3].

Ethereum-based BFastPay Evaluation. The Ethereum smart contract supported functions have "gas" cost depending on how many computational steps and storage space it requires. The cost is computed as (costed gas)×(gas price). In the experiments, the gas price is set to be 2 gwei, where 1 gwei= 10^{-9} ETH. The cost of Ethereum-based BFastPay may come from five parts: (1) Pre-operations (including deploy contract operation and add security deposit operation), (2) Ethereum tx (send BFPayReq), (3) Ethereum tx (send NonPaymentProof), (4) Ethereum tx (send PaymentChallenge), and (5)Ethereum tx (transfer security deposit to the payee).

Table 4 summarizes the average cost for each part. The fee for pre-operations is a one-time cost, so we do not consider them in calculating the operation cost per fast Bitcoin transaction. There are many use cases of BFastPay, resulting in different operation costs. We consider the two most common use cases to evaluate the operation cost of BFastPay for a Bitcoin transaction.

- **Case 1** (no dispute arises): In this case, the cost only comes from Ethereum tx (send BFPayReq). The operation cost per Bitcoin transaction is 5.56×10^{-4} ETH (or $0.167, 300 $/ETH[4]).
- **Case 2** (dispute arises): In this case, we consider the payee sends out NonPaymentProof and the payer sends out PaymentChallenge. If the payer wins, the cost comes from (1) Ethereum tx (send BFPayReq), (2) Ethereum tx (send NonPaymentProof), and (3) Ethereum tx (send PaymentChallenge). The total operation cost per Bitcoin transaction is 1.02×10^{-3} ETH (or $0.306, 300 $/ETH). If the payee wins, the cost additionally includes (4) Ethereum tx (transfer security deposit). The total operation cost per Bitcoin transaction is 1.12×10^{-3} ETH (or $0.336, 300 $/ETH).

In practice, case 1 (no dispute arises) is more frequent than case 2 (dispute arises) because there are no attacks in the vast majority of real-world Bitcoin payments. In summary, the operation cost of Ethereum-based BFastPay is low.

EOSIO-based BFastPay Evaluation. The EOSIO blockchain allocates blockchain resources based on the amount of EOS token staked. We set the use frequency of BFastPay service to be up to 10 times per day. Table 5 summarizes the EOS token needed to stake for different operations. It shows that the user needs to stake at most 7.7 EOS (or $38.5, 5$/EOS) to use 10 times of BFastPay service per day no matter whether there is a dispute or not during the BFastPay service life cycle. Because the deployer can revoke the smart contract to reclaim the staked EOS token later, we treat the EOSIO-based BFastPay service to be free of charge.

8 COMPARISON

We compare Ethereum-based BFastPay and EOSIO-based BFastPay with the classic escrow-based solution: Lightning Network [31]. Lightning Network is representative because most escrow-based solutions exploit a similar mechanism with Lightning Network. Lightning Network provides users with fast payment services by establishing some off-chain payment channels. Specifically, two parties (e.g., a payer and a payee) first open a secure payment channel by depositing some Bitcoin to a 2-of-2 multi-signature Bitcoin address. The parties interact directly to make payments by adjusting the respective ownership shares of the deposited fund. In cases where no direct payment channel exists between two parties, parties have to rely on intermediate peers to route transactions. The comparison results between BFastPay and Lightning Network are summarized in Table 6. In the table, Lightning Network is called *intra-blockchain escrow* approach because the security deposit is escrowed on the Bitcoin blockchain itself.

Remarks. We have three remarks about Table 6.

[3]Note that the double-spending probability in the 6-confirmation-waiting approach is also less than 0.1% in the presence of an adversary with 10% of the global hash power [28].

[4]The price is from Sep. 2020.

Table 6: The comparison between the intra-blockchain escrow approach (Lightning Network) and the inter-blockchain escrow approach (BFastPay) (●: yes, ◐: partial, ○: no).

