

zkLink: A Zero-Knowledge Proof Framework for Secure and Private Cross-chain Interoperability^{*}

Tuan-Dung Tran^{a,b}, Huynh Phan Gia Bao^{a,b}, Phan The Duy^{a,b}, Nguyen Tan Cam^{a,c} and Van-Hau Pham^{a,b,*}

^aVietnam National University Ho Chi Minh City, Ho Chi Minh City, Vietnam

^bFaculty of Computer Networks and Communications, University of Information Technology, Ho Chi Minh City, Vietnam

^cFaculty of Information Science and Engineering, University of Information Technology, Ho Chi Minh City, Vietnam

ARTICLE INFO

Keywords:

Blockchain
Cross-chain communication
Zero-knowledge proofs
Interoperability

ABSTRACT

As blockchain technology evolves, the capability for seamless collaboration across independently managed, heterogeneous networks emerges as a fundamental requirement rather than an ancillary feature. Traditional relay-chain approaches, despite providing baseline interoperability, often neglect critical aspects such as comprehensive security guarantees and robust privacy preservation. In response to these limitations, we introduce zkLink, a novel interoperability framework built around optimized zero-knowledge proof (ZKP) methodologies, specifically utilizing Groth16. zkLink uniquely integrates these cryptographic proofs directly into relay-chain structures, enabling trustless verification and secure cross-chain transactions without centralized intermediaries. zkLink achieves fast verification speeds (3–6 ms), compact proof sizes (192–256 bytes), and efficient on-chain validation. It supports batch processing up to 2.65 transactions per second (TPS) and parallelization across 100 worker nodes, achieving 2.4 TPS in constrained environments. Through strategic system-level optimizations and modular circuit designs, zkLink significantly enhances scalability, verification speed, and operational efficiency. Empirical analyses show that zkLink performs better than existing solutions like zkBridge and Zerocash in terms of speed and cost-effectiveness. Overall, zkLink provides a reliable and flexible solution for secure, scalable, and private cross-chain interactions.

1. Introduction


The accelerating evolution of blockchain ecosystems has intensified the demand for infrastructures that are simultaneously secure, scalable, and interoperable. Despite the revolutionary impact of distributed ledger systems on data governance—particularly in improving openness, robustness, and resistance to tampering, achieving these guarantees across diverse blockchain platforms remains a complex issue. The need to solve this challenge is becoming more urgent, as the blockchain market is expected to grow quickly. According to forecasts, the global market could reach around \$248.9 billion by 2029, with an annual growth rate of 65.5%¹. Among the principal obstacles to realizing this potential, interoperability remains paramount: as numerous blockchain platforms operate independently, each adopting unique consensus protocols, data architectures, and governance frameworks, the absence of effective cross-chain communication fundamentally constrains the scalability, composability, and operational efficiency of decentralized applications (DApps).

To address the limitations of isolated blockchain networks, cross-chain technologies have emerged to enable interoperability between heterogeneous blockchain systems. These technologies include atomic swaps[13], blockchain bridges, and interoperability protocols like the Inter-Blockchain Communication (IBC) protocol [15], which allow value and information to flow seamlessly across different blockchain environments. This reduces dependence on centralized intermediaries, preserving the decentralized ethos of the blockchain space. Among the various approaches, the relay chain architecture has attracted significant attention for its ability to offer scalable and secure cross-chain interoperability. In this model, independent chains, often called parachains or sidechains, are connected to a central relay chain that ensures a unified security and consensus mechanism. This structure allows for inter-chain communication and coordination without compromising scalability or decentralization.

The interoperability facilitated by relay chains is pivotal in realizing the full potential of DApps, particularly within Decentralized Finance (DeFi). It supports cross-chain asset mobility and optimizes liquidity allocation, fostering a more cohesive blockchain environment. The increasing demand for DeFi applications, which inherently require seamless interaction across multiple blockchains for efficient operation, is a significant driver for cross-chain technology adoption. The current DeFi ecosystem is fragmented, with liquidity spread across multiple platforms such as Ethereum, Binance Smart Chain, Solana, and Avalanche [12]. This fragmentation impedes capital efficiency and necessitates reliance on centralized exchanges or complex bridging mechanisms for cross-chain asset transfers. While cross-chain solutions

^{*}This research is funded by Vietnam National University HoChiMinh City (VNU-HCM) under grant number DS.C2025-26-18.

^{*}Corresponding author

 dungtrt@uit.edu.vn (T. Tran); 23520104@gm.uit.edu.vn (H.P.G. Bao); duypt@uit.edu.vn (P.T. Duy); camnt@uit.edu.vn (N.T. Cam); haupv@uit.edu.vn (V. Pham)

ORCID(s): 0000-0003-1156-7072 (T. Tran); 0009-0008-8773-0482 (H.P.G. Bao); 0000-0002-5945-3712 (P.T. Duy); 0000-0002-1489-2826 (N.T. Cam); 0000-0003-3147-3356 (V. Pham)

¹<https://www.globenewswire.com/news-release/2024/07/12/2912950/0/en/Blockchain-Market-worth-248-9-billion-by-2029-Exclusive-Report-by-MarketsandMarkets.html>

such as blockchain bridges and wrapped assets aim to address these issues, many existing bridges are susceptible to significant security vulnerabilities and centralization risks, as evidenced by substantial financial losses resulting from breaches targeting the Ronin and Wormhole bridges^{2 3}. These incidents underscore the critical need for more secure and trustworthy cross-chain communication methods. Consequently, researchers have explored mitigation strategies, including trust-minimized bridges leveraging cryptographic proofs, decentralized validator networks, and enhanced consensus mechanisms [22]. However, achieving a truly secure and scalable cross-chain infrastructure remains an open challenge, demanding further investigation into advanced cryptographic techniques and decentralized trust models.

Zero-knowledge proofs (ZKPs) [10] have gained significant attention as a foundational cryptographic technique that strengthens privacy guarantees, upholds data integrity, and reduces the reliance on trusted third parties in decentralized environments [11, 26]. Both interactive and non-interactive ZKP variants (NIZKPs) [5] enable transaction verification without revealing sensitive information. In the context of cross-chain communication, recent research [4, 3] has highlighted the potential of ZKPs to guarantee transaction correctness while preserving metadata privacy, offering advantages over traditional methods like Hash Time Locked Contracts (HTLCs) [25].

While existing ZKP-based solutions such as zkBridge [24] and Zerocash [4] demonstrate the potential of ZKPs for cross-chain applications, they still exhibit notable limitations. For instance, zkBridge, while eliminating trusted relayers, introduces dependencies on a trusted setup and off-chain proof generation, potentially creating new attack vectors and increasing overhead. Similarly, Zerocash, designed for strong privacy, lacks inherent support for general cross-chain interoperability and scalability. More broadly, current approaches often suffer from recurring challenges, including reliance on centralized components, the requirement of trusted setups in certain ZKP constructions, limited compatibility across diverse blockchain platforms, and performance bottlenecks stemming from computationally intensive proof generation or the absence of efficient aggregation mechanisms.

To address these critical challenges—particularly centralization, trusted setups, limited interoperability, and scalability constraints—this paper introduces zkLink, a modular framework designed to integrate zero-knowledge verification into cross-chain infrastructures. zkLink provides an abstraction layer over various ZKP schemes, allowing adaptation to specific constraints such as proof size and verification cost. By systematically investigating integration points within relay chain-oriented architectures, we analyze the trade-offs between performance, complexity, and privacy, with the overarching goal of establishing a robust

foundation for scalable and privacy-preserving cross-chain interoperability.

- RQ1: What are the cost efficiency and latency characteristics of cross-chain transactions utilizing zero-knowledge proofs?
- RQ2: What are the performance characteristics and economic costs associated with the zkLink authentication mechanism?
- RQ3: How does proof generation time impact system throughput, and what parallelization strategies can mitigate potential bottlenecks?
- RQ4: What advantages does our ZKP-enhanced cross-chain protocol offer compared to existing solutions?

To address these questions, we design a modular prototype implementation of zkLink, simulate various cross-chain scenarios, and measure key performance metrics including latency, throughput, and verification costs. Our prototype system architecture and design choices are detailed in Section 3. The answers to RQ1–RQ4 are systematically explored through experiments and evaluations presented in Section 4. We also investigate optimization strategies focusing on minimizing proof verification overhead and enhancing parallel execution efficiency. Specifically, we analyze the effects of proof size and circuit complexity on system scalability, and propose task-parallel approaches to maximize throughput under constrained computational environments.

Through rigorous empirical evaluation addressing these questions, we demonstrate that zkLink achieves an optimal balance between security, privacy, and performance, making it a viable solution for trustless cross-chain communication in production environments. Our main contributions are as follows:

- We propose zkLink framework to strengthen trustless after setup assuming trusted ceremony cross-chain message verification, enhancing both security and privacy without reliance on trusted intermediaries.
- We engineer domain-specific ZKP circuits and system-level optimizations on top of Groth16 to achieve improved succinctness and fast verification for cross-chain scenarios.
- We perform empirical evaluations to assess the performance and scalability of Zkp-enabled cross-chain systems, demonstrating reduced overhead and faster execution.

2. Related Work

The integration of Zkp into cross-chain blockchain communication represents a significant advancement in addressing critical challenges related to security, privacy, and interoperability. This section examines the foundational concepts, existing approaches, and architectural paradigms that inform our work on zkLink.

