Hedge Funds on a Swamp: Analyzing Patterns, Vulnerabilities, and Defense Measures in Blockchain Bridges [Experiment, Analysis & Benchmark]

Poupak Azad
University of Manitoba
Canada
azad@myumanitoba.ca

Jiahua Xu
University College London
United Kingdom
jiahua.xu@ucl.ac.uk

Feng Yebo
Nanyang Technological University
Singapore
yebo.feng@ntu.edu.sg

Cuneyt Gurcan Akcora
University of Central Florida
USA
cuneyt.akcora@ucf.edu

ABSTRACT

Blockchain bridges have become essential infrastructure for enabling interoperability across different blockchain networks, with more than $24B monthly bridge transaction volume. However, their growing adoption has been accompanied by a disproportionate rise in security breaches, making them the single largest source of financial loss in Web3. For cross-chain ecosystems to be robust and sustainable, it is essential to understand and address these vulnerabilities. In this study, we present a comprehensive systematization of blockchain bridge design and security. We define three bridge security priors, formalize the architectural structure of 13 prominent bridges, and identify 23 attack vectors grounded in real-world blockchain exploits. Using this foundation, we evaluate 43 representative attack scenarios and introduce a layered threat model that captures security failures across source chain, off-chain, and destination chain components.

Our analysis at the static code and transaction network levels reveals recurring design flaws, particularly in access control, validator trust assumptions, and verification logic, and identifies key patterns in adversarial behavior based on transaction-level traces. To support future development, we propose a decision framework for bridge architecture design, along with defense mechanisms such as layered validation and circuit breakers. This work provides a data-driven foundation for evaluating bridge security and lays the groundwork for standardizing resilient cross-chain infrastructure.

1 INTRODUCTION

Blockchain bridges have become essential infrastructure in the blockchain ecosystem, enabling interoperability between otherwise isolated networks. In its most basic asset-based bridge type, a bridge locks or burns assets on a source chain and mints or releases corresponding assets on a destination chain. This coordination preserves total supply across networks while facilitating asset mobility and composability. Bridges play a foundational role in enabling cross-chain decentralized applications, unifying fragmented liquidity pools, and allowing users to leverage features across heterogeneous platforms. For instance, monthly bridge transaction volume has exceeded $24B [1], underscoring their systemic significance.

However, this growing importance has been matched by an alarming trend: bridges represent the single largest source of financial loss in Web3 security breaches. As of mid 2025, 13 out of 39 bridges on l2beat.com are already labelled as insecure.[1] Of the top six bridges in total-value-locked as ranked by [2] in 2022, three (*Multichain* [3], *Ronin* [4], and *Rainbow* [5]) have already been hacked for more than $750M. A critical vulnerability in the sixth, *Polygon's Plasma Bridge* [6], could have exposed $850M; it was patched just in time after a white-hat disclosure that earned a $2M bounty.

Vulnerabilities range from smart contract bugs and improper verification logic to compromised multisig keys and failure-prone trusted validators. The result is a persistent threat model with catastrophic consequences: stealthy attacks, rapid fund drainage, and minimal recourse for victims. The threat of large-scale, unpredictable failure continues to undermine user trust and impede the adoption of cross-chain systems. With billions of USD locked in bridges, their current security risks make them resemble hedge funds operating atop a swamp; highly valuable, yet dangerously unstable and exposed to unpredictable attacks.

Despite a growing body of research and security reviews, there is still no unified framework to evaluate bridge vulnerabilities. Existing literature tends to focus on isolated case studies [7, 8], postmortem hack reports [9], or abstract protocol taxonomies [9, 10]. This fragmentation limits our ability to draw general conclusions, compare implementations, or develop preventive standards.

In this work, we take a data interoperability perspective to present a comprehensive security and privacy systematization of blockchain bridges, grounded in both formal modeling and empirical analysis. We argue that bridge security is fundamentally a data transfer security problem, concerned with state consistency, transaction ordering, and trust-minimized replication across chains. We believe that it must be addressed with the same rigor as traditional interoperability standards.

We approach the study of bridge security through three orthogonal lenses:

[1]The 13 bridges marked as vulnerable (indicated with a red cross and / or brown shield badge on the website) due to unverified contracts and past hacks are: *Aptos (LayerZero)*, *LayerZero v2 OFTs*, *Multichain*, *Connext*, *Omnichain (LayerZero)*, *Hyperlane Nexus*, *Socket*, *Chainport*, *Allbridge*, *StarGate (LayerZero)*, *Symbiosis*, *Everclear*, and *Orbit Bridge*.

First, we formalize the operational layers of bridge systems, spanning the source chain, off-chain intermediaries, and destination chain components. This layered model allows us to define three core security priors, such as cross-chain causality, and to reason about how different bridge designs instantiate or violate them.

Second, we perform a large-scale static analysis of bridge smart contracts deployed on Ethereum [11]. Using program analysis and custom metrics, we map the usage of role-based access control, defensive programming patterns, error handling, and call structures across 13 major bridge protocols. Next, we extract bridge-related transaction networks from on-chain data and apply graph machine learning to analyze usage and attacker behavior in real-world incidents. Our analysis spans 43 major attacks and reveals patterns in exploit phases, laundering strategies, and response delays.

Most importantly, we aim to answer four research questions to evaluate proactive goals: **RQ1** (Prevention): To what extent are bridge attacks preventable? Which defense techniques, such as static code analysis, formal verification, or trusted execution environments, are effective? Are some bridge types inherently more secure? **RQ2** (Detection): If prevention is not guaranteed, can attacks be detected in real time? What mechanisms (e.g., anomaly detection, on-chain monitoring) support timely identification? **RQ3** (Mitigation): Once an attack is detected, what mechanisms exist to mitigate damage and minimize asset loss? How do mitigation strategies differ across bridge types? **RQ4** (Outlook): What is the long-term outlook for bridge security? What foundational steps are needed to design formally verifiable or provably secure bridge protocols?

By unifying theoretical foundations with large-scale empirical analysis, this work offers the first comprehensive, data-driven systematization of blockchain bridge security. Our findings serve as a baseline for future research, standardization efforts, and the secure evolution of bridge protocols.

Our contributions are as follows:

- We formalize a layered model of blockchain bridge architectures and define core security priors relevant to cross-chain behavior.
- We develop a vulnerability taxonomy and introduce a formal notion of bridge attack surfaces based on trust assumptions and implementation details.
- We perform the first large-scale static analysis of deployed bridge contracts, quantifying security patterns across access controls, call structures, and guard mechanisms.
- We extract and analyze transaction-level behavior from bridge exploit incidents, identifying behavioral patterns across phases of use, compromise, and fund laundering.
- We provide security benchmarks and design recommendations to guide future bridge development, formal analysis, and regulatory evaluation.

2 RELATED WORK

Recent literature reflects an active effort to reconcile decentralization, trust minimization, and performance in blockchain bridges. We summarize case studies and surveys here and refer the reader to Appendix A in the supplementary material (available at our repository URL) for additional details on interoperability and benchmark studies.

Transaction analysis for bridges can identify operational patterns, including usage behaviors, anomalous activity, and indicators of compromise. Huang et al. [12] present an in-depth empirical analysis of Stargate, a prominent Layer-0 bridge with one of the highest total value locked (TVL) figures in the ecosystem. Using on-chain data from six EVM-compatible blockchains (Ethereum, Polygon [13], BSC [14], Avalanche [15], Arbitrum [16], Optimism [17]), they examine Stargate's transaction volume, user adoption, and operational patterns. This case study provides a data-driven benchmark for analyzing cross-chain liquidity transfers in practice. However, the study does not identify bridge priors nor carry out a static code analysis.

Subramanian et al. [18] benchmark the performance of blockchain bridge aggregators: services that route transfers across multiple bridge protocols for optimal speed or cost. They develop a framework to test popular aggregators (e.g., LI.FI, Socket, deBridge) by executing hundreds of cross-chain swaps and recording metrics such as fees, slippage, and latency. Their findings quantify differences in cost-efficiency and reliability, providing performance benchmarks for user-centric interoperability services. Our work differs in its broader focus and the development of a taxonomy to offer a holistic view of bridges.

Augusto et al. [19] introduce *XChainDataGen*, a framework for generating large-scale bridge transaction datasets. Using this tool, they collect approximately 35 GB of data from five major bridges deployed on 11 blockchains during the second half of 2024, extracting over 11.2 million bridge transactions that moved over $28B in token value. They compare protocols in terms of security, cost, and performance, contrasting, for example, full source-chain finality with "soft" finality, alongside fee models and emerging paradigms such as cross-chain intents. However, the study focuses on financial aspects and does not study bridge vulnerabilities that hinder bridge adoption in finance.

Recent surveys and taxonomies have focused on identifying systemic vulnerabilities in bridge designs. Li et al. [10] provide a comprehensive review of blockchain bridges, classifying architectures (e.g., lock-mint vs. notary schemes) and documenting common vulnerabilities such as smart contract flaws, centralization risks, liquidity issues, and oracle manipulation. While the study presents a broad taxonomy and useful design insights, it does not analyze transaction data or investigate real-world exploits. In a recent work, Augusto et al. [20] analyze bridge exploits; however, they do not conduct a direct analysis of what happens to stolen funds after bridge hacks, nor carry out static code analysis themselves. Instead, they rely heavily on systematic literature review, audit reports, bug bounty disclosures, and gray literature to collect information about vulnerabilities and attacks. Our analysis covers both the code and transaction analysis.

In contrast, Belenkov et al. [9] present a Systematization of Knowledge focused on major bridge hacks, including the $600M Axie *Ronin* exploit. They categorize failure modes such as compromised keys, multisig errors, and validator logic flaws, and propose best practices for prevention. However, the analysis is limited to static perspectives and excludes transactional behavior. Our work complements and extends these efforts by combining both static and transaction analysis perspectives and grounds them in a unified formal model.

Most security and forensic analyses of bridge hacks have been retrospective machine learning studies of transaction networks, typically conducted on a per-case basis and without developing generalizable vulnerability taxonomies or formalizing bridge-specific assumptions. For example, Augusto et al. [7] propose *XChainWatcher*, a Datalog-based monitoring system that detects anomalous token flows across bridges in real time. While it successfully reconstructs the *Ronin* and *Nomad* [21] attacks, its scope is limited to two case studies and does not address broader questions about interoperability security. Similarly, Lin et al. [22] develop *ABCTracer*, an automated framework for linking bridge transaction legs. Combining event log mining with machine learning inference, *ABCTracer* achieves 91.75% F1-score in identifying transaction pairs across 12 DeFi bridges. While valuable for forensics, the work does not define a vulnerability taxonomy nor investigate design assumptions underpinning bridge protocols.

Finally, Wu et al. [8] focus on identifying and classifying bridge-specific attacks. Using a dataset of 49 bridge exploits totaling nearly $4.3B in losses, they propose *BridgeGuard*, a graph-based detection system that models bridge transactions and flags anomalous behavior. Evaluated on 203 known attacks and 40,000 benign transactions, *BridgeGuard* outperforms prior methods in recall and detection precision. However, their work does not incorporate static analysis of contract logic or propose a general framework for understanding bridge security.

While these studies contribute valuable empirical and forensic insights, they tend to focus narrowly on individual case studies, static audits, or retrospective detection. In contrast, our work presents the first unified and multi-dimensional study that combines a formal model of bridge semantics, a systematic static analysis of deployed bridge contracts, and a large-scale transaction analysis across multiple chains.

3 BRIDGE MODELING AND FORMALIZATION

A blockchain b is an immutable ledger where transactions are appended chronologically. It comprises a set of peer-to-peer network nodes N_p, a set of address nodes N_a, a chronological history of transactions $\mathbb{H}$ between nodes in N_a, and consensus rules R that govern transaction creation, where $N_p \cap N_a \neq \emptyset$ and $b = (N_p, N_a, \mathbb{H}, R)$.

Asset Types. A clear distinction exists between coins, tokens, and wrapped assets. Coins are native digital currencies intrinsic to their own blockchains, such as Bitcoin (BTC) on the Bitcoin network [23], and are essential for paying transaction fees and participating in consensus mechanisms. A smart contract-based token θ represents a type of blockchain currency, with its state defined by the chronological history of transactions $\mathbb{H}_\theta$ in which θ has been involved. All blockchains issue native coins (e.g., Ether on Ethereum), but only some blockchains allow users to issue smart contract-based tokens (e.g., Storj on Ethereum).

Wrapped tokens are a subset of tokens designed to represent tokens from one blockchain on another, facilitating interoperability across disparate blockchain networks. For instance, Wrapped Bitcoin (WBTC) is issued on Ethereum, enabling BTC to be utilized within Ethereum's decentralized finance ecosystem. Despite enhancing bridge functionality, wrapped tokens inherit the limitations of standard tokens: they cannot be used to pay transaction fees on the host blockchain (i.e., Ethereum for WBTC).

A transaction $tx \in \mathbb{H}$ on b transfers a value v of a token θ from an address $a_1 \in N_a$ to an address $a_2 \in N_a$, and is associated with a timestamp t_{tx} indicating when the transaction occurred: $tx = (\theta, v, a_1, a_2, t_{tx})$. If the transaction is understood, we will simplify t_{tx} to t. Each transaction is assumed to be valid under the consensus rules R of b.

Blockchain Layers. Blockchain systems are structured into hierarchical layers to manage scalability, functionality, and specialization.

- **Layer 1 (L1):** Base layer of a blockchain network, encompassing the core protocol responsible for transaction validation, consensus mechanisms, and data storage. L1 blockchains operate independently and are the foundation upon which other layers are built. Examples include Bitcoin, Ethereum, and Solana [24].
- **Layer 2 (L2):** Built atop L1 blockchains, L2 protocols aim to enhance scalability and transaction throughput without altering the base protocol. They achieve this by processing transactions off-chain or in parallel, subsequently settling them on the L1. Notable L2 implementations include Optimism and Arbitrum on Ethereum.
- **Layer 3 (L3):** L3 represents an emerging concept focusing on application-specific functionalities. These are protocols or networks constructed on L2 solutions, offering tailored environments for decentralized applications (dApps). L3s aim to provide enhanced scalability, interoperability, and customization, facilitating complex applications like decentralized finance platforms and gaming ecosystems. Examples of L3 projects include Orbs Network [25] and XAI Games [26] on Arbitrum.

3.1 Blockchain Bridges

A blockchain bridge facilitates the transfer of tokens across distinct blockchain domains. While many bridges connect L1 blockchains, others operate across layers, such as the *Arbitrum Canonical Bridge* between Ethereum (L1) and Arbitrum (L2). Bridges that involve L3 layers are an emerging area with early examples such as *Arbitrum Orbit*, *zkSync Hyperchains* [27], and *Orbs Network*. Despite their growing importance, blockchain bridges lack a consistent formalization in the literature, which we aim to define as follows.

We consider two blockchain domains (i.e., chains or protocols), $b_1 \in \{L1, L2, L3\}$ and $b_2 \in \{L1, L2, L3\}$, which may differ in node sets, transaction histories, and consensus rules. We define a bridge between the domains b_1 and b_2 through an implementation mechanism $\mathcal{I}$ that governs the operational logic of the bridge:

$$\mathbb{B}_{1\leftrightarrow 2} = (\{b_1, b_2\}, \mathcal{I}) \tag{1}$$

We categorize bridges along two orthogonal axes: their *operational trust model* and their *functional type*. The trust model refers to how source domain activity is verified and includes *trusted*, *trust-minimized*, and *trustless* designs, formalized shortly in Section 3.5. The functional type captures how value is transferred and includes:

- **Asset Bridges:** In *asset bridges* (also known as burn-and-mint models), the bridge locks or burns an asset θ_1 on the source domain b_1 and creates a representative asset θ_2 on the destination

domain b_2. These assets are not inherently equivalent; their linkage is established solely through the semantics of the bridge protocol. They include:

- *Externally-Verified Asset Bridges*, which depend on multisigs, notaries, or sidechains to verify lock events (e.g., *Wormhole* [28], *Ronin*, *Multichain*).
- *Rollup-Native Asset Bridges*, which leverage L1 consensus to verify L2 state transitions using fraud or validity proofs (e.g., Arbitrum, Optimism, Loopring [29]). These are considered the most trustless.

The settlement process on asset-based bridges may be slow and require specialized verification and challenge code. For example, optimistic rollups like Arbitrum or Optimism impose a 7-day challenge period on withdrawals.

- **Liquidity Networks:** In contrast to asset-based bridges, *liquidity network-based bridges*, such as Connext[30] or Across[31], avoid minting and instead fulfill user requests through liquidity providers (LPs) who maintain reserves on both domains. LPs are economically incentivized via transfer fees and, in some protocols, additional yield or reward mechanisms. This design enables faster settlement and circumvents the latency of on-chain verification and challenge periods.
- **Hybrid Bridges:** These support both functional models, often depending on the specific chain pair or asset type. For instance, *Multichain* combines canonical minting with liquidity provisioning.

A bridge transaction tx moves a value v of token θ_1 from an address $a_1 \in N_a$ on b_1 to an address $a_2 \in N_a$ on b_2:

$$tx = (\theta_1, v, a_1, a_2, t_{tx}) \tag{2}$$

The transformation of a bridge transaction state χ under the bridge's operation is defined as:

$$\chi_0 \xrightarrow{\mathbb{B}_{1\leftrightarrow 2}} \chi_t \tag{3}$$

where χ_0 and χ_t represent the initial and final states of the transaction across chains.

The implementation $\mathcal{I}$ of the bridge includes mechanisms on the source chain, off-chain components, and the destination chain. Additionally, the bridge relies on a node trust set $\mathbb{T}$, comprising entities responsible for validating and securing bridge transactions. The composition of $\mathbb{T}$ is determined by the specific implementation $\mathcal{I}$ and the security assumptions of $\{b_1, b_2\}$.

3.2 Bridge Mechanism

Consider that Alice wants to move some assets from the source chain b_1 to the destination chain b_2. For simplicity in exposition, we will assume that at a given time, the bridge is used by one user only and for bridging one token only. The token Alice will be moving is θ_1, and the representation of the token on b_2 is θ_2. We track the movement of value across the bridge $\mathbb{B}_{1\leftrightarrow 2}$ as a function of time and fee structure. The state of the bridge transaction χ is described by four accounts: Alice's addresses a_1 on b_1 and a_2 on b_2 along with the bridge addresses c_1 on b_1 and c_2 on b_2. The state of the bridge transaction is represented as $\chi = (a_1, c_1, a_2, c_2)$.

