Horizon: A Gas-Efficient Trustless Bridge for Cross-Chain Transactions

Rongjian Lan[1], Ganesha Upadhyaya[1], Stephen Tse[1], and Mahdi Zamani*[2]

[1] *Harmony,* [2] *Visa Research*

Contents

1 Introduction

A unifying vision of the distributed ledger technology is to connect multiple distributed systems together. While the disruption of financial infrastructure is still early, now is the time to bring many protocols together to scale their innovations for broader adoption. Cross-chain bridges provide broader access of users and assets to decentralized finance. In particular, lending Bitcoin for high-yield financial instruments on Ethereum has surpassed US $1B on-chain management. Cross-chain transactions are more than asset transfers via atomic swaps with hash time locked contracts.

A relay bridge implements a bi-directional relay of block headers between two blockchains. In such a bridge, block headers of blockchain A are constantly being submitted to a smart contract in blockchain B, which implements a light client logic to verify the validity of the headers. And analogously headers from blockchain B are submitted to a smart contract in blockchain A.

A simple example is the BTC relay which implements a uni-directional relay from Bitcoin to Ethereum. There, the Ethereum smart contract computes the difficulty of the submitted Bitcoin headers. A proof for the validity of each header amounts to checking it resides on the longest chain of submitted headers.

Light clients [1–3] can be used to perform cross-chain transactions efficiently. Such transactions can happen when exchanging cryptocurrencies [4, 5], transferring assets to sidechains [2, 6, 7], or sharding blockchains [8, 9]. Such clients enable a user to generate short cryptographic inclusion proofs about past events recorded on a blockchain. In a *proof-of-work (PoW) blockchain* such as

*This work was done when the author was affiliated with Yale University.

Bitcoin [10] and Ethereum [11], this can be done by providing the sequence of all block headers to prove to any verifier that the event is recorded on the honest chain that is the longest, or more precisely, the most computationally-difficult chain.

An important characteristic of many light client proofs is portability, i.e., they can be forwarded to other nodes on other blockchain networks to convince them that a certain event was recorded on the source blockchain. While light clients for PoW chains have been proposed starting with Bitcoin's SPV client [10] and later in PoPoW [2, 12] and FlyClient [3], no light client has been proposed for proof-of-stake chain that rely on Byzantine fault tolerance (BFT) for consensus. Designing a light client for such chains is challenging. While aggregate signatures contain a sufficient number of attestations, we do not know if those attestations actually come from validators that actually have stake. The only way to know this is if we know the balance (i.e., state) of every validator which is what a full node does.

1.1 Our Contributions

We propose a gas-efficient, cross-chain bridge protocol to transfer assets from a BFT blockchain to another blockchain (e.g., Ethereum) which supports basic smart contract execution. To achieve this, our paper makes the following contributions:

- We construct **a super-light client for BFT chains** that allows a client to prove to any external entity that a transaction has been recorded on the BFT chain by providing a cryptographic proof that is constant size in the length of the chain.
- We construct **a bridge smart contract** on the destination chain for atomic verification of super-light client proofs that guarantee a certain amount of tokens are locked on the BFT chain. The contract also can unlock/mint an equal amount of tokens on the destination blockchain once the verification succeeds.
- We construct **a relay node** which periodically transmits to the contract constant-size, checkpoint information as commitments to the BFT chain. This information allows the contract to later verify super-light client proofs submitted by the client to the contract. While the total amount of information submitted by the relay to the contract for all checkpoints is linear to the chain length, the frequency of checkpoints could be adjusted in practice to curb this overhead.
- We propose **an efficient chain commitment** mechanism that allows the client to prove inclusion of a block in a blockchain with a constant-size commitment and logarithmic blockchain inclusion proofs.
- We further propose a **stateless bridge contract** design that allows the client to send a small, self-sufficient cross-chain transaction to the contract that does not require any pre-relayed checkpoint information. Our solution requires the client to include only a logarithmic-size (in the chain length) inclusion proof in its message, making it the first BFT bridge protocol that requires logarithmic-size, cross-chain proofs.

