HotGate: Trustless Cross-chain Settlement Protocol

by

Mahyar Daneshpajooh

B.Sc., Sharif University of Technology, 2020

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF APPLIED SCIENCE

in

THE COLLEGE OF GRADUATE STUDIES
(Electrical Engineering)

The University of British Columbia
(Okanagan)

December 2022

The following individuals certify that they have read, and recommend to the Faculty of Graduate and Postdoctoral Studies for acceptance, the thesis entitled:

HotGate: Trustless Cross-chain Settlement Protocol

submitted by **Mahyar Daneshpajooh** in partial fulfillment of the requirements for the degree of **MASTER OF APPLIED SCIENCE**

Examining Committee:

Chen Feng, School of Engineering
Supervisor

Anas Chaaban, School of Engineering
Supervisory Committee Member

Ahmad Al-Dabbagh, School of Engineering
Supervisory Committee Member

Abstract

The concept of blockchain was introduced by Nakamoto in the Bitcoin whitepaper [29]. Bitcoin enabled users to make payments in a fully-decentralized manner. In the past decade, researchers and engineers leveraged the Bitcoin idea and proposed blockchains with different characteristics (such as level of security, privacy, and decentralization), each focusing on a particular area. One way to classify blockchains is based on their programmability. A programmable blockchain [17] enables developers to run their programs on top of it. This way, the execution of programs becomes decentralized so no one can manipulate the result. Currently, many Decentralized Application (DAPP) are built on top of programmable blockchains such as exchanges, lending protocols, etc. Bitcoin, although its promising security, doesn't have the programming capability to build dApp on it.

In this work, I introduce and implement HotGate, a trustless cross-chain settlement protocol that enables users to move Bitcoin (BTC) from Bitcoin to other programmable blockchains. So users can leverage their BTC in dApps of other blockchains. Moreover, HotGate enables users to exchange their BTC for assets that exist on these blockchains. HotGate uses Relay [18], a fully-decentralized interoperability solution that connects blockchains directly together. This makes Hotgate a trustless protocol that is aligned with blockchain values. Moreover, HotGate introduces two mechanisms that enable users to move BTC in a fast and private manner, which are the weaknesses of the Bitcoin blockchain.

Lay Summary

Bitcoin is a blockchain that enables users to make payments in a fully-decentralized manner. However, Bitcoin is non-programmable which prevents developers to build any applications on top of it. HotGate is a cross-chain settlement protocol that enables users to move BTC from Bitcoin to other programmable blockchains, so they can leverage decentralized applications built on them. HotGate does this in a fully-decentralized manner where no third party is included. Moreover, since Bitcoin is a slow blockchain that doesn't preserve the privacy of users, HotGate introduces two mechanisms that enable users to move BTC in a fast and private manner.

Preface

All the work presented in this thesis was conducted under the supervision of Prof. Chen Feng. The dissertation is based on research done on blockchain interoperability and decentralized exchange protocols. Chapter 2 is a joint work of Niusha Moshrefi and myself. The rest of the thesis is mostly done and written by me. Chen Feng was involved in revising the system design, incentive design, and choosing the material.

Table of Contents

List of Tables

List of Figures

Glossary

DAPP Decentralized Application

AMM Automated Market Maker

CEX Centralized Exchange

DEX Decentralized Exchange

BTC Bitcoin

EVM Ethereum Virtual Machine

NFT Non-Fungible Token

HTLC Hash Time-Locked Contract

Acknowledgments

It was an incredible experience to do research under the supervision of Chen Feng. I want to thank him for supporting and mentoring me during my studies. I would also like to thank Niusha Moshrefi, who was by my side all along this path.

Chapter 1

Introduction

1.1 Blockchain

Blockchain is a distributed system that maintains a ledger and allows individuals who do not trust each other to agree on the updates to that ledger. The shared ledger data structure is a chain of blocks where each block points to its previous block (see figure 1.1). Each block contains two main parts: block header and transactions. To attach a new block to the blockchain, nodes called miners collect transactions, verify them, create the block and propagate it in the network. Miners often use a peer-to-peer protocol to distribute transactions and blocks. Blockchains are categorized as decentralized systems since there is no centralized authority that miners rely on it. Miners update the state of the ledger by running a consensus algorithm.

Researchers proposed several consensus algorithms [25]. A class of consensus algorithms is fault-tolerant algorithms. These algorithms work correctly even if a portion of nodes are faulty. There are two types of node failures: crash failures and byzantine failures. In the former, a node may stop working if its system crashes. In the latter, a node may behave maliciously in the network (e.g. send conflicting messages to different nodes) [26]. Under the byzantine failure assumption, nodes cannot trust each other.

Before introducing blockchain, byzantine tolerant algorithms were proposed that solve the consensus problem in a permissioned network [20]. A permissioned

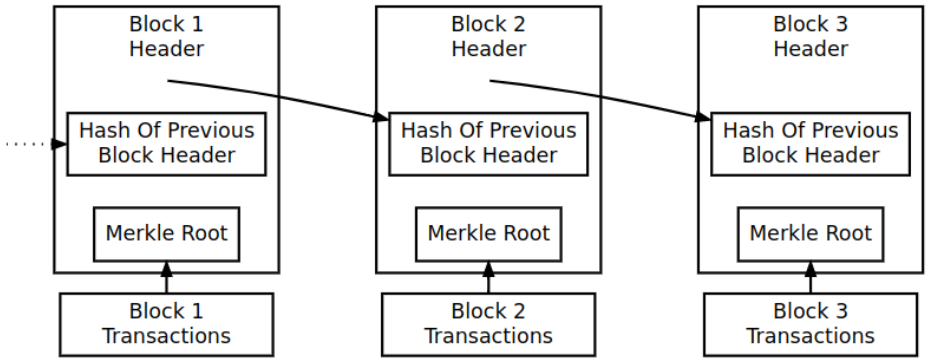

Figure 1.1: Blockchain structure [2]

network is a network where nodes know each other. If a new node wants to join the network, it should get the needed permissions and notice other nodes. The main innovation of blockchain is enabling miners to reach consensus in a permissionless byzantine network where miners can join and leave the network without permission.

What made it possible to reach a consensus in such a network was designing a proper economic incentive mechanism. The incentive mechanism motivates miners to follow the protocol rules to maximize their profit. The resulting decentralized system has several advantages over traditional centralized systems [40]:

- **Immutability**: Blockchain's ledger is updated when nodes reach consensus, so a single node doesn't have the power to rewind the ledger. This is not true for a centralized system where the system admin can modify the ledger without others' permission.
- **Anti-censorship**: No one can prevent a user from interacting with the blockchain. If a miner ignores a user's transactions, the user can send transactions to other miners to include them in the blockchain. In a centralized system, the system admin can censor a user's activity permanently.
- **Resilient System**: Blockchain is maintained by a distributed set of nodes that are independent of each other. This makes blockchain resilient against network attacks since there is no single point of failure. On the other hand, a

centralized system is controlled by a single entity which makes it vulnerable to such attacks.

- **Anonymity**: Users interact with the blockchain without revealing their real identity, so users' privacy is preserved. Anonymity also helps the system to be anti-censorship, i.e, if miners ignore transactions of a specific user, the user can send transactions using a new identity.

The blockchain concept was first introduced in the Bitcoin whitepaper [29] by Satoshi Nakamoto. Satoshi leverages the blockchain to create a decentralized payment network. His main goal was to introduce an alternative to traditional banking systems. In Bitcoin, users can send payment transactions without the need for a centralized entity to facilitate the process.

Bitcoin has a stack-based programming language called Bitcoin script. A user sends money to a receiver by creating a locking script. To spend that money, the receiver should provide a valid unlocking script. Bitcoin script is not a Turing complete language so a limited range of applications can be written on it. In the literature, Bitcoin is categorized as under non-programmable blockchain.

Bitcoin has gained massive popularity over the past decade, and many other cryptocurrencies have been created since then. One notable example is Ethereum [17], a programmable blockchain that introduced the concept of smart contracts. Ethereum extends the idea of bitcoin to a network where computer programs are executed in a decentralized way, i.e, Ethereum offers a decentralized computing network. Solidity is the Ethereum turning-complete programming language that enables users to write any application on top of it. Such applications run on Ethereum Virtual Machine (EVM). Today many other blockchains (Avalanche [32], Fantom [30], etc.) leverage EVM to attract Ethereum's developers to build on them.

Applications deployed on top of a decentralized system are called dApp. The advantage of dApps is that users don't trust the execution environment to run the application correctly. Each blockchain miner runs the application separately to generate the result. Then miners reach a consensus on the new state of the blockchain based on the execution result.

Many application has been built on top of programmable blockchains such as exchanges, lending and borrowing platform, stablecoins, Non-Fungible Token

(NFT), etc. One of the applications that have gained a lot of attention is Decentralized Exchange (DEX) [42]. Currently, notable portions (around %15) of Ethereum transactions are to interact with DEX dApps [11].

1.2 Decentralized Exchange

DEX is an exchange protocol built on top of a programmable blockchain to leverage its decentralized nature. In traditional Centralized Exchange (CEX), the CEX company has full control over the users' assets. Users deposit their assets to their CEX account and then trade them on the CEX servers. However, using DEX, users exchange their assets directly on the blockchain without giving the assets' custody to third parties. DEXs have several advantages over CEXs:

- **Anti-censorship**: A CEX can ban users' trading activities or prevent them from depositing/withdrawing their assets. There are lots of examples where a CEX froze users' assets without even noticing them. However, users are confident that no one can censor their trading activities in DEX since it is running on a blockchain. Users can trade at any time without permission.