We track the mechanisms in three stages: source chain mechanism, off-chain mechanism, and destination chain mechanism.

3.2.1 Source Chain Mechanism. Alice initiates the transfer on b_1. Initially, at $t = 0$, Alice holds v_1 units of the token θ_1 at her address a_1, with a total value $v_1 \cdot price(\theta_1, t)$, where v_1 is the amount of the token, and $price(\theta_1, t)$ represents the price of the token θ_1 in a fiat currency such as USD at time t. We will use $a_x \mapsto v$ to show that the balance of address a_x is v. Hence, the initial state of balances is $\chi_0 = (a_1 \mapsto v_1, c_1 \mapsto 0, a_2 \mapsto 0, c_2 \mapsto 0)$.

If Alice wants to transfer an amount $v_x \leq v_1$ to her address a_2 on the other blockchain, she initiates the transaction $tx_{b_1} = (\theta_1, v_x, a_1, c_1, t_{tx_{b_1}})$ on b_1.

She also pays a bridge fee F_{forward}, which consists of the transaction processing fee f_1 on b_1 and f_2 on b_2, as well as the bridge operation fee (f^*). Hence the total fee $F_{\text{forward}} = f_1 + f_2 + f^*$. For simplicity, we assume that the bridge operation fee is charged on the initiating blockchain b_1. The sent amount v_x and fees are deducted from the initial balance of $a_1 \mapsto v_1$, hence $a_1 \mapsto (v_1 - v_x - f_1 - f^*)$. This leaves only f_2 to be paid on b_2.

After b_1's block confirmation duration d_{b_1}, the bridge smart contract on b_1 acknowledges tx_{b_1} and locks $(c_1 \mapsto v_x)$ or burns $(c_1 \mapsto 0)$ the received assets depending on the implementation $\mathcal{I}$. We follow the locked asset scenario, and the state is updated as $\chi_{t_{tx_{b_1}}} = (a_1 \mapsto (v_1 - v_x - f_1 - f^*), c_1 \mapsto v_x, a_2 \mapsto 0, c_2 \mapsto 0)$. The blockchain state changes as $b_1 = (N_p, N_a, \mathbb{H} \cup \{tx_{b_1}\}, R)$.

3.2.2 Off-chain Mechanism. An off-chain communicator observes transactions to bridge address c_1, specifically transactions of the form: $tx_{B_1} = (\theta, v, a, c, t)$. Upon detection, it instructs c_2 to mint $v_x - f_2$ worth of θ_2. The off-chain mechanism takes time d_{off} to notice the first transaction and send a signal to the b_2. Thus, at $t_{\text{off}} = t_{tx_{b_1}} + d_{\text{off}}$, the off-chain mechanism signals the start of the token transfer process on the second blockchain b_2. Specifically, the mechanism signals the smart contract on b_2 accordingly. There are multiple mechanisms to create this signal, and we refer the reader to [9] for a comprehensive list. In a simple oracle-based solution, a network of oracles monitors transactions on blockchain b_1. These relayers are either permissioned (managed by a specific entity) or decentralized (e.g., a set of staked validators). When a bridge transaction tx_{b_1} is observed, relayers submit proofs or messages (through transactions that push data to the blockchain) to a contract on b_2 to trigger the minting or unlocking of assets. The node trust set $\mathbb{T}$ validates these processes based on the implementation $\mathcal{I}$.

3.2.3 Destination Chain Mechanism. The transfer completes when $v_x - f_2$ worth of θ_2 is minted on b_2 and sent to Alice's address a_2: $tx_{b_2} = (\theta_2, v_2 = 0, a_2 \mapsto v_x - f_2, c_2 \mapsto 0, t_{\text{off}})$. After b_2's block confirmation duration D_{b_2}, the complete bridge process ends at $t = t_{tx_{b_1}} + d_{\text{off}} + d_{b_2}$. The final state is: $\chi_t = (a_1 \mapsto (v - v_x - f_1 - f^*), c_1 \mapsto 0, a_2 \mapsto v_x - f_2, c_2 \mapsto 0)$.

The destination domain is $b_2 = (N_p, N_a, \mathbb{H} \cup \{t_{b_2}\}, R)$. The smart contract on b_2 handles the transaction and updates the blockchain state accordingly.

3.2.4 Reverse Process. The reverse transfer follows the same mechanism as the forward transfer but in the opposite direction. Alice initiates a transaction on b_2 to send v_x of θ_2 back to b_1. The bridge smart contract on b_2 locks $(c_2 \mapsto v_x)$ or burns $(c_2 \mapsto 0)$ token θ_2, and after confirmation, an off-chain mechanism signals b_1 to release

$(v_x - F_{\text{reverse}})$ of θ_1 to Alice's address a_1. The final state transition mirrors the forward process, with updated fees and timestamps.

3.3 Formalization of Bridge Implementations

Blockchain bridges can be categorized based on their trust assumptions, yielding three classes: trustless, trusted, and trust-minimized. A trustless bridge has no external trusted entities, with $\mathbb{T}_{trustless} = \{\}$, relying exclusively on the security guarantees of the underlying blockchains. A trusted bridge introduces external dependencies, requiring $\mathbb{T}_{trusted} \neq \{\}$. A trust-minimized bridge is a subset of trusted bridges in which all trusted entities $\mathbb{T}_{trustmin}$ consist of deterministic, publicly auditable algorithms (for example, on-chain smart contracts, oracles, or relays). While trust-minimized bridges rely on external components, their correctness can be independently verified by users.

The bridge trust set is expressed as $\mathbb{T} = \mathbb{T}_{src} \cup \mathbb{T}_{off} \cup \mathbb{T}_{dest}$, where $\mathbb{T}_{src}$ captures trust assumptions on the source blockchain, $\mathbb{T}_{off}$ captures trust in off-chain mechanisms (such as relayers or validators), and $\mathbb{T}_{dest}$ captures trust in the destination blockchain. Two key metrics, $Size(\mathbb{T})$ and $cost(\mathbb{T})$, characterize the scale of the trusted set and its overall fee impact, respectively. These metrics influence the security, decentralization, and efficiency of a bridge.

3.3.1 Source Chain Implementation for Trust Set $\mathbb{T}_{src}$. The source chain component locks or burns assets before triggering $\mathbb{T}_{off}$. It can be realized via smart contracts (so $\mathbb{T}_{src} = \{SCs\}$), validator-controlled mechanisms ($\mathbb{T}_{src} = \{\}$), or hybrid approaches that combine both. Smart contract mechanisms require trusting the correctness of the contract logic, whereas validator-based methods rely on the blockchain's consensus, eliminating additional trust. Hybrid methods usually trade off fees or time for scalability and security. As shown in Table 1, these components appear on both source and destination chains, but since their role is more critical on the destination side, where bridged tokens are released, we omit their detailed explanation here and defer it to the destination chain discussion to avoid repetition.

3.3.2 Off-chain Implementation for Trust Set $\mathbb{T}_{off}$. Bridges typically use either a set of notaries (N) or a set of light clients (L) for off-chain processing.

Notaries. Notaries introduce additional trust in external parties ($\mathbb{T}_{off} \neq \{\}$) that monitor transactions and ensure bridge causality, with security depending on the number of notaries $Size(N)$ and their operational costs $cost(N)$. As $Size(N)$ and $cost(N)$ grow, security improves, but so do fees and transaction delays. The expected loss costs $\mathbb{E}(C)$, bridge fees F, and transaction time delay D follow $\mathbb{E}(C) \propto 1/Size(N)$ and $F, D \propto Size(N), Cost(N)$.

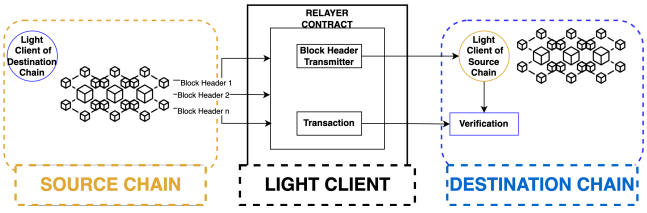

Figure 1: Light Client Implementation

Light Clients. Light clients rely on Merkle proofs (M) and perform verification on-chain via light client contracts (L). These systems do not depend on external entities, so the off-chain trust set is $\mathbb{T}_{off} = \{L, M\}$. These bridges are trust-minimized because no external entities are needed beyond the blockchain itself. Verification time t_{proof} affects expected loss costs $\mathbb{E}(C)$ and bridge cost F, D according to $\mathbb{E}(C) \propto 1/t_{proof}$, $F, D \propto t_{proof}$. Although frequent proof generation boosts security, it also increases fees. Light clients must run on each chain, reducing overall scalability. Figure 1 illustrates a standard light client implementation used in many trustless bridge designs.

Some bridges blend the two approaches. A security-optimized hybrid requires that either the light client or the notary set remain honest, so $\mathbb{T}_{\text{off}} = \{L, M\} \cap \{N\}$. Another hybrid uses light clients where available and defaults to notaries otherwise: $\mathbb{T}_{\text{off}} = \{L, M\} \vee \{N\}$.

Sidechains. A third option is using sidechains, which function as independent blockchains that can facilitate bridge transfers. Sidechains add a secondary blockchain with its own consensus R'. A bridge passing through a sidechain depends on $\mathbb{T}_{\text{off}} = (\{L, M\} \vee \{N\} \vee (\{L, M\} \cap \{N\})) \cup R'$.

In other words, instead of relying exclusively on a notary network or on-chain light clients to handle the bridging, a sidechain can be set up with built-in asset-transfer features. Bridge transactions then pass through that sidechain, inheriting its consensus and security properties.

While the idea of a sidechain can still incorporate notaries or light clients under the hood, the main distinction is that a sidechain is an entire blockchain in its own right, rather than a narrowly defined tool like a light client or a notary service. This extra layer can increase scalability (since the sidechain can process many transactions internally) but also introduces new trust assumptions. Specifically, the correctness and security of the sidechain's consensus mechanism. Figure 2 illustrates a representative sidechain-based bridge design.

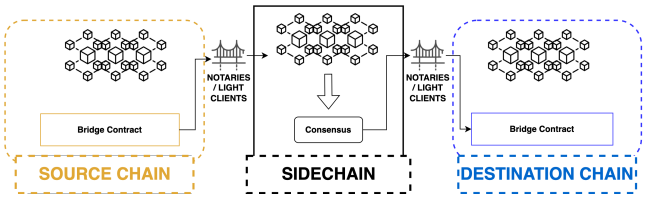

Figure 2: Sidechain Implementation

3.3.3 Destination Chain Implementation $\mathbb{T}_{dest}$. The destination side enforces token minting, final validation, and recipient assignment. Possible methods include smart contracts ($\mathbb{T}_{dest} = \{SCs\}$), validator-control ($\mathbb{T}_{dest} = \{\}$), custodian-based ($\mathbb{T}_{dest} = \{C\}$), or a hybrid that integrates these techniques. The choice depends on constraints related to cost, security, and overall efficiency.

Smart Contracts. A smart contract can be deployed on the destination chain to handle final token transfers. After receiving a valid signal and data from $\mathbb{T}_{off}$, the contract locks, mints, or releases

Table 1: Cross-chain bridge classification for the highest volume bridges. ●: trusted, ◐: trust minimized, ○: trustless.

	Bridge	Source Chain	Destination Chain	Trust
Notaries	Nomad Bridge	Hybrid	Smart Contracts	●
	Allbridge Classic	Smart Contracts	Smart Contracts	●
	deBridge	Smart Contracts	Smart Contracts	●
	Multichain	Smart Contracts	Smart Contracts	●
	Wormhole Bridge	Smart Contracts	Smart Contracts	●
	Avalanche Bridge	Smart Contracts	Validator Control	●
	Ronin Bridge	Smart Contracts	Smart Contracts	●
	Wanchain Bridge	Smart Contracts	Smart Contracts	●
Light Client	BTC Relay	Validator Control	Smart Contracts	◐
	zkBridge	Validator Control	Validator Control	◐
	Rainbow Bridge	Validator Control	Validator Control	◐
	PeaceRelay	Smart Contracts	Smart Contracts	◐
Hybrid	Connext	Smart Contracts	Smart Contracts	●
	Optics Bridge	Smart Contracts	Smart Contracts	●
Sidechain	Cosmos IBC	Validator Control	Validator Control	○
	Gravity Bridge	Smart Contracts	Smart Contracts	◐
	zkRelay	Smart Contracts	Smart Contracts	◐
	Cactus	Smart Contracts	Smart Contracts	◐
	Celer cBridge	Smart Contracts	Smart Contracts	◐
	Orbit Bridge	Smart Contracts	Smart Contracts	◐
	Axelar	Smart Contracts	Smart Contracts	◐
	Chainswap	Smart Contracts	Smart Contracts	◐
	PolyBridge	Smart Contracts	Smart Contracts	◐
	pNetwork	Smart Contracts	Smart Contracts	◐
	Meter Passport	Smart Contracts	Smart Contracts	◐
	QANX Bridge	Smart Contracts	Smart Contracts	◐
	Binance Bridge	Smart Contracts	Smart Contracts	◐
	Horizon Bridge	Smart Contracts	Smart Contracts	◐
	Plasma Bridge	Smart Contracts	Smart Contracts	◐

tokens to the designated address. Because users must trust the correctness of the contract code, we have $\mathbb{T}_{dest} = \{SC\}$.

Validator Control. If the destination blockchain's existing consensus participants validate the bridging process, no additional trusted entities are introduced. The bridge may still have predefined logic to identify the destination address, but final checks are performed by decentralized validators. Thus $\mathbb{T}_{dest} = \{\}$.

Hybrid. A hybrid model combines smart contracts with validator-based consensus to balance security, cost, and time. The exact composition of $\mathbb{T}_{dest}$ depends on how responsibilities are allocated between on-chain code and validator consensus.

3.4 Bridge Security Priors

We formalize three key security properties that bridges must satisfy to ensure token parity, transaction causality, and value preservation across blockchains. A fourth foundational requirement is the liveness of the bridge, but we assume that the underlying blockchains are live and capable of processing transactions in a timely manner, allowing us to focus on security properties specific to the bridge itself.

Bridge Peg. Blockchain domains b_1 and b_2 often have disparate implementations of tokens. For users to reliably use token θ_1 on b_1 as a representation of token θ_2 on b_2, the bridge must establish that the asset amounts remain equivalent, i.e., $v_1 \equiv v_2$. This holds only if a peg exists between θ_1 and θ_2, ensuring that one θ_1 can always be exchanged for one θ_2. For simplicity, we ignore the total fee $F_{forward}$ that would be incurred to use the bridge, and state that given token prices at time t, denoted as $price(\theta_1, t)$ and $price(\theta_2, t)$, the bridge must ensure

$$v_1 \cdot price(\theta_1, t) \equiv v_2 \cdot price(\theta_2, t), \quad \forall t. \tag{4}$$

To uphold this peg, the bridge must maintain both liveness and security. If the bridge experiences technical failure or an attack, equivalency in (4) may break down, leading to price divergence between the native and bridged tokens, which can disrupt user trust and capital efficiency.

Bridge Causality. A secure bridge must enforce causality, ensuring that no user extracts value that was never created or loses value that was not meant to be lost. Formally:

$$\begin{aligned} &\forall tx_{B_2} = (\theta_2, , , c_2, t_2), \quad \exists! tx_{B_1} = (\theta_1, , , c_1, t_1) \\ &\forall tx_{B_1} = (\theta_1, , , c_1, t_1), \quad \exists! tx_{B_2} = (\theta_2, , , c_2, t_2) \\ &\text{such that } t_1 < t_2. \end{aligned} \tag{5}$$

This ensures a bijective mapping between transactions on b_1 and b_2, preventing double-minting or loss of funds. Additionally, the bridge must preserve temporal ordering, enforcing $t_1 < t_2$ so that a transaction on b_1 must occur before its corresponding transaction on b_2.

Bridge Consistency. The bridge must guarantee that tokens locked on b_1 remain inaccessible until the corresponding tokens on b_2 are either burned or locked. This consistency constraint ensures that the system does not create or destroy value arbitrarily.

$$v_2 \neq 0 \Rightarrow \neg \exists tx_{B_1}(\theta_2, v_1, , c_1,) \tag{6}$$

Equation (6) prevents a user from simultaneously withdrawing locked assets on b_1 and holding minted tokens on b_2, which could result in infinite money creation.

$$v_2 > 0 \Rightarrow \exists! \theta_1 \text{ such that } c_1 \mapsto a_2. \tag{7}$$

Equation (7) strengthens this by enforcing a strict one-to-one correspondence between locked and minted tokens.

Value at Risk. Failures in bridge security can lead to asset loss with custodial, contract, and economic risks. In custodial risk, an externally controlled address or multi-sig wallet is compromised, and funds are permanently stolen. In contract risk, a smart contract-based bridge has a vulnerability (e.g., reentrancy attack) where attackers can extract tokens illicitly. In economic risk, a bridge relies on economic assumptions (e.g., optimistic rollups or bonded validators), but adversaries manipulate incentives to destabilize the system.

Losses may occur at i) the bridge contract level, where contract c_1 on b_1 and contract c_2 on b_2 impact all users of the bridge, or ii) at the user level, where losses occur for individual user addresses a_1 or a_2. User-level losses (e.g., a_2 is not a valid address) are orthogonal to bridge security as long as they are not caused by flaws in bridge design.

A bridge's trust model (custodian-based, validator-based, or smart contract-based) influences its risk profile and determines how funds are secured.

3.5 Formalization of Bridge Attack Surfaces

Current research on bridge attacks is fragmented, lacking a consistent definition of attack surfaces and vectors. We present a formalized model for analyzing bridge security.

Definition 3.1 (Attack). *If* (4), (5), (6), *or* (7) *are violated due to malicious agents, the bridge is under attack.*

Definition 3.2 (Failure). *If* (4), (5), (6), *or* (7) *are violated due to technical errors but without malicious intent, the bridge is experiencing failure.*

Theorem 1. *If a bridge experiences an attack or failure, then* (4) *is always violated.*

Due to space limitations, the proof is given in Appendix B.

Definition 3.3 (Attack Vector V). *A vulnerability, pathway, or method that a malicious agent can exploit to launch an attack on a bridge* $\mathbb{B}_{1\leftrightarrow 2}$.