On PoW-to-BFT Transfers. Our bridge protocol further allows a client to transfer assets from a PoW chain (such as Bitcoin or Ethereum) to a BFT chain using FlyClient [3] logarithmic-size proofs. This, however, requires certain chain commitments (in the form of Merkle roots) already being included in every block header, which unfortunately, is not possible until a soft fork on Bitcoin and Ethereum includes these commitments in all future block headers. Until then, our

bridge protocol adopts the SPV approach of Rainbow bridge [13], where the relay node periodically sends all recent Bitcoin/Ethereum block headers to the smart contract on the BFT chain. While this incurs a significantly higher storage and computation overhead on the contract, we expect the significantly lower gas cost of most BFT chain (such as Harmony [14] and NEAR [13]) could justify such overhead until chain commitments become available on Bitcoin and Ethereum. Moreover, one may .

1.2 Our Model

System Model. Consider a blockchain protocol A, where a group of *validators* agree on a chain via a BFT blockchain protocol such as [15–17], where validators participate in a BFT consensus protocol to agree on each block of transactions. In a permissionless setting, such a protocol typically proceeds in *epochs*, where in each epoch, one or more BFT executions are followed by a reconfiguration step to randomly select a new group of validators to drive the next epoch. To obtain voting power in the BFT protocol, the reconfiguration protocol may establish identities through any Sybil-resistance mechanism such as PoW (as in [8, 15, 18]) or proof-of-stake (PoS) (as in [14, 16, 19]).

Let B denote another blockchain network that grows a valid chain based on any consensus mechanism (e.g., Nakamoto [10] or BFT) and provides basic smart contract support to execute arbitrary programs. We say a block B of transactions added to B's chain is final when B is permanently recorded on the chain. For example, in Bitcoin and Ethereum, this means that a sufficient number of blocks are appended to the chain after B, formalized as the most difficult chain principle by Garay et al. [20, 21].

Our bridge system consists of (1) a *client* C who can submit cross-chain transactions to A and B, (2) a *relay* R which periodically submits information about A's chain to B, and (3) a *full node* F_A that maintains an up-to-date copy of A's chain at any time and responds to queries from C and R about the chain.

Threat Model. We consider a probabilistic polynomial-time Byzantine adversary who wishes to prevent our protocol from reaching its goals by corrupting a subset of nodes involved. The corrupt nodes may not only collude with each other but can also deviate from the protocol in any arbitrary manner, e.g., by sending invalid or inconsistent messages, by remaining silent, etc. We allow the adversary to control both C and R to prevent our protocol from achieving its goals.

We assume that, at any time, both chains maintain standard blockchain immutability and no double spending properties. In PoW chains, for example, this requires that, at any time, less than 1/2 of the computational mining power (aka, hash rate) be controlled by the adversary. To ensure security against selfish mining [22], one may alternatively assume less than 1/4 adversarial computational power. Alternatively, a PoS chain may require less than 1/3 of the total stake associated with validators in the network be controlled by the adversary.

Problem Definition. Client C wants to perform a cross-chain transaction $X_{\mathsf{A}\to\mathsf{B}}$ to transfer an amount of x tokens from A to B. The transaction consists of two on-chain transactions T_{burn} and T_{unlock} to be recorded on A and B respectively. A cross-chain bridge protocol is secure if and only if it guarantees the following atomicity property:

> **Atomicity:** Transaction $X_{\mathsf{A}\to\mathsf{B}}$ is either committed or aborted. We say $X_{\mathsf{A}\to\mathsf{B}}$ is committed if and only if T_{burn} is committed (i.e., permanently recorded) on A's chain and T_{unlock} is committed on B's chain. We say $X_{\mathsf{A}\to\mathsf{B}}$ is aborted if and only if T_{burn} and T_{unlock} are both aborted (i.e., not committed on the corresponding chains).

2 Background and Related Work

2.1 Light Clients

To verify that a blockchain is valid without participating in the mining process, a client may choose to download blocks from a miner or a *full node* who holds a copy of the entire chain. Currently, downloading and verifying all blocks in Bitcoin or Ethereum requires a node to download more than 200 GB of data, taking hours to synchronize the node's local blockchain [23]. Such a requirement causes long delays for regular clients and makes it nearly impossible for storage-limited clients to quickly verify transactions.

The original Bitcoin design [10] describes a faster synchronization mechanism, known as *simplified payment verification*, that allows efficient verification of transactions on the blockchain. In Bitcoin [10] and Ethereum [11], block headers contain enough information to ensure that (1) the PoW is valid, (2) the block includes a certain transaction, and (3) the block is at a certain position on the correct chain. The transaction validation process utilizes a Merkle tree commitment to all transactions in a block which is stored in each block header. A light client does not verify all transactions in the entire chain and essentially relies on the assumption that the chain with the most proof-of-work contains only valid transactions and follows the rules of the system.