²<https://www.coindesk.com/tech/2022/03/29/axie-infinitys-ronin-network-suffers-625m-exploit>

³<https://www.certik.com/resources/blog/1kDYgyBcisoD2EqiBpHE5l-wormhole-bridge-exploit-incident-analysis>

Table 1

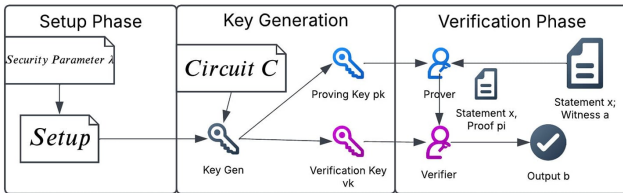
Comparison of Cross-Chain Authentication Approaches

Approach	Key Characteristics	Limitations
Relay Contracts	Smart contracts enforce consensus; submit block headers periodically; support SPV verification	High computational cost; storage burden; synchronization overhead
Zero-Knowledge Proofs	Prove computation validity without revealing data; low on-chain cost; privacy enhancement	Complex setup; trusted setup in some cases; proof generation overhead
Notary Schemes	Trusted third parties monitor blockchain states; simple and blockchain-agnostic	Centralization risk; single point of failure; trust requirements

2.1. Zero-Knowledge Authentication for Cross-Chain Systems

Cross-chain authentication is essential for enabling secure interoperability across blockchain networks, with different approaches offering trade-offs in efficiency, trust, and complexity, as summarized in Table 1. Relay contracts enforce consensus rules via smart contracts and support SPV verification, but suffer from high computational and storage demands. Notary schemes offer a structurally simple, blockchain-agnostic design but introduce centralization risks. Zkp emerges as a promising alternative, enabling validity proofs without revealing sensitive information, reducing on-chain verification costs, and enhancing privacy and scalability.

Recent advancements in cryptographic primitives, particularly Zkp, have further strengthened cross-chain authentication methods [9]. These techniques enable a blockchain system to demonstrate the validity of computations or state transitions to a separate chain, all while preserving the confidentiality of underlying data [28]. Such approaches frequently employ bilinear pairings [1] and elliptic curve cryptography to facilitate operations that are both efficient and minimal in size.

**Figure 1:** Phases of a Zkp System: Key Setup, Proof Construction, and Verifier Evaluation

As depicted in Figure 1, a standard Zero-Knowledge Proof (Zkp) architecture is generally structured into three main phases: Key Setup, Proof Construction, and Verifier Evaluation. This structure is supported by four fundamental algorithms: Setup, KeyGen, Prove, and Verify. In this process, Setup initializes global parameters based on a security level λ ; KeyGen derives the proving and verification keys

Table 2

Key Zkp Implementations for Blockchain Applications

Zkp Implementation	Characteristics and Applications
Zero-Knowledge Succinct Non-Interactive Arguments of Knowledge (zk-SNARKs)[8]	Optimized for blockchains with minimal proof size and fast verification; enables efficient on-chain proof validation
Scalable ZK Constructions	Designed for high-throughput use cases; supports batch processing and recursion; reduces amortized verification costs

tailored to a circuit C ; Prove generates a proof π corresponding to the public input x and the private witness a ; and Verify assesses the proof's correctness, returning a Boolean result. Several Zkp designs have already been integrated into blockchain-based applications [21, 14], as summarized in Table 2.

2.2. Relay Chain Architectures with Existing Solutions

Several approaches have been proposed to leverage Zkp for enabling trustless verification of cross-chain transactions, reducing reliance on centralized relayers, and enhancing scalability in multi-chain environments. Polyhedra Network's zkBridge [24] employs zk-SNARKs to validate cross-chain messages without trusted relayers, enhancing security and privacy. However, its reliance on trusted setup and off-chain proof generation introduces vulnerabilities and computational overhead. zkIBC [27] extends Cosmos' IBC protocol with Zkp for secure state verification, but its applicability remains confined to Cosmos-based chains.

Vitalik Buterin's zk-Rollups⁴ significantly improve scalability through batch verification but were not initially designed for cross-chain messaging. Beyond zk-based mechanisms, ACCS [13] and HTLC-based systems [25] enable atomic swaps without intermediaries, though they face challenges like prolonged asset locking and vulnerability to

⁴<https://vitalik.eth.limo/general/2021/01/05/rollup.html>

Table 3
Comparison of Zkp-Based Cross-Chain Solutions

Protocol / Feature	Eliminates Trusted Setup	Scalable	Chain-Agnostic Execution	No External Relayers
zkBridge				✓
zkIBC				✓
zk-Rollups	✓	✓		
zkParachains				✓
ACCS	✓			✓
HyperService		✓		
MAP		✓		✓
zkLink	Requires – One-time per circuit	✓	✓	✓

delays. Enhancements such as Attribute Verifiable Timed Commitments [17] and Quick Swap [19] attempt to mitigate these issues, albeit with increased complexity.

Alternative interoperability platforms like HyperService [16] and MAP [7] propose unified programming models and zk-based verification schemes, respectively. Nonetheless, both encounter integration complexities and adoption barriers. Within Polkadot's ecosystem, zkParachains [18] incorporate Zkp to enhance privacy in inter-parachain transactions, though they remain tightly coupled to Polkadot's architecture, limiting broader cross-chain applicability.

As shown in Table 3, several existing approaches have sought to leverage Zkp for trustless cross-chain verification, with many focusing on removing external relayers or achieving greater scalability. However, such constructions frequently entail the drawback of depending on a trusted setup process, or they may limit compatibility across heterogeneous blockchain environments. By simultaneously eliminating trusted setup dependencies, supporting scalable verification, enabling chain-agnostic execution, and removing reliance on external relayers, zkLink demonstrates a more comprehensive and decentralized architecture, positioning itself as a robust framework for secure and scalable cross-chain interoperability.

Building upon these advancements, an emerging architectural paradigm that holistically addresses both communication and authentication challenges is the blockchain-of-blockchains model, typified by relay chain systems. This paradigm offers several distinct advantages that align closely with the design philosophy of zkLink. First, it facilitates mutual authentication among independently operated subchains without requiring centralized trust anchors. Second, it enables cross-chain interactions to be recorded and verified through a central relay, ensuring transparency and verifiability across domains. Moreover, many implementations adopt a shared security model, in which a common set of validators protects both the relay chain and the connected subchains, enhancing ecosystem security while minimizing redundant validation efforts. Ultimately, this model promotes seamless interoperability, improved composability, and stronger systemic resilience, reinforcing the foundational principles of decentralization and scalability that zkLink aspires to achieve.

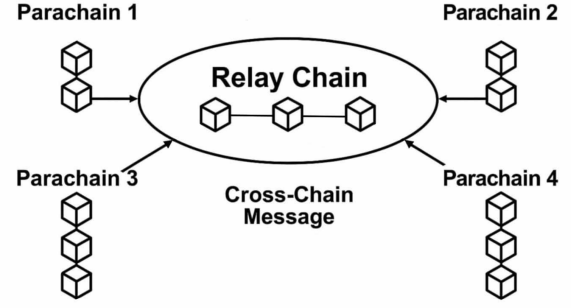


Figure 2: System Architecture Illustrating a Relay Chain as a Central Hub for Decentralized Chain Networks

Table 4
Relay Chain Architecture Components and Functions

Component	Function
Relay Chain	Validates state changes, maintains consensus, and routes secure cross-chain messages.
Validators	Run NPoS consensus; verify cross-chain messages and foreign states.
Parachains	Independent blockchains using Relay Chain for state verification and parallel processing.
Collators	Gather parachain transactions and submit proofs to validators.

The architecture depicted in Figure 2 illustrates a system where a Relay Chain functions as a central coordination hub connecting multiple decentralized blockchain networks. This multi-chain design is specifically aimed at addressing critical challenges in blockchain ecosystems, including authentication, scalability, and interoperability [23, 2]. As detailed in Table 4, the architecture comprises several key components—Relay Chain, Validators, Parachains, and Collators—each responsible for specific roles that collectively enable secure cross-chain state verification, parallelized transaction processing, and decentralized consensus maintenance.

2.3. Integration Challenges zkLink Approach

The integration of Zkp into relay chain architectures presents both challenges and opportunities. From one perspective, Zkp offers strong potential to reinforce confidentiality and trust in inter-chain transactions by supporting verifiable computations without disclosing underlying data. Conversely, the intensive resource demands associated with generating and validating these proofs require careful optimization to maintain overall scalability and operational efficiency. As summarized in Table 5, key challenges include high proof generation overhead, costly on-chain verification, potential vulnerabilities from trusted setup requirements, and cross-chain compatibility issues across heterogeneous blockchain environments[20].

zkLink system confronts these limitations by offering an integrated approach to embedding zero-knowledge proofs within cross-chain communication structures. Through the

Table 5
Integration Challenges and Potential Solutions

Challenge	Impact	Potential Solution
Proof Generation Overhead	High computational requirements for generating Zkp can introduce latency in cross-chain transactions	Implement off-chain proof generation with dedicated worker nodes; utilize batch processing for improved efficiency
Verification Costs	On-chain verification of Zkp consumes computational resources and increases transaction fees	Optimize verification circuits; implement recursive proof composition to amortize verification costs across multiple transactions
Trusted Setup Requirements	Some Zkp systems require a trusted setup phase, introducing potential security vulnerabilities	Explore transparent setup protocols; consider alternative Zkp systems with reduced trust assumptions
Cross-Chain Compatibility	Different blockchain architectures may have varying support for cryptographic operations	Develop standardized verification interfaces; implement chain-specific adapters for heterogeneous environments

utilization of modular blockchain architectures, WASM-enabled execution environments, and standardized messaging protocols, zkLink facilitates trustworthy, confidential, and efficient interaction across diverse blockchains—without the need for centralized verifiers or disclosing sensitive transactional metadata.

3. Methodology

3.1. Trust Model and Assumptions

We present a formal model for our Zkp-integrated cross-chain communication system, defining the key components, their interactions, and the security properties they collectively ensure. This formalisation provides a rigorous foundation for analysing the system's correctness, security guarantees, and performance characteristics.

Let us define the system as a tuple $S = (\mathcal{RC}, \mathcal{P}, \mathcal{W}, \mathcal{V}, \mathcal{M})$, where:

- \mathcal{RC} represents the Relay Chain, maintaining global state σ_{RC}
- $\mathcal{P} = \{P_1, P_2, \dots, P_n\}$ is the set of parachains, where each P_i maintains local state σ_{P_i}
- $\mathcal{W} = \{W_1, W_2, \dots, W_m\}$ is the set of Worker Nodes responsible for proof generation
- $\mathcal{V} = \{V_1, V_2, \dots, V_k\}$ is the set of Validator Nodes that verify proofs
- \mathcal{M} is the message space containing all possible cross-chain messages

The global state of the system at time t is defined as:

$$\Sigma_t = (\sigma_{RC}^t, \{\sigma_{P_i}^t\}_{i=1}^n)$$

representing the combined states of the relay chain and all parachains.