Definition 3.4 (Attack Surface Σ). *The sum of all attack vectors for a given* $\mathbb{B}_{1\leftrightarrow 2}$. *Formally, an attack surface is defined as* $\Sigma = \langle \mathbb{T}, \mathcal{I} \rangle$, *where* $\mathbb{T}$ *represents trusted entities and* $\mathcal{I}$ *represents implementation details.*

An attack surface can be decomposed into sub-surfaces for finer-grained analysis. A sub-surface $\Sigma^{'}_{\mathbb{B}_{1\leftrightarrow 2}} \in \Sigma_{\mathbb{B}_{1\leftrightarrow 2}}$ is defined as:

$$\Sigma^{'}_{\mathbb{B}_{1\leftrightarrow 2}} = \langle \mathbb{T}^{'}, \mathcal{I}^{'} \rangle, \quad \text{where} \quad \mathbb{T}^{'} \in \mathbb{T}, \quad \mathcal{I}^{'} \in \mathcal{I}.$$

Attack vectors in $\Sigma^{'}_{\mathbb{B}_{1\leftrightarrow 2}}$ use all trusted entities $\mathbb{T}^{'}$ and span specific implementations $\mathcal{I}^{'}$.

3.5.1 Examples of Attack Surfaces. In custodial key compromise ($\mathbb{T} = \{C\}$), a centralized custodian's private key is compromised, and all assets in the bridge are at risk. In the reentrancy attack on smart contract-based bridges ($\mathbb{T} = \{SCs\}$), an attacker exploits an improper withdrawal function to repeatedly drain funds. In validator collusion ($\mathbb{T} = \{V\}$), the validators of a PoS-based bridge collude; they approve fraudulent transactions and steal assets.

3.5.2 An Attack Surface-Damage Model.

Definition 3.5 (Damage Potential/Effort Ratio). *The Damage Potential/Effort Ratio of an attack vector* V, *denoted* $der(V)$, *quantifies the feasibility of an attack and is defined as* $der(V) = \frac{I(V)}{E(V)}$, *where* $I(V)$ *represents the expected damage or impact of the attack vector and* $E(V)$ *represents the computational, economic, or procedural effort required to execute the attack.*

An attack vector V is viable if and only if $der(V) > 1$, indicating that the expected damage outweighs the effort required to execute the attack. Hence,

$$der(V) = \begin{cases} 1, & \text{if } der(V) > 1 \\ 0, & \text{otherwise} \end{cases}$$

We define the attack surface area of a bridge $\mathbb{B}_{1\leftrightarrow 2}$, or a subset of it, containing n attack vectors as:

$$Area(\Sigma^{'}_{\mathbb{B}_{1\leftrightarrow 2}}) = \sum_{i=1}^{n} der(V_i) \tag{8}$$

We apply this framework to analyze security across distinct bridge implementation layers, including i) on-chain contracts (handling token locking, minting, and withdrawal), ii) off-chain relayers (notary or validator networks responsible for cross-chain verification) and iii) finality mechanisms (ensuring transaction irreversibility and bridging safety). This layered security model enables targeted risk mitigation by isolating attack surfaces based on implementation constraints.

3.6 Attack Surfaces of Blockchain Bridges

Blockchain bridges rely on multiple components (on-chain smart contracts, off-chain relayers, and destination chain mechanisms), each introducing distinct security risks. We formalize these risks through attack surface analysis and summarize them in Table 2. Vectors are defined in Appendix E.

3.6.1 Source Chain Layer Security. The attack surface of a bridge's source chain implementation is given by $\mathcal{I}_{src} = \langle \mathbb{T}_{src}, R_{src} \rangle$. For a bridge deploying n smart contracts on the source chain, we model its attack surface as $\Sigma_{src} = \langle \{SC_1, SC_2, \ldots, SC_n\}, \mathbb{T}_{src} \rangle$ (i.e., contract implementations and trusted entities), with an attack surface area $Area(\Sigma_{src}) = \sum_{i=1}^{n} der(SC_i)$. For validator-controlled implementations ($SC_i = \emptyset$), the attack surface consists solely of the trust set $\mathbb{T}_{src}$ (i.e., the validators). While the contractual surface area is zero, the system remains exposed to validator-level risks such as economic incentives, collusion, or key compromise.

3.6.2 Off-Chain Layer Security. The off-chain layer, which handles bridge communication, consists of notaries, light clients, or sidechains, each with unique trust assumptions as we discuss next.

Notary-Based Mechanisms. For notary-based bridges ($\mathbb{T}_{off}$ = notaries), security improves with more notaries, but decentralization is costly. If a sufficient fraction of notaries turn malicious, they can control the bridge for a duration $t > t^*$, enabling asset theft.

Light Client Mechanisms. For light-client bridges ($\mathbb{T}_{off} = \{LC, MP\}$), security depends on Merkle proofs (MP). If an attacker controls the light client, they can alter transaction verifications. The attack surface reduces to $Area(\Sigma_{off} \mid \mathbb{T}_{off} = \text{light client}) = der(\text{light client})$

Sidechain-Based Mechanisms. Sidechains use independent consensus rules (i.e., R_{src}). If R_{src} is compromised, an attacker can mint arbitrary tokens or modify the bridge state. If the sidechain guarantees R_{src} security, the attack surface vanishes $Area(\Sigma_{off} | \mathbb{T}_{off} = \text{sidechain}) = 0 \Leftrightarrow der(R_{src}) = 0$.

3.6.3 Destination Chain Layer Security. The destination chain's attack surface depends on its reliance on smart contracts (SC) and custodians $\Sigma_{dest} = \langle \tau_{dest}, \mathbb{T}_{dest} \rangle$, $\tau_{dest} = \{SC_1, SC_2, \ldots, SC_n\} \cup \{custodian\}$. Attack surface area is $Area(\Sigma_{dest}) = n + 1$, assuming a custodian is present.

3.6.4 Total Attack Surface of a Blockchain Bridge. The total bridge attack surface is $\Sigma_{\mathbb{B}_{1\leftrightarrow 2}} = \Sigma_{src} \cup \Sigma_{off} \cup \Sigma_{dest} \cup \Sigma_{other}$. Other attack vectors include updates to bridge protocols, governance failures, or rug-pulls. We validate our framework by classifying past major bridge exploits under this model.

Table 2: Categorized attack/disruption vectors across bridge layers. A dagger (†) marks vulnerabilities that only arise when a bridge architecture actually employs the relevant component (for example, an oracle).

Attack Vector	Source Chain	Off-Chain	Destination Chain
Contract Logic & Code Vulnerabilities			
V1: Reentrancy attacks	✓		✓
V2: Integer and arithmetic errors	✓		✓
V3: Access control and forged account flaws	✓		✓
V4: Race condition attacks	✓		✓
V5: Unsafe external call exploits	✓		✓
V6: Malicious event log manipulation	✓		✓
V7: Contract upgrade risks	✓		✓
Authentication & Authorization Failures			
V8: Fake burn/lock proofs	✓	✓	✓
V9: Malicious transaction modification	✓	✓	✓
V10: Light-client verification flaws	✓	✓	✓
V11: Oracle manipulation	✓†	✓	✓†
V12: Malicious custodian manipulation	✓†	✓	✓†
V13: Private key leakage or theft	✓†	✓†	✓†
Replay, Race, and Timing–Based Attacks			
V14: Timestamp manipulation	✓		✓
V15: Replay attacks		✓	✓
Consensus & Infrastructure Risks			
V16: Consensus failure (51% attack)	✓		✓
V17: Delayed finality exploitation	✓		✓
V18: Validator equivocation or misbehavior	✓†	✓	✓†
V19: Denial of Service attacks	✓	✓	✓
V20: Deep chain reorganization	✓		✓
V21: Unbounded withdrawal limits	✓		✓
V22: Rugpull		✓	✓†
Front-End and Off-Chain Manipulation			
V23: Front-end deception		✓	

4 BRIDGE DESIGN PATTERNS AND IMPLEMENTATION LANDSCAPE

To contextualize our formalism and threat model, we survey representative bridge implementations, analyzing their design across three key layers: source chain, off-chain coordination, and destination chain. Appendix Table 9 summarizes their implementation patterns, trust assumptions, functional types, and blockchain coverage. Full protocol descriptions appear in Appendix C and the ecosystem is described in Appendix D.

4.1 Static Analysis of Bridges

Table 3: Access control and code structure metrics of bridge smart contracts

Bridge Name	Local vars	Inheritances	Modifier Count	RoleBased	Standard Libs
Avalanche BridgeToken	2	1	0	Yes	1
Wormhole BridgeImplementation	58	2	1	No	3
Arbitrum L1GatewayRouter	9	5	2	No	0
Arbitrum L1ERC20Gateway	3	1	1	No	1
Arbitrum L1CustomGateway	5	2	2	No	1
Arbitrum L1WethGateway	0	1	0	No	2
Stargate Router	30	3	1	No	3
DeBridge DeBridgeGate	44	5	4	Yes	1
Across HubPool	50	5	3	No	3
Stargate Bridge	35	3	1	No	1
Allbridge Bridge	12	4	0	No	1
Nomad BridgeToken	6	4	0	No	0
Base L1StandardBridge	1	2	0	No	0
Optimism L1StandardBridge	0	2	0	Yes	0
Hyperliquid Bridge2	66	2	0	Yes	5
Meson BridgeV2	2	2	4	Yes	3

To assess the security robustness of bridge smart contracts, we conduct a *static analysis* across key dimensions drawn from our layered attack surface model. Static analysis focuses on examining the contract's code without executing it, allowing us to detect vulnerabilities in logic, structure, and access control that may compromise the security priors *bridge causality*, *consistency*, and *token peg integrity*. We analyze the bridges in terms of i) access control and code structure metrics (Table 3), ii) function visibility and variable usage (Table 4) and iii) call behavior and defensive programming constructs (Table 5). This analysis maps to several high-impact attack vectors from Table 2, notably: V1: reentrancy attacks, V3: access control and forged account flaws, V5: unsafe external call exploits, V7: contract upgrade and misconfiguration risks. We identify several notable patterns and anomalies that offer insights into the heterogeneous security postures and implementation philosophies across bridges.

Table 4: Lines of code (LOC), function visibility, and variable usage in bridge smart contracts

Bridge Name	LOC	Total lines	Public Funcs	External Funcs	Internal Funcs	Private Funcs	Global vars Declared
Avalanche BridgeToken	155	226	9	0	0	1	6
Wormhole BridgeImplementation	630	776	19	3	14	0	0
Arbitrum L1GatewayRouter	208	305	5	4	1	0	2
Arbitrum L1ERC20Gateway	112	161	5	2	0	0	6
Arbitrum L1CustomGateway	165	243	5	3	0	0	6
Arbitrum L1WethGateway	78	92	2	1	4	0	2
Stargate Router	280	323	0	17	4	0	5
DeBridge DeBridgeGate	905	1110	7	23	15	0	27
Across HubPool	610	1076	24	2	13	0	16
Stargate Bridge	249	310	2	14	4	0	9
Allbridge Bridge	215	320	1	11	2	0	5
Nomad BridgeToken	150	245	7	5	0	0	6
Base L1StandardBridge	260	321	8	8	6	0	2
Optimism L1StandardBridge	220	324	3	9	6	1	3
Hyperliquid Bridge2	570	840	17	15	2	14	18
Meson BridgeV2	167	210	8	8	0	2	4

Table 5: Analysis of external call behavior and defensive programming practices in bridge implementations

Bridge Name	Ext Funcs	Low-level	Untrusted	Reentry Guard	Require/Assert	Custom Errors	Checks/Fn
Avalanche BridgeToken	5	0	0	No	14	0	1.56
Wormhole BridgeImplementation	2	5	0	Yes	21	0	0.58
Arbitrum L1GatewayRouter	3	0	0	No	9	0	1
Arbitrum L1ERC20Gateway	2	1	0	Yes	3	0	0.43
Arbitrum L1CustomGateway	4	0	0	Yes	6	0	0.75
Arbitrum L1WethGateway	0	0	0	No	3	0	0.43
Stargate Router	15	0	0	No	8	0	0.38
DeBridge DeBridgeGate	18	3	3	Yes	26	20	0.87
Across HubPool	13	3	3	Yes	26	0	1
Stargate Bridge	13	0	0	No	8	0	0.5
Allbridge Bridge	5	0	0	No	11	0	0.9
Nomad BridgeToken	4	0	0	No	4	0	0.33
Base L1StandardBridge	4	0	0	No	4	0	0.25
Optimism L1StandardBridge	4	0	0	No	0	0	0
Hyperliquid Bridge2	7	0	0	Yes	19	0	1
Meson BridgeV2	7	1	1	No	7	0	0.58

One immediate observation in Table 5 is the inconsistent use of reentrancy protection mechanisms. Despite having multiple externally callable functions, contracts such as *Stargate Router* (15 external functions) and *Stargate Bridge* (13 external functions) do not implement any form of reentrancy guard. In contrast, similarly sized contracts like *DeBridgeGate* and *Across HubPool* use such guards appropriately. The absence of reentrancy protection in these high-exposure contracts reflects a reliance on architectural assumptions.

We also find a curious disconnect in Table 3 between the use of role-based access control and the deployment of Solidity modifiers. For example, *Hyperliquid Bridge2* and Avalanche *BridgeToken* both report role-based access ("RoleBased = Yes") but have zero modifiers. This suggests that access restrictions may be implemented inline via "require" statements rather than modularized through modifiers, leading to less auditable and reusable control logic. In contrast, *Nomad BridgeToken* and *Allbridge* show zero modifier usage and lack role-based protection, increasing susceptibility to unauthorized

access or logic misbehavior (V3). *Nomad*'s exploit history confirms this: a faulty initialization allowed any user to spoof legitimate senders.

A particularly unusual case in Table 5 is the *Optimism L1Standard Bridge* contract, which lacks any require or assert statements despite being externally callable and declaring role-based control. While this absence would typically raise concerns (since most bridges include basic sanity checks), it reflects a deliberate design choice. Optimism and *Base L1StandardBridge* rely on rollup-native architecture, where Ethereum layer-1 consensus enforces external validation. This structural safeguard reduces reliance on in-contract defensive coding against V3 and V5 vectors, but the lack of internal guards still poses risks if such contracts are reused outside this tightly scoped context.

In terms of architectural modularity, some contracts in Table 4 show a preference for deeply internalized logic structures. *Wormhole BridgeImplementation* and *Hyperliquid Bridge2* contain large numbers of internal or private functions (13-15 each), indicating encapsulated logic. While this may reflect thoughtful separation of concerns, it also complicates external audits and transparency.

We further observe disproportionate uses of access control infrastructure relative to contract size in Tables 3 and 4. *Meson BridgeV2*, with just 167 lines of code, defines four modifiers and a complete role-based access layer. In contrast, *Stargate Router* spans over 280 lines but employs one modifier. Such disparity shows divergent security philosophies among bridge developers, with some opting for granular control at all costs and others prioritizing operational simplicity.

A notable structural pattern in Table 4 appears in the allocation of global versus local variables. *DeBridgeGate* maintains a high number of global state variables (27), reflecting rich on-chain logic and persistent data tracking. Conversely, *Wormhole Bridge* holds 58 local variables but defines no global state, potentially due to reliance on proxy patterns or off-chain state.

Low-level calls (`call`, `delegatecall`, `staticcall`) introduce attack surface via vector V5. *Wormhole*, *DeBridge*, and *Across* use them multiple times, sometimes in the presence of untrusted inputs. While `call` can be necessary (e.g., for token forwarding), its misuse or failure to handle return values correctly has led to major incidents such as the *Poly Network* and *Qubit* [32] exploits. Bridges like *Avalanche BridgeToken*, *Nomad*, and *Allbridge* avoid low-level calls entirely, reducing exposure to unsafe execution paths.

Lastly, we compute the average number of checks per function (#Checks/#Functions) as a coarse proxy for defensive programming density. Contracts such as *Avalanche BridgeToken* (1.56) and *Hyperliquid Bridge2* (1.00) stand out as aggressively guarded, while *Optimism L1StandardBridge* (0.00) again reflects an absence of such measures. These findings indicate varying degrees of maturity in secure smart contract engineering practices.

Static analysis reveals that bridge contracts vary widely in their defensive quality. *DeBridgeGate* sets a high bar for smart contract hygiene, leveraging modifiers, role-based controls, and comprehensive assertions to guard its complex logic. *Wormhole* and *Across* balance complexity with modular defenses but remain susceptible to implementation flaws, as demonstrated in *Wormhole*'s historical signature validation failure. *Stargate*, *Nomad*, and *Allbridge* show critical gaps in protective constructs, leaving them exposed to known attack vectors. These are particularly concerning given their public deployment.

4.2 Transaction Analysis of Bridges

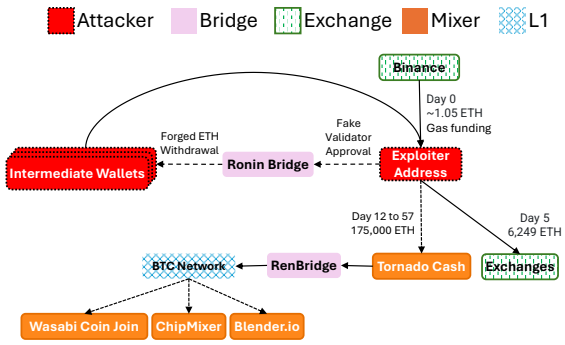

Figure 3: Attack flow of the *Ronin Bridge* exploit. The attacker gained control of 5 out of 9 Ronin Bridge validators and used them to sign and submit forged withdrawals. The stolen funds amounted to 173,600 ETH, along with an additional 25.5 million USDC that was exchanged for 8,564 ETH.

To better understand the on-chain behavior of bridge exploiters, we perform a transaction-level analysis focused on Ethereum addresses used by attackers, as Ethereum is commonly chosen for fund collection due to its exchange access and availability of mixing services. We obtain Ethereum transaction data by running a full node using Geth (https://github.com/ethereum/go-ethereum) and parsing all transactions via Ethereum-ETL (https://github.com/blockchain-etl/ethereum-etl). We visualize the attack patterns in transaction graphs to capture attacker behavior. The analysis includes both externally owned account transactions and internal transactions, covering pre-attack, attack, and post-attack phases, where the phase lengths span the full Ethereum history up to March 2025.

