# MAP the Blockchain World: A Trustless and Scalable Blockchain Interoperability Protocol for Cross-chain Applications

Yinfeng Cao
The Hong Kong Polytechnic University
Hong Kong, China
csyfcao@comp.polyu.edu.hk

Jiannong Cao
The Hong Kong Polytechnic University
Hong Kong, China
csjcao@comp.polyu.edu.hk

Dongbin Bai
The Hong Kong Polytechnic University
Hong Kong, China
dong-bin.bai@connect.polyu.hk

Long Wen[*][†]
The Hong Kong Polytechnic University
Hong Kong, China
Derivation Technology Limited
Hong Kong, China
long.wen@derivation.info

Yang Liu
MAP Protocol
Singapore
phil@maplabs.io

Ruidong Li[*]
Kanazawa University
Kanazawa, Ishikawa, Japan
liruidong@ieee.com

## Abstract

Blockchain interoperability protocols enable cross-chain asset transfers or data retrievals between isolated chains, which are considered as one of the core infrastructure for Web 3.0. However, existing protocols either face severe scalability issues due to high on-chain and off-chain cost, or suffer from trust concerns because of centralized architecture.

In this paper, we propose MAP, a trustless blockchain interoperability protocol that relays cross-chain transactions across heterogeneous chains with high scalability. First, within MAP, we develop a novel *cross-chain relay* architecture, which integrates a unified relay chain and on-chain light clients of source chains, allowing the trustworthy retrieval and verification of heterogeneous cross-chain transactions. Furthermore, we reduce cross-chain verification cost by incorporating an optimized on-chain light client scheme that adaptively decouples signature verification overheads from inefficient smart contract execution and offloads them to off-chain provers. For experiments, we conduct the first large-scale evaluation on existing blockchain interoperability protocols. With MAP, the required number of on-chain light clients is reduced from $O(N^2)$ to $O(N)$, with decreasing on-chain cost 35% and 25% off-chain cost when verifying cross-chain transactions.

To demonstrate the effectiveness, we deployed MAP in the real world. By 2025, we have supported over six popular public chains, 50 cross-chain applications and relayed over 200K cross-chain transactions worth over 640 million USD. Based on the deployment records, we construct the first real-world cross-chain dataset to further advance blockchain interoperability research.

[*]Corresponding authors.
[†]This work was fulfilled when Long Wen was with The Hong Kong Polytechnic University.

## CCS Concepts

• **Networks** → **Peer-to-peer protocols**; • **Security and privacy** → **Cryptography**; *Web application security*.

## Keywords

Blockchain; Web 3.0; Interoperability; Cross-chain; Cryptocurrency; Decentralized Finance

**ACM Reference Format:**
Yinfeng Cao, Jiannong Cao, Dongbin Bai, Long Wen, Yang Liu, and Ruidong Li. 2025. MAP the Blockchain World: A Trustless and Scalable Blockchain Interoperability Protocol for Cross-chain Applications. In *Proceedings of the ACM Web Conference 2025 (WWW '25), April 28-May 2, 2025, Sydney, NSW, Australia.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3696410.3714867

## 1 Introduction

Blockchain is a decentralized ledger technology that uses cryptographic techniques and consensus mechanisms to achieve Byzantine Fault Tolerance (BFT), enabling decentralized trust and secure data sharing. Leveraging the philosophy of blockchain, the next generation of the web, known as Web 3.0, is being built. In recent years, a wide range of Web 3.0 applications are emerging, including cryptocurrencies, which revolutionize digital money, Decentralized Finance (DeFi) protocols that disrupt traditional financial systems, immersive virtual environments in the Metaverse, and various Decentralized Applications (DApps) [16, 17, 20].

**The Problem**. With the rapid development of Web 3.0, on-chain data and assets are increasingly being distributed across multiple blockchains. According to statistics, there are already over 1,000 public blockchains in the market, hosting more than 10,000 types of on-chain assets [41]. This extensive distribution creates a critical need for blockchain interoperability protocols, which enable the retrieval and transfer of on-chain data and assets between source and destination chains through cross-chain transactions [33, 40]. With interoperability, conventional DApps could leverage data and assets from multiple chains simultaneously, thereby supporting a wider range of applications. For example, cross-chain DeFi services can increase liquidity and offer diversified financial services by depositing and exchanging assets from different chains, such as cryptocurrencies, Non-Fungible Tokens (NFTs), and Real-world Assets (RWAs)[42]. Likewise, an interoperable Metaverse platform could enable users to access various virtual worlds, thus enriching their experiences [22].

However, there are three major technical challenges when making chains interoperable: *trust requirement*, *expensive verification*, and *chain heterogeneity*.

**Trust Requirement**. When processing cross-chain transactions, the interoperability protocol must maintain the same level of BFT security as typical public blockchains to avoid compromising overall security. This implies that the protocol should be decentralized and trustless. However, achieving this level of security is challenging, as the protocol must handle complex tasks such as cross-chain transaction retrieval, processing, and verification, while maintaining consistency and liveness. As a result, many solutions are centralized or semi-centralized, such as notary schemes and committee-based protocols [28, 35]. These are widely used by cryptocurrency exchanges but are vulnerable to internal corruption and attacks due to their reliance on trust. For example, one of the largest multi-party computation (MPC)-based cross-chain bridges, Multichain, was severely exploited due to allegedly compromised secret keys, leading to a financial loss of over 120 million USD [37, 38, 49].

**Expensive Verification**. As different blockchains do not trust each other, they must verify incoming cross-chain transactions to ensure they are valid and confirmed on the source chains. However, this verification process is expensive and inefficient, particularly when it is performed on-chain, as it involves numerous complex cryptographic operations and the storage of block headers. For example, verifying an Ethereum Virtual Machine (EVM)-compatible transaction through an on-chain Light Client (LC) consumes approximately 18 million gas, which is equivalent to about 60 USD on Ethereum at the time of writing [19]. Such high cost is mainly due to the storage of public keys and the signature verification process. Although cutting-edge solutions aim to reduce on-chain cost by zk-SNARKs, they still require significant off-chain computational resources for proof generation [19, 43, 46].

**Chain Heterogeneity**. Connecting heterogeneous chains via interoperability protocols presents additional challenges. Heterogeneous chains differ in their underlying components, such as smart contract engines, supported cryptographic primitives, parameters, and transaction formats. As a result, they cannot directly verify and confirm transactions from one another. For instance, an EVM chain like Ethereum cannot directly verify transactions from Solana because the EVM lacks support for the multi-signature scheme used in Solana transactions. Therefore, existing solutions either only support specific chain types [18, 44], or require significant modifications on the underlying components of chains to achieve compatibility [24], which are both not feasible for in-production public chains. LC-based bridges may suffer less from compatibility issues, but still need to redundantlh deploy LC contracts on each chain [19, 46, 48], as shown in Figure 1. This approach incurs quadratic complexity $O(N^2)$ when extending to additional chains, thus posing huge gas consumption and development burdens.

**Our Approach**. In this paper, we introduce MAP, a scalable and trustless blockchain interoperability protocol. At a high level, MAP aims to minimize the computational cost when scaling to new chains while maintaining decentralized security, without any underlying modifications on chains. Specifically, MAP employs a novel relay chain architecture as the intermediary to relay cross-chain transactions from source chains to destination chains. This architecture eliminates the need of deploying pairwise chain-to-chain light clients. Moreover, to reduce both the on-chain and off-chain cost when verifying transactions, we propose an optimized zk-based light client scheme, *hybrid light client*, which adaptively decouples the signature verification workloads [4] according to their diverse performance in on-chain smart contracts and off-chain circuits.