- **Transparency**: How an exchange matches the users' orders affects the users' revenue. In a CEX, the CEX owner has the power to change the matching algorithm without informing users. Moreover, the CEX owner has the authority for adding/removing trading pairs. In most cases, these processes are not transparent to the users. DEX's policy for matching orders and adding/removing pairs is hard-coded in the code which is available on the blockchain.

- **Higher Security**: In CEX, users send their assets to the CEX's custodial address. If attackers gain access to the CEX's account, they access all users' assets that are in the custody of the CEX. There are several cases where a CEX was hacked and users lost their assets. In DEX, users have complete control over their assets since they don't transfer their assets to a third party.

The above advantages of DEX make it fairer than CEX, i.e, no one has an advantage over others.

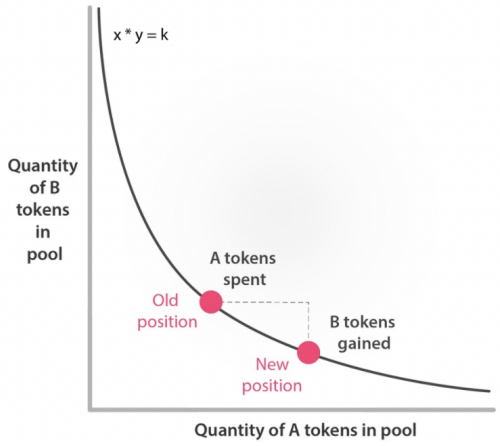

Figure 1.2: Constant product Automated Market Maker (AMM) [1]

The biggest challenge of DEX is its cost. Due to the scalability issues of the existing blockchains, the cost of submitting data and executing programs on them is high. In an order-book exchange, users submit buy-orders and sell-orders, then, the exchange matches these orders. A naive order-book implementation on a blockchain is a costly solution that demotivates users to use the DEX.

Automated Market Maker (AMM) [13] is a solution that tackles the cost problem. This solution has a lower cost in comparison to an order-book DEX. In the AMM DEX, users called market makers deposit their assets in liquidity pools. Then, users exchange their assets through liquidity pools. For example, when a user wants to exchange asset A for asset B, the DEX finds the exchange rate using the liquidity pool of assets A/B and sends asset B to the user. Here the DEX doesn't match buying/selling orders for discovering the exchange rate.

AMM DEXs use different strategies to discover the exchange rate. One of the popular approaches is the constant product strategy. In this strategy, DEX keeps the product of assets' quantities constant in a liquidity pool. Figure 1.2 shows how a constant product AMM works. When a user buys an asset, the quantity of that asset in the liquidity pool decreases. So the price of the bought asset in terms of the sold asset increases.

1.3 Blockhain Interoperability

Bitcoin led to the blockchain revolution with a large number of promising blockchain projects developed in the past decade. These projects have different characteristics (such as underlying consensus algorithms, levels of security, and privacy), each focusing on a particular area. For example, Bitcoin is a non-programmable chain that focuses on payment. Monero [15] and Zcash [37] are payment networks that offer higher levels of privacy. Flow [23] is a blockchain focused on NFT and gaming but it is not decentralized as Bitcoin.

The current state of the blockchain industry is siloed blockchains, preventing the blockchain revolution from reaching its full potential. Therefore, the future success of blockchain heavily lies in interoperability that enables different blockchains to cooperate. Having blockchain interoperability [18], users can move assets and data between blockchains. This also creates the opportunity to build cross-chain dApps, applications that are built on top of multiple chains.

Motivated by this, various solutions have been proposed for blockchain interoperability [16] in the past few years, but each has its own limitation. Below, we review the main solutions of blockchain interoperability and discuss their strengths and weaknesses.

- **Custodians**: Custodians are a set of nodes in charge of verifying data transmission between chains. For example, assume that a user wants to move her assets from blockchain A to blockchain B. First, she sends a request on blockchain A. Then she asks the custodian to confirm her action and gives her assets on blockchain B. Custodians may also provide data of blockchain A on blockchain B so users and apps of blockchain B have access to it. The weakness of the custodian solution is centralization: custodians control the data flow between chain A and chain B. They can censor a user by not verifying her cross-chain data.

- **HTLC**: Hash Time-Locked Contract (HTLC) helps users to exchange their assets between blockchains. This solution cannot be used for moving assets or data between chains. In this solution, a user creates and shares a puzzle with an exchange counterparty. Then both of the users create transactions

using that puzzle. When the puzzle creator reveals the puzzle's solution to claim the exchanged asset, her counterparty finds the solution and uses it to claim the exchanged asset. Although the limited use case, since the exchange is one of the most important blockchain dApps, this solution is categorized as one of the main interoperability solutions.

- **Relay**: Relay [4] is a smart-contract-based solution for the interoperability problem. In this solution, Parties called Relayers submit data from the source chain to the target chain, then, the smart contract on the target chain checks the correctness of the submitted data. Here, the smart contract verifies all cross-chain data, so no one can submit invalid data. This makes Relay as secure as the underlying chain. Also, anyone can become a Relayer and transfer data between chains which makes it fully decentralized. The weakness of this solution is its cost since verifying data on the blockchain is costly.

In general, interoperability solutions have three important aspects:

- **Decentralization**: An interoperability solution can involve trusted parties, or it can completely rely on the underlying blockchains.
- **Cost**: The cost of transferring assets and data is an important factor that motivates users to use a protocol.
- **Universality**: Some solutions move the whole data of one chain to another, and some solutions are limited to a specific application.

Solution	**Decentralization**	**Cost**	**Universality**
Custodian	Centralized	Medium	No
HTLC	Decentralized	Medium	Only swap
Relay	Decentralized	High	Universal

Table 1.1: Interoperability solutions comparison

1.3.1 Related Works

Relay [44] is a universal interoperability solution with the highest level of decentralization, however, the cost of applying it is more than other approaches. What makes Relay a high-cost solution is an on-chain verification. Often, on-chain computation has more cost than off-chain computation since all miners run the same code to generate the final result. However, in off-chain computation, only a limited set of nodes (mostly one node) run the code.

There are a few attempts to reduce the Relay cost. The main idea is to verify part of the data by off-chain entities instead of verifying all the data by smart contracts. The main challenge of these solutions is keeping the protocol trustless since they rely on the off-chain entities' verification, i.e, off-chain entities should not be able to submit the wrong data to the protocol.

The first attempt to solve the Relay cost problem was Testimonium [21]. In this solution, instead of verifying every submitted block header on the Relay smart contract (which is an expensive task), Testimonium optimistically accepts block headers until some complaints arise. In the case of a complaint, the Relay contract verifies the block header validity. If a block header passes a period called the challenge period without getting challenged, the Relay contract accepts it. Although Testimonium reduced the cost, it decreased the Relay speed since a block header should pass the challenge period to consider valid. Another approach to reducing the Relay cost is using zero-knowledge poof [37]. In zkRelay [39], the Relay contract verifies a zero-knowledge proof of the data validity rather than verifying the whole data on-chain. So by using zero-knowledge proofs, most of the computation is done by the off-chain entities and only a tiny part of it is done on-chain. zkRelay protocol can be applied for PoW blockchains. A recent solution called zkBridge [41] extends the idea to other consensus mechanisms such as Proof of Stake.

1.4 Cross-chain Settlement

Currently, multiple blockchains exist in the network but they do not interact with each other properly. By connecting these siloed chains, the range of applications that users can access expands. This connection is necessary for Bitcoin, the leading cryptocurrency in the market that isn't programmable. Currently, Bitcoin users

cannot exchange their BTC in a trustless manner. They need to move BTC to centralized exchanges to do so. Also, they cannot earn interest on top of their BTCs in a decentralized way. Again, their only option is to use centralized entities to do so. Building a fully decentralized cross-chain settlement protocol between Bitcoin and other programmable blockchains unlocks a huge value for Bitcoin users.

Cross-chain settlement protocol [43] aims to solve the above problem. Using such a protocol, users can move or exchange assets between chains [36]. When an asset is moved from one chain (called source chain) to another (called target chain), the user receives an asset with an equal value called a wrapped asset. The wrapped asset is the representation of the native asset on the target chain.

Several examples show the importance of such an application. Suppose that Alice has token A on the source chain. Now, consider the following scenarios:

- Alice wants to pay Bob for her purchase. Bob only accepts token B on the target chain.
- Alice wants to use a target chain application that only accepts token B.
- Alice wants to exchange token A for token B but no DEX exists on the source chain (the source chain is a non-programmable blockchain).
- None of the DEXs of the source chain has the trading pair A-B, however, a DEX on the target chain has this trading pair.
- A DEX on the target chain offers a better rate for the trading pair of A-B in comparison to the source chain's DEX.

In all the above cases, a cross-chain settlement protocol is needed. There are few attempts to build such an application, but the proposed solutions suffer from weaknesses. In the next chapter, we review the main goals of the cross-chain settlement protocol and examine the existing solutions.

1.5 Contributions

In this thesis, I proposed Hotgate, a cross-chain settlement protocol between Bitcoin and EVM blockchains. Using Hotagte, users can transfer and exchange assets between Bitcoin and other blockchains. The main contribution of this work is:

- Proposing a fully-decentralized protocol for wrapping BTC that doesn't rely on third parties.
- Introducing a mechanism for transferring BTC to other chains in a time less than the Bitcoin finalization time.
- Designing the private transfer mechanism which unlinks transferred assets from the original assets to preserve users' privacy.
- Implementing and testing Hotgate between Bitcoin and Ethereum testnets.