Approaches	**Intra-blockchain escrow**	**Inter-blockchain escrow**	
Protocols	Lightning Network	Ethereum-based BFastPay	EOSIO-based BFastPay
Required waiting time	<1 second	≈ 3 mins	<1 second
Extra operation cost	<$0.01/tx	< $0.34/tx	$0/tx
Time required for refuel/reuse security deposit	60 mins	95 mins	95 mins
Decentralization is preserved?	◐	●	◐
Mainly using Bitcoin?	●	●	●
No requirements on the payment channel?	○	●	●
Routing-free?	○	●	●
Not only support micropayments?	○	●	●

(1) Both Lightning Network and EOSIO-based BFastPay reduce decentralization. Because Lightning Network introduces payment hubs [9] and EOSIO is a permissioned blockchain in which the miners need permission to join [5].
(2) Both Lightning Network and EOSIO-based BFastPay can reduce waiting time to be less than 1 second. The key reason is that both of them trade decentralization for fast transaction validation.
(3) Lightning Network directly uses the security deposit for fast transactions. If the security deposit is insufficient, the payer needs to trigger a Bitcoin transaction to *refuel* the security deposit, which takes 60 mins on average. In contrast, BFastPay does not directly use the security deposit for fast transactions. If there are no attacks, the security deposit will be free automatically after 95 mins[5].

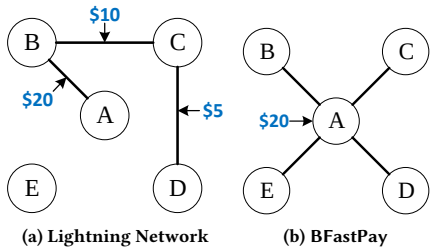

Figure 5: (a) In Lightning Network, the escrowed fund is associated with a payment channel. The maximum allowed transaction amount between A and D does not exceed $\min\{20, 10, 5\} = \$5$. (b) In BFastPay, the escrowed fund is associated with a party. The maximum allowed transaction amount between A and any party is $20.

Advantages. Compared with Lightning Network, BFastPay has several advantages.

(1) **BFastPay has no requirements on the payment channel.** In contrast, Lightning Network has requirements on the payment channel. It requires establishing a payment channel or a routing path between the payer and the payee before using it. Figure 5a shows an example. Because there are no payment routing paths between A and E, the Lightning Network service is unavailable between them.
(2) **BFastPay is routing-free.** In contrast, Lightning Network requires cooperative intermediate peers to help to route transactions. Figure 5 shows an example. In Lightning Network, the escrowed fund is associated with a payment channel between two parties, so a payer can only use the escrowed fund for fast payment for *only one payee* (paying for other payees should rely on routing). However, in BFastPay, the escrowed fund is associated with the payer. The payer can use it to pay for *any party without routing*.
(3) **BFastPay does not only support micropayments.** Lightning Network only supports micropayments, whereas BFastPay can support any payment with an amount less than the escrowed fund. Figure 5 shows an example. Suppose that there are three established payment channels: (1) between the party A and the party B ($20 escrowed), (2) between the party B and the party C ($10 escrowed), and (3) between the party C and the party D ($5 escrowed). If the party A wants to send a fast Bitcoin payment to the party D via the routing path A→B→C→D, then the maximum allowed amount does not exceed $\min\{10, 20, 5\} = \$5$. Generally speaking, the longer the path, the smaller the transaction amount allowed. Thus, Lightning Network suffers from a high probability of routing failure. A recent study [8, 27] shows that ***anyone who uses Lightning Network to transfer Bitcoin over $5.5 has 50% chance of routing failure***.

Disadvantages. There are several disadvantages of BFastPay over Lightning Network. First, BFastPay requires payers to exchange other tokens (e.g., ETH, EOS) to be the security deposit. Moreover, BFastPay requires the payer agent and the payee agent to have access to the Internet for a short period of time (i.e., during the arbitration time window). Last, BFastPay requires a slightly longer time to reuse/refuel the security deposit.