Kiayias et al. [2, 12] propose an interactive proof mechanism, known as proofs of proof-of-work (PoPoW) that allows a prover to convince a verifier with high probability in logarithmic time and communication that a chain contains a sufficient amount of work. The PoPoW protocol suffers from multiple drawbacks described by Bünz et al. [3] who propose a new solution, known as FlyClient, that overcomes the limitations of PoPoW.

FlyClient uses a probabilistic sampling technique to randomly sample a logarithmic number of block headers (in the chain length) from a PoW-based blockchain with variable block difficulty. FlyClient uses an efficiently-updatable commitment mechanism, known as *Merkle mountain range (MMR)* [24], that allows provers to commit to an entire blockchain with a small (constant-size) commitment while offering logarithmic block inclusion proofs with position binding guarantees.

2.2 Cross-Chain Bridges

Rainbow Bridge [13]. NEAR's Rainbow bridge uses light clients to transfer ERC-20 tokens from Ethereum to NEAR's PoS blockchain and vice versa. For each chain, the Rainbow protocol deploys a smart contract that implements a light client and relay nodes that regularly sends block headers to the light client. Namely, Ethereum relays (aka, ETH-2-NEAR relays) sends every single Ethereum header to the NEAR contract, while the NEAR relays (aka, NEAR-2-ETH relays) sends one header every 4 hours to the Ethereum contract. As a result, both contracts can independently verify the inclusion of any event on the other chain.

While the Rainbow bridge allows trustless cross-chain transfers, it has multiple performance and security drawbacks. First, the relays need to constantly forward Ethereum and NEAR headers to the smart contracts and this comes at a large gas cost on both chains. Second, to avoid an ever-growing state of the NEAR contract, the bridge limits the number of synced headers to only seven days. If the contract also limits the verification to only the seven-day log, then this significantly lowers the security of the light client as a malicious Ethereum relay can now create a fake chain of only seven days worth of headers appended to a valid prefix and present it to the NEAR contract. If the contract also incorporates previous seven-day logs from old blocks on the NEAR blockchain, then the large gas cost overhead would again come into play. Third, Rainbow's Ethereum contract does not verify all validator signatures on every NEAR header submitted to the contract.

XCLAIM [25]. XCLAIM is a framework for trustless, cross-chain exchanges, where a smart contract on each chain governs the exchanges between the two chains (e.g., Bitcoin and Ethereum) and punishes malicious parties by seizing their collateral in favor of honest parties. The XCLAIM model consists of three main entities: A client who wishes to move funds from Bitcoin to Ethereum, a vault that locks the Bitcoin funds received from the client, and an Ethereum relay contract known as BTCRelay [26] which stores Bitcoin block headers to allow verification of SPV proofs.

The protocol starts with the vault locking up sufficient collateral on the Ethereum smart contract. The client then sends her Bitcoins to the vault and submits a proof to the contract showing that the transaction has been recorded on the Bitcoin blockchain. The chain relay verifies this proof and confirms to the contract that the lock has been executed correctly. Finally, the contract releases Ethereum tokens to Alice.

tBTC [27]. tBTC is a multi-wallet, multi-signer protocol that provides a BTC-backed bearer asset on Ethereum. tBTC attempts to remove single points of failure by geographically distributing signers and aiming for a multi-wallet scheme. The signers use a multi-party threshold ECDSA protocol to collectively create a signing group wallet which is created by randomly selecting a set of signers from the eligible pool of signers. tBTC relies on collateral to prevent signers from deviating the protocol as in the following cases: (1) To liquidate deposits in case they are in danger of undercollateralization; (2) To punish a signing group if it signs an unauthorized piece of data is once distributed key generation is complete; (3) To punish a signing group that fails to produce a signature for the system when requested; and (4) To ensure a depositor is refunded if the signing group fails to form. As a result, tBTC requires a complicated mechanism for detecting and dealing with undercollateralized signers.

Waterloo [28]. Kyber's Waterloo is a cross-chain bridge between Ethereum and EOS [29]. An Ethereum smart contract serves as a light client that only verifies EOS block headers. The consensus protocol of EOS is based on the delegated PoS mechanism [30], where EOS token holders continuously vote (i.e., delegate) for their favorite block producers. Instead of relaying all EOS block headers, the Waterloo relay only relays the changes in the set of EOS block producers.