Chapter 2

Design Goals

In this chapter, I define the design goals of a cross-chain settlement protocol, a protocol that enables users to move or exchange assets between chains. I also review the existing cross-chain settlement solutions to see their advantages and disadvantages.

2.1 Trustless

A cross-chain settlement protocol works between two blockchains. Since decentralization is the main value of blockchains [40], the cross-chain settlement protocol should be also decentralized.

Being trustless means that the protocol only facilitates moving users' assets between chains, without taking custody of their assets. In custodian solutions, a set of custodians take control of users' assets. So if they get hacked, users will lose their funds. However, in a trustless protocol, users have full control over their funds which makes the protocol way more secure.

Furthermore, users can use a trustless protocol without permission. In custodian solutions, custodians may ban a user by ignoring her requests. However, no one can censor a user's activities in trustless protocols.

Design goal: The cross-chain settlement protocol should not depend on trusted third parties (AKA custodians) for processing users' requests.

2.2 Speed

Speed plays an important role in cross-chain settlement. Users of such a protocol prefer to receive their assets as soon as possible. If the protocol is slow, it may cause a huge loss for users. For example, if the cross-chain exchange takes a long time for a user, the exchange rates may change a lot which makes a huge loss for the user. Another example is an arbitrager who wants to move his assets to another chain to sell and gain revenue. If this process takes a long time, the arbitrage opportunity may be lost.

In cross-chain settlements, two chains are involved in the process: the source chain and the target chain. On each chain, a transaction should be submitted to perform the cross-chain settlement. Therefore, users have to wait at least for the confirmation time of the source blockchain plus the confirmation time of the target chain.

Executing cross-chain requests before they get finalized on the source chain has a risk of transaction reversion, which is a threat to protocol security. For example, assume that a protocol moves BTC from Bitcoin to a target chain before requests get finalized on Bitcoin. An attacker can leverage the protocol to move some BTC to the target chain. Then she forks the Bitcoin blockchain to exclude his request. Here, the attacker received the BTC without sending any request to Bitcoin.

In general, if the source chain or the target chain is slow in confirming transactions, this causes inconveniences for the protocol users. For example, transferring BTC from Bitcoin to Ethereum takes more than one hour due to Bitcoin's slow finality. This delay is not acceptable for a user who needs to get BTC urgently.

Design goal: Moving assets between chains in a time less than the sum of the source chain and target chain finalization time without harming the protocol security.

2.3 Privacy

Privacy is a major concern for many blockchain users. They want to preserve their privacy while interacting with the blockchain. Privacy-preserving is particularly difficult for public blockchains since all the data is recorded on-chain [14]. Although blockchain users use anonymous identities, their real identities

can be leaked. For example, if a Bitcoin user buys a coffee in a coffee shop, the owner of the coffee shop can trace back all the user's transactions recorded on the blockchain.

The privacy in the cross-chain settlement is to unlink users' assets on the source chain from their assets on the target chain. Since blockchains are transparent, it is easy to link users' transactions together, so anyone can find who moved assets from one chain to another. This can reveal some information about users' intentions. For example, if a user has moved assets between chains for arbitrages, moving new assets signals to users that there is an arbitrage opportunity on the target chain. So they can front-run her and capture the revenue.

Design goal: When a user moves assets from the source chain to the target chain, no one should be able to link the moved assets to the received assets.

2.4 User Experience

Cross-chain settlement is a multi-step process that involves two blockchains. Often working with a single chain is a complex process for many users, and introducing another blockchain that needs another wallet to interact with make thing more complex.

Sending transactions on two blockchains requires users to have a wallet with enough native coins on each chain. These requirements make the cross-chain settlement a complex process that is not suitable for regular users.

In centralized protocols, third parties can improve the **UX!** (**UX!**). They can send transactions on behalf of the users so users don't need to interact with different blockchains. However, being trustless is a priority in our design. We are looking for a solution that improves the UX without harming the decentralization of the protocol.

Design goal: Users can execute a cross-chain settlement by only interacting with the source blockchain.

2.5 Existing Solutions

Below, I explain three existing solutions for cross-chain settlement and examine whether they achieve our design goals or not.

CEX-DEX. One possible solution for a cross-chain settlement is using a CEX as an intermediary. In this case, a user sends token A to the CEX, withdraws token A on the target chain, then, exchanges token A for token B using a DEX on the target chain. This solution has four disadvantages:

1. A centralized entity (CEX) is involved in the process, so it is not a trustless solution. A CEX may freeze the users' assets when the user deposits them.

2. It causes a lot of delays: depositing in and withdrawing out of exchange often is a slow process. The exchange may decide to give token A on the target chain with a long delay to the user.

3. This solution doesn't provide privacy for the user since CEX knows who deposited and withdrew in the system.

4. It creates a bad user experience: the user needs to perform three different actions (send, withdraw, exchange), which require users to interact with both chains and have the native tokens of both chains.

Custodian Solutions. Some solutions use a network of custodians to enable cross-chain settlements. On the source chain, users send their assets to the custodians and custodians give them their desired assets on the target blockchain. These solutions are similar to the CEX-DEX solution, however, the custodians' network is more decentralized.

To perform an action in such systems, a consensus should be reached among custodians. So these systems are safe as long as the majority of their custodians are honest. In reality, a custodian solution has a few custodians which makes it vulnerable to collision or getting hacked.

HTLC. As we discussed before, HTLCs [24] are used for exchanging assets between two blockchains. This solution is trustless since no one expects users who are willing to exchange their assets are involved in the process. Before creating HTLCs on blockchains, users need to agree on the exchange rate. Using HTLC for a cross-chain settlement has the below downsides:

1. HTLC is only useful for exchanging assets between chains, however, moving assets between chains are not possible using HTLC.

2. When a user wants to perform a cross-chain exchange for a trading pair, she needs to find another user who wants to perform the reverse cross-chain exchange for the same trading pair.

3. For an HTLC to be completed, users need to wait for the finalization of the transaction on the slower blockchain.

Solution	**Trustless**	**Speed**	**Privacy**	**UX**
CEX-DEX	Centralized	Slow	No	Bad
Custodian	Semi-decentralized	Medium	No	Good
HTLC	Decentralized	Medium	Partial	Medium

Table 2.1: Cross-chain settlement solutions comparison

As we have seen, non of the existing solutions satisfied all the design goals. In the next chapter, I introduce HotGate, a cross-chain settlement protocol that works between Bitcoin and EVM chains.

Chapter 3

System Architecture

In this chapter, we introduce components of HotGate, a cross-chain settlement protocol that works between Bitcoin and EVM-compatible blockchains. Using HotGate, users can move or exchange assets between Bitcoin and an EVM chain. HotGate was designed based on the goals mentioned in the previous chapter.

Although Bitcoin has the biggest market cap among all cryptocurrencies, it doesn't have a Turing complete programming language, so it is not possible to build dApps on it. One of the most-used dApps of programmable blockchains is DEX. HotGate offers a DEX for Bitcoin users who want to exchange their BTC in a decentralized manner.

The main components of HotGate are Relay, asset pools, and network participants. Relay connects Bitcoin to other chains in a trustless manner. Assets pools help users to exchange their assets and get their assets with a short delay. Network participants provide the needed data and help the protocol run smoothly.

3.1 Relay

A bridge between two blockchains connects them. Some bridges only enable moving assets from one blockchain to another, and some of them move the whole data of one blockchain to another. The second type covers the first one.

Relay [27] is an interoperability solution that enables users to access the whole data of the source chain on the target chain. Figure 3.1 shows the summary of this

solution. Parties called Relayers submit block headers of the source chain on the target chain. The Relay contract checks the validity of submitted block headers by checking the consensus mechanism of the source chain. Then, it decides which blocks should be finalized according to the finalization rule of the source chain. Therefore, the Relay smart contract that exists on the target chain, acts as a light client of the source chain. In summary, Relay works in three phases:

- **Data submission**: Relayers submit some data of the source chain on the target chain.
- **Data verification**: Relay contract verifies the correctness of source chain data.
- **Data finalization**: Relay contract decides on the finalization of submitted data so users can access it.

HotGate uses Relay as its interoperability solution. Since the consensus mechanism of the source chain and all the cross-chain data is verified by the Relay contract, the security of the HotGate is equal to the security of its underlying blockchains.

In the case of Proof-of-Work source blockchains such as Bitcoin, the consensus check includes verifying that the submitted block header shows enough work has been done, re-targeting the difficulty has been done correctly, etc. If the submitted header is invalid (i.e. it does not belong to the main fork of the source chain), the Relay contract will reject it.

After adding the new block header, HotGate checks for any previous block header that got finalized. In the case of Bitcoin, this gets done by using the longest chain rule. This means that if a block header gets buried under six valid block headers, Relay considers it finalized.

If a block header is finalized by the HotGate contract, users can refer to it on the target chain for proving the inclusion of some data on the source chain. As block headers include Merkle roots [28] of all transactions and states, HotGate can perform any state or transaction inclusion verification using Merkle proofs generated by users. This means that the target chain has access to all the data of

the source chain. Any dApp on the target chain that wants to read some data from the source chain can leverage the Relay.