Due to space limitations, an example transaction subgraph is shown in Figure 3, and seven other notable attacks are visualized in Appendix F. In the figure, the *Ronin bridge* to Ethereum was victim to a validator key compromise in an attack now attributed to North Korea's Lazarus Group. *Ronin* used a 5-of-9 multisig validation for bridge withdrawals. Attackers compromised five private keys through social engineering. Once in control, they issued two fraudulent transactions, draining 173,600 ETH and 25.5M USDC (worth approximately $624 million at the time) from the *Ronin bridge* in a single stroke. This event, the largest DeFi hack ever, was essentially a failure of the bridge's trust model: the off-chain validators were assumed honest, but the minimal quorum and centralized key management (a single entity controlled 4 of 9 validators) made it easy for an attacker to breach. The breach went unnoticed for six days until a user discovered it.

Most exploiter addresses have little to no previous on-chain activity. In many cases, the wallets were newly created and funded with just a sufficient amount to cover gas fees. The average number of pre-attack transactions was 28, suggesting that even the more

"active" wallets were only lightly used. Some of these initial funds even originated from mixers, most notably Tornado Cash [33] in the *Wormhole* bridge hack, where the exploiter received ETH from Tornado Cash before launching the attack, an indication of deliberate obfuscation of the funding source. In other instances, such as the *Nomad* and *Horizon bridge* [34] hacks, attackers used unlabeled intermediate wallets, further complicating traceability, and as a strategy to hide the funding sources.

During the attack period, the behavior is highly focused. The attacker often interacts directly with the bridge contract via a few function calls, most of the time, only one forged withdrawal or proof verification. In attacks such as *Nomad*, we observe many small transactions from multiple addresses exploiting the same vulnerability after it became public, showing replication once the vulnerability was made public.

Internal transactions are important for creating the contract call sequences. The contract call traces highlight how the attack is carried out at the smart contract level, often using call stacks involving specific functions like *withdrawERC20For()* or *verifyProof()*. This helps verify the exploit strategies and confirms the role of compromises or logic flaws.

After the exploit, the stolen funds are moved rapidly to avoid detection. Some exploiters transferred stolen funds through multiple intermediate wallets before returning them to a central exploiter address, for instance, in the *Ronin Bridge* hack. Some others bridged funds to other blockchains such as Ethereum, Polygon, and Avalanche using bridges. Many proceeds are laundered through privacy tools like Tornado Cash or converted via decentralized exchanges (DEXs).

After initial laundering, several attacker addresses remain active. In our analysis, 6 out of 18 wallets continued transacting for more than one year after the exploit, indicating possible long-term usage or reactivation of compromised wallets.

Although certain early detection remains challenging, our analysis identifies behavioral clues that may serve as early warning signs of an upcoming exploit. A common pattern is the initial funding of a wallet with just enough ETH to cover gas fees, which is sourced from mixers. These wallets usually indicate little to no prior activity. Moreover, in order to verify system reactions or contract behavior, numerous low-value probing transactions are occasionally issued prior to the main exploit. These patterns could be leveraged to flag suspicious activity in near-real time.

5 SECURITY ANALYSIS OF BRIDGE ATTACKS

Our analysis reveals that bridge causality and consistency priors are violated in bridge attacks. Most attacks involve violations of the cross-chain causality prior. While Equation (4), the peg prior, is formally violated as a result of these attacks, the peg itself is not the target of the attacks. That is, attackers do not manipulate token prices or attempt to create price divergence across chains. Instead, the violations arise indirectly because tokens are minted or released without proper backing, breaking the assumption of value parity that the peg prior encodes. Due to space limitations, we describe the attacks in Appendix F, and show a meta-analysis in Table 8.

Attack Vector Analysis. Table 8 shows that V3 (access control) and V13 (key leakage) have been exploited 10 times each. The exploits reveal two broad categories of failures: (1) Off-chain trust failures and (2) On-chain validation failures. The off-chain trust failures encompass all incidents where the bridge's security relied on individuals or off-chain systems that were compromised, notably the multisig key compromises (*Ronin*, *Harmony*, *Multichain*, *Orbit*) and related cases. In *Ronin* and *Harmony*, the sources of failure were outside the blockchain: hackers penetrated the organizations controlling the validators and obtained the private keys needed to sign fake transactions. These attacks did not exploit a bug in code; they exploited insufficient decentralization and operational security. Essentially, the assumption that a small set of validators would remain honest was violated. When 5 of 9 *Ronin* validators and 2 of 5 *Harmony* signers turned malicious (via key theft), the bridge smart contracts on-chain duly obeyed the malicious signatures, an example of how improperly validated inputs led the system to execute unintended behavior. Similarly, *Multichain*'s collapse was an off-chain failure: the system's architecture secretly concentrated too much trust in one individual. These illustrate that trusted or trust-minimized bridges are vulnerable by design. They introduce new trust points (keys, signers, servers) that attackers target through phishing, insider collusion, malware and more. The on-chain validation failures, on the other hand, cover exploits like *Poly Network*, *Wormhole*, *Nomad*, *BSC Token Hub*, *Qubit*, where the bridge smart contracts or crypto verification routines on one of the chains had a flaw. In these cases, the attacker manipulated the code logic (e.g., forging a message that the contract erroneously accepted as valid). For *Poly*, the bug was an unchecked external call in a privileged contract; for *Wormhole*, a bypassed signature verification on Solana; for *Nomad*, an incorrect initialization setting that trusted everyone; for BSC, a flawed light-client proof verification.

Despite different mechanics, these represent software/security bugs in the bridge implementation. The failures occurred either on the source chain contract (for *Poly* on Ethereum) or the destination chain contract (for *Wormhole*, *Nomad*, *Qubit*) or the intermediate relay logic (as in *BSC*'s light client). Crucially, these attack vectors map to points of validation in the bridge architecture.

In several cases (*Wormhole*, *Nomad*, *BSC*), the violated core assumption was that the smart contract correctly validates the cross-chain proof. These were not fundamental cryptographic failures (algorithms such as *secp256k1* were sound), but errors in how the algorithms were applied. This suggests many bridge exploits are avoidable with more rigorous software engineering, such as comprehensive audits, formal verification of bridge contracts, and in-depth defense (e.g., requiring multiple independent checks of a proof).

Importantly, many of these incidents exhibit high damage-to-effort ratios. The impact of a successful exploit has often reached hundreds of millions of dollars, while the effort required, whether key theft, phishing, or exploiting a poorly audited contract, remains relatively low. These high $\mathrm{der}(V)$ values underscore why such attack vectors persist across different bridge designs and why they dominate the historical incident landscape.

RQ1: Are bridge attacks structurally preventable, or can they be mitigated by design? Yes, the evidence suggests that certain designs are far more robust. Notably, **no major exploits have occurred on fully trustless light-client bridges or rollup bridges**. For example, Cosmos's L1-to-L1 IBC channels have operated without

incident between dozens of chains since 2021 (a lossless reentrancy bug was discovered early [35]), and Ethereum's (young) rollup L1-to-L2 bridges have not been breached to date. Avalanche L1-to-L1 bridge to Ethereum continues functioning without any incidents. These systems benefit from minimal attack surface: there are no external signers to target, and the verification logic is often simpler or inherited from consensus (IBC uses tendermint-like light clients, which are well-vetted; rollups use post validity/fraud proofs checked by L1). This suggests that trustless designs inherently eliminate entire classes of attacks (notably, attack vector V3 of Table 2). These architectures inherently exclude common attack vectors like compromised keys or poorly controlled access mechanisms, offering significantly lower der(V) values.

By contrast, most attacks struck bridges that added a new layer of validation on top of the two blockchains. Each additional component (e.g., a multisig, an off-chain oracle network, or a novel smart contract) introduced complexity and potential weakness. Our analysis shows evidence that the more a bridge can lean on intrinsic blockchain security (native verification), the safer it will be. In contrast, trust-minimized L1-to-L1 bridges are often only as strong as their validator set, which could be much weaker than either chain's consensus. A quantified framework by LI.FI found in 2022 that many so-called trust-minimized bridges still effectively rely on a majority of a few validators, leaving them vulnerable to 51% attacks or key compromises [36]. Indeed, *Ronin* and *Harmony* hacks demonstrated that a <10-member validator set offers poor resilience. Increasing the set (e.g., Axelar has 70+ validators) improves security, but unless those validators are as robust as a full blockchain consensus (with hundreds of nodes and economic security), the bridge remains a tempting honeypot for hackers. Therefore, trust-minimized bridges offer better resilience than naive trusted models, but they still concentrate trust among a group.

Properly decentralized networks with bonding (to slash malicious validators) are harder to attack; for instance, a hack of the scale of *Ronin* on a larger bonded validator bridge has not occurred yet. However, Ethereum-to-Solana (L1-to-L1) *Wormhole*'s Feb 2022 case reminds us that even absent collusion, a bug in the validation code can be just as catastrophic. This blurs the line between *fundamental* and *implementation* issues: we argue that the complexity of blockchain bridge protocols is a fundamental weakness, as more complex logic yields a higher chance of errors.

Finally, the prevalence of hacks has spurred research into failure-resistant bridges. For example, designs like "circuit breaker" bridges propose that if an unusually large withdrawal is attempted, the bridge only allows it after a delay or community vote, to mitigate instant draining. Others suggest blending multiple validation mechanisms, e.g., requiring both an off-chain multi-sig and an on-chain light client to agree, which could guard against either one failing alone (at the cost of added complexity). From a benchmarking perspective, we argue that diversity in validation (multi-layer security) might drastically reduce the probability of a successful exploit, albeit with performance trade-offs. Moreover, as bridges adopt techniques like zero-knowledge (zk) proofs for validation (e.g., *zkBridge* frameworks), some traditional attack vectors might be closed (a zk proof can attest to a source chain state without any trusted signer at all). Such bridges could offer the holy grail: trustless interoperability between any two chains, with cryptographic guarantees and no reliance on human validators. Early prototypes (like *zkRelay* and light-client circuits) are promising, but they require heavy computation and are just entering practical deployment.

RQ2: Are bridge attacks detectable in real-time? A surprising inability of the blockchain ecosystem is the lack of a strong analytics foundation that could monitor bridges for potential attacks. In the *Ronin* hack, it took days (and a user complaint) to notice that $620M had vanished. An obvious insight here is that real-time auditing and alerts are essential, but it raises a more fundamental question: why have such systems not been developed, despite the substantial capital invested in decentralized finance?

We outline four reasons. First, many DeFi projects operate with lean teams focused primarily on product development and protocol maintenance, with their expertise originating from smart contract development. Allocating resources to build and maintain sophisticated monitoring systems often falls outside their immediate capabilities. Second, implementing real-time detection systems requires a scalable analytics infrastructure capable of handling large volumes of blockchain data. This necessitates expertise in data engineering and analytics, skills that may not be prevalent among smart contract developers. Third, analyzing blockchain data presents unique challenges, including model scalability and the accuracy of anomaly detection. Traditional data analytics methods are often ill-suited to address the decentralized and heterogeneous structure of blockchain networks. Moreover, only a limited number of research efforts have focused on foundational questions specific to blockchain, such as identifying influential addresses within transaction networks (e.g., Alphacore [37]; see [38] for a recent survey). Lastly, investing in detection systems does not directly generate revenue, making it less appealing for projects to allocate funds toward such initiatives. While white-hat hacking offers some incentives, it often involves high effort with uncertain rewards. Nonetheless, advancements in automated security testing, particularly fuzzing [39], are emerging as promising solutions. Fuzzing involves providing invalid, unexpected, or random data inputs to smart contracts to uncover vulnerabilities.

The anomaly detection effort must eventually fall on bridge managers. Some bridges (e.g., *Gravity*, *Chainlink CCIP* [40]) are now deploying independent, on-chain monitoring tools that can pause a bridge if suspicious activity is detected (for example, if an invariant is violated or if an unrecognized validator signature appears). In centralized finance, analogous systems flag large transfers for manual review; bridges could implement similar safeguards or circuit-breakers, but this reintroduces trust if not done in a decentralized way.

RQ3: What mechanisms exist to mitigate damage and minimize asset loss? There is no standard set of mechanisms for limiting damage once a bridge is under attack, and mitigation strategies remain ad hoc and uneven across protocols. *Gravity* and *Chainlink* are developing anomaly detection tools that can halt bridge operations in the face of an attack. In Celer, there is a buffer delay period during which transactions can be independently verified before a transfer is finalized. If inconsistencies are detected, the transfer can be halted, offering an added layer of fraud prevention. Post-factum, blockchain transparency does aid forensics; after these hacks, investigators (e.g., *ZachXBT* on Twitter) traced funds through addresses and

often identified suspects or recovered portions. But prevention and rapid response are clearly preferable to relying on clawbacks. Benchmark evaluations must incorporate attack detection latency as a metric; i.e., how quickly a breach is noticed and halted. Shorter detection times (as in BSC's case, hours) can significantly limit losses compared to delayed discovery (as in *Ronin*).

The bridge attacks of 2021–2025 underscore a core tension in blockchain interoperability: security vs. connectivity. Bridges expand what users can do (moving assets across chains), but they also expand the attack surface beyond any single blockchain. Our survey of exploits indicates that while many were due to low-hanging fruit bugs or a lack of operational security, there are also intrinsic risks whenever custody of funds shifts to a separate system.

RQ4: What is the long-term outlook for bridge security? What foundational steps are needed to design formally verifiable or provably secure bridge protocols? If truly trustless (or provably secure) bridge mechanisms can be standardized, we expect far fewer catastrophic failures. Until then, however, bridge designers must assume that attacks are a matter of when, not if, and design accordingly. This means employing rigorous audits, fail-safes, decentralized validation, and continuous monitoring as part of any cross-chain system. Past events suggest that increasing decentralization and using established, simple verification methods (light clients, chain consensus) correlate with higher security, whereas highly complex or centralized bridges have correlated with the largest failures.

On the other hand, the multisig key hacks raise the question of whether those bridges were destined to fail, i.e., was it an inevitability that eventually some signer's key would get compromised? One can argue that any valuable honeypot (whether a contract with billions of locked value or a key controlling billions) will, over time, face relentless attack until a weakness is found. Even with multiparty validators, if the set is small or the keys are not protected by robust hardware security modules and operational security, attackers have a focal point to strike. Thus, fundamental architecture choices (like using a 2-of-5 multisig with hot keys) can be seen as structural vulnerabilities, not in the theoretical sense, but in practical terms of presenting an irresistibly weak link (humans with keys) compared to the surrounding blockchain's security.

A lesson learned with pain is that trusted bridges must be avoided due to significant management risks [2]. Rollups offer a trustless solution but are limited to chains with compatible runtimes. New protocols like the *IBC* protocol of Cosmos and *Chainlink*'s Cross-Chain Interoperability Protocol aim to bridge heterogeneous chains. However, even those new-generation solutions introduce varying trust assumptions. IBC uses light clients when both chains natively support it. However, when adapted for non-IBC-compatible chains, additional trust assumptions are required. CCIP, on the other hand, relies on Chainlink's decentralized oracle network to facilitate cross-chain interactions, introducing a trust-minimized model where users depend on the honesty and reliability of the oracle nodes.

6 CONCLUSION

Blockchain bridges have become indispensable infrastructure for cross-chain interoperability, yet they remain among the most vulnerable components of the decentralized ecosystem. Our study presents the first comprehensive, data-driven systematization of bridge security, combining formal modeling, large-scale static analysis, and empirical transaction-level investigations. We show that the majority of bridge attacks violate core security priors, particularly causality and consistency, without breaching the value peg itself.

Our analysis reveals that most successful exploits stem from two dominant classes: off-chain trust failures and on-chain validation bugs. These vectors frequently exhibit high damage-to-effort ratios, which we formalize through the $\mathrm{der}(V)$ metric. We also find that architectural design plays a decisive role in resilience. Trustless models, such as light-client and rollup-native bridges, have so far withstood real-world adversarial conditions, while trusted and loosely trust-minimized bridges remain disproportionately vulnerable. Emerging defense mechanisms like circuit breakers, buffer delays, and hybrid validation schemes offer promise but are inconsistently implemented and lack formal standardization.

Finally, we identify critical gaps in real-time detection and mitigation. Despite billions in locked value, most bridges lack robust monitoring infrastructure or containment protocols, and responses to attacks are often delayed and improvised. Bridging this gap will require new research into on-chain anomaly detection, decentralized fail-safe mechanisms, and benchmarking frameworks that account for detection latency and systemic risk.

We hope this work provides a foundation for rethinking how cross-chain systems are built, evaluated, and secured. As bridges continue to evolve, their long-term viability will depend not only on throughput and composability but also on their ability to withstand failure in adversarial conditions. Bridging across blockchains should not mean bridging across trust assumptions.

REFERENCES

[1] "Bridge Volume - DefiLlama." https://defillama.com/bridges. Accessed: 2025-05-29.

[2] B. Kiepuszewski, "Rollups are the most secure bridges." https://archive.devcon.org/devcon-6/rollups-are-the-most-secure-bridges/, 2022. Devcon 6, Bogota, October 13, 2022.

[3] "Multichain." https://multichain.org/. Accessed: 2025-04-24.

[4] "Ronin bridge." https://docs.roninchain.com/apps/ronin-bridge, 2025. Accessed: 2025-05-21. Official documentation for the Ronin Bridge, facilitating ERC-20 token and NFT transfers between Ethereum and the Ronin chain.

[5] "Rainbow bridge." https://rainbowbridge.app/, 2025. Accessed: 2025-05-21. Official interface for the Rainbow Bridge, facilitating trustless and permissionless asset transfers between Ethereum, NEAR, and Aurora blockchains.

[6] "Polygon plasma bridge." https://docs.polygon.technology/, 2025. Accessed: 2025-05-21. Official documentation for the Polygon Plasma Bridge, enabling secure asset transfers between Ethereum and Polygon using Ethereum's Plasma scaling framework.

[7] A. Augusto, R. Belchior, J. Pfannschmidt, A. Vasconcelos, and M. Correia, "Xchainwatcher: Monitoring and identifying attacks in cross-chain bridges," *arXiv preprint arXiv:2410.02029*, 2024.

[8] J. Wu, K. Lin, D. Lin, B. Zhang, Z. Wu, and J. Su, "Safeguarding blockchain ecosystem: Understanding and detecting attack transactions on cross-chain bridges," *arXiv preprint arXiv:2410.14493*, 2024.

[9] N. Belenkov, V. Callens, A. Murashkin, K. Bak, M. Derka, J. Gorzny, and S.-S. Lee, "Sok: A review of cross-chain bridge hacks in 2023," *arXiv preprint arXiv:2501.03423*, 2025.