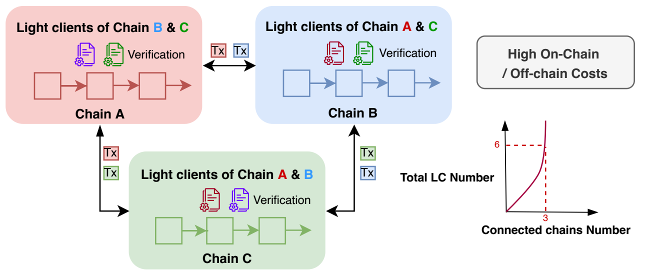
Light clients of Chain B & C
Verification
Chain A
Tx
Tx
Light clients of Chain A & C
Verification
Chain B
High On-Chain / Off-chain Costs
Tx
Tx
Tx
Tx
Light clients of Chain A & B
Verification
Chain C
Total LC Number
6
3
Connected chains Number

**Figure 1: The redundant number of on-chain light clients in LC-based solutions. To connect three chains A, B, and C, LC-based protocols must deploy the chain-to-chain LCs of chains B and C on chain A to allow chain A to verify transactions from those chains (and same for chains B and C), resulting in total $3 * 2 = 6$ ($O(N^2)$) LCs needed and posing heavy on-chain or off-chain workloads**

**Contributions**. In summary, MAP makes the following technical contributions:

- MAP introduces a unified relay chain to facilitate cross-chain transactions between heterogeneous chains, achieving decentralized security while reducing the required number of on-chain LCs from $O(N^2)$ to $O(N)$. Furthermore, the relay chain renders MAP chain-agnostic. When extending to new chains, only corresponding on-chain light clients are required to deploy.
- We develop a hybrid light client scheme based on zk-SNARKs that reduces both the on-chain and off-chain cost of verifying cross-chain transactions. We adaptively decouple the verification workloads of BLS signatures and proof generation based their performance in on-chain smart contracts and off-chain circuits. This scheme achieves a reduction in on-chain cost by 35% and off-chain cost by 25% compared to the existing state-of-the-art works.
- We evaluate the performance and security of MAP. Specifically, for performance, we are the first to perform large-scale measurements on existing interoperability protocols. For security, besides the cross-chain liveness and consistency proof, we identify and discuss a new security issue named *inter-chain security degradation* between interoperable chains.
- We deployed MAP on six public chains and support over 50 cross-chain applications, relaying over 200K real-world cross-chain transactions, worth over 640 million USD. Base on such practical experiences, we construct the first cross-chain

dataset, *BlockMAP*, containing over 150k cross-chain transactions across six chains. We also open-sourced all the codes of MAP, accompanied by detailed documentations.

## 2 Related Works

**Centralized/Committee-based Protocols**. To enable efficient interoperability, centralized designs are widely adopted by native protocols. Notary schemes directly host clients' tokens in custodial wallets and designate an authority (such as crypto exchanges) to facilitate their exchange efficiently [3, 9]. Similarly, committee-based protocols, such as MPC bridges and vote-oracle bridges[28, 35], appoint a small group of off-chain committees to verify and vote on cross-chain transactions, offering more decentralized features compared to notary schemes. Despite their convenience and efficiency, both solutions rely on trusting off-chain entities, which are usually not transparent and permissioned, making them vulnerable to internal corruption and attacks [28].

**Chain-based Protocols**. To further reduce the needed trust, chain-based protocols are developed, which process cross-chain transactions fully on-chain, thus making the protocols trustless. However, these protocols typically suffer from expensive verification and chain heterogeneity. Hash-Time Lock Contracts (HTLCs) are pioneering peer-to-peer protocols that allow users to deploy paired contracts on two chains to control asset release. However, HTLCs lack efficiency [2] because they require manual peer matching, enforcing users to wait for another user with the same token swap demand. As a result, HTLCs are rarely used to support large-scale cross-chain applications. Polkadot and Cosmos (Blockchain of Blockchain, BoB) employ hubs to process cross-chain transactions efficiently [18, 44], but these hubs only support their own specific homogeneous chains. HyperService [24] proposes a cross-chain programming framework, but it still requires significant modifications to the underlying components of heterogeneous chains, which is not feasible for in-production chains. LC-based bridges [19] are currently the mainstream protocols that deploy light clients (LCs) on each chain to verify cross-chain transactions. However, the internal verification workload of on-chain LCs is extremely expensive. ZKLC-based bridges [43, 46] attempt to reduce on-chain cost by moving verification to off-chain provers using zk-SNARKs. Unfortunately, this requires intensive computing power and multiple distributed servers due to the large circuit size of signature verification. Additionally, all LC-based protocols face high scaling cost due to redundant LCs.

**Cross-Shard**. Another related line of work involves blockchain sharding techniques [15, 21, 29, 31, 34, 47]. In these works, cross-shard processing techniques are developed to retrieve transactions from different shards. While these works share some similarities with cross-chain transaction processing, one key difference is that they only consider single blockchain scenarios, where all nodes trust each other and only simple transaction verification based on Authenticated Data Structure (ADS) is required, such as Merkle proof verification. In contrast, in cross-chain scenarios, blockchains that do not trust each other, and require complicated verification like block header verification.

## 3 System Model and Goals

**Interoperability Model**. In MAP, we consider the most general interoperability model that exists in most cross-chain applications. Within this model, there are typically two types of chains to communicate with each other: the source chain $\mathbb{SC}$ and the destination chain $\mathbb{DC}$. $\mathbb{SC}$ is the initiating entity, which first receives and confirms cross-chain transactions *ctx* from users and DApps. Then, a *blockchain interoperability protocol* is deployed between $\mathbb{SC}$ and $\mathbb{DC}$, responsible for transmitting *ctx* from $\mathbb{SC}$ to $\mathbb{DC}$. Different from interoperability in traditional databases or networking protocols, blockchain interoperability especially focuses on ensuring the verifiability and trustworthiness of *ctx* because of the trustless nature of blockchains.

**Transaction Model**. Interoperability between blockchains is implemented in the form of cross-chain transactions *ctx* in MAP. A *ctx* is a blockchain transaction from $\mathbb{SC}$ to $\mathbb{DC}$ containing the message or asset to be transferred. Formally it is defined as $ctx = \{\mathbb{DC}, payload\}$. The $\mathbb{DC}$ field is the chain id of $\mathbb{DC}$, which identifies the destination of *ctx*. *payload* field represents the types of *ctx*. When a *ctx* is an asset transaction, its *payload* contains the specific asset type, the amount, and the asset operation instructions; when a *ctx* is a message transaction, its *payload* contains the smart contract calls. In MAP, different types of *ctx* are handled in the identical way.

**Design Goals**. MAP has the following design goals:

(1) **Trustless**. Maintaining the same level of BFT security as typical public blockchains.
(2) **Scalability**. Gas-efficient and computationally efficient when processing cross-chain transactions and scaling to new chains.
(3) **Chain-agnostic**. When extending to new chains, no underlying modifications needed except deploying new smart contracts.

## 4 MAP Protocol

### 4.1 Overview

As shown in Figure 2, there are two pipelined phases of cross-chain relay in MAP:
(Phase 1. $\mathbb{SC}$ - $\mathbb{RC}$ relay). First, cross-chain transactions *ctx* are firstly committed by users or DApps and confirmed on the source chain $\mathbb{SC}$ (❶). Then, an off-chain server *prover* will proactively monitor this confirmation event and retrieve the *ctx* with its associated proofs issued by $\mathbb{SC}$, such as headers and Merkle proofs (❷). Then the *ctx* and its proofs are sent to the unified relay chain $\mathbb{RC}$ by *prover* for generating proofs (❸).

The *unified relay chain* $\mathbb{RC}$ is an intermediary blockchain that processes cross-chain transactions between source and destination chains in a unified manner. More specifically, $\mathbb{RC}$ integrates multiple *hybrid on-chain* LCs of each $\mathbb{SC}$ (our zk-SNARKs-based optimized version of LCs, details in §4.3), which receive *ctx*s from *prover* and verify whether they are legal and already confirmed on $\mathbb{SC}$ (❹). After the verification, the *ctx*s are temporarily confirmed and appended to $\mathbb{RC}$.
(Phase 2. $\mathbb{RC}$ - $\mathbb{DC}$ relay). Similar with phase 1, there is another off-chain server *prover* retrieving *ctx*s from $\mathbb{RC}$ (❺). *prover* generates the proofs of *ctx*s for verification on the destination chain $\mathbb{DC}$ (❻).