In summary, there are two main interactions that one can have with the Relay contract:

- **Block header submission**: Relayers submit block headers of Bitcoin and earn rewards (see 1). If the block headers are not valid the submission will revert, and if the block header won't get finalized in the main Bitcoin chain, the Relayer will not earn the reward.
- **Checking data inclusion**: Any user can query the contract to check the inclusion of some data in a finalized block of the source chain (see 2). HotGate will respond with true (in case the data is included in the finalized block), or false (otherwise).

Procedure 1 SUBMITBLOCKHEADER(*oldHeader, newHeader*)

Require $oldHeader.length == newHeader.length == 80$ bytes
Require $oldHeader.difficultyTarget == newHeader.difficultyTarget$
$oldHash \leftarrow sha2(sha2(oldHeader))$
$newHash \leftarrow sha2(sha2(newHeader))$
Require $oldHeader$ has not been finalized
Checks that $newHeader$ has not been submitted before
Require $oldHash == newHeader.parentHash$
Require $newHash < newHeader.difficultyTarget$
$Blockchain[newHeader.height].push(newHeader)$
for $(i = 0, i < 6, i++)$ **do**
    $parentHash \leftarrow oldHeader.parentHash$
    $oldHeader = findHeader(parentHash, oldHeader.height)$
end for
Runs $pruneBlockchain(oldHeader)$
Sends reward to the Relayer

Relay helps us to move data from Bitcoin to other EVM chains in the most secure and decentralized way. The Relay gives us access to the whole data of the Bitcoin on the EVM chains where the Relay is implemented. For a Relay to work properly, nodes called Relayers should provide the needed input data for it.

Procedure 2 CHECKTXINCLUSION(*transactionId, proof, blockNumber*)

1: **Require** $B[blockNumber][0]$ is finalized
2: Gets the Relay fee
3: $transactionRoot \leftarrow B[blockNumber][0].root$
4: $merkleRoot \leftarrow calculateRoot(transactionId, proof)$
5: **return** $transactionRoot == merkleRoot$

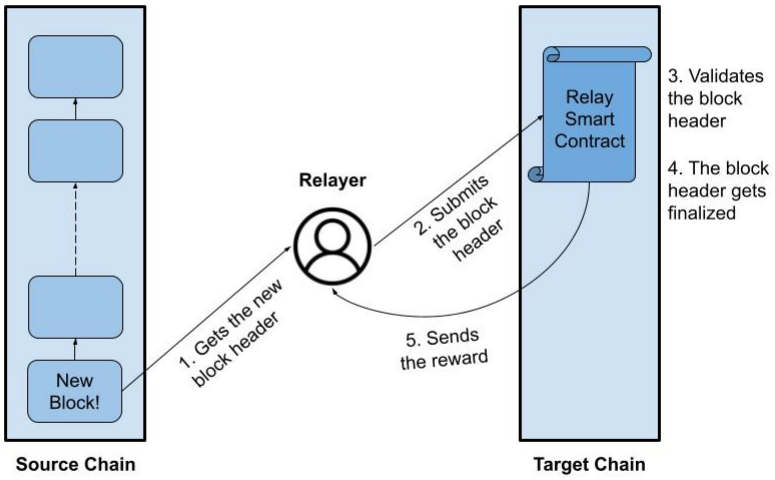

Figure 3.1: How Relay works

3.2 Network Participants

HotGate has different participants who help the protocol run smoothly. These participants are incentivized to participate in the protocol and act honestly. The five main participants of HotGate are Relayer, Keeper, fisherman, liquidity pool provider, and fast pool provider.

Relayers are in charge of transferring data from the source chain to the target chain. Keepers facilitate cross-chain transfer and exchange for users. Fishermen

are responsible for monitoring Keepers' activities. Liquidity pool providers and fast pool providers provide liquidity for exchange pools and fast pools. In the following, we describe each role and its responsibilities.

3.2.1 Relayer

Relayers are nodes that get data from the source chain and submit them to the target chain. These nodes are crucial for HotGate since they provide the input for the Relay contract. In the case of Bitcoin, Relayers provide the contract with Bitcoin's block headers.

To become Relayer, a node runs the Relayer script. This script connects the Relayer to a few full nodes of Bitcoin to get the latest block headers. Relayer script is lightweight so it can be run on a personal computer.

Submitting and verifying block headers on the blockchain costs for Relayers. To compensate them for this cost, Relay takes fees from users and pays them to the Relayers. The fee covers the gas amount used for submitting and verifying the block header, plus %10 extra as a reward for the Relayer.

The Relay contract checks the correctness of the submitted data and sends a reward to the Relayer who provided the valid data faster. If a Relayer provides a false or staled block header, this header gets rejected by the Relay contract and the Relayer loses the paid gas fee. Therefore, a malicious Relayer cannot cause any harm to the protocol.

HotGate's Relayers network is a decentralized network. Anyone can become a Relayer in HotGate and compete to collect rewards. Having a high bandwidth can help a Relayer to get more rewards. It is important to note that in any case, HotGate is secure as the underlying blockchain, and as long as one honest Relayer exists in the system, HotGate has liveness.

3.2.2 Keeper

Keepers are nodes that facilitate users to perform cross-chain transfer and cross-chain exchange requests. They have two main roles in the systems:

- **Submitting requests**: Users send their cross-chain settlement requests on Bitcoin. Keepers collect and submit these requests to the target chain.

- **Asset custody**: No smart contract exists on Bitcoin. So users should send their assets to parties called Keepers. Although Keepers take custody of users' assets, their behaviours are monitored by fishermen so they cannot act maliciously.

In the following, we describe each of these responsibilities.

Submitting Requests Keepers make the HotGate user experience smooth. They collect users' cross-chain requests from Bitcoin and submit them to the target chain. So, users perform cross-chain settlements by only interacting with Bitcoin (the chain they already have assets on it). This eliminates the need of having both chains' native tokens for cross-chain settlements.

Keepers watch Bitcoin for users' cross-chain settlement requests. For requests that Keepers collect, they create proofs of inclusion and submit them to the HotGate contract on the target chain. HotGate contract verifies the validity of the proof using the Relay and then executes the request. Checking request inclusion by Relay makes the system trustless, i.e., Keepers cannot manipulate users' requests. They only move requests between chains.

Submitting a transaction on the target chain and verifying data using Relay cost Keepers. Keepers pay these costs on behalf of users. In return for the service they provide, Keepers take fees from users, which compensates them for the costs (Relay fee and blockchain transaction fee) they have paid plus an extra reward for the Keeper. The Keeper fee is reduced from users' requests automatically.

In HotGate, anyone can become a Keeper. As long as one active Keeper exists in the system, the liveness of the system is guaranteed. There is a competition between active Keepers to collect and submit requests faster to earn more fees. It is also possible for users to move their requests from Bitcoin to the target chain by themselves.

Asset Custody Another responsibility of Keepers is taking custody of users' assets in a fully-decentralized manner. They help users to transfer or exchange their assets between chains. When a user wants to perform a cross-chain settlement, she sends her assets on Bitcoin to a Keeper. Then the user provides proof of locking

her assets on the target chain to complete the cross-chain settlement. Keepers guarantee that every wrapped asset on the target chain has an equal amount of backed BTC on Bitcoin.

Each Keeper has a Bitcoin wallet. All activities of this wallet are controlled by the smart contract on the target chain. This makes the HotGate protocol trustless. A node becomes a Keeper in the system by locking some amount of bond. This bond guarantees the system security, i.e., if the Keeper acts maliciously, the bond will get slashed. The Bitcoin wallet address of Keepers is recorded in the HotGate smart contract so users access it.

Users move their assets from Bitcoin to the target chain by sending their assets to the Keeper's wallet address. After sending, Keepers provide proof of locking users' assets to the HotGate contract. HotGate contract uses Relay to verify the correctness of the proof. If the proof is valid, the HotGate contract mints the wrapped BTC for the user.

Users can also receive BTC by burning their wrapped BTC on the target chain. The HotGate contract records users' burn requests and gives limited time to the Keeper to send BTC to the user. After sending BTC, the Keeper needs to provide proof of it. Otherwise, the Keeper's bond gets slashed.

A Keeper is not allowed to move BTCs that have been sent to the wallet unless a user makes a burning request. This ensures that every wrapped BTC has equally backed BTC, so the price of these assets is equal. If a Keeper steals BTC, the Keeper's bonds will be slashed.

In summary, Keepers are responsible for keeping the users' assets safe and responding to them quickly. The above mechanisms ensure that the Keeper behaves appropriately and makes the HotGate protocol trustless.

Keepers' bonds play a crucial role in aligning Keepers' incentives to act responsibly. If the value of the bond becomes lower than the amount of BTC that is held in the Keeper's wallet, the Keeper has the incentive to steal all BTCs. To avoid this, HotGate has a liquidation mechanism, which ensures that Keeper's bonds are always greater than the value held by the wallet.

If the value of the Keeper's bond becomes lower than the wallet balance (because of the price fluctuations), the liquidation mechanism gets activated. Any user can buy Keeper's bond using the wrapped BTC with a discount. Then, the

collected wrapped BTC is burnt in the system and the Keeper will be able to move that amount of BTC from the wallet.