Discussions. We discuss two issues below. (1) **Why not directly use other tokens?** Because Bitcoin has dominated in practical usage [1], our goal is to develop a solution to support fast Bitcoin payment while keeping Bitcoin as the major payment currency. *Note that in the vast majority of cases (both parties are honest and never launch attacks), the payer can enjoy a one-time deposit and permanent BFastPay fast payment services.* Thus, BFastPay is different from

[5]The choice of 95 mins is explained in Section 7.2.

the solution that requires users to exchange Bitcoin to other tokens every time before using the fast payment. (2) **How to mitigate the online requirement issue?** After the payment, the payer agent and payee agent can also delegate the arbitration operations to an always-online cloud server. This solution can mitigate the online requirement issue.

Summary. BFastPay and Lightning Network are competing approaches. Each approach has its appropriate application scenarios. For instance, Lightning Network is more suitable for micropayments, while BFastPay is more suitable for relatively large payments.

9 RELATED WORK

We omit the related work on solutions [21, 22, 24, 30, 31, 35] that support fast Bitcoin payment since they have been covered in the introduction section. In this section, we review two projects that work on inter/cross-blockchain transaction validation: BTC Relay project [2] and Summa project [19]. (1) BTC Relay project lets relayers relay Bitcoin blockchain headers to the Ethereum blockchain. The Bitcoin blockchain headers can be used by Ethereum smart contracts to validate Bitcoin transactions based on simple payment verification (SPV) [28] method. BTC Relay has two limitations. First, because of the lack of relayers, BTC Relay only works intermittently and hence it is not reliable. Second, it is very expensive to store the entire Bitcoin blockchain headers in Ethereum blockchain. Compared with BTC relay, BFastPay is a much more reliable and low-cost solution. (2) The Summa solution validates a Bitcoin transaction in an Ethereum smart contract only checking the total PoW carried by a proof. However, this solution is not sufficiently secure because an adversary with a small portion of hash power can fabricate a long enough block header chain by extending the mining time. In contrast, based on the novel PoW-based arbitration mechanism, BFastPay can support a much higher level of confidence for inter/cross-blockchain Bitcoin transaction validation.

10 CONCLUSION

We propose BFastPay, the first inter-blockchain and routing-free protocol, to support fast payment in Bitcoin network. BFastPay can be built based on any PSC-supported blockchain, therefore, any further improvement on the transaction validation mechanism of the PSC-supported blockchains can directly lead to the improvement of BFastPay. Moreover, we develop a PoW-based arbitration mechanism to enable BFastPay to make fair payment arbitration in a payment dispute. Our comparative study shows that BFastPay and Lightning Network are competing approaches. We hope that our study can stimulate more future research endeavors on this crucial problem in blockchain.

11 ACKNOWLEDGMENT

This work is supported in part by the National Science Foundation under Grants No. CNS-1815636 and CNS-1814551. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors only and do not necessarily reflect those of the National Science Foundation.