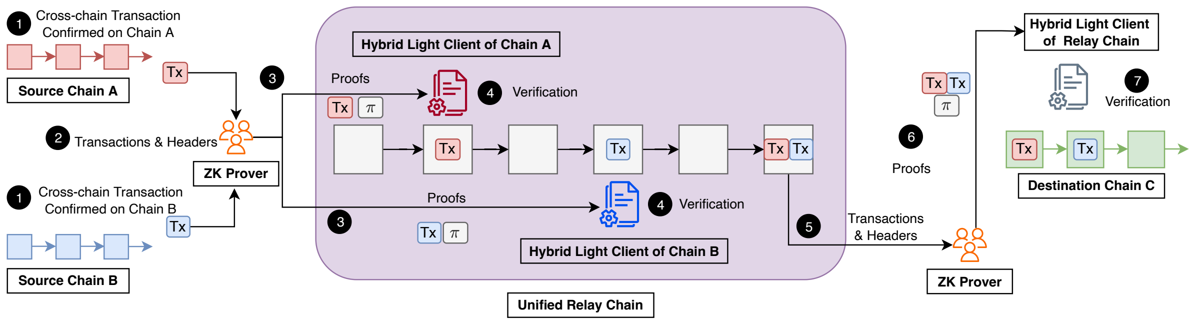

**Figure 2: Overview of MAP: We introduce a unified relay chain architecture to facilitate cross-chain communications, which continually retrieves and verifies cross-chain transactions from source blockchains. Transactions are verified by hybrid light clients, which are implemented by smart contracts and off-chain provers.**

On each $\mathbb{DC}$, an *identical* hybrid on-chain LC of $\mathbb{RC}$ is deployed, which verifies whether *ctxs* confirmed on $\mathbb{RC}$.

Recalling the overall procedures, the *ctxs initially committed* to $\mathbb{SC}$ are eventually confirmed on $\mathbb{DC}$, thus finalizing the entire cross-chain transaction relay (❼).

Note that there could be multiple $\mathbb{SC}$ and $\mathbb{DC}$ pairs in MAP, and the relay process is executed in the same way for each pair. Besides, $\mathbb{SC}$ and $\mathbb{DC}$ are relative, and they could be reversed during relay processes.

## 4.2 Unified Relay Chain

**Insights**. To address the challenges of trust and heterogeneity, we present two key insights for designing the architecture of blockchain interoperability protocols: (1) Only a BFT system can maintain the same security level with connected blockchains, thus avoiding degradation of overall security. Therefore, the overall architecture must be BFT-secure, such as a blockchain. (2) For decentralized protocols such as (ZK)LC-based bridges, the number of LCs on each chain is actually *overlapping and redundant*. That is, each chain only considers how to verify other chains from its own perspective (i.e., deploying other chains' LCs linearly), ignoring that the same type of LC can be deployed multiple times from a global perspective. For example, as shown in Figure 1, each type of LC is actually deployed twice. Therefore, if a new entity is able to verify transactions regardless of whether they come from different heterogeneous chains, the heterogeneity challenge could be resolved.

**Architecture**. To consolidate the above insights, we introduce the relay chain $\mathbb{RC}$ as the cross-chain intermediary in MAP. First, $\mathbb{RC}$ is also a blockchain that primarily responsible for receiving transactions from the source chain, verifying them, and forwarding verified transactions to the destination chain. This relay chain fundamentally ensures that MAP maintains decentralized security and trustworthiness.

Moreover, to address the challenge of chain heterogeneity, we adopt a *unified processing* strategy that enables $\mathbb{RC}$ to efficiently verify *ctx* from different heterogeneous chains, thus minimizing the number of LCs on $\mathbb{SC}$ and $\mathbb{DC}$. Specifically, MAP uses the on-chain LCs for cross-chain transaction verification. However, unlike existing LC-based bridges that require each of the LCs to be deployed on every other chain, we instead integrate the LCs of different chains into a single $\mathbb{RC}$. Consequently, all on-chain LCs $\Pi^{sc}_{hlc} = \langle \Pi^{sc_1}_{hlc}, \Pi^{sc_2}_{hlc}, \ldots, \Pi^{sc_i}_{hlc} \rangle$ are built on $\mathbb{RC}$ (the internal process of $\Pi^{sc}_{hlc}$ will be introduced in §4.3).

**Cross-Chain Relay**. The general process of relaying *ctx* from source chain $\mathbb{SC}_i$ to $\mathbb{DC}$ works as follows. As shown in the Algorithm 1, there are two pipelined phases.

First, for the $\mathbb{SC}_i - \mathbb{RC}$ phase, after *ctx* is committed and confirmed on $\mathbb{SC}_i$, it will emit a confirmation event by outputting the block header $bh^{sc_i}$ with the Merkle tree root $r^{sc_i}_{mkl}$ (line 2). Then a *prover* between $\mathbb{SC}_i$ and $\mathbb{RC}$ will monitor this confirmation event and proactively retrieve the *ctx* and generate the proofs $\langle ctx, bh^{sc_i}, \pi^{sc_i}_{mkl}, \pi^{sc_i}_{zk} \rangle$ (line 4-5) from $\mathbb{SC}_i$ and transmit them to $\mathbb{RC}$ (line 6). Then $\mathbb{RC}$ verifies these transactions against the corresponding $\Pi^{sc_i}_{hlc}$ of $\mathbb{SC}_i$ built on $\mathbb{RC}$. After verification, the *ctx* are confirmed on $\mathbb{RC}$ as *intermediary cross-chain transactions* $\widehat{ctx}$.

Then, in the second $\mathbb{RC} - \mathbb{DC}$ phase, $\widehat{ctx}$ will also emit a confirmation event to $\mathbb{RC}$ by outputting the block header $bh^{rc}$ with the Merkle tree root $r^{rc}_{mkl}$ (line 9). Then a *prover* between $\mathbb{RC}$ and $\mathbb{DC}$ will get the $\widehat{ctx}$ and generate its proofs $\langle \widehat{ctx}, bh^{rc}, \pi^{rc}_{mkl}, \pi^{rc}_{zk} \rangle$ (line 11-12). These proofs are transmitted to $\mathbb{DC}$ for further verification (line 13). The key difference here is that only one identical type of $\Pi^{rc}_{hlc}$ needs to be deployed on each $\mathbb{DC}$ (line 15) to verify $\widehat{ctx}$. This is because all $\widehat{ctx}$ are now from $\mathbb{RC}$, even though they were originally from different $\mathbb{SC}_i$. After passing the verification of $\Pi^{rc}_{hlc}$, the $\widehat{ctx}$ are confirmed on $\mathbb{DC}$ as the finalized cross-chain transactions *ctx*.

**Consensus**. To ensure the decentralized security of the relay process on $\mathbb{RC}$, we make $\mathbb{RC}$ run a BFT consensus (e.g., a Proo-of-Stake(PoS) BFT consensus like IBFT [26]). Generally, it enforces honest nodes with economic incentives, while punishing malicious behavior by slashing staked tokens. Validators will be motivated to participate and behave honestly because of token rewards from staking and processing cross-chain transactions.

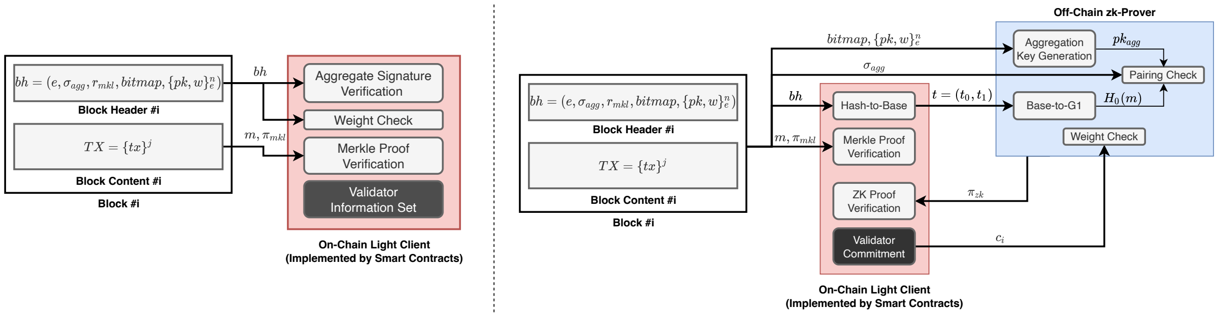

**Figure 3: Our hybrid light client overperforms conventional light clients by adoptive offloading. We move the on-chain verification workloads to off-chain provers through zk-SNARKs. Meanwhile, we keep the hash operations on-chain to minimize the circuits size and proof generation time.**