[10] N. Li, M. Qi, Z. Xu, X. Zhu, W. Zhou, S. Wen, and Y. Xiang, "Blockchain cross-chain bridge security: Challenges, solutions, and future outlook," *Distributed Ledger Technologies: Research and Practice*, vol. 4, no. 1, pp. 1–34, 2025.

[11] "Ethereum." https://ethereum.org/, 2025. Accessed: 2025-05-21. Official website of Ethereum, a decentralized platform supporting smart contracts and decentralized applications (dApps).

[12] C. Huang, T. Yan, and C. J. Tessone, "Seamlessly transferring assets through layer-0 bridges: An empirical analysis of stargate bridge's architecture and dynamics," in *Companion Proceedings of the ACM Web Conference 2024*, pp. 1776–1784, 2024.

[13] "Polygon." https://polygon.technology/, 2025. Accessed: 2025-05-21. Official website of Polygon, an Ethereum scaling platform offering Layer 2 solutions such as the PoS chain, zkEVM, and other modular frameworks for building interoperable blockchain networks.

[14] Binance, "Binance smart chain whitepaper," 2020.

[15] "Avalanche." https://avax.network/, 2025. Accessed: 2025-05-21. Official website of Avalanche, a high-performance Layer 1 blockchain platform developed by Ava Labs, featuring a unique three-chain architecture (X-Chain, C-Chain, and P-Chain) and the Avalanche Consensus Protocol for rapid transaction finality.

[16] "Arbitrum." https://arbitrum.io/, 2025. Accessed: 2025-05-21. Official website of Arbitrum, a Layer 2 scaling solution for Ethereum utilizing optimistic rollups to enhance transaction throughput and reduce fees while maintaining security.

[17] "Optimism." https://www.optimism.io/, 2025. Accessed: 2025-05-21. Official website of Optimism, a Layer 2 scaling solution for Ethereum utilizing optimistic rollups to enhance transaction throughput and reduce fees while maintaining security.

[18] S. Subramanian, A. Augusto, R. Belchior, A. Vasconcelos, and M. Correia, "Benchmarking blockchain bridge aggregators," in *2024 IEEE International Conference on Blockchain (Blockchain)*, pp. 37–45, IEEE, 2024.

[19] A. Augusto, A. Vasconcelos, M. Correia, and L. Zhang, "Xchaindatagen: A cross-chain dataset generation framework," *arXiv preprint arXiv:2503.13637*, 2025.

[20] A. Augusto, R. Belchior, M. Correia, A. Vasconcelos, L. Zhang, and T. Hardjono, "Sok: Security and privacy of blockchain interoperability," in *2024 IEEE Symposium on Security and Privacy (SP)*, pp. 3840–3865, IEEE, 2024.

[21] "Nomad." https://app.nomad.xyz/, 2025. Accessed: 2025-05-21. Official website of Nomad, a security-first cross-chain messaging protocol enabling secure and gas-efficient communication between blockchains.

[22] D. Lin, Z. Zheng, J. Wu, J. Yang, K. Lin, H. Xiao, B. Song, and Z. Zheng, "Track and trace: Automatically uncovering cross-chain transactions in the multi-blockchain ecosystems," *arXiv preprint arXiv:2504.01822*, 2025.

[23] "Bitcoin." https://bitcoin.org/, 2025. Accessed: 2025-05-21. Official website of Bitcoin, the first decentralized cryptocurrency and blockchain network.

[24] "Solana." https://solana.com/, 2025. Accessed: 2025-05-21. Official website of Solana, a high-performance blockchain platform supporting decentralized applications and smart contracts.

[25] "Orbs network." https://www.orbs.com/, 2025. Accessed: 2025-05-21. Official website of Orbs Network, a decentralized Layer-3 blockchain infrastructure designed to enhance the capabilities of existing smart contracts by providing decentralized backend services.

[26] "Xai games." https://xai.games/, 2025. Accessed: 2025-05-21. Official website of Xai Games, a Layer 3 blockchain platform built on Arbitrum, designed to bring AAA gaming experiences to Web3 by enabling seamless integration of blockchain technology into games.

[27] "zksync." https://zksync.io/, 2025. Accessed: 2025-05-21. Official website of zkSync, a Layer 2 Ethereum scaling solution utilizing zero-knowledge rollups to enhance scalability and reduce transaction costs while maintaining high security.

[28] "Wormhole." https://wormhole.com/. Accessed: 2025-04-24.

[29] "Loopring." https://loopring.org/, 2025. Accessed: 2025-05-21. Official website of Loopring, a Layer 2 zkRollup protocol on Ethereum that enables scalable, secure, and low-cost decentralized trading and payments.

[30] Connext, "Connext: The interoperability protocol for cross-chain applications." https://www.connext.network/, 2025. Accessed: 2025-06-03.

[31] Across, "Across: Cross-chain bridge for ethereum and layer 2 networks." https://across.to/, 2025. Accessed: 2025-06-03.

[32] "Qubit finance." https://qbt.fi/, 2025. Accessed: 2025-05-21. Official website of Qubit Finance, a decentralized money market platform on Binance Smart Chain (BSC) that connects lenders and borrowers efficiently and securely.

[33] "Tornado cash." https://tornado.cash/, 2025. Accessed: 2025-05-21. Official website of Tornado Cash, a decentralized, non-custodial privacy protocol on Ethereum that uses zero-knowledge proofs to enable anonymous transactions.

[34] "Horizon bridge." https://bridge.harmony.one/, 2025. Accessed: 2025-05-21. Official website of Horizon Bridge, a cross-chain bridge developed by Harmony that facilitates the transfer of crypto assets between Ethereum, Binance Smart Chain, and Harmony networks.

[35] B. Lindrea, "Cosmos patches 'critical' ibc protocol bug, saving $126m," 2024. Accessed: 2025-05-19.

[36] A. Chand and M. Murdock, "Li.fi: With bridges, trust is a spectrum," July 2022. LI.FI Knowledge Hub.

[37] F. Victor, C. G. Akcora, Y. R. Gel, and M. Kantarcioglu, "Alphacore: data depth based core decomposition," in *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pp. 1625–1633, 2021.

[38] P. Azad, C. Akcora, and A. Khan, "Machine learning for blockchain data analysis: Progress and opportunities," *Distributed Ledger Technologies: Research and Practice*, 2024.

[39] M. Rodler, D. Paaßen, W. Li, L. Bernhard, T. Holz, G. Karame, and L. Davi, "Ef/cf: High performance smart contract fuzzing for exploit generation," in *8th IEEE European Symposium on Security and Privacy, EuroS&P 2023, Delft, Netherlands, July 3-7, 2023*, pp. 449–471, IEEE, 2023.

[40] Chainlink Labs, "Chainlink cross-chain interoperability protocol (ccip)." https://chain.link/cross-chain, 2025. Accessed: 2025-04-24.

[41] P. McCorry, C. Buckland, B. Yee, and D. Song, "Sok: Validating bridges as a scaling solution for blockchains," *Cryptology ePrint Archive*, 2021.

[42] Han'guk T'ongsin Hakhoe, IEEE Communications Society, Denshi Jōhō Tsūshin Gakkai (Japan). Tsūshin Sosaieti, Institute of Electrical, and E. Engineers, *ICTC 2018 : the 9th International Conference on ICT Convergence : "ICT Convergence Powered by Smart Intelligence" : October 17-19, 2018, Maison Glad Jeju, Jeju Island, Korea.* 2018.

[43] S.-S. Lee, A. Murashkin, M. Derka, and J. Gorzny, "SoK: Not Quite Water Under the Bridge: Review of Cross-Chain Bridge Hacks," in *2023 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, pp. 1–14, IEEE, 10 2023.

[44] G. Wang, Q. Wang, and S. Chen, "Exploring Blockchains Interoperability: A Systematic Survey," *ACM Computing Surveys*, vol. 55, pp. 1–38, 12 2023.

[45] R. Belchior, A. Vasconcelos, S. Guerreiro, and M. Correia, "A Survey on Blockchain Interoperability: Past, Present, and Future Trends," 11 2022.

[46] A. Zamyatin, M. Al-Bassam, D. Zindros, E. Kokoris-Kogias, P. Moreno-Sanchez, A. Kiayias, and W. J. Knottenbelt, "SoK: Communication Across Distributed Ledgers," tech. rep., 2019.

[47] I. A. Qasse, M. A. Talib, and Q. Nasir, "Inter blockchain communication: A survey," in *ACM International Conference Proceeding Series*, Association for Computing Machinery, 10 2019.

[48] S. K. Niranjan, REVA University, Institute of Electrical, E. E. B. Section, Institute of Electrical, and E. Engineers, *Proceedings of the International Conference on Smart Technologies in Computing, Electrical and Electronics (ICSTCEE 2020) : October 9-10, 2020, Virtual Conference.* IEEE, 2020.

[49] S. Schulte, M. Sigwart, P. Frauenthaler, and M. Borkowski, "Towards Blockchain Interoperability," in *Business Process Management: Blockchain and Central and Eastern Europe Forum: BPM 2019 Blockchain and CEE Forum, Vienna, Austria, September 1–6, 2019, Proceedings 17*, pp. 3–10, 2019.

[50] T. Hardjono, A. Lipton, and A. Pentland, "Toward an Interoperability Architecture for Blockchain Autonomous Systems," *IEEE Transactions on Engineering Management*, vol. 67, pp. 1298–1309, 10 2020.

[51] P. Lafourcade and M. Lombard-Platet, "About Blockchain Interoperability," tech. rep., Université Clermont Auvergne, 2020.

[52] L. Duan, Y. Sun, W. Ni, S. Member, W. Ding, J. Liu, and W. Wang, "Attacks Against Cross-Chain Systems and Defense Approaches: A Contemporary Survey," *IEEE*, 2023.

[53] M. Borkowski, D. Mcdonald, C. Ritzer, and S. Schulte, "Towards Atomic Cross-Chain Token Transfers: State of the Art and Open Questions within TAST," tech. rep., TU Wien, 2018.

[54] G. Caldarelli, "Wrapping trust for interoperability: A preliminary study of wrapped tokens," 10 2022.

[55] M. Borkowski, P. Frauenthaler, M. Sigwart, T. Hukkinen, H. Hladk´y, and S. Schulte, "Cross-Blockchain Technologies: Review, State of the Art, and Outlook," tech. rep., TU Wien, 2019.

[56] M. Herlihy, B. Liskov, and L. Shrira, "Cross-Chain Deals and Adversarial Commerce," in *Proceedings of the VLDB Endowment*, vol. 13, pp. 100–113, 2019.

[57] G. Malavolta, P. Moreno-Sanchez, C. Schneidewind, A. Kate, and M. Maffei, "Anonymous Multi-Hop Locks for Blockchain Scalability and Interoperability," in *Network and Distributed System Security Symposium (NDSS)*, 2019.

[58] S. A. K. Thyagarajan, G. Malavolta, and P. Moreno-Sanchez, "Universal Atomic Swaps: Secure Exchange of Coins Across All Blockchains," in *2022 IEEE Symposium on Security and Privacy (SP)*, pp. 1299–1316, 2022.

[59] B. Bünz, L. Kiffer, L. Luu, and M. Zamani, "FlyClient: Super-Light Clients for Cryptocurrencies," in *2020 IEEE Symposium on Security and Privacy (SP)*, pp. 928–946, 2020.

[60] A. Kiayias, A. Miller, and D. Zindros, "Non-Interactive Proofs of Proof-of-Work," in *Financial Cryptography and Data Security (FC)*, vol. 12059 of *LNCS*, pp. 505–522, Springer, 2020.

[61] G. Scaffino, L. Aumayr, M. Bastankhah, Z. Avarikioti, and M. Maffei, "Alba: The Dawn of Scalable Bridges for Blockchains," in *Network and Distributed System Security Symposium (NDSS)*, 2025.

[62] T. Xie, J. Zhang, Z. Cheng, F. Zhang, Y. Zhang, Y. Jia, D. Boneh, and D. Song, "zkBridge: Trustless Cross-Chain Bridges Made Practical," in *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pp. 3003–3017, ACM, 2022.

[63] A. Zamyatin, D. Harz, J. Lind, P. Panayiotou, A. Gervais, and W. Knottenbelt, "XCLAIM: Trustless, Interoperable, Cryptocurrency-Backed Assets," in *2019 IEEE Symposium on Security and Privacy (SP)*, pp. 193–210, 2019.

[64] Z. Liao, Y. Nan, H. Liang, S. Hao, J. Zhai, J. Wu, and Z. Zheng, "SmartAxe: Detecting Cross-Chain Vulnerabilities in Bridge Smart Contracts via Fine-Grained Static Analysis," in *Proceedings of the 2024 ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE)*, 2024.

[65] "Near protocol." https://near.org/, 2025. Accessed: 2025-05-21. Official website of NEAR Protocol, a scalable, developer-friendly Layer 1 blockchain platform utilizing sharding (Nightshade) and a unique consensus mechanism (Doomslug)

to support decentralized applications and smart contracts.
[66] "Aptos." https://aptosfoundation.org/, 2025. Accessed: 2025-05-21. Official website of Aptos, a high-performance Layer 1 blockchain developed by former Meta engineers. Aptos leverages the Move programming language and the Block-STM parallel execution engine to achieve high throughput and low latency, aiming to bring mainstream adoption to Web3.
[67] "Sui." https://sui.io/, 2025. Accessed: 2025-05-21. Official website of Sui, a high-performance Layer 1 blockchain designed for scalability and low latency. Sui utilizes the Move programming language and features an object-centric data model, enabling parallel transaction execution and instant settlement.
[68] "Cosmwasm." https://cosmwasm.com/, 2025. Accessed: 2025-05-21. Official website of CosmWasm, a WebAssembly-based smart contract platform designed for the Cosmos ecosystem, enabling secure, interoperable, and high-performance decentralized applications across multiple blockchains.
[69] "Base." https://base.org/, 2025. Accessed: 2025-05-21. Official website of Base, a secure, low-cost, builder-friendly Ethereum Layer 2 (L2) blockchain incubated by Coinbase. Base is designed to bring the next billion users onchain by offering fast transactions and low fees. It is built on the Optimism Superchain and aims to progressively decentralize over time.
[70] "Celo." https://celo.org/, 2025. Accessed: 2025-05-21. Official website of Celo, a mobile-first, carbon-negative blockchain platform that transitioned from an independent Layer 1 to an Ethereum Layer 2 solution. Celo focuses on real-world applications, offering low-cost transactions and supporting stablecoins like cUSD, cEUR, and cREAL.
[71] "Litecoin." https://litecoin.org/, 2025. Accessed: 2025-05-21. Official website of Litecoin, a peer-to-peer cryptocurrency enabling instant, near-zero cost payments to anyone in the world. Litecoin is an open-source, global payment network that is fully decentralized without any central authorities.
[72] "Blocknet." https://blocknet.co/, 2025. Accessed: 2025-05-21. Official website of Blocknet, a decentralized interoperability protocol that enables communication, exchange, and service delivery between different blockchains through a cross-chain architecture.
[73] "Colossusxt." https://www.colossusxt.io/, 2025. Accessed: 2025-05-21. Official website of ColossusXT, a community-oriented, energy-efficient cryptocurrency focused on decentralization, privacy, and real-world implementation.
[74] "Terra blockchain." https://www.terra.money/, 2025. Accessed: 2025-05-21. Official website of Terra, an open-source blockchain protocol that supports algorithmic stablecoins and decentralized applications. Terra experienced a significant collapse in May 2022, leading to the devaluation of its native tokens, LUNA and UST, and the subsequent launch of a new blockchain known as Terra 2.0.
[75] "The complete platform for onchain finance - wormhole." https://wormhole.com/institutions, 2025. Accessed: 2025-06-02.
[76] "Bridges | flow developer portal." https://developers.flow.com/ecosystem/bridges, 2025. Accessed: 2025-06-02.
[77] "Cardano." https://cardano.org/, 2025. Accessed: 2025-05-21. Official website of Cardano, a decentralized proof-of-stake blockchain platform built on peer-reviewed research, designed for secure and scalable smart contracts and decentralized applications.
[78] Cosmos Developer Portal, "What is IBC?." https://tutorials.cosmos.network/academy/3-ibc/1-what-is-ibc.html, 2024. Accessed: 2025-04-24.
[79] "Ibc/tao - clients - developer portal." https://tutorials.cosmos.network/academy/3-ibc/4-clients.html, 2025. Accessed: 2025-06-02.
[80] "Trustless interoperability between rollups: Landscape, constructions, and challenges." https://medium.com/1kxnetwork/trustless-interoperability-between-rollups-landscape-constructions-and-challenges-8ff195ea92cc, 2024. Accessed: 2025-06-02.
[81] L. T. Thibault, T. Sarry, and A. S. Hafid, "Blockchain scaling using rollups: A comprehensive survey," *IEEE Access*, vol. 10, pp. 93039–93054, 2022.
[82] "Axelar network." https://axelar.network/. Accessed: 2025-04-24.
[83] LayerZero Foundation, "Layerzero protocol." https://docs.layerzero.network/v2/developers/solana/getting-started, 2025. Accessed: 2025-04-24.
[84] "Celer cbridge." https://cbridge.celer.network/. Accessed: 2025-04-24.
[85] "Bnb chain." https://www.bnbchain.org/, 2025. Accessed: 2025-05-21. Official website of BNB Chain, a decentralized blockchain ecosystem supporting smart contracts and decentralized applications (dApps).
[86] "Bridging the gap: Ethereum-binance smart chain bridge (eth-bep)." https://www.binance.com/en/square/post/891550, 2023. Accessed: 2025-06-02.
[87] "Osmosis." https://osmosis.zone/, 2025. Accessed: 2025-05-21. Official website of Osmosis, a decentralized exchange (DEX) and Layer 1 blockchain in the Cosmos ecosystem, enabling cross-chain asset swaps and liquidity provision through the Inter-Blockchain Communication (IBC) protocol.
[88] "Cronos." https://cronos.org/, 2025. Accessed: 2025-05-21. Official website of Cronos, an open-source, EVM-compatible Layer 1 blockchain developed by Crypto.com. Built on the Cosmos SDK and powered by Ethermint, Cronos supports the Inter-Blockchain Communication (IBC) protocol, enabling interoperability with Ethereum and Cosmos ecosystems. It aims to scale decentralized applications (dApps) in DeFi, NFTs, and the Metaverse, offering low transaction fees and fast finality.
[89] "Gravity bridge." https://www.gravitybridge.net/, 2025. Accessed: 2025-05-21. Official website of Gravity Bridge, a decentralized, neutral Cosmos SDK-based blockchain facilitating trustless interoperability between Ethereum and Cosmos ecosystems through the Inter-Blockchain Communication (IBC) protocol.
[90] "Exploring cross-chain solutions: How to bridge into the cosmos ecosystem." https://www.axelar.network/blog/cosmos-bridge-explained, 2023. Accessed: 2025-06-02.
[91] "Fantom." https://fantom.foundation/, 2025. Accessed: 2025-05-21. Official website of Fantom, a high-performance, scalable, and secure smart-contract platform designed for decentralized applications (dApps) and digital assets.
[92] deBridge Foundation, "debridge finance." https://debridge.finance, 2025. Accessed: 2025-04-24.
[93] "Polkadot." https://polkadot.com/, 2025. Accessed: 2025-05-21. Official website of Polkadot, a scalable, interoperable, and secure multi-chain blockchain platform designed to connect various blockchains into a unified network.
[94] Polkadot Documentation, "Introduction to xcm (cross-consensus messaging)." https://docs.polkadot.network/develop/interoperability/intro-to-xcm/, 2025. Accessed: 2025-04-24.