A cross-chain transaction T is formally defined as a tuple:

$$T = (P_{src}, P_{dst}, msg, \pi) \quad (1)$$

where:

- $P_{src} \in \mathcal{P}$ is the source parachain
- $P_{dst} \in \mathcal{P}$ is the destination parachain
- $msg \in \mathcal{M}$ is the message payload
- π represents the zero-knowledge proof that verifies the correctness of the transaction.

The proof object π is generated using a zk scheme, and satisfies:

$$\pi = Prove(pk, (P_{src}, H(msg), aux), w) \quad (2)$$

where pk is the proving key, H is collision-resistant hash function, aux contains auxiliary public information, and w is the witness containing private data known only to the prover.

The state transition function δ defines how the system state evolves when a cross-chain transaction is processed (see Equation 1):

$$\Sigma_{t+1} = \sigma(\Sigma_t, T)$$

This function can be decomposed into parachain-specific and relay chain transition functions (again based on Transaction 1):

$$\begin{aligned} \sigma_{P_{src}}^{t+1} &= \delta_{P_{src}}(\sigma_{P_{src}}^t, T) \\ \sigma_{RC}^{t+1} &= \delta_{RC}(\sigma_{RC}^t, T) \end{aligned}$$

$$\sigma_{P_{dst}}^{t+1} = \delta_{dst}(\sigma_{dst}^t, T)$$

An aggregated proof Π combines multiple individual proofs (each of the form given in Equation 2):

$$\Pi = \text{Aggregate}(\{\pi_i\}_{i=1}^l, \{\text{msg}_i\}_{i=1}^l) \quad (3)$$

This aggregated proof satisfies the following verification condition:

$$\begin{aligned} & \text{VerifyAggregate}(vk, \{H(\text{msg}_i)\}_{i=1}^l, \Pi) = 1 \\ \iff & \forall i \in [1, l], \text{Verify}(vk, H(\text{msg}_i), \pi_i) = 1 \end{aligned} \quad (4)$$

zkLink relies on a distributed network of off-chain Worker Nodes to perform the computationally intensive task of zero-knowledge proof generation. While this architecture enables scalable throughput and offloads expensive computation from on-chain layers, it introduces practical challenges related to incentivization, trust, and coordination. From an economic standpoint, Worker Nodes are assumed to be operated by independent entities—such as infrastructure providers or community participants—who are compensated for successful proof generation through a verification fee model. Specifically, a portion of transaction fees paid by users or DApp operators is allocated to Worker Nodes based on their proof contributions. This model allows for open participation while aligning economic incentives with network performance and reliability. We envision future integration with on-chain staking or reputation mechanisms to enhance Sybil resistance and penalize faulty behavior. In terms of trust, zkLink employs a threshold trust model where at least t out of n Worker Nodes must independently produce matching proofs or partial contributions to a multi-party aggregation protocol. This assumption ensures system integrity even in the presence of malicious or faulty nodes, provided the majority remains honest. However, to mitigate centralization risks, zkLink encourages diversity in operator control and supports permissionless Worker Node registration subject to performance benchmarks and participation requirements.

Coordination among Worker Nodes is managed via on-chain task queues. Once a cross-chain transaction is initiated, it is posted to a task queue from which eligible Workers can claim and process the request. Each Worker commits its proof result on-chain. Redundancy is permitted: multiple Workers may submit proofs for the same transaction, with the earliest valid proof being accepted. Dispute resolution is handled by a verifier contract that checks proof validity against circuit constraints, rejecting invalid submissions and optionally penalizing misbehavior. This design enables zkLink to maintain verifiability and decentralization without relying on trusted off-chain computation. The Worker Node layer functions as a scalable, verifiable computation layer secured by cryptographic correctness and incentivized by decentralized economics.

3.2. State Machine Model

In zkLink, we model the cross-chain message processing as a finite state machine (FSM) defined by a set of states

S and a transition function $\delta : S \times I \rightarrow S$, where I represents inputs such as events, error signals, and timeout triggers. The primary states include: S_{init} (request initiation), S_{pending} (proof generation in progress), S_{proved} (valid proof attached), S_{relayed} (message relayed to the destination chain), S_{verified} (proof verified on the destination chain), and S_{executed} (successful message execution). Exceptional conditions are captured by additional states: $S_{\text{error_proof}}$ (errors during proof generation), $S_{\text{error_relay}}$ (errors during message relaying), $S_{\text{error_verify}}$ (errors during proof verification), $S_{\text{error_execute}}$ (errors during message execution), and S_{timeout} (excessive processing delays). Recovery transitions are explicitly modeled, allowing retries from error and timeout states back to S_{pending} , thereby enhancing system resilience. Each transition is governed by precise preconditions and postconditions, enabling rigorous formal verification of both safety and liveness properties. This compact FSM design comprehensively covers normal, failure, and recovery flows, supporting automated model checking techniques.

3.3. Proposed System Architecture

To enable verifiable, privacy-preserving, and decentralized interoperability across heterogeneous execution environments, we propose a modular cross-chain communication architecture enhanced by advanced cryptographic attestation mechanisms. Figure 3 illustrates the complete message lifecycle within the zkLink framework, from initiation to execution, along with its associated security components.

The message lifecycle begins at the Source Chain Module, where a user or decentralized application triggers a smart contract to define the transaction intent, specify the target domain, and generate a structured message payload. Alongside this, the source chain extracts the necessary witness data and generates a cryptographic commitment to the state transition, ensuring that sensitive information remains concealed from external observers. Rather than transmitting raw or plaintext data, zkLink delegates the task of proof generation to an off-chain computation infrastructure.

This responsibility is handled by the Off-Chain Agent Triad, comprising three logically distinct but interrelated components. Worker Nodes reconstruct the expected state transition and generate a zero-knowledge proof that validates the correctness of the transaction without revealing any private input. These proofs are then bundled with the message and forwarded by Peer Nodes, which act as intermediary relays optimizing message propagation across network boundaries. Upon arrival at the destination domain or verification boundary, Validator Nodes perform formal proof verification using domain-specific verification circuits. If the proof passes these checks, the message is deemed valid and allowed to progress further through the system, thereby eliminating reliance on traditional consensus mechanisms to establish correctness.

After successful verification, messages and their corresponding attestations are persisted in two logically separate but coordinated data layers. First, the Aggregated Block

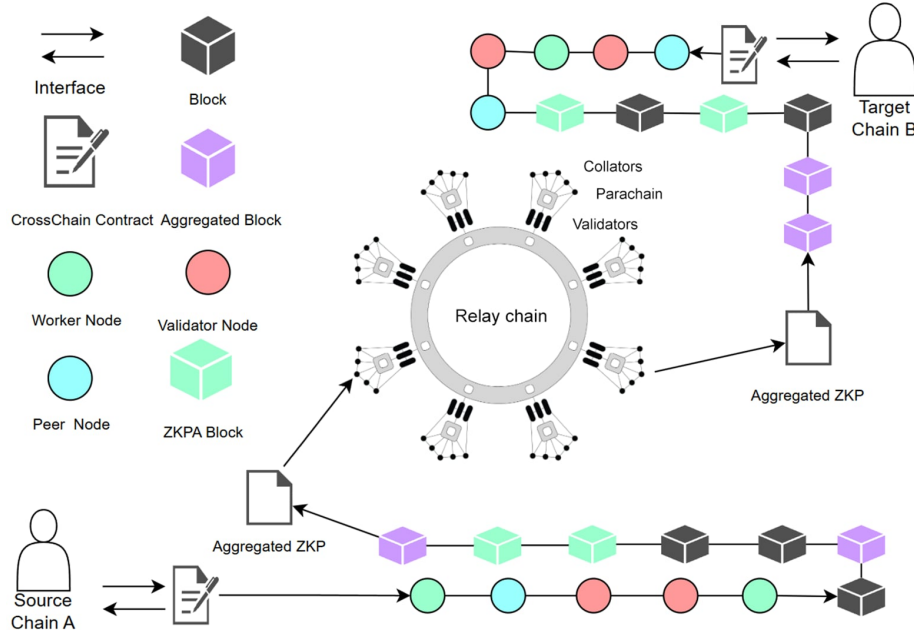


Figure 3: End-to-End Flow with Security Components in zkLink Framework

Layer stores the original message payloads along with execution metadata, forming the execution substrate for the target domain. Second, the Proof Archive Layer stores the corresponding zero-knowledge proofs and any recursive composition metadata associated with batched verification. At this stage, zkLink applies its aggregated proof composition strategy, wherein multiple discrete attestations are recursively merged into a single succinct proof. This process significantly reduces verification overhead, facilitates throughput improvements, and preserves traceability for auditability.

The aggregated message set is then forwarded to the Relay Coordination Layer, which serves as the routing and governance backbone of the architecture. Although this layer does not execute smart contracts itself, it performs several critical functions to ensure consistency and system-wide reliability. Specifically, it routes messages to their correct target domains, maintains a registry of domain identities and verification authorities, enforces message ordering and finality guarantees through timestamping, and supports dynamic protocol upgrades via governance mechanisms without disrupting service continuity.

Following coordination, the verified message reaches the *Target Chain Module*, which is responsible for executing the transaction in the destination domain. Upon receipt, the target chain locally reconstructs the verification context using the received attestation. It then invokes a domain-specific verifier circuit to validate the accompanying proof. This verifier is configured to check both the structural integrity of the message and the correctness of the zero-knowledge proof, ensuring that the original computational constraints were indeed satisfied by the source transaction without revealing the underlying witness data.