**Algorithm 1:** Unified Relay Chain in MAP

```
Input: A cross-chain transaction $ctx$ from $\mathbb{SC}_i$ to $\mathbb{DC}$
Output: Updated $\mathbb{DC}$ by $ctx$
Procedure SourceChain($ctx$):
    $(bh^{sc_i}, r^{sc_i}_{mkl}) \leftarrow$ confirm($ctx$, $\mathbb{SC}_i$) ▷ $ctx$ is firstly committed and confirmed on $\mathbb{SC}_I$
    for prover between $\mathbb{SC}_i$ and $\mathbb{RC}$ do
        retrieves $(bh^{sc_i}, r^{sc_i}_{mkl})$ emitted by $ctx$ from $\mathbb{SC}_i$
        $\pi^{sc_i}_{mkl}, \pi^{sc_i}_{zk} \leftarrow$ genProof $(bh^{sc_i}, r^{sc_i}_{mkl}, ctx)$
        return transmit($ctx, bh^{sc_i}, \pi^{sc_i}_{mkl}, \pi^{sc_i}_{zk}, \mathbb{RC}$)
    end
Procedure RelayChain($ctx, bh^{sc_i}, \pi^{sc_i}_{mkl}, \pi^{sc_i}_{zk}$):
    if $\Pi^{sc_i}_{hlc}(ctx, bh^{sc_i}, \pi^{sc_i}_{mkl}, \pi^{sc_i}_{zk}) == True$ then
        $(bh^{rc}, r^{rc}_{mkl}) \leftarrow$ confirm($ctx$,$\mathbb{RC}$) ▷ $ctx$ is verified and confirmed on $\mathbb{RC}$ by corresponding $\mathbb{SC}_i$'s light client
        for prover between $\mathbb{RC}$ and $\mathbb{DC}$ do
            retrieves $(bh^{rc}, r^{rc}_{mkl})$ emitted by $ctx$ from $\mathbb{RC}$
            $\pi^{rc}_{mkl}, \pi^{rc}_{zk} \leftarrow$ genProof $(bh^{rc}, r^{rc}_{mkl}, ctx)$
            return transmit($\widehat{ctx}, bh^{rc}, \pi^{rc}_{mkl}, \mathbb{DC}$)
        end
    end
Procedure DestinationChain($\widehat{ctx}, bh^{rc}, \pi^{rc}_{mkl}, \pi^{rc}_{zk}$):
    if $\Pi^{sc}_{hlc}(\widehat{ctx}, bh^{rc}, \pi^{rc}_{mkl}, \pi^{rc}_{zk}) == True$ then
        $(bh^{dc}, r^{dc}_{mkl}) \leftarrow$ confirm($\widehat{ctx}$, $\mathbb{DC}$) ▷ $ctx$ is finally verified and confirmed on $\mathbb{DC}$ by $\mathbb{RC}$'s light client
        return $\mathbb{DC}$
    end
```

## 4.3 Hybrid Light Client

Although introducing the relay chain can effectively reduce the required number of on-chain LCs through unified processing, the heavy on-chain LC verification workload remains a bottleneck [46, 48].

**On-chain Verification**. To explore potential optimization spaces, we analyze the cost of each procedure in normal EVM-PoS light clients. After a transaction $tx$ is committed and finalized by consensus, a block $B$ and its header $bh$ will be produced and appended on chain [11]. To prove that such $tx$ is included in $B$, the following major content needs to be inputted to normal light client $\Pi_{lc}$:

- a receipt message $m$ emitted by $tx$ inside $B$.
- a Merkle proof $\pi_{mkl}$ for $m$ extracted from $B$, which is usually provided by full nodes.
- a header $bh = (\{pk, w\}^n, \sigma_{agg}, bitmap, r_{mkl})$ that consists of:
  - an epoch number $e$.
  - a current validator information set $vs_e = \{pk, w\}^n_e$ that contains $n$ validator public keys and corresponding voting weights corresponding to $e$. When consensus entering a new epoch, a new validator information set will be updated.
  - an aggregate signature $\sigma_{agg}$ from validators signing $B$.
  - a mapping value $bitmap$ that indicates which validator actually signed $B$.
  - a root hash of receipt trie $r_{mkl}$ that is computed from $m$.
- other auxiliary information such as timestamp and epoch size $E$

With above input content, the normal $\Pi_{lc}$ is defined as three algorithms (Setup, Update, Verify), as shown in Figure 3 (left):

- $vs_g \leftarrow$Setup($para$): given system parameters, $para$, initialize $\Pi_{lc}$ in terms of the epoch size, $E$, the vote threshold, $T$, and the initial validator information, $\{pk, w\}^n_g$. Then output a validator set, $vs_g = \{pk, w\}^n_g$, that indicates the current validator set stored in $\Pi_{lc}$.
- $vs_{e+1} \leftarrow$Update($e, vs_e, bh$): given a header $bh$ with an epoch change, verify the aggregate signature $\sigma_{agg}$ inside $bh$ and update the current validator set $vs_e$ to a new validator set $vs_{e+1} = \{pk, w\}^n_{e+1}$.
- $\{0, 1\} \leftarrow$Verify($vs_e, m, bh, \pi_{mkl}$): given a message, $m$, emitted from $tx$ and its header, $bh$, check whether $tx$ is successfully included in $B$ through its aggregate signature $\sigma_{agg}$, vote weights, and its Merkle proof $\pi_{mkl}$. Output $\{0, 1\}$ as the result. The incremental increase in the epoch number, $e$, is also verified during the signature verification.

**Efficiency Optimization Space**. Computation and storage are the main overheads when triggering `Update` and `Verify`. For computation, aggregate signature verification is frequently performed, which is essentially operations on elliptic curves, including hashing (i.e., the Hash-to-Curve algorithm), equation evaluations, and pairing checks[1, 6, 13]. These operations are inefficient in EVM due to their relatively high complexity when calculating underlying fields via curve equations. For instance, currently verifying one EVM cross-chain transaction with full BLS signatures can cost up to $1 \times 10^6$ gas[48] (approximately 30 USD on ETH). For storage, $\Pi_{lc}$ needs to store $vs_e = \{pk, w\}_e^n$ persistently and frequently read them. Since most of PoS blockchains have more than 100 validators, storing and updating these data at the end of each epoch on smart contracts requires a large amount of storage space, thus consuming a expensive gas fees. Storing one validator information set requires $0.1 \times 10^6$ gas.

**Hybrid Verification**. To estimate the high gas fee consumption, we develop a hybrid verification scheme $\Pi_{hlc}$ to reduce on-chain cost using off-chain zk-SNARKs.

First, we aim to efficiently prove the two functions `Update` and `Verify` using zk-SNARKs. We compress the validator information $vs$ into a single commitment: $vs = \text{commitment}(\{(pk_0, w_0), (pk_1, w_1), \ldots, (pk_n, w_n)\})$ to reduce the on-chain storage overhead. In this way, the validator aggregate signatures of $bh$ and the corresponding voting weights must satisfy this commitment value to pass verification. One native approach to implementing zk-SNARKs for proving is to program and compile all verification procedures into circuits, i.e., input the entire block header into the circuit along with all signature verification algorithms [39, 46]. Then deploy an off-chain prover to generate the zk-proofs based on this circuit and submits them to $\Pi_{hlc}$ for verification.

However, we observe that despite $\Pi_{hlc}$ improving efficiency by shifting on-chain workloads to off-chain provers, generating zk-proofs for verifying the entire aggregate signature instead requires substantial off-chain storage and computational resources for the prover. Specifically, in this way, the circuit size for an aggregate signature verification is extremely large due to multiple complex operations such as *Hash-to-Curve* and pairing checks (e.g., typically exceeding $2 \times 10^7$ gates and 100 GB for a block with eight signatures[12]). These factors also increase the proof generation time.