REFERENCES

[1] Bitcoin dominance. https://coinmarketcap.com/charts/#dominance-percentage.
[2] Btc relay. http://btcrelay.org/.
[3] Coinmarketcap. https://coinmarketcap.com/.
[4] Confirmation. https://en.bitcoin.it/wiki/Confirmation.
[5] Eos.io white paper. https://github.com/EOSIO/Documentation/blob/master/TechnicalWhitePaper.md.
[6] Eosio.cdt (contract development toolkit) is a suite of tools used to build eosio contracts. https://github.com/EOSIO/eosio.cdt.
[7] Ethereum white paper. https://github.com/ethereum/wiki/wiki/White-Paper.
[8] Lightning strikes, but select hubs dominate network funds. https://diar.co/volume-2-issue-25/.
[9] Mathematical proof that the lightning network cannot be a decentralized bitcoin scaling solution. https://bit.ly/2tg3sxe.
[10] Microsoft adds bitcoin payments for xbox games and mobile content. https://bit.ly/1vIvjLP.
[11] Neo white paper. http://docs.neo.org/en-us/whitepaper.html.
[12] On slow and fast block times. https://blog.ethereum.org/2015/09/14/on-slow-and-fast-block-times/.
[13] Remix. https://github.com/ethereum/remix.
[14] Rinkeby. https://rinkeby.etherscan.io/.
[15] Samsung stores in three baltic states accept cryptocurrencies. https://bit.ly/2LAPFHz.
[16] The solidity contract-oriented programming language. https://github.com/ethereum/solidity.
[17] Solidity documentation. https://solidity.readthedocs.io/en/v0.4.24/.
[18] The stellar consensus protocol: A federated model for internet-level consensus. https://www.stellar.org/papers/stellar-consensus-protocol.pdf.
[19] summa project. https://medium.com/summa-technology/cross-chain-auction-technical-f16710bfe69f.
[20] Bamert, T., Decker, C., Elsen, L., Wattenhofer, R., and Welten, S. Have a snack, pay with bitcoins. In *IEEE International Conference on Peer-to-Peer Computing (P2P)* (2013), pp. 1–5.
[21] Decker, C., and Wattenhofer, R. A fast and scalable payment network with bitcoin duplex micropayment channels. In *Symposium on Self-Stabilizing Systems (SSS)* (2015), pp. 3–18.
[22] Dmitrienko, A., Noack, D., and Yung, M. Secure wallet-assisted offline bitcoin payments with double-spender revocation. In *ACM on Asia Conference on Computer and Communications Security (AsiaCCS)* (2017), pp. 520–531.
[23] Goldfeder, S., Bonneau, J., Gennaro, R., and Narayanan, A. Escrow protocols for cryptocurrencies: How to buy physical goods using bitcoin. In *International Conference on Financial Cryptography and Data Security* (2017), Springer, pp. 321–339.
[24] Karame, G. O., Androulaki, E., and Capkun, S. Double-spending fast payments in bitcoin. In *ACM conference on Computer and communications security (CCS)* (2012), pp. 906–917.
[25] Kiayias, A., Russell, A., David, B., and Oliynykov, R. Ouroboros: A provably secure proof-of-stake blockchain protocol. In *Annual International Cryptology Conference (CRYPTO)* (2017).
[26] Mavroudis, V., Wüst, K., Dhar, A., Kostiainen, K., and Capkun, S. Snappy: Fast on-chain payments with practical collaterals. In *Network and Distributed System Security Symposium (NDSS)* (2020).
[27] Mercan, S., Erdin, E., and Akkaya, K. Improving transaction success rate via smart gateway selection in cryptocurrency payment channel networks. *arXiv preprint arXiv:2003.10877* (2020).
[28] Nakamoto, S. Bitcoin: A peer-to-peer electronic cash system.
[29] Panwar, G., Misra, S., and Vishwanathan, R. Blanc: Blockchain-based anonymous and decentralized credit networks. In *ACM Conference on Data and Application Security and Privacy (CODASPY)* (2019), pp. 339–350.
[30] Pérez-Solà, C., Delgado-Segura, S., Navarro-Arribas, G., and Herrera-Joancomartí, J. Double-spending prevention for bitcoin zero-confirmation transactions. *IACR Cryptology ePrint Archive* (2017).
[31] Poon, J., and Dryja, T. The bitcoin lightning network: Scalable off-chain instant payments. *https://lightning. network/lightning-network-paper.pdf* (2016).
[32] Roos, S., Moreno-Sanchez, P., Kate, A., and Goldberg, I. Settling payments fast and private: Efficient decentralized routing for path-based transactions. In *Network and Distributed System Security Symposium (NDSS)* (2017).
[33] Rosenfeld, M. Analysis of hashrate-based double spending. *arXiv preprint arXiv:1402.2009* (2014).
[34] Sivaraman, V. *High-efficiency cryptocurrency routing in payment channel networks.* PhD thesis, MIT, 2019.
[35] Takahashi, and Otsuka, A. Secure offline payments in bitcoin. In *International Conference on Financial Cryptography and Data Security (FC)* (2019).