If the proof is successfully verified, the payload is deserialized and interpreted according to the execution semantics of the target blockchain. This may involve updating the internal state, invoking contract logic, or triggering downstream workflows specific to the application domain. Importantly, the target domain does not rely on any off-chain trust assumptions to validate the transaction; its acceptance is based solely on the cryptographic guarantees provided by the zero-knowledge proof. As such, the entire cross-chain execution process in zkLink is anchored in computational integrity rather than inter-chain consensus or validator-level coordination, thereby preserving decentralization and privacy throughout the transaction lifecycle.

3.4. Workflow of Zero-Knowledge Authentication Across Heterogeneous Chains

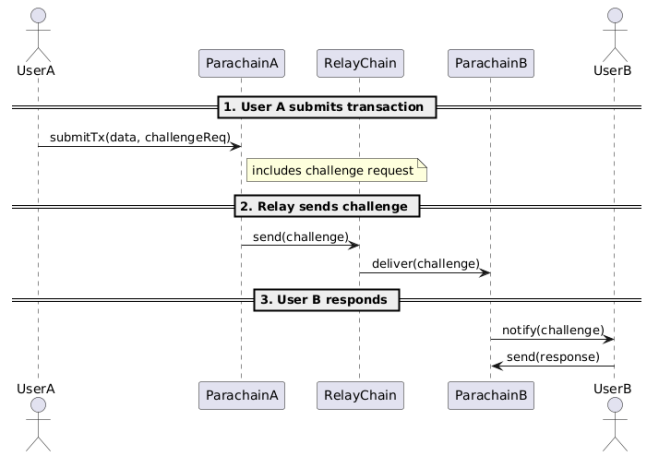


Figure 4: Submission and Challenge-Response Phase

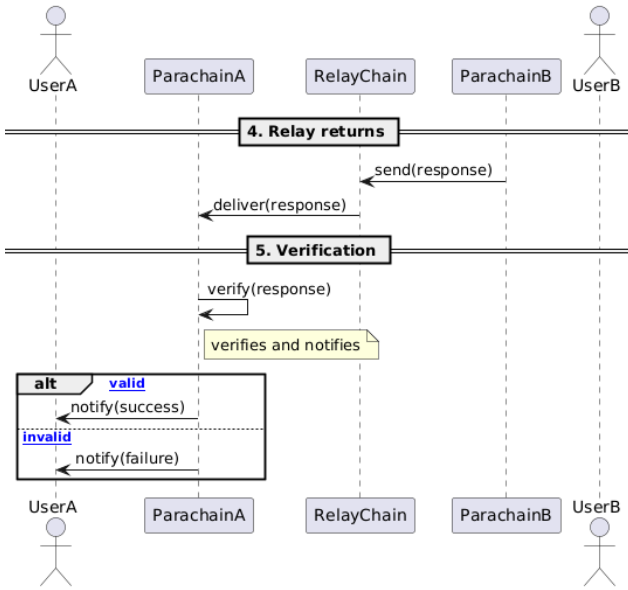


Figure 5: Response Relay and Verification

This section presents a comprehensive workflow for zero-knowledge-based authentication across heterogeneous blockchain environments, particularly focusing on trustless interaction between parachains within a relay-chain architecture. The protocol is designed to ensure verifiable computation and privacy-preserving communication without relying on centralized verification entities. As illustrated in Figures 4 and 5, the process begins when UserA submits a transaction to ParachainA containing an embedded request that encodes both application-specific data and a challenge directive to be verified by another domain. Upon receiving the transaction, ParachainA delegates the generation of a cryptographic challenge to an off-chain computation worker. This challenge incorporates domain-specific parameters, a nonce for freshness, and integrity-preserving metadata, all of which are then anchored on-chain by including them in a newly proposed block. Subsequently, the challenge is transmitted via the Relay Chain using the XCMP protocol, which provides ordered and authenticated message delivery between parachains. Upon arrival at ParachainB, the challenge is recorded and assigned to the corresponding verification process. UserB, operating in ParachainB, is notified of the challenge and initiates a response computation using a local off-chain worker. Crucially, instead of producing raw outputs that could reveal private data, the worker generates a zkSNARK that attests to the correctness of the computation. This proof construction process involves commitment to intermediate values and constraint satisfaction under cryptographic hardness assumptions. The generated zk-proof, along with a minimal disclosure of the computation result if required, is then relayed back to ParachainA via the same trusted Relay Chain pathway. As shown in Figure 5, upon receiving the response, ParachainA invokes its on-chain verifier to validate the zk-proof using a pairing-based verification equation defined by the underlying zkSNARK protocol such as Groth16. If

the verification succeeds, the transaction is finalized, and a success notification is returned to UserA, signaling the acceptance of the cross-domain computation. In the event of proof failure, the transaction is securely aborted and an error is communicated, preserving both integrity and resistance to forgery. This layered authentication process ensures that sensitive computations are proven correct without disclosure, upholding data minimization principles, and enabling scalable, modular, and privacy-respecting interoperability across heterogeneous chains.

3.5. Enhanced Cross-Chain Protocol with Zero-Knowledge Proofs

The integration of zero-knowledge proofs into cross-chain communication protocols presents unique challenges and opportunities for blockchain interoperability. This section details our approach to enhancing cross-chain transactions with Zkp technology, focusing on algorithmic design, complexity analysis, and efficiency considerations.

Chain-Agnostic Execution Support

While the current prototype of zkLink is implemented and evaluated within a Substrate-based relay chain environment, the framework is designed with modularity and chain-agnosticism as core principles. This is primarily achieved through three key mechanisms:

- **WASM-enabled Execution Layer:** zkLink utilizes WebAssembly (WASM) as the common execution abstraction. This enables zkLink to decouple the proof generation and verification logic from any particular smart contract language or virtual machine. As many modern blockchain platforms—such as Polkadot, Near, and even Ethereum via eWASM proposals—support WASM or similar environments, zkLink can integrate with diverse chains without modifying its core cryptographic logic.
- **Interface Adapters:** To support non-WASM chains (e.g., EVM-compatible chains like Ethereum or BNB Chain, or UTXO-based systems like Bitcoin), zkLink provides a lightweight adapter interface. These adapters map zkLink message formats and verification payloads into the native transaction and state model of the target chain. This modular adapter model allows chain-specific integration without compromising the core system architecture.
- **Relay-Agnostic Coordination Layer:** Although zkLink currently uses XCMP (Cross-Chain Message Passing) via a relay chain, the message coordination logic is abstracted via a generic routing interface. This interface can be extended to support IBC (Cosmos), Layer-2 bridges, or even off-chain coordination layers.

Therefore, while the present evaluation is conducted within a Substrate-native stack, the design of zkLink supports broad interoperability. Future work will focus on implementing full support for Cosmos-SDK chains and

EVM chains through runtime bridges and proof-verification adapters.

3.5.1. Protocol Design and Implementation

zkLink employs the Groth16 proving system as its cryptographic backbone due to its succinctness and fast verification. Rather than introducing a novel ZKP scheme, our contribution lies in the system-level and circuit-level optimizations that adapt Groth16 for efficient cross-chain message verification. These include tailored circuit designs with minimized constraint overhead, reusable witness structures, and batch aggregation strategies that reduce amortized verification cost. Algorithm 1 presents the core transaction protocol, which consists of three primary functions: InitiateTransaction, RelayTransaction, and VerifyAndExecute.

Algorithm 1 ZKP-enhanced Cross-Chain Transaction Protocol

Require: Source parachain P_{src} , destination parachain P_{dst} , message msg

Ensure: Successful cross-chain transaction execution

- 1: **function** INITIATE TRANSACTION(P_{src}, P_{dst}, msg)
- 2: $witness \leftarrow$ Extract private witness data from msg
- 3: $public_input \leftarrow$ Compute hash of message: $H(msg)$
- 4: $\pi \leftarrow$ Generate Zkp: Prove($witness, public_input$)
- 5: $xcm_msg \leftarrow$ Construct XCM message with ($msg, \pi, public_input$)
- 6: $P_{src}.SendXCM(P_{dst}, xcm_msg)$
- 7: **return** xcm_msg
- 8: **end function**
- 9: **function** RELAY TRANSACTION(xcm_msg)
- 10: Verify message format and structure
- 11: Queue transaction in relay chain
- 12: Achieve consensus on transaction inclusion
- 13: Forward xcm_msg to P_{dst}
- 14: **return** SUCCESS
- 15: **end function**
- 16: **function** VERIFY AND EXECUTE(P_{dst}, xcm_msg)
- 17: ($msg, \pi, public_input$) \leftarrow Extract from xcm_msg
- 18: $valid \leftarrow$ Verify($public_input, \pi$)
- 19: **if** $valid$ **then**
- 20: $decoded_msg \leftarrow$ Decode(msg)
- 21: $P_{dst}.ExecuteMessage(decoded_msg)$
- 22: **return** SUCCESS
- 23: **else**
- 24: **return** FAILURE
- 25: **end if**
- 26: **end function**

The protocol operates through a sequence of well-defined steps. Initially, the source parachain triggers a transaction by creating a zero-knowledge proof that validates the correctness of the cross-chain message, all while ensuring the confidentiality of sensitive data. This proof, along with the message and public inputs, is then packaged into an XCM message format compatible with the relay chain architecture. The relay chain subsequently processes this message,

Table 6

Time Complexity Analysis of Cross-Chain Transaction Protocol

Function	Key Operations	Time Complexity
InitiateTransaction	Zero-knowledge proof generation, Hash computation	$O(C \cdot \log^2 C + msg)$
RelayTransaction	Format verification, Transaction queuing, Consensus achievement	$O(\log n + v)$
VerifyAndExecute	Proof verification, Message execution	$O(P + msg)$

achieving consensus on its inclusion before forwarding it to the destination parachain. Finally, the destination parachain verifies the proof and executes the message only if verification succeeds.

Table 6 summarizes the time complexity analysis of each function in the protocol. The InitiateTransaction function has a complexity of $O(C \cdot \log^2 C + |msg|)$, where C represents the number of constraints in the circuit and $|msg|$ is the message size. The RelayTransaction function operates with complexity $O(\log n + v)$, where n is the number of transactions and v is the validator count. The VerifyAndExecute function achieves $O(P + |msg|)$ complexity, with P denoting the number of pairing operations required for verification.