To optimize the off-chain cost of generating zk-proofs, we try to decouple the aggregate signature verification process and handle it separately. Specifically, the *Hash-to-Curve* algorithm in the BLS scheme hashes the message $m$ to curve points in $\mathbb{G}$, which typically consists of two steps in practical implementations:

(1) *Hash-to-Base.* Input a a message $m$ and map it to possible coordinates (base field elements) through hash functions. This returns a field element $t$.
(2) *Base-to-G*. Input a field element $t$ and calculate the curve point $(x, y)$ through the curve equations.

Since *Hash-to-Base* mainly consists of multiple hash operations, it can be efficiently computed through smart contract but inefficiently compiled into circuits due to its large size. In contrast, *Base-to-G* performs arithmetic operations in the finite field through elliptic curve equations, which can be relatively briefly and efficiently expressed into circuits. In this way, we improve the off-chain efficiency of zk-SNARKs based aggregate signature verification, further speed up the entire $\Pi_{hlc}$.

With the above optimization, the $\Pi_{hlc}$ is defined as the following algorithms, as shown in Figure 3(right):

- $vs_g \leftarrow$`Setup`$(para)$: given the system parameters, $para$, initialize $\Pi_{hlc}$ with the hard-coded epoch size, $E$, the vote threshold, $T$, and the initial validator information commitment, $vs_g = C(\{pk, w\}_g^n)$. Then, output a validator set, $vs_g = \{pk, w\}_g^n$, that indicates the current validator set stored in $\Pi_{hlc}$.
- $vs_{e+1} \leftarrow$`Update`$(vs_e, h, \pi_{zk})$: given header $bh$ during an epoch change, verify the aggregate signature, $\sigma_{agg}$, of $bh$. First, compute the base field elements $t = (t_0, t_1)$ in $G_1$ by hash function $H_0(bh)$, and send $t$ to the prover. After receiving $\pi_{zk}$ that satisfied $c$, update the current validator set, $vs_e$, with the new validator set, $vs_{e+1} = C(\{pk, w\}_{e+1}^n)$.
- $\pi_{zk} \leftarrow$`GenZK`$(bitmap, vs_e, \sigma_{agg}, t, )$: given extracted $bitmap$, $vs_e = \{pk, w\}_e^n$, $\sigma_{agg}$, validator set commitment $c$ from $vs_e$ and $t$ from `Update`, run a zk-SNARKs system and generate a zk-proof, $\pi_{zk}$, for $c$.
- $\{0, 1\} \leftarrow$`Verify`$(vs_e, m, h, \pi_{mkl})$: given message $m$ emitted from $tx$ and its header $bh$, verify whether $tx$ is successfully included in $B$ through its aggregate signature, $\sigma_{agg}$, and there are sufficient weights according to the stored $vs_e$ and its Merkle proof $\pi_{mkl}$. Then output $\{0, 1\}$ as the result.

## 5 Performance Evaluation

**Experiment Setup**. We set up two Google Compute Engine machine type c2d-highcpu-32 instance (32 vCPUs with 64GB RAM, ~800 USD per month) as provers. For the relay chain, the hardware configuration for validator is similar with e2-standard-4 (4 vCPUs with 16 GB RAM).

**Baselines and Workloads**. Since very few works provide quantitative performance evaluation results, it is difficult to find an available and common baseline to ensure fairness [33, 36, 40]. To this end, we perform the first comprehensive measurement and comparison of existing blockchain interoperability protocols. As shown in Table 1, we measure five key security and scalability metrics across six representative types of protocols. We set the cross-chain transactions from *Polygon to Ethereum* as workloads for comparison, which is mostly supported by existing works. For protocols that do not support such workloads (such as Polkadot), we select their popular source-destination chain pair for evaluation. For each type of workload, we measure 100 transactions and record the average result or cost per cross-chain transaction.

### 5.1 Evaluation Results

**On-chain Costs**. For on-chain cost, we mainly refer to the LC-based bridges as baselines, because they are the most common decentralized solutions [48]. For each cross-chain transaction verification, on-chain LCs require ~$1 \times 10^6$, while MAP requires only ~0.65M, saving ~35%. These cost are deterministic in repeated tests on smart contracts [11, 45].

| Evaluation Metrics | Centralized | Committee | Chain | | | | |
|---|---|---|---|---|---|---|---|
| Solutions | Binance, CoinBase[3][9] | Multichain, Celer[28][10] | EthHTLC[7], Lighting[2] | Polkadot, Cosmos[44][18] | Horizon, LayerZero,[19][48] | zkRelay, zkBridge[43][46] | MAP |
| Types | Notary | MPC | HTLC | BoB | LC | ZKLC | **ZKLC+Relay** |
| Security Models | Trusted | Semi-Trusted | Trustless | Trustless | Trustless | Trustless | **Trustless** |
| On-chain Costs (gas) | N/A | $0.5 \times 10^6$ | $1.5 \times 10^6$ | $0.8 \times 10^5$ | $1 \times 10^6$ | $0.3 \times 10^6$ | **$0.65 \times 10^6$ (35% ↓)** |
| Off-chain Costs (gates) | N/A | N/A | N/A | N/A | N/A | $2 \times 10^7$ | **$1.57 \times 10^7$ (25% ↓)** |
| Latency | 1s | 310s | N/A | 13s | 227s | 153s | **210s** |
| Complexity | $O(N)$ | $O(N^2)$ | $O(N^2)$ | $O(N)$ | $O(N^2)$ | $O(N^2)$ | **O(N)** |

**Table 1: Performance Comparisons of MAP and Existing Blockchain Interoperability Protocols. Results are approximate and mainly from Polygon to Ethereum transaction workload (per cross-chain transaction cost).**

**Table 2: Circuit size of provers for verifying different number of validator signatures**

| Number of Sig. per ctx | Circuit Size |
|---|---|
| 4 | $0.9 \times 10^6$ gates |
| 8 | $15.7 \times 10^6$ gates |
| 16 | $25.2 \times 10^6$ gates |
| 32 | $49.3 \times 10^6$ gates |

**Off-chain Costs**. For off-chain cost caused by zk-SNARKs, we refer to the standard implementation using snarkjs Groth16 to prove the signature verification scheme in transaction verification as the baseline [12, 39, 46]. As shown in Table 2, for eight signatures, the circuit size of the MAP prover is ~$1.57 \times 10^7$ gates, which is reduced by ~25% compared to the aforementioned baselines ($2 \times 10^7$ gates). Correspondingly, the proof generation time is also reduced by ~25% due to its linear relationship with circuit size. Moreover, MAP only needs a single sever to generate proofs rather than multiple server settings [46], which further reduces the off-chain cost in deploying and maintaining processes.

**Number of On-chain Light Clients (Scaling Up Cost)**. According to statistics[1], a PoS-BFT EVM light client requires approximately ~100K gas per validator information storage. Assuming the number of validators is 100 for each chain, then for connecting $N$ chains, LC-based bridges need to spend $10^7 \times N(N-1)$ gas to deploy LCs. In contrast, for MAP, it is ~100K gas fixed per LC for validator information set commitment storage (no matter how many validators), which means only $2 \times 10^5 \times N$ gas is needed. Moreover, it avoids establishing communication channels with every other chain. Instead, each chain only needs to ensure the communications with the relay chain, which further makes MAP more practical.

**Cross-chain Latency**. We measure the end-to-end latency of cross-chain transactions relayed in MAP, from the confirmation timestamp on source chains until the confirmation timestamp on destination chains, including transaction transmission between chains, proof generation, and on-chain LC verification. As shown in Table 1, the results indicate that MAP's cross-chain latency is ~210 seconds. Compared to existing works, these results suggest that despite introducing provers and relay chain will slightly increase latency. However, the overall impact is negligible, as the latency for cross-chain applications is not prioritized like conventional chains in practice.

[1]https://github.com/shresthagrawal/poc-superlight-client

## 6 Security Analysis

We thoroughly analyze the security of MAP. Particularly, as previous works have extensively proved the transaction liveness and consistency within a single PoS-BFT chain [30, 32], we focus on demonstrating the newly introduced components (i.e., provers and relay chain) in MAP will maintain the cross-chain liveness and consistency under various attacks.

### 6.1 Assumptions

MAP works under several basic and common security assumptions [36] [46].