3.2.3 Fisherman

Fishermen monitor the Keepers' activity. If a Keeper acts irresponsibly, a fisherman will call the contract and slashes the Keeper. The incentive of the fisherman is to earn a slashing reward. There are two main scenarios in which a fisherman slashes the Keeper:

- Lazy Keeper: The Keeper has a limited time to send BTC to the user who made a burning request. If the Keeper doesn't execute the request before the determined deadline, a slasher calls the HotGate contract to slash the Keeper. Then, part of the Keeper's bond with a value equal to the user request is sent to the user and a percentage of that value is also sent to the slasher as a reward.
- Thief Keeper: If the Keeper steals BTC from the wallet, a slasher calls the HotGate contract. The slasher provides the transaction in which Keeper has stolen BTCs. HotGate contract checks that the transaction doesn't belong to any burn request and finds out the stolen amount. Then, part of the Keeper's bond is slashed and goes on sale. The goal of this sale is to collect a number of wrapped BTCs equal to the stolen amount. Anyone can participate to buy the bond with a discount, paying in the wrapped BTC. The collected wrapped BTCs are then burnt by the HotGate contract to make sure that every wrapped BTC has equally backed BTC.

Fishermen play a crucial role in the system since they prevent Keepers, parties who have users' assets custody, to act maliciously.

3.2.4 Liquidity pool providers

These parties provide liquidity for the liquidity pools of HotGate. Anyone can become a liquidity pool provider in our system and collect exchange fees from users using that liquidity pool. To provide liquidity for a pool, a user should have both tokens of that pool.

When a liquidity provider provides liquidity for a pool, it receives some amount of token called liquidity pool token which shows her share in that liquidity pool. Whenever the liquidity provider decides, she can remove liquidity from the pool and get the deposited asset plus collected fees.

3.2.5 Fast pool providers

These parties provide liquidity for the Bitcoin fast pool of HotGate. Anyone can become a liquidity pool provider in our system and collect fees from users using the fast pool. To provide liquidity for the fast pool, a user should have wrapped BTC on the target chain.

When a fast provider provides liquidity, it receives some amount of token called a fast pool token which shows the share of her in the pool. Whenever the fast provider decides, she can remove the liquidity from the pool and get the deposited asset plus collected fees.

Becoming a liquidity pool provider or fast pool provider has a different risk/reward profile. The fast pool offers higher income for nodes but has a risk of loss of funds. We will explore this risk in the next chapter.

3.3 Asset Pools

There are two types of asset pools in HotGate which helps users to exchange their assets and also get their assets faster on the target chain. The first is done using liquidity pools, and the second is done using fast pools. Liquidity pool providers provide liquidity for liquidity pools, and fast pool providers provide liquidity for fast pools.

Liquidity pools Liquidity pools are double asset pools that are used for exchanging tokens. In an AMM DEX, the relative price of token A to token B is calculated based on the amount of deposited token A and deposited token B in the liquidity pool of the A-B trading pair. The amounts are calculated based on a constant product AMM scheme.

Liquidity pool providers add liquidity to the liquidity pools to earn exchange fees. In return, they receive liquidity pool tokens that indicate their share of the

total collected fees.

Fast pool This pool enables fast exchange and fast transfer in HotGate. It accepts wrapped BTC as liquidity. Fast pool providers provide liquidity for this pool. They receive fast pool tokens that indicate their share of the total collected fees.

Chapter 4

Main Scenarios

In this chapter, we explain the main functionalities of HotGate:

- **Cross-chain transfer**: A user moves BTC from Bitcoin to the target chain
- **Cross-chain exchange**: A user exchanges her BTC for other assets that exist on the target chain.

4.1 Cross-chain Transfer

In cross-chain transfer, a user who has BTC on Bitcoin wants to move her BTC to the target chain. This can happen for several reasons:

- The user wants to use her BTC on some dApp that exists on the target chain.
- The user wants to pay a merchant that only accepts tokens on the target chain.
- Transaction fees on the target chain are lower than Bitcoin and the user wants to make multiple payments.

The cross-chain transfer has three modes:

- **Normal**: The user will receive BTC on the target chain after her request gets finalized on Bitcoin. This mode is useful for users who are not in hurry to receive their assets.

- **Private**: The user wants to hide the link between her BTC on Bitcoin and wrapped BTC on the target chain. This mode is useful for users who don't want their assets to be easily tracked.

- **Fast**: The user will receive BTC on the target chain before her request gets finalized on Bitcoin. Users who rush to receive BTC on the target chain can take advantage of this mode by paying an extra fee.

In the following, we explore the above modes.

4.1.1 Normal Cross-chain Transfer

To perform a normal cross-chain transfer, a user sends a request that locks BTC on Bitcoin. To lock BTC, the user sends it to the Keeper wallet on the Bitcoin. The user gets the address of the Keeper wallet from the HotGate smart contract on the target chain. Figure 4.1 shows the workflow.

In the request, the user determines the recipient address, Keeper fee, etc. The recipient address is the address of the user on the target chain. Also, the user determines the fee amount she will pay to the Keeper to execute her request. All data is written in the Bitcoin transaction payload. Bitcoin payload is a special type of Bitcoin output that allows users to write arbitrary data on the Bitcoin blockchain.

The Keeper watches the target chain for transactions that send BTC to his wallet and include users' requests. After the request gets finalized on Bitcoin (it receives six confirmations), the Keeper submits it to the HotGate contract. If the Keeper submits a request before it gets finalized, the request will be rejected by the HotGate contract and the Keeper lose the paid transaction fee.

To submit a request, the Keeper generates Merkle proof for it. The Merkel proof shows that the request has been included in a specific block. If the Keeper submits a request that was not included in the given block, the Merkle proof will be rejected by HotGate smart contract, and the Keeper lose the paid transaction fee.

After checking the Merkle proof, the contract extracts the needed data from the request. Also, it finds how the amount of BTC the user has sent to the Keeper address. Then, the contract mints an equal amount of wrapped Bitcoin for the user. Also, it pays the Keeper fee from the minted BTC.

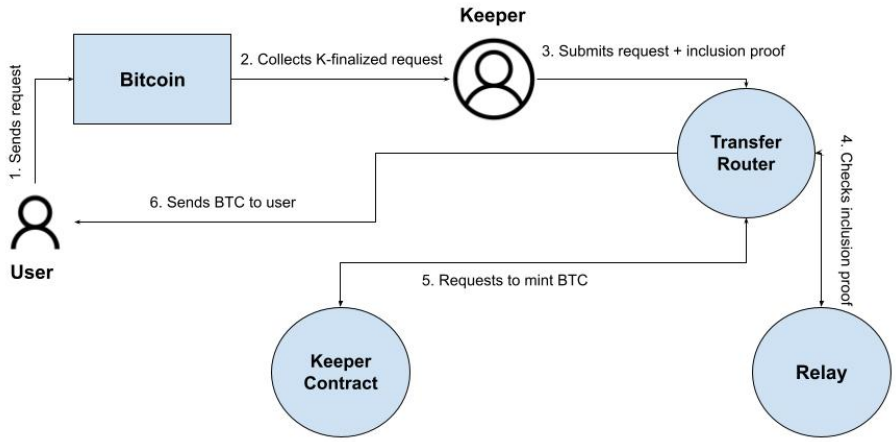

Figure 4.1: Normal cross-chain transfer scenario

The wrapped BTC is a token that has equal value to the BTC. It is the representation of BTC on the target chain. Each wrapped BTC has an equally backed BTC, so the price of the wrapped BTC and BTC are the same. Also, whenever a user wants, she can burn the wrapped BTC and get an equal amount of BTC on the Bitcoin. This is another factor in pegging the price of BTC and wrapped BTC. The user can use wrapped BTC in the target chain dApps. For example, she can lend the wrapped BTC to another user through a lending dApp.

4.1.2 Private Cross-chain Transfer

In the normal cross-chain transfer, the user determines the recipient address in the request. So anyone can link the BTC locked on Bitcoin to the BTC that is minted for the user. This means that the user doesn't have privacy when moving BTC between chains.

As discussed before, Bitcoin transaction doesn't provide full privacy. It is easy for users to track the transactions of a user to find the history of all the user payments. There is a solution in Bitcoin called Coinjoin [34] which adds privacy to Bitcoin payments (see 4.2). Assume two users who want to send one BTC. If each user does it separately (creates a separate transaction), the link between the recipient and the receiver address is clear. However, by combining the transactions into

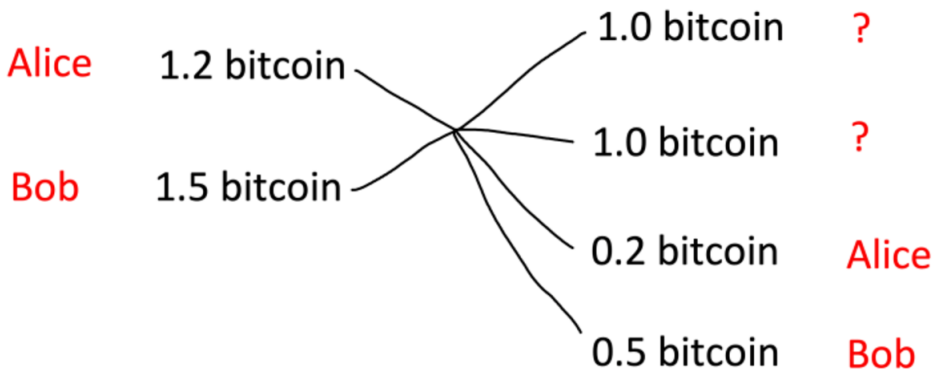

Figure 4.2: How Coinjoin works [5]

a single transaction, the link between recipients and receivers is disappeared.