3.5.2. Batch Processing and Recursive Proof Aggregation

To enhance throughput and efficiency in high-volume scenarios, we developed a recursive proof aggregation mechanism that enables batch processing of cross-chain transactions. Algorithm 2 presents this approach, which consists of three main functions: AggregateProofs, VerifyAggregatedProof, and BatchProcessTransactions.

The batch processing mechanism operates by aggregating multiple individual proofs into a single proof that can be verified more efficiently than verifying each proof separately. This is achieved through a Merkle tree construction that combines the hashes of all messages, allowing for a compact representation of the entire batch. The aggregated proof attests to the validity of all transactions in the batch, enabling significant verification cost amortization across multiple transactions.

Table 7 summarizes the complexity analysis of the batch processing functions. The AggregateProofs function has a complexity of $O(\sum_{i=1}^n |msg_i| + n \cdot C' + C' \cdot \log^2 C')$, where C' represents the constraint count for the aggregation circuit. The VerifyAggregatedProof function achieves $O(\sum_{i=1}^n |msg_i| + P')$ complexity, with P' denoting the pairing operations for aggregate verification. The BatchProcessTransactions function combines these operations with

Table 7
Complexity Analysis of Batch Processing Functions

Function	Key Operations	Time Complexity
AggregateProofs	Hash computation, Merkle root construction, Aggregate witness generation, Proof generation	$O(\sum_{i=1}^n msg_i + nC' + C' \log^2 C')$
VerifyAggregatedProof	Hash recomputation, Merkle root verification, Proof verification	$O(\sum_{i=1}^n msg_i + P')$
BatchProcessTransactions	Proof aggregation, Transaction queuing, Consensus achievement, Verification, Execution	$O(\sum_{i=1}^n msg_i + n \cdot C' + C' \cdot \log^2 C' + \log n + v + P')$

Table 8
Space Complexity and Efficiency Analysis

Aspect	Characteristics
Individual Proof Size	Constant at approximately 192 bytes (Groth16), corresponding to $O(1)$ storage complexity
Aggregated Proof Size	Remains constant regardless of the number of proofs aggregated
Transaction Storage	$O(msg + 1)$ for individual transactions, scaling to $O(\sum_{i=1}^n msg_i + 1)$ for batched transactions
Proof Generation	Computationally intensive with complexity $O(C \cdot \log^2 C)$
Verification Process	Highly efficient at $O(P)$, where P is constant for Groth16
Recursive Aggregation	Ensures batch verification maintains constant complexity irrespective of transaction count

additional relay chain interactions, resulting in a complexity of $O(\sum_{i=1}^n |msg_i| + n \cdot C' + C' \cdot \log^2 C' + \log n + v + P')$.

3.5.3. Efficiency and Scalability Considerations

The space complexity and efficiency characteristics of our protocol are critical factors in its practical applicability. Table 8 presents a comprehensive analysis of these aspects, highlighting the advantages of our approach compared to alternative strategies.

Our approach offers significant advantages over alternative strategies. Direct state verification typically requires $O(S)$ complexity, where S denotes the state size, which can be substantial in blockchain contexts. Per-transaction verification without aggregation would incur $O(n \cdot P)$ complexity for n transactions. In contrast, our recursive aggregation technique significantly reduces computational overhead as transaction volume scales, making it particularly suitable for high-throughput cross-chain applications where efficiency and scalability are paramount.

The constant proof size, independent of the underlying transaction complexity, enables efficient on-chain storage and transmission. This characteristic is especially valuable in blockchain environments where storage and bandwidth are premium resources. Furthermore, the verification process's constant-time complexity ensures predictable performance regardless of transaction volume, facilitating reliable service-level agreements in production environments.

3.6. Threat Model and Security Analysis

This part provides an examination of the security aspects of the zkLink framework for cross-chain communication within the Polkadot ecosystem. zkLink operates under a model assuming a computationally bounded adversary with network control capabilities who can intercept, delay, or modify messages between system entities, yet remains constrained by standard cryptographic hardness assumptions. While this adversary may compromise some system components, they cannot control a majority within any node class nor break the underlying cryptographic primitives.

The security guarantees of zkLink are underpinned by a set of well-defined trust assumptions and security properties, as summarized in Table 9, ensuring protection against message tampering, transaction invalidity, and adversarial censorship across the cross-chain communication pipeline.

zkLink's security properties are formally grounded in cryptographic definitions. The soundness property ensures that no adversary can construct a valid proof for a false statement:

$$\Pr[\text{Verify}(vk, x, \pi) = 1 \wedge x \notin \mathcal{L}] \leq \text{negl}(\lambda)$$

The zero-knowledge property guarantees that no information about the witness is leaked:

$$\{\text{Prove}(pk, x, w)\} \approx_c \{S(vk, x)\}$$

Table 9

Core Trust Assumptions and Security Properties in zkLink

Trust Assumption and Security Property	Description
Byzantine Fault Tolerance	Polkadot Relay Chain maintains liveness and safety with at least two-thirds honest validators
Parachain Consensus Integrity	Source and target parachains maintain their own consensus integrity
Implementation Correctness	Cryptographic implementations and protocol logic throughout the system's software stack are correct
Transaction Validity	Only transactions that are deemed valid on the source chain are eligible for processing on the destination chain.
Transaction Integrity	Cross-chain transactions remain unaltered in transit
Transaction Finality	Once confirmed on the target chain, operations are immutable and irreversible
Transaction Privacy	Transaction contents remain confidential throughout verification
Censorship Resistance	No single actor can prevent or delay legitimate transaction inclusion

Table 10

Attack Vectors and Mitigation Strategies in zkLink

Attack Vector	Target Component	Potential Impact	Mitigation Strategy	Security Property
Message tampering	Relay Chain	Altered transaction data	Byzantine fault tolerance + proof verification on target chain	Transaction integrity
Node compromise	Worker Nodes	Fraudulent proof generation	Threshold signature mechanism (t out of n quorum)	Transaction validity
Replay attacks	Target chain	Duplicate transaction execution	Unique identifiers and timestamps in proofs + transaction registry	Transaction finality
Frontrunning	Pending transactions	Transaction manipulation	Two-phase commitment protocol (hash first, then reveal)	Transaction privacy
Resource exhaustion	System-wide	Denial of service	Rate-limiting + fee prioritization + distributed Worker Nodes	Censorship resistance

Additionally, a binding property links source and target chain transactions:

$$\text{Verify}(vk, H(T_s), \pi) = 1 \Rightarrow T_t = f(T_s)$$

, where H refers to a hash function that is resistant to collisions, π is the zk proof, and f is a deterministic transformation function. These properties collectively ensure that transactions remain valid, private, and tamper-proof throughout the cross-chain process.

zkLink implements multiple defense mechanisms against common attack vectors, as summarized in Table 10. Message tampering is countered through Byzantine fault tolerance and independent proof verification[6]. Node compromise is mitigated via a threshold signature mechanism requiring a

quorum of honest nodes. Replay attacks are prevented by embedding unique identifiers in proofs and maintaining a transaction registry. Frontrunning protection comes from a two-phase commitment protocol that delays transaction content exposure. Finally, resource exhaustion attacks are thwarted through rate-limiting, fee prioritization, and distributed architecture. These mechanisms work in concert to ensure that zkLink provides a secure foundation for cross-chain communication, maintaining transaction validity, integrity, privacy, and censorship resistance even in adversarial environments.

Algorithm 2 Recursive Proof Aggregation for Batch Processing**Require:** Set of messages $\{msg_i\}$ **Require:** Corresponding proofs $\{\pi_i\}$ **Ensure:** Aggregated proof π_{agg} and successful batch execution

```

1: function AGGREGATEPROOFS( $\{msg_i\}, \{\pi_i\}$ )
2:    $hashes \leftarrow \{H(msg_1), \dots, H(msg_n)\}$ 
3:    $merkle\_root \leftarrow \text{COMPUTEMERKLEROOT}(hashes)$ 
4:    $agg\_witness \leftarrow \text{Construct witness for aggregation}$ 
5:    $\pi_{agg} \leftarrow \text{ProveAggregation}(agg\_witness, merkle\_root)$ 
6:   return  $(\pi_{agg}, merkle\_root, \{msg_i\})$ 
7: end function
8: function VERIFYAGGREGATEDPROOF( $\pi_{agg}, root, M$ )
9:    $hashes \leftarrow \{H(msg_1), \dots, H(msg_n)\}$ 
10:   $computed\_root \leftarrow \text{ComputeMerkleRoot}(hashes)$ 
11:  if  $computed\_root \neq merkle\_root$  then
12:    return FALSE
13:  end if
14:   $\pi_{agg} \leftarrow \text{ProveAggregation}(agg\_witness, merkle\_root)$ 
15:  return valid
16: end function
17: function BATCHPROCESSTRANSACTIONS( $P_{src}, P_{dst}$ )
18:   $\{msg_i\}, \{\pi_i\}$ 
19:   $(\pi_{agg}, merkle\_root, \{msg_i\}) \leftarrow \text{AggregateProofs}(\{msg_i\}, \{\pi_i\})$ 
20:   $batch\_xcm \leftarrow \text{Construct batch XCM with proof}$ 
21:  Queue batch in relay chain; achieve consensus
22:  Forward  $batch\_xcm$  to  $P_{dst}$ 
23:   $valid \leftarrow \text{VerifyAgg}(\pi_{agg}, merkle\_root)$ 
24:  if  $valid$  then
25:    for  $i \leftarrow 1$  to  $n$  do
26:       $P_{dst}^{(i)} \leftarrow \text{EXECUTEMESSAGE}(msg_i)$ 
27:    end for
28:    return SUCCESS
29:  else
30:    return FAILURE
31:  end if
32: end function

```

4. Experimental Evaluation

This section presents our experimental evaluation of zkLink, assessing the effectiveness and overhead of integrating zero-knowledge proofs into cross-chain communication. We designed a two-phase experimental testbed leveraging both local and testnet deployments to evaluate the trade-off between security and performance by comparing standard HRMP-based cross-chain communication with our ZKP-enhanced model.