APPENDIX

A RELATED WORK ON SCALABILITY, INTEROPERABILITY AND BRIDGE DESIGNS

Due to space constraints in the main text, this appendix outlines three additional research areas relevant to our work. We will start this section by analyzing scalability and interoperability studies.

A.1 Blockchain Scalability and Off-Chain Protocols

Recent bridge proposals aim to shift trust off-chain and reduce exposure by using succinct proofs, validator diversity, or formal verification. The literature reflects an active effort to reconcile decentralization, trust minimization, and performance in cross-chain systems. For example, McCorry et al. [41] and Kim et al. [42] explore off-chain protocols and categorize scalability solutions, respectively. Blockchain interoperability, the ability for different blockchain systems to interact, is a core focus of Lee et al. [43], Wang [44], Belchior et al. [45], Zamyatin et al. [46], Qasse et al. [47], Monika et al. [48], Schulte et al. [49], Hardjono et al. [50], Lafourcade et al. [51], and Duan et al. [52]. These studies highlight the challenges in ensuring ACID properties across diverse blockchains [44], the limitations due to blockchain isolation [47], and the lack of interaction between different ledgers and legacy systems [48]. Methods vary: Lee et al. [43] focus on bridges, Zamyatin et al. [46] on communication protocols, Schulte et al. [49] suggest transitioning from closed to open systems, while Hardjono et al. [50] propose standardized architectures. Lafourcade et al. [51] advocate for a unified blockchain.

A.2 Security and Privacy Concerns in Blockchain Interoperability

Security risks inherent to cross-chain protocols are discussed in Lee et al. [43], Zamyatin et al. [46], and Duan et al. [52]. Lee et al. examine bridge-specific vulnerabilities, Zamyatin et al. propose a trust-evaluation framework, and Duan et al. categorize systemic attack vectors. Borkowski et al. [53] provide a comprehensive review of cross-blockchain technologies through the TAST research project. Caldarelli [54] introduces wrapped tokens for interoperability, though with reintroduced trust assumptions. Borkowski et al. [55] explore atomic cross-chain transfers as a means to link otherwise isolated systems.

A.3 Protocol Design, Relays, and Bridges

Pioneering work by Herlihy et al. [56] introduced the notion of cross-chain deals, formulating atomic cross-chain transactions under adversarial conditions. Their protocol ensures that mutually distrusting parties exchanging assets across separate blockchains either all succeed or all abort, without a trusted intermediary. Atomic cross-chain swaps and hashed timelock contracts emerged from this foundation [56], with further developments such as Anonymous Multi-Hop Locks [57] and Universal Atomic Swaps [58] generalizing these mechanisms for broader interoperability.

Relay and light-client protocols enable cross-chain state verification. BTC Relay provided a working, though costly, Ethereum-based verifier for Bitcoin transactions. Efficiency enhancements such as FlyClient [59] and NiPoPoW [60, 61] offer logarithmic communication overhead and succinct proofs. *zkBridge* demonstrates state transition verification via *zk-SNARKs* [62], showcasing low-cost, trust-minimized bridging across Ethereum and Cosmos.

Asset-based bridges also remain prominent. XCLAIM [63] introduced trustless, collateral-backed cross-chain assets. While projects like *Wormhole* and *PolyNetwork* use validator-based event notaries, they pose risks if quorum assumptions fail. These are complemented by formal frameworks such as *zkRelay* and *ETH Relay*, and conceptual models like Bitcontracts for Bitcoin–Ethereum smart contract interoperability.

Security of these bridges is increasingly scrutinized. SmartAxe [64] detects Cross-Chain Vulnerabilities through fine-grained static analysis. Attacks like the $320M *Wormhole* and $610M *PolyNetwork* incidents illustrate the cost of flawed validation and inadequate trust assumptions. Alba [61] proposes *Pay2Chain* bridges that combine off-chain payment guarantees with cryptographic proofs to reduce bridge attack surfaces.

B PROOF OF THE FAILURE THEOREM

Theorem 2. *If a bridge experiences an attack or failure, then Equation* (4) *is always violated.*

Proof. Equation (4) formalizes token parity across two blockchains b_1 and b_2, requiring that the locked value on b_1 matches the released or minted value on b_2:

$$v_1 \cdot price(\theta_1, t) \equiv v_2 \cdot price(\theta_2, t), \quad \forall t.$$

Suppose a bridge undergoes an attack or failure. By Definitions 3.1 and 3.2, this implies that at least one of Equations (5), (6), or (7) is violated.

Case 1: Violation of causality ((5)*).* If a token transfer occurs on b_2 without a corresponding event on b_1, or vice versa, then either tokens are created without backing ($v_2 > v_1$), or locked value is unclaimed ($v_1 > v_2$). In both cases, the equivalence in (4) is broken.

Case 2: Violation of consistency ((6) *or* (7)*).* If a user holds tokens on b_2 while also being able to access funds on b_1, or if the one-to-one correspondence between locked and minted tokens is violated, then value duplication or loss occurs. Again, this leads to $v_1 \cdot price(\theta_1, t) \not\equiv v_2 \cdot price(\theta_2, t)$.

In all such cases, the bridge no longer preserves token parity, hence Equation (4) is violated.

□

C CROSS-CHAIN BRIDGE SOLUTIONS

We will discuss several popular cross-chain bridge protocols, cross-referencing them with their formalization. Specifically, we examine i) the implementation choices made by each bridge at different layers, ii) the trust assumptions underlying each bridge, iii) fee and time considerations, including comments on security where relevant, iv) the specific blockchains that each bridge connects. Table 9 provides an overview of cross-chain bridge implementations and architectural components.

Table 6: Notations for Bridge Mechanisms

Notation	Description
b	A blockchain.
N_p	Set of peer-to-peer network nodes in a blockchain.
N_a	Set of address nodes in a blockchain.
$\mathbb{H}$	Chronological history of transactions in a blockchain.
R	Consensus rules governing transaction creation in a blockchain.
θ	A token, with state defined by its transaction history $\mathbb{H}_\theta$.
t	A point in time.
tx	A transaction, defined as $tx = (\theta, v, a_1, a_2, t_{tx})$, transferring value v of token θ from address a_1 to a_2 at time t_{tx}.
a	A user address
b_1, b_2	Source and destination blockchains in a bridge.
$\mathcal{I}$	Implementation mechanism of a bridge.
$\mathbb{B}_{1\leftrightarrow 2}$	A blockchain bridge between b_1 and b_2, defined as $\mathbb{B}_{1\leftrightarrow 2} = \{b_1, b_2, \mathcal{I}\}$.
χt	The state of a cross-chain transaction at time t.
TS	A node trust set
c	A smart contract address
v	Quantity of a token θ.
$a \mapsto v$	An address a holds value v of a token.
F_{forward}	Bridge fee for forward transfer, defined as $F_{\text{forward}} = f_1 + f_2 + f^*$.
F_{reverse}	Bridge fee for reverse transfer.
f_1, f_2	Transaction processing fees on b_1 and b_2, respectively.
f^*	Bridge operation fee.
d	Duration associated with a blockchain operation
$\mathcal{T}_{\text{off}}$	Off-chain Implementation
d_{b_x}	Block confirmation durations for b_x.
d^*	Time for off-chain mechanism to process and signal a transfer.
$\mathbb{T}$	Bridge trust set.
$\mathbb{T}_{src}, \mathbb{T}_{dest}$	Trust assumptions related to the source and destination blockchains.
$\mathbb{T}_{off}$	Trust assumptions in off-chain mechanisms.
L	Light client verifier.
M	Merkle proof.
N	Notary validator set.
F	Total bridge fee for a transaction
D	Delay introduced in a bridge due to proof.
Q	Quantity of token being referenced.
MP	Massage propagation
V	Attack vector
$\mathbb{E}(C)$	Expected loss or cost.
$\mathbb{E}(L)$	Expected value at risk in a *CCB*.
SCs	Set of smart contracts.

C.1 Arbitrum Bridge

Arbitrum is not a bridge itself, but a layer two rollup solution designed to prepare large batches of transactions off-chain, significantly reducing gas costs and scaling Ethereum. However, Arbitrum also includes a native token bridge, which lets users move assets between Ethereum (L1) and Arbitrum (L2). This is commonly referred to as the Arbitrum Bridge, and it's used to deposit and withdraw tokens like ETH or ERC-20s.

As an optimistic rollup, Arbitrum assumes transaction validity by default and relies on economic incentives to detect and correct fraud. A centralized sequencer currently assembles and orders transaction batches, producing a new Arbitrum block for each Ethereum block. These transactions are compressed and posted as calldata (read-only, byte-addressable region) on Ethereum.

A set of smart contracts deployed on Ethereum manages the core protocol logic, including the rollup mechanism, cross-chain messaging, fraud proofs, bridging, and third-party gateway interactions. The trust set is defined as $\tau = \{\text{SCs}\}$, representing these deployed smart contracts, which collectively enforce the security and correctness of the rollup.

Arbitrum's bridge is structured around a gateway bridge contract that delegates to more specific contracts (e.g., ERC20, WETH, or custom contracts). These specialized contracts are responsible for minting the corresponding tokens on layer 2. This modular design enhances security and extensibility by isolating token-specific logic across separate contracts.

To guard against fraud, Arbitrum implements a challenge-response protocol during a designated challenge period. Off-chain participants can dispute the validity of a rollup block through an interactive verification process often described as the "grasshopper game", where the correct and fraudulent segments of execution are iteratively narrowed down. If fraud is detected, the protocol rolls back to the last valid state and removes the invalid block. A successful challenger receives a financial reward, typically derived from the stake of the fraudulent party. Conversely, false challengers lose their stake, discouraging frivolous disputes.

Finality on Arbitrum is delayed by the challenge period, which affects asset bridging. Users must wait until this period ends before safely withdrawing assets to Ethereum. Although this delay is inherent to optimistic security models, it is a practical consideration in bridge comparisons.

C.2 Wormhole protocol

Wormhole defines itself as a cross-chain message-passing protocol that enables communication between heterogeneous blockchains. Bridging protocols are built on top of *Wormhole* to facilitate token and data transfers across chains.

On-chain, *Wormhole* uses a set of smart contracts deployed to both the source and destination chains. The emitter contract on the source chain initiates a transfer by invoking the *'publishMessage'* method on the *Wormhole* core contract. These core contracts are responsible for emitting and verifying cross-chain messages, forming the trust set $\tau = \{\text{SCs}\}$ on both ends. The user relies on the correct and secure execution of these smart contracts for the protocol's integrity.

The off-chain implementation $\mathcal{T}_{off}$ relies heavily on a guardian network, composed of 19 independent nodes operated by various institutional entities. These guardians observe on-chain activity, verify messages, and collectively sign a Verifiable Action Approval (VAA). A quorum of at least 13 of 19 guardians must sign the VAA for it to be considered valid. This introduces an additional trust assumption, so the full trust set includes off-chain actors, i.e., $\tau \neq \{\}$. In practice, this makes *Wormhole* a trusted bridge, as its security hinges on the honest majority of the guardian set.

Once the VAA is generated, a relayer (which may be a user or a third party) delivers the signed message to the destination chain. There, the *Wormhole* core contract verifies the guardian signatures and executes the associated instructions via the target smart contract. As with the source chain, the destination chain also has a trust set $\tau = \{\text{SCs}\}$.

Fee and time considerations in *Wormhole* are variable. Since the relayer role is permissionless, users or third parties may bear the cost of relaying messages across chains, and some applications built on *Wormhole* subsidize or automate these steps. Finality and latency depend on the source chain's block confirmation time, guardian network processing, and delivery to the destination chain. As such, it does not provide instant finality but is generally faster than optimistic rollups that require challenge periods.

Wormhole connects a broad range of blockchains, including Solana, Algorand, Near [65], Aptos [66], Sui [67], and CosmWasm chains [68], as well as EVM-compatible blockchains such as Ethereum, Arbitrum, Avalanche, Polygon, Base [69], and Optimism. This makes it one of the most widely integrated cross-chain messaging protocols.

C.3 Celer *cBridge*

Celer *cBridge* offers two models for cross-chain asset transfers: a liquidity-pool-based model and an OpenCanonical token bridging model. In the pool-based model, liquidity providers lock assets into pools on both the source and destination chains, allowing users to swap assets between chains directly. In the OpenCanonical model, token transfers follow a mint-and-burn approach. On the source chain, tokens are locked in a smart contract called *TokenVault*, initiating the bridging process. The trust set on the source chain is $\tau = \{\text{SCs}\}$, as users must rely on the correct execution of these smart contracts.

The off-chain implementation $\mathcal{T}_{\text{off}}$ involves the State Guardian Network (SGN), a Cosmos-based proof-of-stake blockchain built on Tendermint. SGN acts as a validator and orchestrator of the bridge, confirming transactions off-chain. Since SGN is a third blockchain mediating between the source and destination chains, *cBridge* operates as a sidechain-based bridge. The system's trust assumptions, therefore, include the consensus protocol of SGN, denoted $\mathfrak{R}_{\text{SC}}$, as well as the live correctness of validators (LC), message propagation (MP), and the honest majority of SGN nodes (N). The resulting trust set is $\tau = (\{LC, MP\} \cap \{N\}) \cup \mathfrak{R}_{\text{SC}}$, characterizing *cBridge* as a trust-minimized bridge relative to fully trusted notary-based models.

After validation by SGN, the transaction is relayed to the *PeggedToken* contract on the destination chain, which mints pegged tokens to complete the transfer. The destination chain's trust set is again $\tau = \{\text{SCs}\}$, as users rely on the correct behavior of the deployed smart contracts.

An additional layer of optional security is available through an optimistic-rollup-style model. In this mode, before SGN finalizes a transfer, there is a buffer delay period during which transactions can be independently verified. If inconsistencies are detected, the transfer can be halted, offering an added layer of fraud prevention.

Fee and time considerations for *cBridge* vary by model and configuration. Both *xAsset* (mint-and-burn) and *xLiquidity* (pool-based) bridges impose a base fee and a protocol fee. The base fee includes the destination chain's gas costs, while the protocol fee, which ranges from 0 percent to 2 percent, depends on the source and destination chains and the transfer amount. This fee compensates SGN validators and stakers for securing the protocol.

cBridge currently supports a wide range of chains (41 at the time of this writing), including Ethereum, Arbitrum, Avalanche, BNB Chain, Celo [70], Polygon, and Fantom, among others.

C.4 Avalanche Bridge

The *Avalanche Bridge* is designed to enable fast, secure, and low-cost asset transfers between Avalanche and external networks such as Ethereum and Bitcoin. On the source chain, a smart contract is invoked to initiate the bridging process, forming a trust set $\tau = \{\text{SCs}\}$, where users rely on the correct execution of the deployed smart contracts.

The off-chain implementation $\mathcal{T}_{\text{off}}$ of the *Avalanche Bridge* is distinctive and combines a committee-based relayer design with hardware-based verification. A group of eight relayers, known as Wardens, monitor on-chain events and submit signed messages to a secure enclave powered by Intel SGX. This enclave, running verified and attested code, acts as a trusted execution environment that verifies the correctness of submitted data. Once at least six out of eight Wardens agree on the same transaction, the SGX enclave signs off on the transfer. This architecture introduces off-chain trust assumptions, and the trust set satisfies $\tau \neq \{\}$, making *Avalanche Bridge* a trusted bridge. Although the Warden set is geographically and institutionally distributed, the need for a quorum and reliance on Intel SGX introduces centralization risks in the event of collusion or compromise.

On the destination chain, the implementation $\mathcal{T}_{\text{dest}}$ involves a validator-controlled process. After off-chain validation, a smart contract is called to mint the bridged tokens. While this minting process is smart contract-based, finality is dependent on the decentralized consensus of the underlying blockchain (e.g., Avalanche C-Chain), where validators validate the resulting state. As this process is fully decentralized, the trust set at this stage can be written as $\tau = \{\}$.

Fee and time considerations vary depending on the asset and direction of transfer. Transfers from Ethereum to Avalanche typically take around 15–20 minutes, while Bitcoin transfers may take up to 1 hour due to Bitcoin's block time. On the Avalanche side, confirmations complete within seconds. ERC-20 tokens sent from Ethereum to Avalanche incur a 0.025 percent fee (minimum USD 3, maximum USD 250), while USDC is charged at a reduced rate of 0.02 percent (minimum USD 3, maximum USD 250). Bridging WBTC or WETH from Ethereum to Avalanche incurs a flat fee of USD 3. Transfers in the opposite direction, from Avalanche to

Ethereum, are charged a 0.1 percent fee (minimum USD 12, maximum USD 1000). WBTC and WETH offboarding carries a flat USD 20 fee. Native Bitcoin transfers to Avalanche incur a minimum fee of approximately USD 3 in BTC and result in the minting of BTC.b tokens. Returning BTC.b from Avalanche to the Bitcoin network incurs a fee of approximately USD 20 plus Bitcoin miner fees, with a minimum transfer threshold in place to ensure the transaction is economically viable.

The *Avalanche Bridge* currently supports bi-directional cross-chain transfers between Ethereum or Bitcoin and the Avalanche C-Chain. ERC-20 tokens bridged from Ethereum are wrapped and appear with a `.e` suffix (e.g., `USDC.e`, `WBTC.e`), while Bitcoin is represented as `BTC.b`. One exception is USDC, which now uses Circle's Cross-Chain Transfer Protocol (CCTP) and is natively burned and minted on each side, removing the `.e` suffix. Avalanche-native assets cannot be bridged to other chains using the Avalanche Bridge.