2.3 Harmony Blockchain

We build our solution for transferring assets from a BFT blockchain to a PoW blockchain in the context of the Harmony blockchain [14] as our BFT protocol and Ethereum as our PoW blockchain. Harmony is a sharded PoS blockchain protocol inspired by research results including but not limited to RapidChain [8] and OmniLedger [9]. The protocol execution is divided into predetermined time intervals (e.g., 24 hours), known as *epochs*. In each epoch, the network is partitioned into a set of *shards* each of which maintains a separate blockchain in parallel to other shards.

The validators of shards are randomly sampled using a distributed random generation (DRG) protocol executed at the end of every epoch by a special shard known as the *beacon shard*, which itself gets re-elected in every epoch. The beacon shard is also where the configuration of the network (i.e., validator identities and shard assignments) are stores. Moreover, the beacon shard is where token holders deposit their stakes to become validators in the consensus protocol.

Each epoch consists of multiple executions of a BFT consensus protocol in each shard, where a block of transactions is generated and appended to the shard's chain after every execution. The set of validators and shards is fixed throughout each epoch but may change between epochs based on a shard reconfiguration protocol executed by the beacon shard at the end of each epoch. The last block of the beacon chain in every epoch is called an *epoch block* which stores the identities (i.e.,

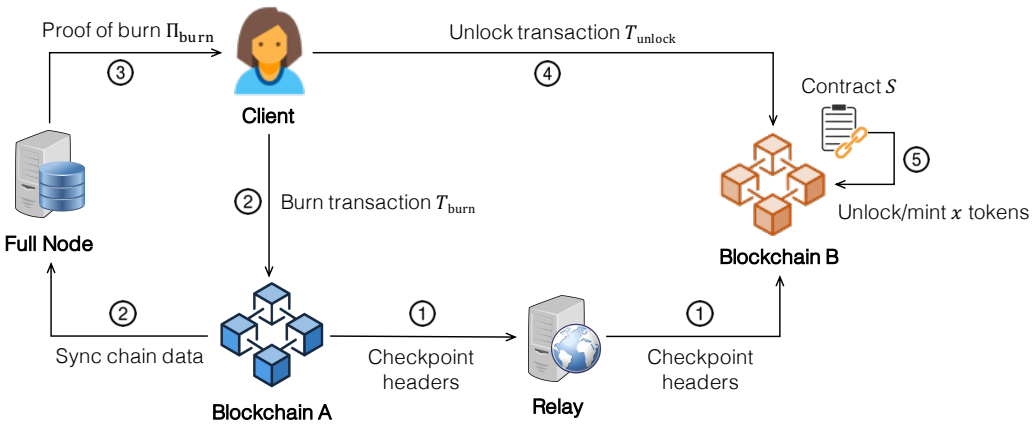

Figure 1: Our Cross-Chain Bridge Protocol

public keys) of all shard members for the next epoch. Each identity consists of the node's ECDSA address, its BLS public key, and its stake in the consensus, represented as a decimal number between 0 and 1.

For every BFT consensus execution, a leader is selected based on the randomness generated by the DRG at the end of the previous epoch.[1] In every run of the consensus protocol, the leader runs the aggregate BLS signature protocol of Boneh et al. [33, 34] to collect the validators' votes in a constant-sized, threshold aggregate signature and then broadcast it to the shard. The aggregate signature is included in the block header for later block verification. After the new block is committed to the shard chain (i.e., the chain maintained by the shard), the shard validators send the block header to the beacon shard who verifies the header contents (i.e., the previous hash and the aggregate signature) and broadcast it to all shards to facilitate future cross-shard verification.

3 Our Solution

Consider a client C who wants to perform a cross-chain transaction $X_{\mathsf{A}\to\mathsf{B}}$ to transfer an amount x of her coins from blockchain A (i.e., the Harmony blockchain) to blockchain B (e.g., the Ethereum blockchain). Our bridge protocol consists of a smart contract S deployed on blockchain B as well as a relay node R who periodically syncs S with epoch block headers from A. As shown in Figure 2, our protocol consists of two parts: (1) Relay/contract sync, and (2) Cross-chain transaction. In the following sections, we describe each part in detail.