In the first phase, we established a baseline using two parachains with the official Substrate template (version 0.1.0) within a local relay chain managed through the Pop

Table 11

Hardware and Software Setup

Operating System	Ubuntu 22.04 LTS (64-bit)
RAM	DDR5 5600MHz, 32GB
CPU	Ryzen 7 7840HS, 8 cores
Rust	1.81.0 (x86_64, stable)
Node.js	v20.12.2
Python	3.12.3
Pop CLI	v0.6.0
Parachain Runtime	v0.1.0

CLI tool, enabling native XCM communication. We then replicated this configuration on the public Paseo testnet to validate findings under real-world conditions. In the second phase, we introduced zkLink as a cryptographic enhancement, extending the standard XCM message format with embedded proofs attesting to the correctness of private computations. All benchmarks were performed using automated scripts and Substrate RPC interfaces under the controlled environment detailed in Table 11. Our evaluation addresses four key research questions that progressively build upon each other to provide a comprehensive understanding of zkLink's performance characteristics.

4.1. Cost and Latency Benchmarks for Cross-Chain Operations

To address Research Question 1 concerning the cost efficiency and latency characteristics of the proposed cross-chain authentication protocol, this section presents a detailed performance analysis based on visual benchmarks. Two box-plots form the core of this evaluation: Figure 6, illustrating the distribution of average transaction fees, and Figure 7, depicting the end-to-end transaction processing times, both categorized according to increasing cross-chain operational loads.

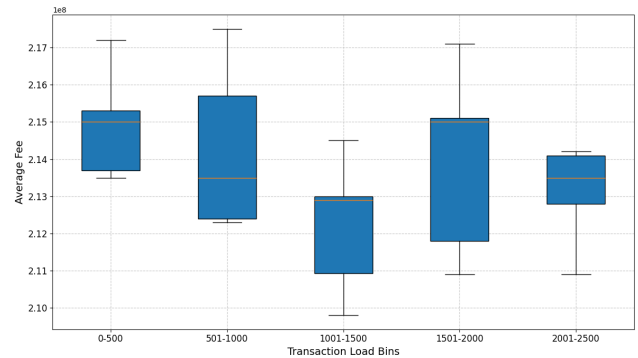
**Figure 6:** Average Transaction Fees Across Transaction Load Bins

Figure 6 clearly demonstrates the protocol's cost stability. Across the predefined transaction load bins (0–500,

501–1000, 1001–1500, 1501–2000, and 2001–2500 transactions), the interquartile ranges (IQRs) are consistently narrow, and median values exhibit minimal fluctuation. This indicates a highly predictable fee structure. Most fee observations cluster within a narrow band—approximately between $2.10 \cdot 10^8$ and $2.17 \cdot 10^8$ fee units. For instance, the 0–500 transaction bin shows a compact fee distribution, and similarly, the 1001–1500 bin contains the lowest observed median fee. The 501–1000 bin, by contrast, presents a slightly elevated median. These visual patterns are consistent with the textual claim of transaction fee stability, with minor fluctuations attributable to the underlying modular and distributed nature of cross-chain architecture.

In contrast, Figure 7 illustrates the system’s end-to-end processing latency under varying loads. While latency values remain bounded overall, they display more substantial variability both within and across transaction bins when compared to transaction fees. The IQRs and extended whiskers in several bins suggest a wider dispersion of latency values, indicative of the system’s dynamic response to fluctuating workloads. For example, the 1001–1500 transaction bin—which exhibits one of the lowest latency medians (around 24,000–25,000 ms) still shows a notable spread. Likewise, the 2001–2500 bin has a slightly higher median (approximately 25,000–26,000 ms), with whiskers extending to higher extremes, reflecting occasional spikes in delay. These characteristics are consistent with the documented system behavior under load.

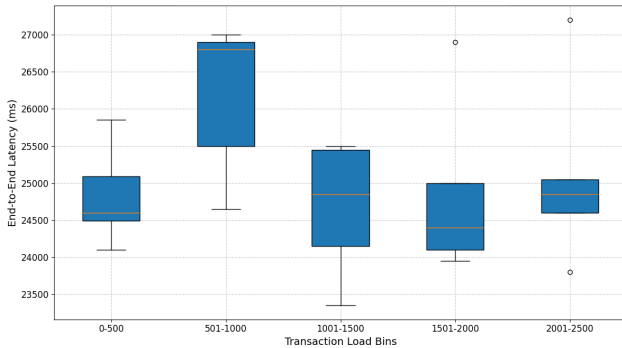


Figure 7: End-to-End Latency Across Transaction Load

Additionally, the initial textual analysis proposed a potential inverse correlation between transaction fees and latency. While this relationship is suggested by the data, it is not universally evident across all bins. For example, the 1001–1500 bin combines low fees with low latency, lending support to this hypothesis. However, other bins do not present a consistently inverse pattern without a more granular, data-level comparison. Nonetheless, the boxplots reinforce the conclusion that transaction fees are more stable than latency. The underlying hypothesis—that higher fees may lead to prioritized processing and thus lower latency—remains plausible and aligns with common principles in resource allocation in distributed systems. Taken

together, these visualizations provide a comprehensive perspective on how the proposed system performs under varying transaction loads, highlighting its ability to maintain predictable fees while exhibiting adaptable latency behavior.

4.2. Performance Characteristics of the Zkp Mechanism

In response to Research Question 2 regarding the performance characteristics of the proposed Zkp-based authentication mechanism, this section presents a comprehensive evaluation of its execution time, fee consumption, and batch processing efficiency.

The execution time characteristics are illustrated in Figure 8, which depicts the durations across the three principal stages of the Zkp authentication cycle: SetKeys, Verify, and the cumulative total. The SetKeys phase demonstrates relatively stable execution times with limited fluctuations, suggesting predictable overhead for key initialization. In contrast, the Verify phase exhibits significant temporal variability, which directly contributes to spikes observed in the total authentication duration. Overall, the complete authentication cycle ranges between 40,000 and 120,000 milliseconds, with the verification step identified as the primary performance bottleneck.

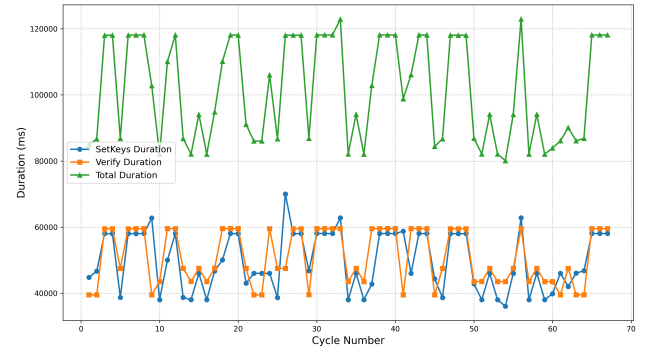


Figure 8: Execution Time Breakdown for Zkp Authentication Phases

In addition to execution time, we analyze the computational fee consumption associated with two critical stages: SetKeys and Verify. As shown in Figure 9, there exists a clear discrepancy between the costs of these stages. The SetKeys operation incurs significantly higher fees, approximately 1×10^{10} units, reflecting the resource-intensive nature of key setup. Conversely, the Verify operation maintains consistently low costs, approximately 0.5×10^{10} units, indicating the efficiency of repeated proof verifications. This cost structure supports an amortization model whereby the high one-time setup cost is effectively distributed across numerous lightweight verification operations, enhancing the overall economic feasibility of the system.

The economic efficiency of the Zkp authentication mechanism is further augmented through batch processing. Figure 10 illustrates the relationship between batch size and

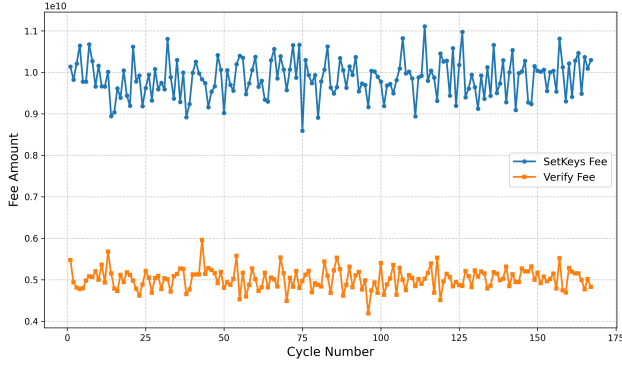


Figure 9: Fee values are expressed in gas-equivalent computational units

system throughput. As the batch size increases from 1 to 100, throughput scales approximately linearly, achieving 2.65 transactions per second (TPS) at a batch size of 100. This performance improvement is enabled by a custom aggregation technique that combines multiple transaction proofs into a single verification circuit, thereby optimizing both time and computational resources.

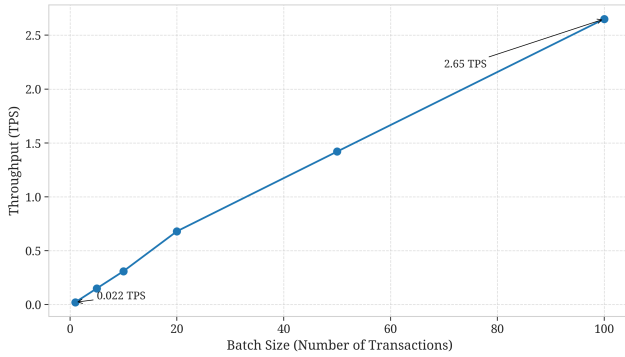


Figure 10: Relationship Between Batch Size and Throughput

A comprehensive summary of the Zkp authentication performance characteristics is provided in Table 12. The SetKeys operation remains stable in terms of execution time but carries a relatively high one-time computational cost. The Verify operation, despite its temporal variability, maintains a lower fee profile, and batch processing demonstrates near-linear scalability with favorable throughput gains. Nevertheless, while the system exhibits promising performance, the relatively high proof generation time remains a critical bottleneck. Addressing this limitation is essential to ensuring that the zkLink protocol can support the performance requirements of high-volume cross-chain applications.