C.5 Multichain

Multichain, formerly known as Anyswap, presents itself as infrastructure for arbitrary cross-chain interactions. On the source chain, the bridge locks the user's assets in a smart contract that operates under a Secure Multi-Party Computation (SMPC) scheme. This smart contract is referred to by *Multichain* as a Decentralized Management Account, which securely holds assets during the bridging process. The source chain trust set is $\tau = \{\text{SCs}\}$, as users must trust the correct behavior of this contract.

The off-chain implementation $\mathcal{T}_{\text{off}}$ relies on a network of SMPC nodes, which collectively validate and sign transactions before initiating actions on the destination chain. These nodes serve as notaries in the bridging process, making the trust set $\tau \neq \{\}$, and thus *Multichain* is categorized as a trusted bridge. Because the off-chain signing controls minting and releasing assets, the integrity of this notary network is critical to the bridge's security.

On the destination chain, once a transaction is validated off-chain by the SMPC network, a smart contract responsible for minting wrapped assets is triggered. The tokens are minted 1:1 relative to the assets locked in the Decentralized Management Account on the source chain. The trust set on the destination chain is again $\tau = \{\text{SCs}\}$, reflecting reliance on the execution of deployed smart contracts. For reverse transfers, the wrapped token contract burns tokens on the destination chain, and the SMPC nodes authorize the release of the original assets from the source chain's smart contract.

Fee and time considerations depend on the chain and transaction size. For non-Ethereum chains, the cross-chain fee ranges from USD 0.9 to USD 1.9. For Ethereum, a 0.1 percent fee applies, with a minimum of USD 40 and a maximum of USD 1000. Minimum transfer amounts are USD 12 for non-Ethereum destinations and USD 50 when bridging to Ethereum. Maximum transfer size is capped at USD 20 million, though transactions above USD 5 million may incur delays of up to 12 hours.

Multichain supports a broad range of blockchains, including Bitcoin, Litecoin [71], Blocknet [72], ColossusXT [73], Terra [74], and many EVM-compatible networks such as Ethereum, Avalanche C-Chain, Binance Smart Chain, Celo, Polygon, and Arbitrum. This wide integration makes *Multichain* one of the most expansive cross-chain bridges currently available.

Table 7: Connectivity matrix of major blockchain networks based on existing bridge protocols. Each cell indicates whether a direct bridge exists and, if so, categorizes the bridge by its trust model: TL: Trustless. Verification relies solely on the blockchains' own consensus (no third-party custodian), TM: Trust-Minimized. Decentralized validators or bonded operators secure the bridge (additional assumptions, but no single custodian), and TR: Trusted. A custodial or centralized entity/committee secures the bridge (users must fully trust this intermediary).

Chains	BTC	ETH	SOL	AVAX	ATOM	NEAR	DOT	XRP	XLM	ALGO	ADA	ARB	OP	MATIC	BNB
BTC	–	TM	TM	TM	TM	TL	TL	TM		TR	TL	TL	TM	TM	TM
ETH	TM	–	TM	TM	TM	TL	TL	TM	TM	TM	TM	TL	TL	TR	TM
SOL	TM	TM	–	TM		TM		TM	TM	TM		TM	TM	TM	TM
AVAX	TM	TM	TM	–	TM	TM		TM	TM	TM	TM	TM	TM	TM	TM
ATOM	TM	TM		TM	–		TL					TM	TM	TM	TM
NEAR	TL	TL	TM	TM		–						TM	TM	TM	TM
DOT	TL	TL			TL		–		TM			TM	TM	TM	TM
XRP	TM	TM	TM	TM				–	TM		TM	TM	TM	TM	TM
XLM		TM	TM	TM			TM	TM	–					TM	TM
ALGO	TR	TM	TM	TM						–		TM	TM	TM	TM
ADA	TL	TM		TM				TM			–	TM	TM	TM	TM
ARB	TL	TL	TM	TM	TM	TM	TM	TM		TM	TM	–	TM	TM	TM
OP	TM	TL	TM	TM	TM	TM	TM	TM		TM	TM	TM	–	TM	TM
MATIC	TM	TR	TM	TM	TM	TM	TM	TM	TM	TM	TM	TM	TM	–	TM
BNB	TM	TM	TM	TM	TM	TM	TM	TM	TM	TM	TM	TM	TM	TM	–

Table 7 shows a connectivity matrix for major blockchains and selected Layer-2 networks. Each cell is color-coded to indicate the nature of the bridge connection (if one exists) between the row and column blockchains: Trustless, Trust-Minimized, or Trusted. For example, *Rainbow Bridge* connecting Ethereum and NEAR is trustless, while *Binance Bridge* (BNB Chain) and *Polygon PoS* Bridge are trusted/custodial solutions. *Cosmos IBC* and Polkadot's upcoming *Snowbridge* are also designed to be trustless. In contrast, multi-signature or externally validated bridges like *Wormhole* and *Multichain* are trust-minimized (they decentralize validation but still rely on additional trust assumptions beyond the base blockchains. Blank cells indicate no widely used direct bridge between those networks. All entries are based on up-to-date bridge protocols (e.g., *Wormhole*, *Multichain*, *LayerZero*, *IBC*, etc.) and public documentation of their trust models.

D STATE-OF-ART FOR BLOCKCHAIN BRIDGES

Blockchain interoperability has created an ecosystem of bridges linking most L1 and L2 networks. For example, the Wormhole protocol alone connects 23+ blockchains across six different smart contract runtimes [75], and the Axelar network bridges over 40 chains [76]. In parallel, Ethereum serves as a hub in this web; many chains deploy bridges to Ethereum to tap its liquidity and user base. Even previously siloed platforms are integrating. For instance, Cardano [77] has explored the Inter-Blockchain Communication (IBC) Protocol [78] to join the Cosmos network.

D.0.1 Trust Models. Classic examples of trustless bridges fall into two categories: L1-to-L1 bridges, such as IBC, and L1-to-L2 bridges, such as in Arbitrum. On IBC channels, light clients on each chain validate each other's consensus state [79], so IBC's security reduces to the underlying chains' security with no additional trusted parties (native chain security). Rollup-based L1-to-L2 bridges fall into optimistic and zero-knowledge-based bridges [80]. A zero-knowledge

rollup bridge includes a validity proof with every state update, so Ethereum only accepts withdrawals with a valid proof, whereas an optimistic rollup bridge relies on fraud proofs with a challenge period. In optimistic rollup bridges, the bridge inherits the source chain's security and does not introduce new trust assumptions, aside from at least one honest watcher in optimistic systems [81]. Indeed, industry analyses consider Ethereum's rollup bridges (L1-to-L2) among the most trustless interoperability solutions available [2].

Rollups cannot facilitate bridging between L1 chains with different runtimes (e.g., Bitcoin and Ethereum). As a result, most L1-to-L1 connectivity relies on trust-minimized intermediaries (i.e., networks of validators, relays, or oracles) that verify transactions across chains. Users must trust that a majority of these actors behave honestly, often incentivized through mechanisms like staked collateral or slashing penalties. Examples include *Axelar* [82], *Wormhole* [28], *LayerZero* [83], *Multichain* [3], and Celer's *cBridge* [84]. These permissioned or federated bridges require a predefined set of entities to authorize transfers and offer stronger assurances than single custodians, but fall short of the security provided by native chain verification. For instance, bridging assets from Ethereum to *BNB Chain* [85] or Solana often involves a validator coalition verifying the source chain deposit and minting a wrapped asset on the destination chain.

At the far end of the spectrum are trusted bridges, which include custodial bridges (e.g., wrapped Bitcoin via *BitGo*'s WBTC, where a single custodian holds the asset reserve) and some early L1-to-L1 bridges run by a single team or exchange. A notable example was Binance's original bridge connecting Ethereum and *BSC* (*BNB Smart Chain*) [86], essentially operated by Binance as a trusted custodian. Similarly, the *Ronin bridge* for Axie Infinity (prior to its 2022 hack) relied on just nine validators controlled by a single organization (Sky Mavis) and its partners, effectively a federated multisig under one entity's control. Such trusted models have minimal decentralization: users must trust that the custodian or signers will never misbehave, as there are no protocol-level penalties or verifications of their actions. While simple and fast to deploy, these bridges have the weakest security guarantees and have often proven fragile under attack, such as the *Ronin bridge* hack of 2022 (see Table 8). Even when the underlying bridge technology is secure, adversaries may compromise physical infrastructure to exfiltrate private keys.

D.0.2 Connectivity Trends. Appendix Table 7 shows the current bridge topology with clear patterns. Ecosystems with homogeneous technology (e.g., shared virtual machine) use trustless native bridges among themselves, whereas heterogeneous connections (e.g., Bitcoin-Ethereum) gravitate toward trust-minimized hubs. The Cosmos ecosystem is a prime example of the former: dozens of Cosmos-SDK chains (e.g., Osmosis [87], Cosmos Hub, Cronos [88], and Juno [88]) all interconnect via IBC channels with no external mediators.

To bridge out to Ethereum and other ecosystems, Cosmos projects deployed separate bridge modules and networks. For example, the *Gravity Bridge chain* [89] and Axelar network act as Cosmos-Ethereum L1-to-L1 bridges by having their validator sets observe events on Ethereum and vice versa [90]. These are trust-minimized solutions (essentially multisig validator bridges) added onto Cosmos to import assets like USDC, DAI, and WETH into the IBC realm.

Table 8: Documented security incidents and failures in cross-chain bridge and routing architectures. Most attacks involve violations of the cross-chain causality prior.

Bridge	Architecture Type	Date (↓)	Loss (USD)	Technique	Violation	Target	Vector
Thorchain	Sidechain/Relay	2021-06-29	140K	False top-up	Causality	Source chain	V3
ChainSwap	Notary	2021-07-02	800K	Contract vulnerability	Consistency	Source chain	V3
ChainSwap 2	Notary	2021-07-11	4M	Contract vulnerability	Consistency	Source chain	V3
Multichain	MPC (custodial)	2021-07-10	7.9M	Private key compromised (Bad ECDSA)	Causality	Source chain	V13
Thorchain 2	Sidechain/Relay	2021-07-15	7.6M	False top-up	Causality	Source chain	V9
Thorchain 3	Sidechain/Relay	2021-07-22	8M	Refund logic exploit	Causality	Source chain	V3
Thorchain 4	Sidechain/Relay	2021-07-24	76K	Phishing attack	Causality	Off-chain relayer	V23
Poly Network	Notary	2021-08-10	611M	Trusted state root exploit	Causality	Source chain	V9
pNetwork	Notary	2021-09-20	13M	Inconsistent event parsing	Consistency	Off-chain relayer	V6
Plasma Bridge	Sidechain/Relay	2021-10-05	None	Reused burn proof	Consistency	Destination chain	V15
Optics Bridge	Notary	2021-11-23	None	Multi-signature permission vulnerability	Causality	Off-chain relayer	V3
Multichain 2	MPC (custodial)	2022-01-18	1.4M	Token contract vulnerability	Consistency	Source chain	V3
Qubit Finance	Relay	2022-01-28	80M	Deposit function exploit	Causality	Source chain	V3
Wormhole	Validator-based	2022-02-02	326M	Signature exploit	Causality	Destination chain	V12
Meter	Sidechain/Relay	2022-02-06	7.7M	Deposit function exploit	Causality	Source chain	V3
Ronin	Sidechain	2022-03-23	624M	Private key compromised (social engineering)	Causality	Off-chain relayer	V13
Marvin Inu	Custodial	2022-04-11	350K	Private key compromise	Causality	Source chain	V13
QANX Bridge	Notary	2022-05-18	2.2M	Deploy false bridge contract onto BSC	Consistency	Source chain	V3
Horizon Bridge	Notary	2022-06-23	100M	Private key compromised (unknown method)	Causality	Off-chain relayer	V12
Nomad	Relay	2022-08-01	190M	Trusted state root exploit	Causality	Destination chain	V10
Celer	Sidechain/Relay	2022-08-18	240K	DNS cache poisoning attack on frontend UI appprox	Causality	Off-chain relayer	V23
Omni Bridge	Notary	2022-09-18	290K	Possible ChainID vulnerability	Consistency	Destination chain	V15
Binance Bridge	Custodial	2022-10-06	570M	Proof Verifier Bug	Consistency	Off-chain relayer	V9
QANX Bridge 2	Notary	2022-10-11	2M	Weak address key vulnerability	Consistency	Destination chain	V13
Thorchain 5	Sidechain/Relay	2022-10-28	None	Network interruption	Causality	Off-chain relayer	V17
pNetwork	Hash-locking	2022-11-04	10.8M	Misconfiguration of the pNetwork-powered bridge	Consistency	Token contract	V3
Multichain 3	MPC (custodial)	2023-02-15	130K	Rush attack	Consistency	Source chain	V13
Dexible	Trade router	2023-02-20	2M	Unchecked destination address	Consistency	Source chain	V5
Allbridge	Sidechain/Relay	2023-04-02	570K	Logic flaw in withdraw function	Consistency	Destination chain	V2
Celer	Hybrid	2023-05-24	None	Double-voting vulnerability	Causality	Off-chain relayer	V18
Poly Network	Notary	2023-07-02	10M	Private key compromised	Causality	Off-chain relayer	V13
Poly Network	Notary	2023-07-03	4.4M	Compromised multisig	Causality	Off-chain component	V13
Multichain 4	MPC (custodial)	2023-07-07	126M	Compromised address	Causality	Off-chain component	V13
Multichain 5	MPC (custodial)	2023-07-14	unknown	State seized multisigs	Causality	Off-chain component	V12
Shibarium	L2 Sequencer Bridge	2023-08-17	None	Frozen withdrawals due to L2 bug	Causality	Off-chain component	V17
HTX	MPC (custodial)	2023-11-22	12.5M	Hot wallet compromised	Causality	Off-chain component	V13
HECO Bridge	MPC (custodial)	2023-11-22	86.6M	Hot wallet compromised	Causality	Off-chain component	V13
HYPR Network	Sidechain/Relay	2023-12-13	220K	Vulnerability in external code dependency	Consistency	Destination chain	V7
Socket's Bungee	Aggregator (router)	2024-01-16	3.3M	Compromised multisig	Consistency	Source chain	V12
Alex	Notary	2024-05-14	4.3M	Compromised deployer	Causality	Source chain	V12, V7
LifiJumper	Aggregator (meta-router)	2024-07-16	9.73M	Buggy function	Consistency	Source chain	V2
Ronin 2	Sidechain	2024-08-06	12M	Whitehat MEV attack (parameter error in update)	Causality	Source chain	V4
Feed Every Gorilla	Notary (Wormhole-based)	2024-12-30	1.3M	Message spoofing	Causality	Source chain	V9

Ethereum and its orbiting chains illustrate a different connectivity paradigm. Virtually every major L1 (Solana, BNB Chain, Avalanche, Polygon, Fantom [91], etc.) has at least one bridge to Ethereum, given the high value of assets and liquidity on Ethereum. For example, *Wormhole* operates as a cross-chain message network connecting Ethereum with Solana, BSC, Avalanche, Polygon, and others, using a guardian consensus (a set of 19 nodes) to validate transfers. When a user moves ETH from Ethereum to Solana via the L1-to-L1 *Wormhole*, the ETH is locked in a *Wormhole* contract on Ethereum, and a wrapped ETH is minted on Solana, based on a signed attestation by the guardian network. This design is trust-minimized (no single custodian, but users trust the majority of guardians).

Other popular Ethereum bridges follow similar patterns: *Multichain* (Anyswap) uses a rotated set of Multi Party Computation signers to custody and mint assets between Ethereum and EVM chains; Celer *cBridge* uses a Proof-of-Stake validator set to oversee transfers. In all these cases, Ethereum-to-L1 bridges tend to introduce a separate middleware layer.

Outside the Ethereum/Cosmos spheres, other ecosystems have pursued their own bridging approaches with mixed strategies. Solana, for instance, developed ties to Ethereum and others mainly through third-party bridge networks. *Wormhole*'s origin was as the Solana-Ethereum L1-to-L1 bridge (also known as *Portal*), and it remains the primary connector between Solana and multiple other chains.

One constraint Solana faced is that its very high throughput (2.9K - 65K tx/sec) and non-EVM-based execution environment meant fewer independent bridge options. The ecosystem largely relied on *Wormhole*, which introduced a single point of failure for connectivity. Recent projects are working on light-client bridges (e.g., using Solana's clients), and even Cosmos's IBC was adapted in 2023 to work from Solana via an adapter chain. However, Solana's

cross-chain connectivity is primarily driven by multi-signature bridge protocols rather than native, trustless channels. The BNB Chain ecosystem likewise began with a highly centralized bridging model. BNB Chain comprises a dual-chain system, consisting of an application-focused Beacon Chain and an EVM-compatible Smart Chain, bridged by the "Token Hub". The BSC Token Hub initially operated under the tight control of a few validators, as evidenced by the fact that a successful attack in October 6, 2022 was able to forge just two messages and mint 2 million BNB (see Table 8, an exploit that even forced a temporary halt to the chain. In response, BNB Chain has moved to increase the security of its bridges (e.g., raising multisig thresholds and improving key management) and has also encouraged the use of external bridges.

Nowadays, BNB Chain is connected to Ethereum and others via both Binance's official L1-to-L1 bridge (still centrally governed) and alternatives like *Celer*, *deBridge* [92], and *LayerZero* [83], which are more decentralized. Nevertheless, BNB's connectivity strategy remains cautious: its native bridge prioritizes speed and user experience (at the cost of trust), whereas third-party solutions offer decentralization but introduce dependency on external validator sets. We observe a similar pattern in other ecosystems, such as Polygon (the *Polygon PoS* bridge utilizes a 5-of-8 multisig to checkpoint withdrawals, a somewhat trusted design, alongside newer trust-minimized options) and Tron (which relies on custodial bridges for assets like BTC/ETH via BitGo, or exchange-mediated transfers). In contrast, Polkadot [93] takes a route closer to Cosmos: all parachains in a Polkadot parachain cluster are natively interoperable via the XCM (Cross-Consensus Message) format [94], which is "trustless and secure" for in-network messaging. Polkadot's relay chain ensures the validity of messages between parachains, so within this ecosystem, cross-chain actions (such as token transfers and remote contract calls) don't require external trust. However, bridging Polkadot to external chains still entails separate bridge projects (often with their own validator sets or light clients). Efforts are underway (e.g., Snowfork and Darwinia for Ethereum-Polkadot L1-to-L1 bridges) to make those as trust-minimized as possible, potentially even integrating light-client verification.