3.1 Relay/Contract Sync

At the end of each epoch (i.e., every 24 hours), R sends to the contract the most recent epoch block header B_i which is maintained by the beacon shard. This block contains sufficient information to allow the contract to later verify the inclusion of any transaction on A. In our bridge protocol, the contract verifies the inclusion of a *burn transaction*, denoted by T_{burn}, submitted by the client. T_{burn}

[1] Harmony's DRG uses a verifiable random function (VRF) [31] in conjunction with a verifiable delay function (VDF) [32] construction to generate randomness with linear communication complexity and delay the revelation of the randomness generated by the VRF to prevent a malicious leader from biasing the randomness by cherry-picking a subset of the VRF random numbers initially generated by the validators [14].

Cross-Chain Bridge Protocol (Horizon)

Client C performs a cross-chain transaction $X_{\mathsf{A}\to\mathsf{B}}$ to transfer an amount x of her coins from blockchain A to blockchain B. Smart contract S on B verifies and unlocks the transferred coins, and relay R periodically syncs S with checkpoint data from A.

Relay/Contract Sync

Let addr_R denote R's wallet address on B. The following protocol is executed between R and S:

1. At every checkpoint block header B_i, relay R sends a sync transaction T_{sync} to S, where
$$T_{\mathsf{sync}} : (\mathsf{addr}_\mathsf{R} \to \texttt{null}; B_i).$$
B_i includes block height i, quorum signature $B_i.\mathsf{sig}$, quorum public keys $B_i.\mathsf{pks}$, and Merkle root $B_i.\mathsf{cmr}$ created on all block header between $B_{i-\Delta}$ and B_i, where Δ is the block distance between two checkpoint blocks.
2. Upon receiving T_{sync} from R, contract S replaces R and aborts if any of the followings are true:
   (a) $i \neq j+1$, where j is the height of the last checkpoint block received, or
   (b) $\mathsf{QuorumVerify}(B_i.\mathsf{sig}, B_j.\mathsf{pks}) = 0$.

   Otherwise, S stores B_i in the contract's state.

Cross-Chain Transaction

Let addr_C denote client C's wallet address on both A and B. The client performs the following protocol with full node F and contract S:

Stage I: Burn

1. C sends a burn transaction $T_{\mathsf{burn}} : \mathsf{addr}_\mathsf{C} \xrightarrow{x} \texttt{null}$ to A to burn x of her coins on A.
2. C sends $[\texttt{PoB-Request}, h]$ to F, where $h = \mathsf{H}(T_{\mathsf{burn}})$.
3. Upon receiving $[\texttt{PoB-Request}, h]$ from C, full node F performs the following steps:
   (a) Find block header B_k on A that includes the hash of transaction h;
   (b) Generate a Merkle proof Π_B using checkpoint Merkle root $B_i.\mathsf{cmr}$, where $k \in (i-\Delta, i]$;
   (c) Generate a Merkle proof Π_T using transaction Merkle root $B_k.\mathsf{tmr}$;
   (d) Send $[\mathsf{PoB}, \Pi_{\mathsf{burn}}, i, B_k]$ to C, where $\Pi_{\mathsf{burn}} = (\Pi_B, \Pi_T)$.
4. Upon receiving $[\mathsf{PoB}, \Pi_{\mathsf{burn}}, i, B_k]$ from F, client C sends an unlock transaction T_{unlock} to S, where
$$T_{\mathsf{unlock}} : (\texttt{null} \xrightarrow{x} \mathsf{addr}_\mathsf{C};\ T_{\mathsf{burn}};\ \Pi_{\mathsf{burn}};\ i;\ B_k).$$
5. If T_{unlock} fails, then C picks a different full node and repeats from Step I–2.

Stage II: Unlock

6. Upon receiving $T_{\mathsf{unlock}} : (\texttt{null} \xrightarrow{x} \mathsf{addr}_\mathsf{C};\ T_{\mathsf{burn}};\ \Pi_{\mathsf{burn}};\ i;\ B_k)$, contract S does the following steps:
   (a) Parse Π_{burn} as (Π_B, Π_T);
   (b) Fetch B_i from contract's state;
   (c) Abort if any of the following conditions are true:
   - $T_{\mathsf{burn}}.\mathsf{sender} \neq T_{\mathsf{unlock}}.\mathsf{receiver}$, $T_{\mathsf{burn}}.\mathsf{amount} \neq x$,
   - $\mathsf{MerkleVerify}(\Pi_B, B_k, B_i.\mathsf{cmr}) = 0$, where $B_i.\mathsf{cmr}$ is checkpoint Merkle root,
   - $\mathsf{MerkleVerify}(\Pi_T, T_{\mathsf{burn}}, B_k.\mathsf{tmr}) = 0$, where $B_k.\mathsf{tmr}$ is B_k's transactions Merkle root.