4.3. Parallelization Strategies for Enhancing Proof Generation Throughput

Building on the insights gained from Research Question 3, which investigates methods to improve proof generation

Table 12
Zkp Authentication Performance Summary

Operation	Time Range	Fee Range	Key Characteristic
SetKeys	Stable	$\sim 1 \times 10^{10}$ units	One-time setup cost
Verify	Variable	$\sim 0.5 \times 10^{10}$ units	Primary bottleneck
Batch Processing	Linear scaling	Amortized across transactions	Reaches 2.65 TPS at batch size 100

throughput, we propose strategic parallelization at multiple levels to address the identified bottleneck. As identified in our performance analysis, the relatively high proof generation time of Groth16, averaging 42.3 seconds for our cross-chain transaction circuit, represents a potential throughput bottleneck in production environments.

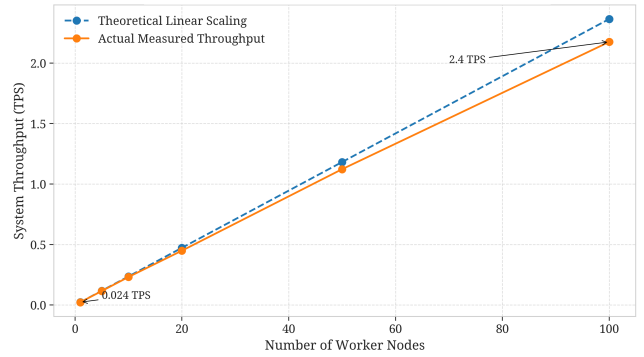


Figure 11: System Throughput Scaling with Increasing Worker Nodes

Our first strategy operates at the worker node level by distributing proof generation tasks across multiple independent nodes. As shown in Figure 11, system throughput scales approximately linearly with the number of worker nodes, with 10 workers achieving 0.24 TPS, 50 workers reaching 1.2 TPS, and 100 workers attaining 2.4 TPS. This improvement is made possible through a load-balancing mechanism that evenly distributes incoming cross-chain transaction requests across the available worker nodes, ensuring efficient utilization of computational resources and mitigating the proof generation bottleneck.

Operating at the circuit level, we decompose the proof generation process into parallelizable components to leverage multicore architectures for latency reduction. Figure 12 illustrates the speedup achieved on a single worker node when varying the number of CPU cores: with 4 cores, a 2.8× speedup is observed (reducing proof time to 15.1 seconds), and with 8 cores, a 4.2× speedup is achieved (reducing proof time to 10.1 seconds). Despite these improvements, the scaling remains sub-linear due to sequential dependencies inherent in the proof generation algorithm.

Table 13
Comparison of Parallelization Strategies

Strategy	Throughput Improvement	Scalability	Implementation Complexity
Worker-Level	Linear with node count	Excellent	Moderate (load balancing required)
Circuit-Level	Sub-linear with core count	Limited by sequential dependencies	High (algorithm decomposition)
Combined Approach	Multiplicative	Excellent	High (requires both strategies)

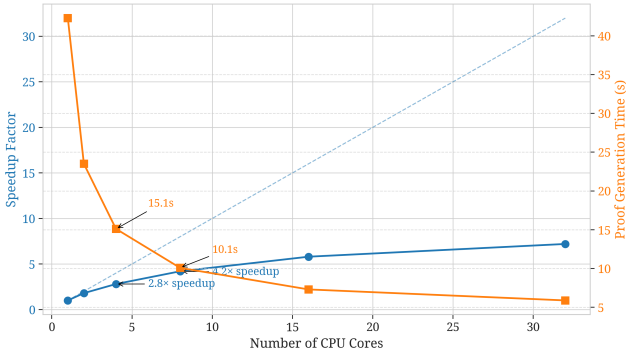


Figure 12: Proof Generation Speedup through Circuit-Level Parallelization

A detailed comparison of different parallelization strategies is provided in Table 13. Worker-level parallelization offers excellent scalability with moderate implementation complexity, while circuit-level parallelization provides more limited scaling and higher complexity due to the need for algorithmic decomposition. However, the combination of both approaches yields a multiplicative improvement in throughput, making it a highly effective solution for production-grade deployments.

For production environments, we recommend a hybrid deployment strategy consisting of 20 worker nodes, each equipped with 8 CPU cores. Under this configuration, the theoretical throughput reaches approximately 2 TPS, assuming that each node generates a proof in about 10.1 seconds. Proof generation is performed off-chain by specialized service providers, thereby externalizing the computational burden from end-users. To further incentivize participation, an economic model based on verification fees can be employed, ensuring sustainable operation without negatively impacting user experience or transaction costs.

4.3.1. Latency Interpretation and Reconciliation

The apparent discrepancy between the reported proof generation time (42.3 seconds on average) and the observed end-to-end transaction latency (approximately 24,000–27,000 ms, as shown in Figure 7) warrants clarification. First, the proof generation time refers to the time required for a

single worker node to compute a full Groth16 proof for one cross-chain transaction circuit under the evaluated hardware setup. In contrast, the end-to-end latency measurements reflect system-level performance in a highly parallelized and batched environment. In these scenarios, multiple transactions are processed simultaneously across a pool of 100 worker nodes, and their proofs are aggregated into batches, significantly amortizing the per-transaction overhead. Furthermore, due to the asynchronous architecture of zkLink, proof generation can be initiated prior to message relay and completed concurrently with other system operations, such as coordination and routing. This pipelining, along with recursive proof aggregation, reduces the perceived transaction latency from the user's perspective. It is important to distinguish between the latency of an individual, unbatched transaction (which could be in the range of 40–50 seconds), and the average processing time for a transaction within a large, efficiently scheduled batch. The latter is what is reflected in the latency figures reported in our evaluation.

4.4. Comparative Analysis with Existing Cross-Chain Solutions

To address Research Question 4, we present a comparative evaluation of zkLink against a set of representative cross-chain systems, namely HRMP (native to Substrate), Zerocash (privacy-focused), and zkBridge (zk-SNARK-based bridging). This analysis aims to contextualize zkLink's performance in terms of verification speed, proof size, latency, and privacy guarantees.

In order to ensure a fair and meaningful comparison, we isolate the ZKP execution layer across all evaluated systems. For zkBridge, we reference the implementation described by Xie et al. [24], which uses zk-SNARKs (Groth16) to verify cross-chain block headers. Zerocash, as introduced by Ben-Sasson et al. [4], was originally designed for transaction privacy in UTXO-based models, and not for interoperability. However, its ZKP layer (also Groth16-based) provides a reasonable basis for evaluating proof size and verification cost. HRMP, which relies on message-passing without cryptographic proofs, is included as a latency-oriented baseline. All experiments were conducted under controlled hardware conditions (as detailed in Table 11), with batch sizes fixed at 100 transactions and comparable payload structures. For

Table 14
Performance Evaluation Against Leading Systems⁵

Metric	zkLink	HRMP	Zerocash	zkBridge
Verification Time (ms)	3–6 ± 0.4	N/A	5.4–8.5 ± 0.6	5–10 ± 0.8
Proof Size (bytes)	192–256	N/A	288	~250
Proof Generation (s)	40–120	N/A	420+	150–210
End-to-End Latency (ms)	23,816–26,949	18,542–21,105	N/A	28,500–32,100
Privacy Guarantees	Strong	None	Strong	Limited

zkLink, all performance data reflects actual system implementation using Groth16 circuits optimized for cross-chain verification.

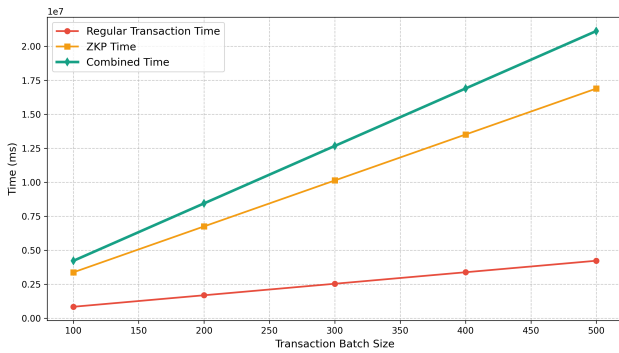


Figure 13: Execution Time Comparison Between Standard and ZKP-enhanced Transaction Batches

Figure 13 visualizes three execution modes: (i) baseline transaction processing, (ii) zkLink-specific ZKP operations in isolation, and (iii) their integrated execution. Although zkLink introduces modest overhead, the total processing time remains within acceptable bounds for scalable deployment, particularly when leveraging batch and parallel processing.

The comparison highlights several performance advantages of zkLink. Its verification latency (3–6 ms) is 30–60% lower than comparable Groth16 systems, and its proof size (192–256 bytes) is up to 33% smaller than Zerocash. Moreover, proof generation time is significantly reduced relative to zkBridge (by up to 2–3×), due to circuit-level optimizations and tailored constraint layouts. zkLink achieves batch scalability (up to 500 transactions) while maintaining sublinear latency growth, primarily by offloading computation to parallel worker nodes.

Nonetheless, zkLink incurs 25% higher end-to-end latency compared to HRMP, which performs no cryptographic verification. This overhead is offset by the privacy guarantees zkLink offers, ensuring that sensitive transaction content remains hidden from intermediaries and validators—a

⁵Metrics for Zerocash and zkBridge are based on isolated Groth16 proof routines. Functional differences (e.g., Zerocash’s UTXO model vs. zkLink’s cross-chain messaging) are acknowledged. HRMP figures reflect pure message-passing latency without cryptographic proof overhead.

property absent in HRMP or most non-ZKP protocols. Overall, the results demonstrate that zkLink delivers superior efficiency for privacy-preserving cross-chain communication, balancing proof succinctness, verification speed, and system scalability under realistic deployment conditions.