To do so, users send their desired transaction inputs and outputs to a coordinator. The coordinator creates a transaction with the given inputs and outputs and sends it back to users for signing. After each user signed the transaction, it gets broadcast on the network. Now when an external party looks at this transaction, it is equally possible that each sender sends money to a receiver.

We leverage this technique to add privacy to HotGate. Users who want to move BTC from Bitcoin to another chain can collaborate to add privacy to their transactions. Users choose a standard amount of BTC that they are willing to move (e.g. one BTC). Then, each sends the desired transaction input and the recipient address to a coordinator. The coordinator creates a transaction that has multiple same-value outputs that go to the Keeper's address. Also, the coordinator writes the given recipient addresses in the transaction payload. Then the coordinator sends this transaction to users to sign. Since payloads are aggregated in a single payload and users send BTC to the Keeper in a single transaction, the external user cannot link the user's Bitcoin address to the desired recipient address.

Although this solution provides a level of privacy for users, it has two limitations:

- The amount of BTC that the user wants to move should be a standard number.

- There is a need for a few counterparties who are willing to do the private cross-chain transfer at the same time.

By running the protocol for a while, we can find the most common number of BTC that is moved by users to choose it as the standard number. Also, we can give a discount for the private transaction to incentive most users to use this kind of transfer instead of normal cross-chain transfer to increase the number of private transfer users.

4.1.3 Fast Cross-chain Transfer

When a user wants to transfer an asset from Bitcoin to the target chain, first she locks the asset on Bitcoin. Then, she mints the wrapped BTC on the target chain by providing proof for the inclusion of the lock transaction. Because of Bitcoin's slow finality, it takes around sixty minutes for the user to be able to provide the inclusion proof for the lock transaction, so, the cross-chain transfer process will be slow.

The number of confirmation blocks "K" needed for a block to get finalized is called the finality parameter. In fast transfer, the user receives the wrapped BTC on the target chain after her lock transaction gets buried under F (less than K) blocks. In other words, the user receives her transferred asset before her lock transaction becomes finalized on Bitcoin, reducing the waiting time for the user.

For example, suppose that Alice wants to transfer some amount of BTC from Bitcoin to the target chain (see figure 4.3). She locks this amount on Bitcoin. After this transaction gets F confirmations, she sends the fast transfer request to the HotGate contract. Leveraging the Relay, the HotGate contract checks whether the lock transaction has gotten F confirmations or not. If the condition is satisfied, the contract sends an equivalent amount of BTC from the Bitcoin fast pool to Alice, extracting the fastFee. The fastFee is the fee that the fasts pool gets from a user for the provided service.

After the user request gets finalized on Bitcoin, HotGate contract mints wrapped BTC for it. Then, it sends the minted tokens to Bitcoin fast pool to compensate it for the transferred amount.

The above mechanism exposes fast pool providers to a risk: since the user

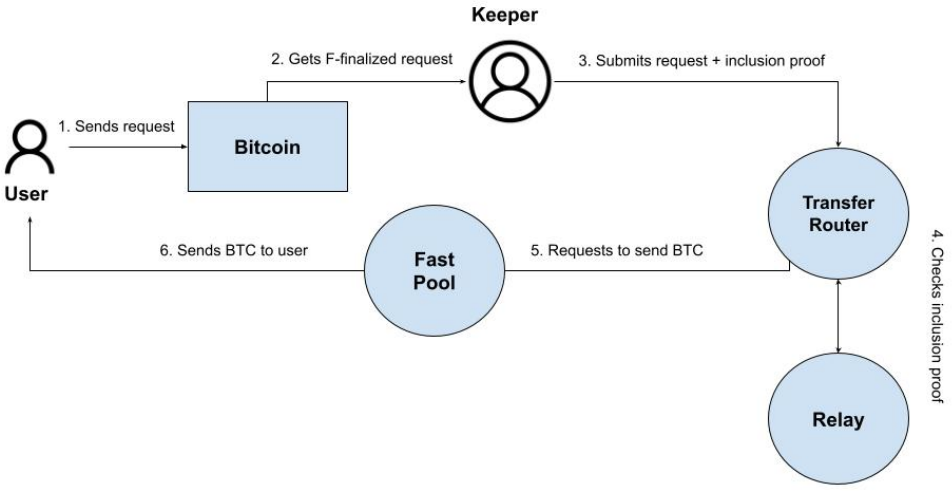

Figure 4.3: Fast cross-chain transfer scenario

request is not finalized when the tokens get transferred from the Bitcoin fast pool, there is a probability that the transaction gets reverted [31] and the fast pool doesn't get compensated. In this case, the fast pool providers suffer loss.

There are two cases where a block with F confirmations gets rewound from the canonical chain. First, it can rewind because of a natural fork in the blockchain. Second, an attacker can rent a mining system and try to rewind that block. To mitigate these risks, we need to choose proper values for the below parameters:

- **fastFee**: A user who performs a fast transfer should pay a percentage of the transferred amount as a fee to the Bitcoin fast pool.
- **fastLimit**: There is a limit on the amount that can be transferred in a single Bitcoin block using the fast feature.

To determine the above parameters, we need to calculate the probability and cost of reversion of a block with F confirmations:

- **reversionProbability**: The probability of reversion of an F-confirmed transaction.
- **reversionCost**: The cost of reverting an F-confirmed block for an attacker.

To make the fast transfer protocol secure, the below conditions should be satisfied:

- $fastFee > reversionProbability$: It motivates fast pool providers to participate in the protocol.
- $reversionCost > fastLimit$: It demotivates attackers to revert an F-confirmed block as it won't be profitable for them to perform such an attack.

4.2 Cross-chain Exchange

In cross-chain exchange, a user who has BTC on Bitcoin wants to exchange her BTC for another token on the target chain. Currently, users move their BTC to centralized exchanges, swap them for the desired asset and then withdraw it on the target chain. This solution is not in favor of users who care about decentralization. Also, it is a three-step process that makes the user experience poor.

Cross-chain exchange has two modes:

- **Normal**: The user will receive the exchanged token on the target chain after her request on Bitcoin gets finalized. This mode is useful for users who are not in hurry to receive their assets.
- **Fast**: The user will receive the exchanged token on the target chain before her request on Bitcoin gets finalized. This mode is useful for users who rush to receive their assets on the target chain.

In the following, we explore the above modes:

4.2.1 Normal Cross-chain Exchange

To perform a normal cross-chain exchange, a user sends an exchange request on Bitcoin. In the request, the user determines the below information:

- Recipient address: This is the address that the exchanged tokens are sent to it.
- Percentage fee: This is the amount of fee that the user wants to pay the Keeper.

- Exchange token: Address of exchange token that the user is willing to receive.
- Output amount: The minimum amount of exchange token that the user expects to receive.
- Deadline: The deadline for executing the exchange request.

By submitting this request, the user also locks some amount of BTC. This BTC will be then exchanged for the desired exchange token. After the user request gets finalized on Bitcoin, a Keeper collects it and submits it on the target chain contract by providing Merkle proof. HotGate contract then checks the inclusion proof of the transaction and executes it. The contract first extracts the information from the request, then mints the wrapped BTC for the user, and after that exchanges the wrapped BTC for the desired exchange token using the corresponding liquidity pool.

In the exchange process, three conditions are checked:

- Liquidity pool existence: To exchange wrapped BTC for the exchange token, the liquidity pool of this pair should exist. The user might submit an exchange token address that doesn't have a liquidity pool with the wrapped BTC.
- Deadline: When the contract wants to exchange assets, the deadline for exchanging should not be passed. This can happen if the Keeper submitted the request with a long delay, or if the deadline was too tight.
- Output amount: The user expects to receive a minimum amount after exchanging her wrapped BTC. If the amount that the user can receive is less than the expected amount, the exchange will be failed. This can happen because of price fluctuations.

If these conditions are satisfied, the exchange will be successful and the exchanged tokens are sent to the determined recipient address. Otherwise, the exchange will not take place. In this case, the HotGate contract sends the wrapped BTC to the recipient's address. This ensures that in both cases the user's fund won't get stuck in the protocol.

4.2.2 Fast Cross-chain Exchange

The fast cross-chain exchange process is the combination of fast cross-chain transfer and normal cross-chain exchange. The user receives wrapped BTC from the fast pool and the received wrapped BTC will be exchanged through the liquidity pool. Here, by separating fast pools from liquidity pools, we separate the risks of users who provide liquidity for fast pools from users who provide liquidity pools.

Fast cross-chain exchange is a suitable option for users who need to receive their exchange assets before the finalization of their request on Bitcoin. For example, when the price of BTC drops quickly, many users want to exchange their BTC for stablecoins as fast as possible. This feature helps users to reduce their losses.

Chapter 5

Incentive Design

In this chapter, we describe the incentives of network participants for collaborating in the protocol. The incentive mechanisms help us to achieve our design goals, otherwise, nodes will not provide their services and the protocol will stop working.

5.1 Relayer

As discussed before, Relayers get block headers from Bitcoin's full nodes and submit them on the target chain. So running such a node costs the Relayer. If Relayers don't get compensate properly, they will leave the network which harms the liveness of the protocol, i.e, users cannot prove that their requests have been recorded on Bitcoin.