   (d) Unlock x to addr_C's balance.

Figure 2: Cross-Chain Bridge Protocol

essentially transfers x coins on A to a `null` address, i.e., deletes the coins permanently. We refer to an inclusion proof for a burn transaction as a *proof of burn (PoB)*.

For ease of presentation and without loss of generality, we consider only one shard (i.e., the beacon shard) and describe our protocol for the case when T_{burn} is recorded on the beacon chain.[2]

Proofs of Burn (PoB). An inclusion proof for T_{burn} should convince the verifier (i.e., the contract) that (1) The transaction is included in a block with header B_k, and (2) The block corresponding to B_k is included on A's chain. The former can be proved using a transaction Merkle proof and verified using the Merkle root stored in B_k. The latter, however, requires a new chain commitment added to the epoch block B_i. Inspired by FlyClient [3], we use a Merkle tree variant, known as a *Merkle Mountain Range (MMR)* [24], over all block headers added to the blockchain between two epoch blocks. This allows S to verify the inclusion of B_k within the epoch using the root of the MMR stored at B_i.

Checkpoint Blocks. The confirmation latency of a cross-chain transaction depends mainly on the rate at which R submits epoch block headers to S. Since one epoch block is created every 24 hours, then a cross-chain transaction would take about 12 hours to be confirmed in expectation. To reduce the confirmation latency, we propose to create periodic *checkpoint blocks* in the middle of epochs on A. A block is called a checkpoint block if its header contains an MMR root calculated over all block headers added to the chain since the previous checkpoint block. Therefore, epoch blocks are considered checkpoint blocks.

Every checkpoint block header B_i includes the following fields the first four of which are included in all blocks:

1. Block height i,
2. Quorum signature $B_i.\mathsf{sig}$, an aggregate BLS signature created by the consensus validators,
3. Quorum public keys $B_i.\mathsf{pks}$, which lists the public key address of every consensus validator,
4. Transaction Merkle root $B_i.\mathsf{tmr}$ created on all transaction included in B_i,
5. Checkpoint Merkle root $B_i.\mathsf{cmr}$ created on all block header between $B_{i-\Delta}$ and B_i, where Δ is the block distance between two checkpoint blocks.

Checkpoint Verification. At every checkpoint block header B_i, the relay sends a sync transaction T_{sync} to S as defined in Figure 2. Upon receiving T_{sync} from R, contract S verifies that the checkpoint information is valid, otherwise it replaces R with another relay and aborts. Namely, S does the replacement and aborts if any of the following conditions are true:

1. $i \leq j$, where j is the height of the last checkpoint block received,
2. $\mathsf{QuorumVerify}(B_i.\mathsf{sig}, B_i.\mathsf{pks}) = 0$,

Otherwise, S stores B_i in the contract's state for future cross-chain verification requests.

3.2 Cross-Chain Transaction

This part of the protocol is initiated by the client C submitting T_{burn} to blockchain A. Once the transaction is confirmed on A, C sends a request to a full node F who always maintains a full,

[2] In the Harmony blockchain, the beacon shard also grows a transaction chain similar to other shards.

up-to-date copy of A's chain. Upon receiving the request, F finds the block B_k that includes T_{burn}. Then, it generates two Merkle proofs Π_B and Π_T that together form the PoB, denoted by Π_{burn}. The former is generated using the checkpoint Merkle root cmr_i stored at the first checkpoint block after B_k, denoted by B_i. The latter is generated using the transaction Merkle root $B_k.\mathsf{tmr}$. Finally, F sends Π_{burn}, i, and B_k to C who creates an unlock transaction T_{unlock} and submits it to blockchain B. Since the full node is not trusted, this transaction may fail, in which case the client replaces F with another full node and repeats the process.

PoB Verification and Asset Unlocking. Upon receiving T_{unlock}, contract S fetches B_i from its state (checkpoint block height i is specified by C in T_{unlock}) and verifies the validity of the burn transaction against the unlock transaction through the checks listed in Figure 2. If all checks are successful, then the contracts unlocks x tokens on B (maintained on S's state) and delivers them to C's address on B.