Assumption 1. ***(PoS-BFT Threshold).*** *For $\mathbb{RC}$, more than $\tau = \frac{2S}{3}$ of the stakes are controlled by honest validators, where $S$ is the total stakes. This group of honest validators is always live, i.e., they will confirm ctx in a timely manner.*

Assumption 2. ***(Secure Cryptographic Primitives).*** *The cryptographic primitives used in MAP, including the BLS signature, the Groth16 zk-SNARKs, and the hash functions, are secure against probabilistic polynomial-time (PPT) adversaries. That is, no PPT adversary can generate incorrect proofs or signatures that would be accepted.*

Assumption 3. ***(Minimal Prover and Reachable Communication).*** *At least one prover is available and honest in MAP, i.e., they will correctly generate the proofs $\pi_{mkl}$ and $\pi_{zk}$ and transmit cross-chain transactions ctx between chains, i.e., $\mathbb{SC}$, $\mathbb{RC}$, and $\mathbb{DC}$. Additionally, we assume that the communication channels between the provers and the chains are reachable (i.e., no network partitions, though they may be insecure).*

Remark 1. *If Assumption 3 does not hold, chains will be isolated and not interoperable in any sense.*

### 6.2 Liveness and Consistency

Theorem 1. ***(Cross-chain Liveness).*** *If a valid ctx is committed to and confirmed on $\mathbb{SC}$, then it will eventually be confirmed on $\mathbb{DC}$ via MAP, assuming the above assumptions hold.*

Proof. Given a committed $ctx$ from $\mathbb{SC}$, there are two potential cases that could prevent it from being confirmed on $\mathbb{DC}$: *Case 1*: A faulty or compromised *prover* refuses to generate proofs and transmit $ctx$ between SC-RC or RC-DC. *Case 2*: Sufficient validators of RC are corrupted to force $\mathbb{RC}$ to withhold $ctx$, preventing it from being sent to $\mathbb{DC}$. For Case 1, by Assumption 3, at least one *prover* will transmit $ctx$ to $\mathbb{RC}$ and $\mathbb{DC}$ (a single *prover* is sufficient for processing transactions from any number of chains). Therefore, even if other *provers* are faulty or compromised (e.g., via DDoS attacks), $\mathbb{RC}$ and $\mathbb{DC}$ can still receive and verify $ctx$ from the reliable *prover*. For Case 2, previous works have proven that any liveness attacks on PoS-BFT chains involving the refusal to verify transactions require at least $\frac{1S}{3}$ stakes [30, 32], which is prevented by Assumption 1. Even in the case of DDoS attacks on partial the $\mathbb{RC}$ validators, since the honest validators are live and control over $\frac{2S}{3}$, they will always confirm the $ctx$ in time. As a result, $\Pi_{hlc}$ run by the validators will eventually verify $ctx$ and confirm it on both $\mathbb{RC}$ and $\mathbb{DC}$, thereby guaranteeing the overall cross-chain liveness. □

Theorem 2. ***(Cross-chain Consistency)****. If a valid $ctx$ is committed and confirmed on $\mathbb{SC}$ and a $\underline{ctx}$ is finally confirmed on $\mathbb{DC}$ via* MAP*, then $ctx = \underline{ctx}$, assuming the above assumptions hold.*

Proof. Given a $ctx$ from $\mathbb{SC}$, there are two potential cases for consistency attacks: *Case 1*: A malicious *prover* generates a tampered $\underline{ctx}$ with its proofs and tries to get them accepted by $\mathbb{RC}$. *Case 2*: Adversaries directly corrupt $\mathbb{RC}$ to force it to accept a tampered $\underline{ctx}$. For Case 1, in order to pass $\Pi_{hlc}$ verification, the malicious *prover* would need to forge block headers (including the corresponding signatures and Merkle proofs) to generate incorrect proofs. However, by Assumption 2, this is highly unlikely to succeed. Therefore, $\Pi_{hlc}$ will not accept $\underline{ctx}$ as a valid cross-chain transaction on $\mathbb{RC}$. For Case 2, corrupting $\mathbb{RC}$ to accept a tampered $\underline{ctx}$ requires controlling at least $\frac{2S}{3}$ of the validators, which is prevented by Assumption 1. Therefore, any tampered $\underline{ctx}$ will not be accepted on $\mathbb{RC}$, thus ensuring cross-chain consistency. □

## 6.3 Inter-Chain Security

Despite the analysis in §6.2 proving that cross-chain transaction verification is secure under Assumptions 1, 2, and 3, it does not fully match cross-chain scenarios. Specifically, within a single chain, the profit-from-corruption can hardly be higher than cost-to-corruption because they are calculated by the relative token value. That is, within a chain A with security threshold $\tau_A = \frac{2S_A}{3}$, it is unlikely to see a transaction with value over $\tau_A$.

We identify a new security issue when connecting multiple chains with interoperability protocols that may converse the above situation, which also applies to other chain-based protocols but not well discussed before. We name this issue *Inter-Chain Security Degradation*. This issue indicates that the overall security of interoperable multi-chain networks is as strong as the least secure chain. For example, given three interoperable PoS-BFT chains A, B, and C, with their BFT security boundaries as $\tau_A = \frac{2S_A}{3}$, $\tau_B = \frac{2S_B}{3}$, and $\tau_C = \frac{2S_C}{3}$, the security of the entire network is $\min(\tau_A, \tau_B, \tau_C)$. This can be justified by considering the following situation: assume $\tau_B = \min(\tau_A, \tau_B, \tau_C)$. If a $ctx$ from chain A to chain B has

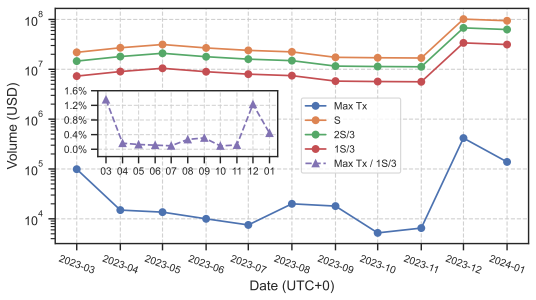

**Figure 4: Historical statistics of MAP: The maximum value of any single cross-chain transaction is significantly smaller than the security boundary of the relay chain**

an extremely large value $V_{extreme} > \tau_B$, the validators of chain B will be sufficiently motivated to manipulate $ctx_{extreme}$ (such as double-spending), even if they were honest before (Assumption 1) and run the risk of being slashed by all the staked. This is because their profit-from-corruption is now explicitly higher than cost-to-corruption. In other words, the security of chains A, B, and C is *degraded to* $V_{extreme} < \tau_B$ due to the existence of interoperability.

**Discussion** Regarding MAP, this degradation requires the security of the relay chain to be strong enough (i.e., high staked value) to support cross-chain transactions. To examine the risk, we provide real-world statistics from a period of MAP. As shown in Figure 4, the most valuable cross-chain transaction was a 100K USDC transfer from NEAR in March 2023[2], worth 1.3% of the total MAP stakes (7M USD), far away from the security threshold. This also indicates MAP could still support transactions worth up to 4.67M USD.

In summary, although inter-chain security degradation is unavoidable due to the mismatched economic security level of different connected chains, MAP is still highly secure in practical scenarios.

## 7 Conclusion

This paper introduces MAP, a trustless and scalable blockchain interoperability protocol with practical implementations. MAP strikes a balance between security and scalability by introducing a unified relay chain architecture and optimized zk-based hybrid light clients (LCs). We conducted extensive experiments to comprehensively evaluate its performance and analyze its security. We envision MAP as a practical solution for interoperable data and networking infrastructure in the Web 3.0 era.

## Acknowledgments

The work described in this paper was partially supported by a grant from HK RGC Research Impact Fund (No. R5034-18) and the Research Institute for Artificial Intelligence of Things, The Hong Kong Polytechnic University. We also thank the anonymous reviewers for their valuable comments and suggestions.

[2]https://maposcan.io/cross-chains/565

## References

[1] Zachary Williamson Antonio Salazar Cardozo. 2018. EIP-1108: Reduce alt-bn128 precompile gas costs. https://eips.ethereum.org/EIPS/eip-1108 Last accessed 14 Oct 2024.