To compensate Relayers, HotGate takes fees from users who want to use the submitted block headers. HotGate calculates the required amount of fee in epochs of block headers, based on the cost of block header submission and Relay usage rate. The number of blocks in one epoch is such that we can estimate the number of queries sent to the Relay in an epoch using the number of queries in the previous epoch. Therefore, the fee that a user needs to pay for verifying some data decreases if the total number of requests had increased in the previous epoch.

To avoid too much increase in the Relay fee, we assume a base number for queries in an epoch. The excess fee from extra usages in previous epochs gets accumulated in the Relay contract. This amount is used to pay the Relayers whose

block headers have been in an epoch with less than the base number of queries usages. The amount of fee that a user pays is calculated as follows:

$$Fee = \frac{G * P * M}{\max(N, B)}$$

- G: block header submission gas usage
- P: gas price
- M: number of blocks in an epoch
- N: previous epoch queries
- B: base number of queries

The Relayer reward is the amount of fee that the Relayer paid for block header submission multiplied by a factor greater than 1. This ensures that the Relayer gets fully compensated and earns an extra reward.

We decouple the fee of a block header with the number of requests belonging to that block header. Otherwise, if a block header doesn't include any request no one has the incentive to submit that header which harms the protocol's liveness.

5.2 Keeper

Keepers lock collateral in the system which enables users to mint and burn BTC. Also, they move users' requests from Bitcoin and submit them on the target chain. To execute the request, they need to pay the Relay fee. In summary, keepers have three main costs:

- Target chain transaction fee: Keepers submit users' cross-chain requests on the target chain. They pay the gas fee for executing these requests.
- Relay fee: Using Relay to check the inclusion of requests has a fee. This fee is paid by the Keeper who submits the request.
- Capital cost: Keepers lock some amount of collateral that they don't have access to it.

To compensate keepers for the above costs, users who perform cross-chain transfers or exchange pay fees to keepers. This fee consist of two part:

- Fixed fee: target chain transaction fee and Relay fee that keepers pay are independent of the value that the user sent to them on Bitcoin. Keepers pay these fees in the target chain native token. To compensate them, an equal amount of BTC will be taken from the users' locked amount and given to the corresponding keeper.
- Dynamic fee: this fee is a percentage of BTC that a user wants to transfer or exchange. It incentives keepers to lock their collateral in the system so that users are able to mint and burn BTC.

5.3 Fisherman

Fishermen monitor keepers' activity and slash them if they behave maliciously. The incentive of fishermen for monitoring the system is to earn a slashing reward, which is taken from the collateral of the keeper who misbehaves.

5.4 Liquidity providers

Liquidity pool providers and fast pool providers earn fees from the liquidity pools and fast pool. This fee is taken from the users who use a liquidity pool or a fast pool. In the case of a liquidity pool, this fee is a percentage of the input token that a user wants to exchange. In the case of the fast pool, the fee is a percentage of the token that the user receives from it. These fees are evenly distributed between liquidity providers.

5.5 Discussion

The incentive design plays a crucial role in Hotgate. It should motivate Hotgate participants (Relayer, Keeper, Fisherman, and Liquidity providers) to follow the protocol rules and behave honestly. This guarantees the safety and liveness of Hotgate. Regarding safety, the only factor that prevents Keepers to steal users' assets is the economic incentive. Fishermen are expected to monitor Keepers and

report their malicious activities. If Keepers can bribe fishermen to stop monitoring, they can steal users' assets. Regarding liveness, Relayers should have a proper incentive to submit block headers as soon as they are created. If all Relayers leave the protocol, the users' requests get stuck. Or if Relayers submit block headers with a long delay, the wrapping process becomes slow. Therefore, many users stop using Hotgate.

The proposed incentive design has not been fully analyzed. It is a complex problem since several participants with different motivations are involved in the Hotgate protocol. For example, if Keepers behave honestly for a while, fishermen lose their incentives to monitor them and leave the network. In the absence of fishermen, Keepers can steal users' assets. This example shows that more investigation is needed to make sure all participants' incentives are aligned with the protocol goals. The previous works also haven't discussed the incentive design and it is still an open research problem.

Chapter 6

Implementation

The proposed cross-chain settlement protocol (HotGate) described in previous sections is implemented [7] between the Bitcoin testnet and the Ethereum testnet (called Ropsten). Blockchain testnets are networks designed for testing smart contracts and dApps, so users don't have to spend real money to test their written codes. Testnet networks work in the same way as the main networks, except that their consensus mechanism is often modified to make it easy for miners to add new blocks. Smart contracts of HotGate are written in Solidity version 8 [10]. The contracts are tested and deployed to Ropsten using Hardhat [6]. Contracts of the HotGate AMM DEX are forked from Uniswap V2 contracts [12].

Scripts for Relayer and Keeper are written in JavaScript. They use Infura [8] for querying and sending data to Ropsten. Also, they use Blockstream [3] to get Bitcoin data. The protocol is tested by launching one Keeper and one Relayer node on an Amazon web services (AWS) Ubuntu Server 18.04 LTS (with 2 GB memory and 1 core processor).

The tested scenarios are normal and fast cross-chain transfer and normal and fast cross-chain exchange. To choose the fast feature parameters (fastFee and fastLimit), we assume that at most %10 of the Bitcoin miners are malicious. Based on [38], the reversion probability of an F-confirmed block is upper bounded by $(4\beta(1-\beta))^{F+1}$. Here, we set $F=2$, which means users will receive their wrapped assets after 20 minutes (which is an acceptable delay). According to this formula, reversionProbability for $F=2$ is upper bound by 0.046. By setting the fastFee

equal to %5, there is enough incentive for fast pool providers to deposit liquidity. To determine fastLimit, we assume that an attacker rents %10 of Bitcoin mining power. Based on [9], it costs around $10k per block for the attacker to rent the required mining infrastructure. So, if we assume that fastLimit is less than $10k, there is no incentive for the attacker to conduct this attack.

Chapter 7

Conclusion

In this work, we propose HotGate, a cross-chain settlement protocol that enables users to move or exchange their assets between Bitcoin and programmable blockchains. Bitcoin has the highest market cap among all cryptocurrencies but it doesn't offer programmability, so Bitcoin's users don't have access to any dApp. HotGate enables Bitcoin's users to get access to ecosystems of multiple blockchains. The proposed protocol is implemented between the Bitcoin testnet and the Ethereum testnet.

HotGate uses Relay, a fully decentralized interoperability protocol that connects blockchains directly together. So instead of relying on third parties to provide valid data of the source chain on the target chain, Relayers only provide the data, and the validity of it is checked by the smart contract.

We identify two weaknesses of Bitcoin: speed and privacy. Transactions get finalized on Bitcoin after six confirmations which take around 1 hour. To overcome the Bitcoin latency, we introduced fast pools that give users their wrapped BTC before their request gets finalized on the Bitcoin by paying an extra fee. To increase the privacy of users' assets, we unlink users' assets on Bitcoin from their wrapped assets on the target chain using an approach similar to mixing protocols.

We also improve the user experience so that users get their wrapped assets without interacting with both chains. They only need to send their request on Bitcoin and nodes called Keepers move their request to the target chain.

In summary, the main advantages of the Hotgate protocol are:

1. Decentralization: Users can move BTC from Bitcoin to other programmable blockchains without trusting third parties. All the process of moving assets is controlled by smart contracts.

2. High-speed: Instead of waiting one hour (which is the Bitcoin finalization time), users can move BTC from Bitcoin to other programmable blockchains in less than 20 minutes.

3. Privacy: Hotgate protects users' privacy in transferring assets between blockchains. External observers cannot link wrapped BTC to the native BTC.

4. Smooth UX: From the users' perspectives, the process of wrapping BTC happens with one click. This means that users don't need to interact with both blockchains to wrap their assets.

The proposed protocol also has below limitations:

1. High-cost: Hotgate uses Relay to access the data of Bitcoin on other blockchains. Since Relay verifies all the data on-chain, it has a higher cost in comparison to other solutions.

2. Limited Fast Transfer: To avoid attacking the protocol, we limit the amount that can be transferred in a fast way from Bitcoin to another blockchain. If there is a huge demand for this feature, Hotgate cannot process all the requests.

3. Long Wait for Private Transfer: When a user wants to wrap BTC privately, there should be at least two other users who want to wrap the same amount of BTC privately. This makes the waiting time for the private transfer longer than normal transfer.

Chapter 8

Appendix

8.1 Terminology

8.1.1 Wrapped Asset

According to [19], the definition of wrapped asset is: ”The wrapped asset is a token that represents a real-world or crypto asset, and it is backed by the represented asset or assets of equal value. The backing asset is put in a vault called a “wrapper” (hence wrapped asset). It is issued on a blockchain, and it is supposed to keep the same value as the represented asset. Tokenizing an asset means converting its structure to a blockchain-compliant one so that smart contracts can manage it.”

8.1.2 Wrapped Token

According to [19], the definition of the wrapped token is: ”Wrapped tokens are a particular category of wrapped assets that is not intended to create a crypto representation of real-world assets such as gold, stock, or fiat currencies but of other crypto assets instead. The reason for their existence is because, as blockchains are entirely isolated from the external world, they are also isolated from each other.” Wrapped tokens help us to move the native currency of one blockchain to another.

8.1.3 Burning

Burning is the process of destroying some amount of wrapped tokens by its smart contract. Assuming that the current supply of a wrapped token is Y, by burning X wrapped tokens, the total supply reduces to $Y - X$. Users burn their wrapped tokens to get the exact amount of the native token.