3.3 Multi-Relay Model

The single-relay bridge model exposes cross-chain transactions to denial-of-service scenarios which could make the bridge protocol completely non-functional. One may alternatively choose to employ a group of relay nodes which redundantly submit the same checkpoint information to the smart contract. Unfortunately, this significantly increases the gas cost of relaying information to the contract as it needs to compare/store an amount of information that grows linearly with the redundancy factor (i.e., the number of relays).

Instead of multiple relay nodes actively submitting checkpoint information to the contract at the same time, we propose to have only one node relay the information at any time and have the other $n-1$ relays read and verify the contract's state after every regular relay event. If the first relay fails to send proper checkpoint information, the second relay would take its role and so forth.

3.4 Stateless Bridge Contract

In some scenarios, one may prefer to implement a bridge protocol without a relay. For example, if the number of cross-chain transactions is small, it might not justify the overhead (i.e., the gas cost) of constantly sending checkpoint information to the smart contract. That's because the number of checkpoint blocks grows linearly with the chain length.

On the other hand, due to the block gas limit, the client cannot include the list of checkpoint blocks for the entire chain in its unlock transaction T_{unlock}. While the client can split the transaction data into multiple transactions/blocks to avoid the gas limit issue, this strategy likely still results in an unreasonable gas cost for a single cross-chain transactions. Moreover, splitting a transaction across multiple blocks requires waiting for multiple block intervals, and thus significantly increasing the latency of the overall transaction.

To allow relay-free cross-chain transactions, we propose to add another layer of chain commitments to every checkpoint block in the form of an MMR constructed over all checkpoint blocks. This allows the prover (i.e., the client) to prove to the verifier (i.e., the contract) that a checkpoint block is included in a missing chain of checkpoint blocks. The client proof now consists of the following information:

1. The header of the block B_{tx} containing T_{burn}.
2. The header of the first checkpoint block B_{cp} after B_{tx}.
3. A Merkle proof showing that T_{burn} was included in B_{tx}.

4. A Merkle proof showing that B_{cp} was included in the chain of checkpoint blocks.

References

[1] Bitcoin Website. http://www.bitcoin.org/.

[2] Aggelos Kiayias, Andrew Miller, and Dionysis Zindros. Non-interactive proofs of proof-of-work, 2017.

[3] B. Bünz, L. Kiffer, L. Luu, and M. Zamani. FlyClient: super-light clients for cryptocurrencies. In *2020 IEEE Symposium on Security and Privacy (SP)*, pages 928–946, 2020.

[4] Maurice Herlihy. Atomic cross-chain swaps. *arXiv preprint arXiv:1801.09515*, 2018.

[5] ethereum/btcrelay: Ethereum contract for bitcoin spv. https://github.com/ethereum/btcrelay, 2018. (Accessed on 12/14/2018).

[6] Adam Back and Gregory Maxwell. Transferring ledger assets between blockchains via pegged sidechains, Nov 2016. US Patent App. 15/150,032.

[7] Aggelos Kiayias and Dionysis Zindros. Proof-of-work sidechains. Cryptology ePrint Archive, Report 2018/1048, 2018. https://eprint.iacr.org/2018/1048.

[8] Mahdi Zamani, Mahnush Movahedi, and Mariana Raykova. RapidChain: Scaling blockchain via full sharding. In *2018 ACM Conference on Computer and Communications Security (CCS)*, 2018.

[9] Eleftherios Kokoris-Kogias, Philipp Jovanovic, Linus Gasser, Nicolas Gailly, Ewa Syta, and Bryan Ford. OmniLedger: A secure, scale-out, decentralized ledger via sharding. In *2018 IEEE Symposium on Security and Privacy (S&P)*, pages 19–34, 2018.

[10] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system, 10 2008. Available at https://bitcoin.org/bitcoin.pdf.

[11] Vitalik Buterin. Ethereum's white paper. https://github.com/ethereum/wiki/wiki/White-Paper, 2014.

[12] Aggelos Kiayias, Nikolaos Lamprou, and Aikaterini-Panagiota Stouka. *Proofs of Proofs of Work with Sublinear Complexity*, pages 61–78. Springer Berlin Heidelberg, 2016.

[13] Maksym Zavershynskyi. Eth-near rainbow bridge - near protocol. https://near.org/blog/eth-near-rainbow-bridge/, August 2020. (Accessed on 09/28/2020).

[14] Harmony Team. Harmony technical whitepaper – version 2.0. https://harmony.one/whitepaper.pdf, 2018. (Accessed on 10/04/2020).