[2] Andreas M Antonopoulos, Olaoluwa Osuntokun, and René Pickhardt. 2021. *Mastering the Lightning Network.* " O'Reilly Media, Inc.".

[3] Binance. 2023. Binance - Cryptocurrency Exchange for Bitcoin, Ethereum. https://www.binance.com Last accessed 14 Oct 2024.

[4] Dan Boneh, Craig Gentry, Ben Lynn, and Hovav Shacham. 2003. Aggregate and verifiably encrypted signatures from bilinear maps. In *Advances in Cryptology—EUROCRYPT 2003: International Conference on the Theory and Applications of Cryptographic Techniques, Warsaw, Poland, May 4–8, 2003 Proceedings 22.* Springer, 416–432.

[5] Dan Boneh, Craig Gentry, Ben Lynn, Hovav Shacham, et al. 2003. A survey of two signature aggregation techniques.

[6] Vitalik Buterin. 2017. EIP-198: Big integer modular exponentiation. https://eips.ethereum.org/EIPS/eip-198 Last accessed 14 Oct 2024.

[7] chatch. 2024. Hashed Timelock Contracts for ETH, ERC20 and ERC721 on Ethereum. https://github.com/chatch/hashed-timelock-contract-ethereum Last accessed 8 Mar 2024.

[8] Panagiotis Chatzigiannis, Foteini Baldimtsi, and Konstantinos Chalkias. 2022. Sok: Blockchain light clients. In *International Conference on Financial Cryptography and Data Security.* Springer, 615–641.

[9] Coinbase. 2023. Coinbase - Buy and Sell Bitcoin, Ethereum, and more with trust. https://www.coinbase.com Last accessed 14 Oct 2024.

[10] Mo Dong, Qingkai Liang, Xiaozhou Li, and Junda Liu. 2018. Celer network: Bring internet scale to every blockchain. *arXiv preprint arXiv:1810.00037* (2018).

[11] Ben Edgington. 2023. Upgrading Ethereum, A technical handbook on Ethereum's move to proof of stake and beyond. https://eth2book.info/capella/ Last accessed 14 Oct 2024.

[12] Electron-Labs. 2023. ED25519 implementation in Circom. https://github.com/Electron-Labs/ed25519-circom Last accessed 14 Oct 2024.

[13] Christopher Gorman. 2020. EIP-3068: Precompile for BN256 HashToCurve Algorithms. https://eips.ethereum.org/EIPS/eip-3068 Last accessed 14 Oct 2024.

[14] Jens Groth. 2016. On the size of pairing-based non-interactive arguments. In *Advances in Cryptology–EUROCRYPT 2016: 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Vienna, Austria, May 8-12, 2016, Proceedings, Part II 35.* Springer, 305–326.

[15] Zicong Hong, Song Guo, Enyuan Zhou, Wuhui Chen, Huawei Huang, and Albert Zomaya. [n. d.]. GriDB: Scaling Blockchain Database via Sharding and Off-Chain Cross-Shard Mechanism. ([n. d.]).

[16] Junqin Huang, Linghe Kong, Guanjie Cheng, Qiao Xiang, Guihai Chen, Gang Huang, and Xue Liu. 2024. Advancing Web 3.0: Making Smart Contracts Smarter on Blockchain. In *Proceedings of the ACM Web Conference 2024* (Singapore, Singapore) *(WWW '24).* Association for Computing Machinery, New York, NY, USA, 1549–1560. doi:10.1145/3589334.3645319

[17] Gaurish Korpal and Drew Scott. 2023. Decentralization and web3 technologies. *Authorea Preprints* (2023).

[18] Jae Kwon and Ethan Buchman. 2019. Cosmos whitepaper. *A Netw. Distrib. Ledgers* 27 (2019).

[19] Rongjian Lan, Ganesha Upadhyaya, Stephen Tse, and Mahdi Zamani. 2021. Horizon: A gas-efficient, trustless bridge for cross-chain transactions. *arXiv preprint arXiv:2101.06000* (2021).

[20] Ricky Leung. 2023. Leveraging AI and Blockchain for Streamlining Healthcare Payments. In *Proceedings of the 2023 5th Blockchain and Internet of Things Conference* (Osaka, Japan) *(BIOTC '23).* Association for Computing Machinery, New York, NY, USA, 58–62. doi:10.1145/3625078.3625086

[21] Pengze Li, Mingxuan Song, Mingzhe Xing, Zhen Xiao, Qiuyu Ding, Shengjie Guan, and Jieyi Long. 2024. SPRING: Improving the Throughput of Sharding Blockchain via Deep Reinforcement Learning Based State Placement. In *Proceedings of the ACM Web Conference 2024* (Singapore, Singapore) *(WWW '24).* Association for Computing Machinery, New York, NY, USA, 2836–2846. doi:10.1145/3589334.3645386

[22] Taotao Li, Changlin Yang, Qinglin Yang, Shizhan Lan, Siqi Zhou, Xiaofei Luo, Huawei Huang, and Zibin Zheng. 2023. Metaopera: A cross-metaverse interoperability protocol. *IEEE Wireless Communications* 30, 5 (2023), 136–143.

[23] Wanxin Li, Collin Meese, Mark Nejad, and Hao Guo. 2024. ZK-BFT: A Zero-knowledge and Byzantine Fault Tolerant Consensus for Permissioned Blockchain Networks. In *Proceedings of the 2023 6th International Conference on Blockchain Technology and Applications* (Xi'an, China) *(ICBTA '23).* Association for Computing Machinery, New York, NY, USA, 70–77. doi:10.1145/3651655.3651663

[24] Zhuotao Liu, Yangxi Xiang, Jian Shi, Peng Gao, Haoyu Wang, Xusheng Xiao, Bihan Wen, and Yih-Chun Hu. 2019. Hyperservice: Interoperability and programmability across heterogeneous blockchains. In *Proceedings of the 2019 ACM SIGSAC conference on computer and communications security.* 549–566.

[25] Damiano Di Francesco Maesa and Paolo Mori. 2020. Blockchain 3.0 applications survey. *J. Parallel and Distrib. Comput.* 138 (2020), 99–114.

[26] Henrique Moniz. 2020. The Istanbul BFT consensus algorithm. *arXiv preprint arXiv:2002.03613* (2020).

[27] Eduardo Morais, Tommy Koens, Cees Van Wijk, and Aleksei Koren. 2019. A survey on zero knowledge range proofs and applications. *SN Applied Sciences* 1 (2019), 1–17.

[28] Multichain. 2023. Cross-Chain Router Protocol. https://multichain.xyz Last accessed 14 Oct 2024.

[29] Faisal Nawab and Mohammad Sadoghi. 2023. .

[30] Michael Neuder, Daniel J Moroz, Rithvik Rao, and David C Parkes. 2021. Low-cost attacks on Ethereum 2.0 by sub-1/3 stakeholders. *arXiv preprint arXiv:2102.02247* (2021).

[31] Yanqing Peng, Min Du, Feifei Li, Raymond Cheng, and Dawn Song. 2020. FalconDB: Blockchain-based collaborative database. In *Proceedings of the 2020 ACM SIGMOD international conference on management of data.* 637–652.

[32] Gabriel Antonio F Rebello, Gustavo F Camilo, Lucas CB Guimaraes, Lucas Airam C de Souza, Guilherme A Thomaz, and Otto Carlos MB Duarte. 2021. A security and performance analysis of proof-based consensus protocols. *Annals of Telecommunications* (2021), 1–21.

[33] Kunpeng Ren, Nhut-Minh Ho, Dumitrel Loghin, Thanh-Toan Nguyen, Beng Chin Ooi, Quang-Trung Ta, and Feida Zhu. 2023. Interoperability in Blockchain: A Survey. *IEEE Transactions on Knowledge and Data Engineering* (2023).

[34] Pingcheng Ruan, Tien Tuan Anh Dinh, Dumitrel Loghin, Meihui Zhang, Gang Chen, Qian Lin, and Beng Chin Ooi. 2021. Blockchains vs. distributed databases: Dichotomy and fusion. In *Proceedings of the 2021 International Conference on Management of Data.* 1504–1517.