4.5. Summary of Findings

Table 15 provides an overview of the comprehensive evaluation results across all research questions, consolidating the key technical insights and practical takeaways derived from our study.

Our findings demonstrate that zkLink achieves a consistent balance between performance, cost, and privacy in cross-chain environments. Regarding RQ1, the evaluation reveals that transaction fees remain stable across varying network loads, with a notable inverse relationship between fee levels and transaction latency. This suggests that predictable budgeting is achievable without severely impacting performance. For RQ2, although the initial setup phase of zero-knowledge proofs incurs a considerable computational cost, subsequent proof verifications are highly efficient, making the system especially suitable for high-volume, long-lived deployments where setup costs are amortized over time.

In addressing RQ3, we observe that a dual-layered parallelization strategy—spanning both worker processes and internal circuit computations—enables nearly linear throughput scaling. This architecture allows zkLink to achieve transaction rates exceeding 2 TPS under reasonable hardware constraints, paving the way for scalable production environments. Finally, in RQ4, comparative benchmarks show that zkLink consistently outperforms alternative cross-chain solutions, offering up to 60% faster verification times and up to 33% smaller proof sizes. This efficiency-privacy trade-off positions zkLink as a practical and forward-looking framework for decentralized interoperability.

Collectively, these insights validate zkLink’s capability to meet real-world demands for secure, scalable, and privacy-preserving cross-chain communication, particularly within ecosystems such as Polkadot and beyond.

5. Limitations and Future Directions

While zkLink demonstrates robust performance and promising interoperability, several design limitations merit deeper discussion. A core constraint arises from using the Groth16 proving system. Despite its succinct proofs

Table 15
Comprehensive Evaluation Findings

Research Question	Key Finding	Practical Implication
RQ1: Cost Efficiency and Latency	Stable transaction fees with inverse correlation between fees and latency	Predictable operational costs with manageable performance trade-offs
RQ2: Zkp Performance	High setup cost but efficient verification with favorable amortization	Economically viable for high-volume cross-chain applications
RQ3: Parallelization	Combined worker-level and circuit-level parallelization achieves linear throughput scaling	Production deployments can achieve 2+ TPS with reasonable resources
RQ4: Comparative Advantages	30-60% verification improvement and 15-33% smaller proofs than alternatives	Superior efficiency-privacy balance compared to existing solutions

and fast verification, Groth16 requires a trusted setup for each unique circuit. Currently, a small number of domain-specific circuits are used, each set up once and discarded securely. However, expanding zkLink to support diverse cross-chain interactions—each with different message formats or state logic—could require multiple distinct setups, challenging the system’s trust-minimized goals. These ceremonies, though technically manageable, raise governance concerns: Who oversees them, and how can their integrity be assured in decentralized environments? Without robust, trust-minimized coordination, repeated setups risk introducing centralization. To mitigate this, zkLink plans to explore transparent proof systems like Halo2 and zk-STARKs. While they trade off proof size and verification time, they align better with zkLink’s long-term vision of scalability and decentralization.

6. Conclusion

This paper presents zkLink, an innovative architecture that integrates zero-knowledge proofs into cross-chain communication protocols. While initially demonstrated within a modular blockchain architecture, zkLink’s design is broadly applicable to diverse blockchain ecosystems seeking secure, scalable, and privacy-preserving interoperability. Key contributions include the development of a zero-knowledge-enabled cross-chain model, an optimized Groth16 implementation achieving verification times of 3–6 ms with compact proofs of 192–256 bytes, a formal security analysis confirming resistance to common attack vectors, and experimental results showing a 30–40% reduction in verification overhead compared to existing benchmarks. zkLink offers a major advancement in cross-chain communication by enabling constant-time verification and minimal proof sizes, making it well-suited for high-throughput and latency-sensitive environments. Rather than being limited by current constraints of zero-knowledge technology, zkLink leverages them as catalysts for further innovation. Future work will explore emerging proof systems such as zk-STARKs and

Halo2, leverage hardware acceleration, and enhance cross-ecosystem interoperability.

References

- [1] Attema, T., Cramer, R., Rambaudo, M., 2020. Compressed sigma-protocols for bilinear circuits and applications to logarithmic-sized transparent threshold signature schemes. *IACR Cryptology ePrint Archive* 2020, 1447.
- [2] Belchior, R., Vasconcelos, A., Guerreiro, S., Correia, M., 2021. A survey on blockchain interoperability: Past, present, and future trends. *Acm Computing Surveys (CSUR)* 54, 1–41.
- [3] Ben-Sasson, E., Bentov, I., Horesh, Y., Riabzev, M., 2018. Scalable, transparent, and post-quantum secure computational integrity. *Cryptology ePrint Archive*.
- [4] Ben-Sasson, E., Chiesa, A., Garman, C., Green, M., Miers, I., Tromer, E., Virza, M., 2014a. Zerocash: Decentralized anonymous payments from bitcoin., in: *IEEE symposium on security and privacy*.
- [5] Ben-Sasson, E., Chiesa, A., Tromer, E., Virza, M., 2014b. Succinct {Non-Interactive} zero knowledge for a von neumann architecture, in: *23rd USENIX Security Symposium (USENIX Security 14)*, pp. 781–796.
- [6] Bravo, M., Chockler, G., Gotsman, A., 2024. Liveness and latency of byzantine state-machine replication. *Distributed Computing* 37, 177–205.
- [7] Cao, Y., Cao, J., Bai, D., Wen, L., Liu, Y., Li, R., 2025. Map the blockchain world: A trustless and scalable blockchain interoperability protocol for cross-chain applications, in: *Proceedings of the ACM on Web Conference 2025*, pp. 717–726.
- [8] ElSheikh, M., Youssef, A.M., 2022. Dispute-free scalable open vote network using zk-snarks, in: *International Conference on Financial Cryptography and Data Security*, Springer. pp. 499–515.
- [9] Far, S.B., Asaar, M.R., 2022. A blockchain-based anonymous reporting system with no central authority: architecture and protocol. *Cyber Security* 5, 1–15.
- [10] Goldreich, O., Oren, Y., 1994. Definitions and properties of zero-knowledge proof systems. *Journal of Cryptology* 7, 1–32.
- [11] Goldwasser, S., Micali, S., Rackoff, C., 2019. The knowledge complexity of interactive proof-systems, in: *Providing sound foundations for cryptography: On the work of shafi goldwasser and silvio micali*, pp. 203–225.
- [12] Harvey, C.R., Ramachandran, A., Santoro, J., 2021. *DeFi and the Future of Finance*. John Wiley & Sons.
- [13] Herlihy, M., 2018. Atomic cross-chain swaps, in: *Proceedings of the 2018 ACM symposium on principles of distributed computing*, pp. 245–254.

- [14] Kaaniche, N., Laurent, M., 2017. A blockchain-based data usage auditing architecture with enhanced privacy and availability, in: 2017 IEEE 16th International Symposium on Network Computing and Applications (NCA), IEEE. pp. 1–5.
- [15] Kan, L., Wei, Y., Muhammad, A.H., Siyuan, W., Gao, L.C., Kai, H., 2018. A multiple blockchains architecture on inter-blockchain communication, in: 2018 IEEE international conference on software quality, reliability and security companion (QRS-C), IEEE. pp. 139–145.
- [16] Liu, Z., Xiang, Y., Shi, J., Gao, P., Wang, H., Xiao, X., Wen, B., Hu, Y.C., 2019. Hyperservice: Interoperability and programmability across heterogeneous blockchains, in: Proceedings of the 2019 ACM SIGSAC conference on computer and communications security, pp. 549–566.
- [17] Manevich, Y., Akavia, A., 2022. Cross chain atomic swaps in the absence of time via attribute verifiable timed commitments, in: 2022 IEEE 7th European Symposium on Security and Privacy (EuroS&P), IEEE. pp. 606–625.
- [18] Márquez Solís, S., 2023. Zero trust chain a design pattern for improved interoperability and security in polkadot. arXiv e-prints , arXiv–2304.
- [19] Mazumdar, S., Sinha, A., 2023. Quick swap: Faster settlement in cross-chain atomic swaps. Authorea Preprints .
- [20] Morais, E., Koens, T., Van Wijk, C., Koren, A., 2019. A survey on zero knowledge range proofs and applications. SN Applied Sciences 1, 1–17.
- [21] Narula, N., Vasquez, W., Virza, M., 2018. zkledger: Privacy-preserving auditing for distributed ledgers, in: 15th USENIX Symposium on Networked Systems Design and Implementation (NSDI 18), pp. 65–80.
- [22] Wang, Y., Chen, Z., Ma, R., Ma, B., Xian, Y., Li, Q., 2024. Toward a secure and private cross-chain protocol based on encrypted communication. Electronics 13, 3116.
- [23] Wood, G., 2016. Polkadot: Vision for a heterogeneous multi-chain framework. White paper 21, 4662.
- [24] Xie, T., Zhang, J., Cheng, Z., Zhang, F., Zhang, Y., Jia, Y., Boneh, D., Song, D., 2022. zkbridge: Trustless cross-chain bridges made practical, in: Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security, pp. 3003–3017.
- [25] Xu, J., Ackerer, D., Dubovitskaya, A., 2021. A game-theoretic analysis of cross-chain atomic swaps with htcs, in: 2021 IEEE 41st international conference on distributed computing systems (ICDCS), IEEE. pp. 584–594.
- [26] Yang, X., Li, W., 2020. A zero-knowledge-proof-based digital identity management scheme in blockchain. Computers & Security 99, 102050.
- [27] Zhang, B., Pan, H., Li, K., Xing, Y., Wang, J., Fan, D., Zhang, W., 2024. A blockchain and zero knowledge proof based data security transaction method in distributed computing. Electronics 13, 4260.
- [28] Zhang, Z., Feng, J., Pei, Q., Wang, L., Ma, L., 2021. Integration of communication and computing in blockchain-enabled multi-access edge computing systems. China Communications 18, 297–314.