[15] Rafael Pass and Elaine Shi. Hybrid consensus: Efficient consensus in the permissionless model. Cryptology ePrint Archive, Report 2016/917, 2016. http://eprint.iacr.org/2016/917.

[16] Yossi Gilad, Rotem Hemo, Silvio Micali, Georgios Vlachos, and Nickolai Zeldovich. Algorand: Scaling byzantine agreements for cryptocurrencies. In *Proceedings of the 26th Symposium on Operating Systems Principles*, SOSP '17, pages 51–68. ACM, 2017.

[17] Ittai Abraham, Dahlia Malkhi, Kartik Nayak, Ling Ren, and Alexander Spiegelman. Solida: A blockchain protocol based on reconfigurable byzantine consensus. In *Proceedings of the 21st International Conference on Principles of Distributed Systems*, OPODIS '17, 2017.

[18] Andrew Miller, Yu Xia, Kyle Croman, Elaine Shi, and Dawn Song. The honey badger of bft protocols. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, CCS '16, pages 31–42. ACM, 2016.

[19] Aggelos Kiayias, Alexander Russell, Bernardo David, and Roman Oliynykov. Ouroboros: A provably secure proof-of-stake blockchain protocol. In *Annual International Cryptology Conference*, pages 357–388. Springer, 2017.

[20] Juan Garay, Aggelos Kiayias, and Nikos Leonardos. The bitcoin backbone protocol: Analysis and applications. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 281–310. Springer, 2015.

[21] Juan Garay, Aggelos Kiayias, and Nikos Leonardos. The bitcoin backbone protocol with chains of variable difficulty. In *Annual International Cryptology Conference*, pages 291–323. Springer, 2017.

[22] Ittay Eyal and Emin Gün Sirer. Majority is not enough: Bitcoin mining is vulnerable. In Nicolas Christin and Reihaneh Safavi-Naini, editors, *Financial Cryptography and Data Security: 18th International Conference, FC 2014, Christ Church, Barbados, March 3-7, 2014, Revised Selected Papers*, pages 436–454. Springer Berlin Heidelberg, 2014.

[23] Blockchain takes way too long to sync · issue #2394 · ethereum/mist. https://github.com/ethereum/mist/issues/2394, 2017. (Accessed on 11/29/2018).

[24] Peter Todd. Merkle mountain range. https://github.com/opentimestamps/opentimestamps-server/blob/master/doc/merkle-mountain-range.md, 2012.

[25] 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.

[26] Ethereum and Consensys. BTC Relay: A bridge between the Bitcoin blockchain & Ethereum smart contracts. http://btcrelay.org/.

[27] tBTC: a decentralized redeemable BTC-backed ERC-20 token. https://docs.keep.network/tbtc/, 2020. (Accessed on 10/16/2020).

[28] Tal Baneth. Waterloo — a decentralized practical bridge between eos and ethereum by kyber network. https://blog.kyber.network/waterloo-a-decentralized-practical-bridge-between-eos-and-ethereum-1c230ac65524, February 2019. (Accessed on 10/28/2020).

[29] EOS.IO Team. EOS.IO technical white paper v2. https://github.com/EOSIO/Documentation/blob/master/TechnicalWhitePaper.md, March 2018. (Accessed on 10/28/2020).

[30] Bitcoin Wiki. Delegated proof of stake. https://en.bitcoin.it/wiki/Delegated_proof_of_stake, October 2020. (Accessed on 10/28/2020).

[31] Silvio Micali, Salil Vadhan, and Michael Rabin. Verifiable random functions. In *Proceedings of the 40th Annual Symposium on Foundations of Computer Science*, FOCS '99, pages 120–. IEEE Computer Society, 1999.

[32] Dan Boneh, Joseph Bonneau, Benedikt Bünz, and Ben Fisch. Verifiable delay functions. In *Advances in Cryptology – CRYPTO 2018*, volume 10991 of *Lecture Notes in Computer Science*, pages 757–788. Springer, 2018.

[33] Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the weil pairing. In *Proceedings of the 7th International Conference on the Theory and Application of Cryptology and Information Security: Advances in Cryptology*, ASIACRYPT '01, pages 514–532. Springer-Verlag, 2001.

[34] Dan Boneh, Craig Gentry, Ben Lynn, and Hovav Shacham. Aggregate and verifiably encrypted signatures from bilinear maps. In Eli Biham, editor, *Advances in Cryptology — EUROCRYPT 2003*, pages 416–432. Springer Berlin Heidelberg, 2003.