[35] Michael Sober, Giulia Scaffino, Christof Spanring, and Stefan Schulte. 2021. A voting-based blockchain interoperability oracle. In *2021 IEEE International Conference on Blockchain (Blockchain).* IEEE, 160–169.

[36] Xiaoqiang Sun, F Richard Yu, Peng Zhang, Zhiwei Sun, Weixin Xie, and Xiang Peng. 2021. A survey on zero-knowledge proof in blockchain. *IEEE network* 35, 4 (2021), 198–205.

[37] Bitcoin Taxes. 2023. 5 Biggest Crypto Cross-Chain Bridge Hacks in 2022. https://bitcoin.tax/blog/cross-chain-bridge-hacks/ Last accessed 14 Oct 2024.

[38] CHAINALYSIS TEAM. 2023. Multichain Protocol Experiences Mysterious Withdrawals, Suggesting Multi-Million Dollar Hack or Rug Pull [UPDATED 7/19/23]. https://www.chainalysis.com/blog/multichain-exploit-july-2023/ Last accessed 14 Oct 2024.

[39] Psi Vesely, Kobi Gurkan, Michael Straka, Ariel Gabizon, Philipp Jovanovic, Georgios Konstantopoulos, Asa Oines, Marek Olszewski, and Eran Tromer. 2022. Plumo: An ultralight blockchain client. In *International Conference on Financial Cryptography and Data Security.* Springer, 597–614.

[40] Gang Wang, Qin Wang, and Shiping Chen. 2023. Exploring blockchains interoperability: A systematic survey. *Comput. Surveys* (2023).

[41] Ziwei Wang, Jiashi Gao, and Xuetao Wei. 2023. Do NFTs' Owners Really Possess their Assets? A First Look at the NFT-to-Asset Connection Fragility. In *Proceedings of the ACM Web Conference 2023* (Austin, TX, USA) *(WWW '23).* Association for Computing Machinery, New York, NY, USA, 2099–2109. doi:10.1145/3543507.3583281

[42] Sam Werner, Daniel Perez, Lewis Gudgeon, Ariah Klages-Mundt, Dominik Harz, and William Knottenbelt. 2022. Sok: Decentralized finance (defi). In *Proceedings of the 4th ACM Conference on Advances in Financial Technologies.* 30–46.

[43] Martin Westerkamp and Jacob Eberhardt. 2020. zkrelay: Facilitating sidechains using zksnark-based chain-relays. In *2020 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW).* IEEE, 378–386.

[44] Gavin Wood. 2016. Polkadot: Vision for a heterogeneous multi-chain framework. *White paper* 21, 2327 (2016), 4662.

[45] Gavin Wood et al. 2014. Ethereum: A secure decentralised generalised transaction ledger. *Ethereum project yellow paper* 151, 2014 (2014), 1–32.

[46] Tiancheng Xie, Jiaheng Zhang, Zerui Cheng, Fan Zhang, Yupeng Zhang, Yongzheng Jia, Dan Boneh, and Dawn Song. 2022. zkbridge: Trustless cross-chain bridges made practical. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security.* 3003–3017.

[47] Cheng Xu, Ce Zhang, Jianliang Xu, and Jian Pei. 2021. SlimChain: Scaling blockchain transactions through off-chain storage and parallel processing. *Proceedings of the VLDB Endowment* 14, 11 (2021), 2314–2326.

[48] Ryan Zarick, Bryan Pellegrino, and Caleb Banister. 2021. Layerzero: Trustless omnichain interoperability protocol. *arXiv preprint arXiv:2110.13871* (2021).

[49] Jiashuo Zhang, Jianbo Gao, Yue Li, Ziming Chen, Zhi Guan, and Zhong Chen. 2022. Xscope: Hunting for cross-chain bridge attacks. In *Proceedings of the 37th IEEE/ACM International Conference on Automated Software Engineering.* 1–4.

## A Preliminaries

**PoS-BFT Consensus**. Proof of Stake with Byzantine-Fault Tolerance (PoS-BFT) consensus has become a best practice for blockchain development due to its high energy-efficiency and security in recent years. It requires nodes (validators) to deposit funds as stakes to be qualified to participate in the consensus and to guarantee security. PoS-BFT consensus procedures typically operate and iterate in epochs. At the beginning and end of each epoch, validators are rotated and elected as committees by the PoS mechanism. During the epoch, there will be a fixed period of time for the committees to validate, agree and finalize proposed blocks according to BFT algorithms and PoS mechanism [11, 25].

**Light Client**. The light client is designed for resource-constrained devices such as mobile phones running blockchain nodes. It only syncs and stores block headers to reduce storage and computation overheads. Therefore, only partial functions of full nodes are available, such as transaction query and verification, while the costly consensus and mining procedures are usually excluded [8].

**Aggregate Signature**. Aggregate signature refers to the signature scheme that supports batching signatures to reduce overheads[4, 5]. In aggregate signature schemes, multiple signatures are aggregated as one signature, which can be further verified by an aggregated public key. Nowadays, BLS signature and its variants currently are widely used in PoS-BFT chains due to their high efficiency.

**Zero-knowledge Proof**. The Zero-Knowledge Proof (ZKP) is a type of cryptographic system that allows a prover to convince a verifier that a given statement is true or false without disclosing any other information. ZKP systems typically need to express and compile the statement proof procedures into circuits with constraints (gates) to generate proofs, which is computationally expensive [23, 27, 36].

## B Implementations details

For the relay chain, we develop a client software of the unified relay chain node[3]. To overcome the heterogeneity, we integrate the most commonly adopted cryptographic primitives and parameters in existing chains into the smart contract engine of the relay chain. Specifically, supported hashing algorithms include SHA-3, SHA-256, keccak256, and blake2b, while signature algorithms (or elliptic curves) include ed25519, secp256k1, sr25519, and BN256, which covers most public chains. We adopt IBFT in the relay chain, which is also well tested and widely adapted in many chains. We also implement six hybrid LCs for six chains[4] in the form of multiple smart contract pairs. For off-chain provers, we use Groth16[14] to express the BLS signature verification (except *Hash-to-Base*) through Circom, alongside with our optimizations to reduce the size of the circuit[5]. First, we make BLS public keys in $\mathbb{G}_2$, while the signatures are in $\mathbb{G}_1$ to reduce the signature size. Second, as mentioned before, we move two *Hash-to-Base* functions out of the circuit to simplify the constraints in the circuit.

[3]https://github.com/mapprotocol/atlas
[4]https://github.com/mapprotocol/atlas, https://github.com/mapprotocol/map-contracts/tree/main/mapclients/zkLightClient, and https://github.com/zkCloak/zkMapo
[5]https://github.com/zkCloak/zkMapo

## C Supported Chains and Cross-chain Applications

`MAP` supports six major public chains: including EVM chains such as Ethereum, BNB chains, Polygon, and Conflux, and Non-EVM chains such as Klaytn and Near. By 2024, there are over 640M USD assets relayed by over 5M cross-chain transactions with `MAP`[6]. Over 50 industrial cross-chain applications and layer-2 projects are built[7]. Representative cross-chain applications range from cross-chain swap (ButterSwap), crypto payment (AlchemyPay), liquidity aggregation (Openliq), DePINs (ConsensusCore), DeFi solutions development (Unify) [8].

## D Real-world Cross-chain Dataset

Based on the experiments and our deployment statistics, we prune and provide the first public, real-world blockchain interoperability dataset, `BlockMAP`[9], which consists of 150k cross-chain transactions from six popular public chains. The dataset includes several essential attributes, such as transaction direction, start and end timestamps, token types, and amounts. This dataset presents practical measurement of real-world cross-chain transactions, aiming to offer new insights and understandings for future blockchain research.

## E Ethics

All the published cross-chain transaction data are anonymized to protect the privacy of users. That is, it is extremely hard to trace or reveal individual identities from these transaction data by public addresses or amounts.

[6]https://www.maposcan.io
[7]A full list at https://www.mapprotocol.io/en/ecosystem
[8]https://www.butterswap.io/swap, https://alchemypay.org, https://www.consensuscore.com,https://openliq.com, https://unifiprotocol.com
[9]https://zenodo.org/records/13928962