8.1.4 Gas Fee

According to [33]: "The creator of a transaction is responsible for specifying, among other fields, a gas limit and a gas price for the transaction. The gas limit is a measure of the cost (in computation, storage, and so on) imposed on the blockchain by the transaction. The gas price specifies how much the transaction creator is willing to pay per unit of gas". Having gas limit and gas price, the cost of executing a transaction on the blockchain is $gaslimit * gasprice$.

8.1.5 Merkle Tree

According to [35]: "Merkle trees are named after Ralph Merkle. Bitcoin and other cryptocurrencies use Merkle trees, also called hash trees, to encode and encrypt blockchain data efficiently and securely. Merkle trees are created by repeatedly calculating hashing pairs of nodes until there is only one hash left: the root hash or Merkle root, which is a summary value. They are constructed using a bottom-up approach in which each transaction is hashed, then each pair of transactions is concatenated and hashed together, and so on until there is one hash for the entire block."

8.1.6 Proof of Work

According to [22]: " Proof of Work (PoW) contained in Bitcoin is used to protect the ledger blockchain from unwanted changes. PoW on Bitcoin works in the following way: All information contained in a candidate block is calculated by its hash value. This hash value generated must meet the criteria of difficulty level determined by the system. If the hash value does not meet the criteria, then the calculation will be repeated by changing the value of nonce (number once). Nonce is a value that does not have any meaning, but is intentionally added to the block

in order to generate a hash value according to the conditions. If the hash value has not met the value of the rule, the nonce value will be changed again until the miner finds a hash value that meets the criteria."

Bibliography

[1] URL https://www.coindesk.com/learn/2021/08/20/what-is-an-automated-market-maker/. → pages x, 5

[2] . URL https://developer.bitcoin.org/devguide/block_chain.html. → pages x, 2

[3] blockstream, . URL https://blockstream.com/. → page 39

[4] Btc relay, a bridge between the bitcoin blockchain & ethereum smart contracts. URL http://btcrelay.org/. → page 7

[5] URL https://bitcoinmagazine.com/technical/a-comprehensive-bitcoin-coinjoin-guide. → pages x, 29

[6] hardhat. URL https://hardhat.org/. → page 39

[7] implementation. URL https://github.com/MahyarDaneshpajooh/hotgate. → page 39

[8] infura. URL https://www.infura.io/. → page 39

[9] miner rental. URL https://www.miningrigrentals.com/rigs/sha256ab. → page 40

[10] solidity. URL https://docs.soliditylang.org/en/v0.8.17/. → page 39

[11] ultra sound money. URL https://ultrasound.money/. → page 4

[12] uniswap v2. URL https://github.com/Uniswap/v2-core. → page 39

[13] H. Adams, N. Zinsmeister, and D. Robinson. Uniswap v2 core. 2020. → page 5

[14] G. Almashaqbeh and R. Solomon. Sok: Privacy-preserving computing in the blockchain era. pages 124–139, 2022. doi:10.1109/EuroSP53844.2022.00016. → page 12

[15] K. M. Alonso and J. Herrera-Joancomartí. Monero - privacy in the blockchain. *IACR Cryptol. ePrint Arch.*, 2018:535, 2018. → page 6

[16] R. Belchior, A. Vasconcelos, S. Guerreiro, and M. Correia. A survey on blockchain interoperability: Past, present, and future trends, 2021. → page 6

[17] V. Buterin. Ethereum white paper: A next generation smart contract & decentralized application platform. 2014. URL https://cryptorating.eu/whitepapers/Ethereum/Ethereum_white_paper.pdf. → pages iii, 3

[18] V. Buterin. Chain interoperability. 2016. URL https://www.r3.com/wp-content/uploads/2017/06/chain_interoperability_r3.pdf. → pages iii, 6

[19] G. Caldarelli. Wrapping trust for interoperability: A preliminary study of wrapped tokens. *Information*, 13(1), 2022. ISSN 2078-2489. doi:10.3390/info13010006. URL https://www.mdpi.com/2078-2489/13/1/6. → page 43

[20] M. Castro. Practical byzantine fault tolerance. 04 2001. → page 1

[21] P. Frauenthaler, M. Sigwart, C. Spanring, and S. Schulte. Testimonium: A cost-efficient blockchain relay. 2020. doi:10.48550/ARXIV.2002.12837. URL https://arxiv.org/abs/2002.12837. → page 8

[22] I. Gemeliarana and R. Sari. Evaluation of proof of work (pow) blockchains security network on selfish mining. 02 2019. doi:10.1109/ISRITI.2018.8864381. → page 44

[23] A. Hentschel, D. Shirley, and L. Lafrance. Flow: Separating consensus and compute. 2019. doi:10.48550/ARXIV.1909.05821. URL https://arxiv.org/abs/1909.05821. → page 6

[24] M. Herlihy. Atomic cross-chain swaps. 2018. doi:10.48550/ARXIV.1801.09515. URL https://arxiv.org/abs/1801.09515. → page 14

[25] L. Lamport. Paxos made simple. 2001. → page 1

[26] L. Lamport, R. Shostak, and M. Pease. The byzantine generals problem. *ACM Trans. Program. Lang. Syst.*, 4(3):382–401, jul 1982. ISSN 0164-0925. doi:10.1145/357172.357176. URL https://doi.org/10.1145/357172.357176. → page 1

[27] R. Lan, G. Upadhyaya, S. Tse, and M. Zamani. Horizon: A gas-efficient, trustless bridge for cross-chain transactions. 01 2021. → page 16

[28] R. C. Merkle. A digital signature based on a conventional encryption function. In *A Conference on the Theory and Applications of Cryptographic Techniques on Advances in Cryptology*, CRYPTO '87, page 369–378, Berlin, Heidelberg, 1987. Springer-Verlag. ISBN 3540187960. → page 17

[29] S. Nakamoto. Bitcoin: A peer-to-peer electronic cash system. 2009. URL http://www.bitcoin.org/bitcoin.pdf. → pages iii, 3

[30] Q. Nguyen, A. Cronje, M. Kong, E. Lysenko, and A. Guzev. Lachesis: Scalable asynchronous bft on dag streams, 2021. URL https://arxiv.org/abs/2108.01900. → page 3

[31] J. Niu, C. Feng, H. Dau, Y.-C. Huang, and J. Zhu. Analysis of nakamoto consensus, revisited. Cryptology ePrint Archive, Paper 2019/1225, 2019. URL https://eprint.iacr.org/2019/1225. https://eprint.iacr.org/2019/1225. → page 31

[32] T. Rocket, M. Yin, K. Sekniqi, R. Van Renesse, and E. Sirer. Scalable and probabilistic leaderless bft consensus through metastability. 06 2019. → page 3

[33] T. Roughgarden. Transaction fee mechanism design. EC '21, page 792, New York, NY, USA, 2021. Association for Computing Machinery. ISBN 9781450385541. doi:10.1145/3465456.3467591. URL https://doi.org/10.1145/3465456.3467591. → page 44

[34] R. Stütz, J. Stockinger, B. Haslhofer, P. Moreno-Sanchez, and M. Maffei. Adoption and actual privacy of decentralized coinjoin implementations in bitcoin, 2021. URL https://arxiv.org/abs/2109.10229. → page 28

[35] H. Sáez de Ocáriz Borde. An overview of trees in blockchain technology: Merkle trees and merkle patricia tries. 02 2022. → page 44

[36] J. Teutsch, M. Straka, and D. Boneh. Retrofitting a two-way peg between blockchains. 08 2019. → page 9

[37] Y. Tong Lai, J. Prestwich, and G. Konstantopoulos. Flyclient - consensus-layer changes. 2019. URL https://zips.z.cash/zip-0221. → pages 6, 8

[38] P. Viswanath. Lecture 6: Safety of bitcoin. 2021. URL https://courses.grainger.illinois.edu/ece598pv/sp2021/lectureslides2021/ECE_598_PV_course_notes6.pdf. → page 39

[39] M. Westerkamp and J. Eberhardt. zkrelay: Facilitating sidechains using zksnark-based chain-relays. pages 378–386, 2020. doi:10.1109/EuroSPW51379.2020.00058. → page 8

[40] K. Wüst and A. Gervais. Do you need a blockchain? pages 45–54, 2018. doi:10.1109/CVCBT.2018.00011. → pages 2, 11

[41] T. Xie, J. Zhang, Z. Cheng, F. Zhang, Y. Zhang, Y. Jia, D. Boneh, and D. Song. zkbridge: Trustless cross-chain bridges made practical, 2022. URL https://arxiv.org/abs/2210.00264. → page 8

[42] J. Xu, K. Paruch, S. Cousaert, and Y. Feng. Sok: Decentralized exchanges (dex) with automated market maker (amm) protocols, 2021. URL https://arxiv.org/abs/2103.12732. → page 4

[43] A. Zamyatin, M. Al-Bassam, D. Zindros, E. Kokoris-Kogias, P. Moreno-Sanchez, A. Kiayias, and W. Knottenbelt. Sok: Communication across distributed ledgers. *IACR Cryptol. ePrint Arch.*, 2019:1128, 2019. → page 9

[44] A. Zamyatin, D. Harz, J. Lind, P. Panayiotou, A. Gervais, and W. Knottenbelt. Xclaim: Trustless, interoperable, cryptocurrency-backed assets. In *2019 IEEE Symposium on Security and Privacy (SP)*, pages 193–210, 2019. doi:10.1109/SP.2019.00085. → page 8