E DESCRIPTIONS OF †ATTACK VECTORS

In Table 2, we used † to indicate that these vulnerabilities only apply under certain design conditions. We now elaborate on those conditions.

V1: Fake Burn/Lock Proofs. Tokens are locked or burned on the source chain, X, before minting can occur on the destination chain, Y. A cryptographic proof sent from X to Y is required for confirmation before minting can occur. This attack involves forging the proof somehow to trick the Y chain into minting before any proper burn or lock has actually happened.

V2: Malicious Transaction Modification. Transaction details (addresses, number of tokens, gas fees, etc.) are sent from the source chain to the destination chain during a transaction. Malicious transaction modification involves somehow intercepting this detailed payload and asserting some fake payload. This can be done off-chain with protocols that include some centralized sequencer, or on-chain when the destination bridge is looking for the detailed payload.

V3: Reentry Attack Occurs when some external function is called multiple times before the initial execution has completed. This attack vector was used in the famous DAO attack, where the token minting was executed many times before the balance was checked and updated.

V4: Integer Overflow/Underflow When overflow or underflow remain unchecked, attackers can induce attacks by creating unexpected results. A wrap-around can also happen, where the maximum value can wrap around to the minimum value, or vice versa, creating room for exploitation.

V5: Access Control Flaws Access control is very important in maintaining control over a contract, and if unchecked, an attacker can impersonate a contract owner and manipulate the entire contract.

V6: Timestamp Manipulation A large number of blockchain protocols require some timestamp with loose limits to allow for network delays. These loose limits allow attackers to falsify timestamps that will still be considered valid by the protocol, leading to vulnerable behavior in the protocol.

V7: Inconsistent Cross-Chain Transfers Bridges can implement different methods for withdrawing and depositing, such as burning when performing a withdrawal but minting on a deposit. When these operations are different, attackers can take advantage of a lack of one-to-one consistency.

V8: Replay Attacks Attackers can take a message in transit that was validated, then delay and retransmit the original message in order to assume validation but cause effects unintended by the original message.

V9: Oracle Manipulation Oracles being used to provide data, such as token pricing, can be points of vulnerability if an attacker were to take control of the oracle and provide fake data, leading to incorrect fees or token amounts during minting or burning. Oracles can provide more data than just token prices, leading to more points of possible attack.

V10: Consensus Failure Commonly referred to as a 51% attack, if an attacker were to control 51% of the mining or staking power (depending on the protocol) then that attacker would be able to create a new chain segment that builds faster than the legitimate chain. This control can lead to false transactions that could mint tokens on other chains.

V11: Malicious Custodian Manipulation Custodial wallets are commonly used to hold locked tokens or private keys via a centralized source. Compromising or maliciously colluding with this custodial wallet leads to access of locked tokens, private keys, or other sensitive data.

V12: Key Leakage / Private Key Theft Locked assets can exist within bridges accessible by private keys and bridges themselves can hold keys along with their controlling proxy contracts. Any compromising of these keys through phishing or other targeted attacks can create vulnerabilities, potentially giving an attacker full control over a bridge.

V13: Race Condition Attacks A vulnerability is created when two or more processes access the same data, such as when withdrawals and deposits are processed without proper synchronization, and conflicting transactions can occur.

V14: Unsafe External Call Exploits When a smart contract accesses some external call to third-party contracts or off-chain contracts, these new calls create vulnerability in their own code security that can be exploited.
V15: Forged Account Attacks Occurs when an account or address is deliberately created to impersonate the identity of another party. This can lead to the redirection of tokens in a bridge structure if a fake account is validated.
V16: Malicious Event Log Manipulation Key actions are logged by smart contracts, typically off-chain. Falsification of the event log can lead to relaying of some fake events that have not happened. This results in tricking the chain into a false state.
V17: Denial of Service Attacks Bridge contracts can be flooded with transactions and requests sent by an attacker, overwhelming the bridge protocol and slowing processes for all transactions on the bridge. This could potentially lead to the complete halting of a bridge protocol.
V18: Arithmetic Accuracy Deviation Exact calculations are required to maintain consistency across chains. Tiny errors in integer overflow or underflow can cause precision errors, leading to inconsistency between chain states.
Function Visibility and Modifier Use.

F DESCRIPTIONS OF HACKS AND ATTACKS ON BRIDGES

In one of the earliest and largest bridge hacks, attackers exploited a flaw in *Poly Network*'s cross-chain transaction handling contract, "literally tricking the project to hack itself". The *Poly Network bridge* connected multiple chains, including Ethereum, BSC, Polygon, and others. Its core contract (*EthCrossChainManage*) had an insecure design: it could make an external call to an arbitrary target contract with supplied data, without proper authorization checks. The attacker crafted a call payload that misled the manager contract into invoking *Poly*'s own storage contract (*EthCrossChainData*) and updating the keeper of funds as if the attacker were the owner. As a result, approximately $611 million worth of cryptocurrency was drained from *Poly Network* in a single attack, with the hacker subsequently returning the funds. The attack flow is illustrated in Figure 4.

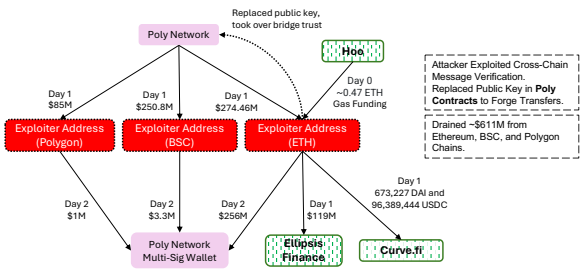

Figure 4: Attack flow of the Poly Network exploit.

The *Ronin bridge* to Ethereum was victim to a validator key compromise in an attack now attributed to North Korea's Lazarus Group. *Ronin* used a 5-of-9 multisig validation for bridge withdrawals. Attackers compromised five private keys through social engineering. Once in control, they issued two fraudulent transactions, draining 173,600 ETH and 25.5M USDC (worth approximately $624 million at the time) from the *Ronin bridge* in a single stroke. This event, the largest DeFi hack ever, was essentially a failure of the bridge's trust model: the off-chain validators were assumed honest, but the minimal quorum and centralized key management (a single entity controlled 4 of 9 validators) made it easy for an attacker to breach. The breach went unnoticed for six days until a user discovered it. Figure 5 summarizes the transaction flow of this exploit.

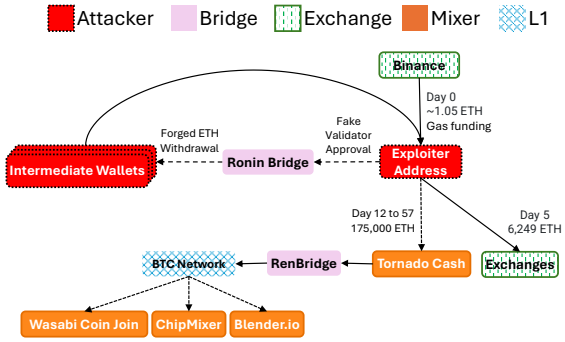

Figure 5: Attack flow of the *Ronin* bridge exploit.

Wormhole's Ethereum–Solana bridge was hacked for 120,000 WETH (worth $326 million) in a smart contract bug on the Solana side. *Wormhole* relies on a set of guardian nodes to attest cross-chain messages, generating a signed VAA (Verifiable Action Approval). On Solana, a *Wormhole* program should verify guardian signatures before minting new tokens. However, a critical mistake was introduced in an update: the Solana verification function was using an outdated system call for signature checks, effectively bypassing the actual verification. The attacker discovered that they could simply fabricate a data account that pretended to be a valid signature set, since the contract wasn't actually validating the signatures against the guardian keys. By exploiting this signature verification bypass, the attacker called *Wormhole*'s complete_wrapped routine on Solana to mint 120k wrapped ETH for themselves with no real backing. The exploit structure is depicted in Figure 6.

Qubit Finance's bridge between Ethereum and BSC was hit for $80 million in an attack that abused improper input validation. *Qubit*'s contract allowed users to deposit an asset on Ethereum and mint a pegged version on BSC. The hack targeted the BSC side's deposit function: due to a logic error in a single overlooked condition, the bridge's smart contract did not verify that a call to transfer ETH actually happened when a deposit message was processed. In simpler terms, an attacker invoked the bridge deposit with zero ETH but bypassed the intended failure check, tricking the contract into believing collateral was provided. This exploit allowed the attacker to mint limitless xETH (the bridged ETH on BSC) without any real ETH locked on Ethereum. The exploit structure is depicted in Figure 7.

The *Nomad* bridge (connecting Ethereum with Moonbeam and other chains) suffered a $190 million loss in one of the most chaotic

Table 9: Cross-Chain Bridge Implementations and Architecture

Bridge	Source Chain (Asset Custody & Locking)	Off-Chain Mechanism (Cross-Chain Validation)	Destination Chain (Asset Release & Minting)
Allbridge Classic	Smart Contracts – Assets locked in source-chain bridge contract[2]	Notaries/Relays – Small validator set observes lock and signs confirmations[3]	Smart Contracts – Target chain bridge contract verifies signature and mints tokens[4]
deBridge	Smart Contracts – Gateway contract escrows assets[5]	Notaries/Relays – Decentralized validator committee signs off-chain[6]	Smart Contracts – Destination contract verifies signatures and releases assets[7]
Multichain	Smart Contracts (MPC custody) – Deposited assets held by MPC-controlled vault[8]	Notaries/Relays – Multi-party computation (MPC) validators co-sign transactions[9]	Smart Contracts – Destination contract mints wrapped assets upon MPC authorization[10]
Wormhole Bridge	Smart Contracts – Core contract locks assets or emits transfer message[11]	Notaries/Relays – Guardian network signs verified action approvals (VAA)[12]	Smart Contracts – Destination chain contract verifies VAA and mints assets[13]
Avalanche Bridge	Validator Control – Ethereum escrow EOA with multi-sig custodian[14]	Notaries/Relays – Intel SGX enclave validators approve transactions[15]	Smart Contracts – Destination bridge contract mints assets upon SGX approval[16]
Ronin Bridge	Smart Contracts – Ethereum bridge contract locks assets[17]	Sidechain – Ronin validators (5 of 9 required) approve transfers[18]	Validator Control – Ronin validators execute mint/burn on sidechain[19]
Wanchain Bridge	Validator Control – Storeman group (25 nodes) holds escrow[20]	Notaries/Relays – Storeman nodes verify transactions using MPC[21]	Validator Control – Storeman group authorizes mint/release on target chain[22]
Connext	Smart Contracts – NXTP protocol escrow locks tokens[23]	Relays – Liquidity routers observe lock and relay signed messages[24]	Smart Contracts – Destination contract releases assets using router liquidity[25]
Axelar	Smart Contracts – Gateway contract on source chain locks assets[26]	Sidechain – Axelar validators reach consensus and use threshold cryptography[27]	Smart Contracts – Destination gateway releases tokens upon validator approval[28]
Binance Bridge	Validator Control – Binance custodial wallet holds assets[29]	Notaries/Relays – Binance-controlled relay system detects deposits[30]	Validator Control – Binance mints BTokens pegged to locked assets[31]
Horizon Bridge	Validator Control – 2-of-5 multi-signature custody on Ethereum[32]	Notaries/Relays – Federated validators approve transfers[33]	Validator Control – Multi-sig bridge mints tokens on Harmony[34]
BTC Relay	Validator Control – Bitcoin transactions are verified by BTC Relay's on-chain light client, but funds must be sent to an externally controlled address[35]	Light Client – BTC Relay runs a Merkle proof verification contract on Ethereum that allows Bitcoin transactions to be verified on-chain[36]	Smart Contracts – Ethereum contracts verify Bitcoin transactions and trigger asset issuance or validation[37]
zkBridge	Smart Contracts – Source chain tokens are locked in a bridge contract before proof generation[38]	Light Client – zk-SNARK proofs verify block headers across chains without external validators[39]	Smart Contracts – Destination chain contracts validate proofs and mint or release tokens[40]
PeaceRelay	Smart Contracts – Ethereum-based contracts escrow tokens and validate state transitions[41]	Light Client – Smart contracts maintain Merkle-Patricia proofs for cross-chain validation[42]	Smart Contracts – Contracts on Ethereum Classic or other chains verify proofs and mint assets[43]
Rainbow Bridge	Smart Contracts – NEAR and Ethereum smart contracts escrow tokens and initiate cross-chain transfers[44]	Light Client – Each chain hosts a light client verifying the other chain's consensus, eliminating external trust[45]	Smart Contracts – Contracts on NEAR or Ethereum validate and execute transactions upon light-client verification[46]
Cosmos IBC	Smart Contracts – The IBC module in Cosmos SDK escrows assets and generates proofs[47]	Light Client – Chains run light clients of each other, verifying block headers using the Tendermint consensus[48]	Smart Contracts – The receiving chain validates IBC messages and mints corresponding assets[49]
Gravity Bridge	Validator Control – Ethereum assets are locked in a Gravity.sol contract controlled by the Gravity Bridge validators[50]	Sidechain – Gravity Bridge validators observe Ethereum transactions and reach consensus on transfers[51]	Validator Control – Tokens are minted on Cosmos chains upon validator consensus[52]
zkRelay	Validator Control – Transactions from PoW chains like Bitcoin are observed, with no native smart contracts on the source chain[53]	Light Client – zk-SNARKs are used to verify large batches of Bitcoin headers efficiently[54]	Smart Contracts – Ethereum or other chains verify the zk-proof and execute transactions accordingly[55]
Cactus	Validator Control – A permissioned set of validators controls funds and relays messages[56]	Notaries/Relays – Cross-chain transactions are notarized by trusted relayers in enterprise settings[57]	Validator Control – The target blockchain executes transactions based on validator approvals[58]
Celer cBridge	Smart Contracts – Liquidity pools and lock-mint contracts facilitate asset transfers[59]	Sidechain/Relayer Network – Celer's State Guardian Network (SGN) validates transfers via PoS consensus[60]	Smart Contracts – cBridge smart contracts process releases based on SGN validators' signed messages[61]
Orbit Bridge	Validator Control – A federation of Orbit Chain validators governs asset custody[62]	Sidechain – Orbit Chain runs a dedicated blockchain that validates cross-chain transactions[63]	Validator Control – Assets are minted or unlocked on the target chain by Orbit validators[64]

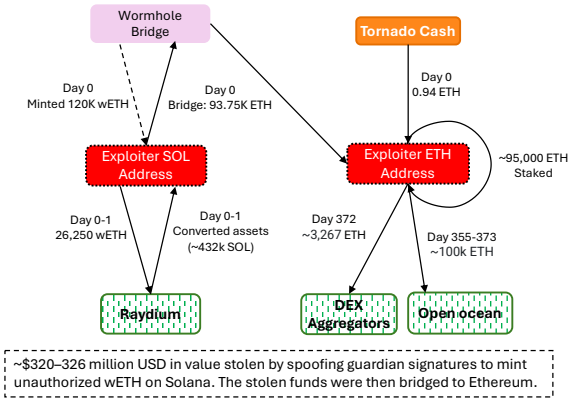

Figure 6: Attack flow of the *Wormhole* bridge exploit.

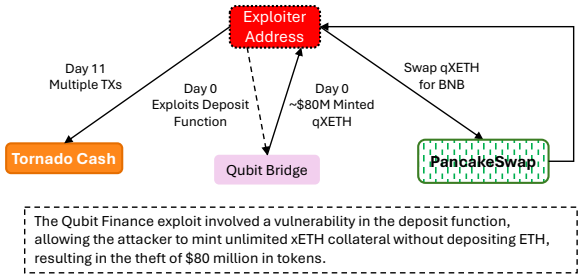

Figure 7: Attack flow of the *Qubit* bridge exploit.

exploits to date. *Nomad* was an optimistic messaging bridge, meaning it had a system of bonded updaters and a fraud challenge window. Unfortunately, after a routine smart contract upgrade, a critical initialization bug was introduced: the *Nomad* Replica contract on Moonbeam was set with a trusted root of 0x00…00 (zero) by mistake, which immediately enabled every message to be accepted as valid. Attackers quickly realized this and began submitting bogus withdrawal messages to drain tokens. The incident became a free-for-all: once the method was public, dozens of copycats (some white-hat, some black-hat) jumped in to also withdraw funds by replaying the attacker's transaction call data. The drain pattern across wallets is illustrated in Figure 8.

Harmony's *Horizon* bridge (connecting Harmony's chain with Ethereum) was hacked for about $100 million in assets, through a straightforward attack on its 2-of-5 multisig validators. Similar to

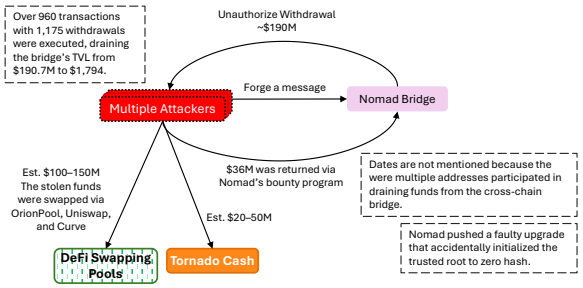

Figure 8: Attack flow of the *Nomad* bridge exploit.

Ronin, the attackers somehow obtained the private keys for at least two of the five signer addresses that controlled the bridge. Figure 9 shows the flow of compromised validator-controlled funds.

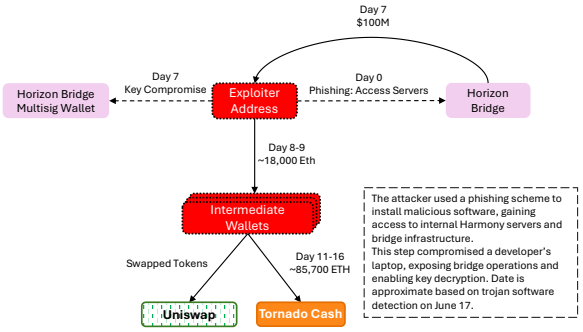

Figure 9: Attack flow of the *Horizon* bridge exploit.

An attacker exploited the Binance Smart Chain's main bridge (between BSC and its Beacon Chain) and managed to mint 2 million BNB (≈ $586 million). The root cause was a bug in the light-client based proof verification on the BSC side. The bridge used an internal light client (IAVL tree-based) to verify bridge messages. The attacker created a fake proof for a past block that the BSC light client would accept as valid, bypassing the normal transaction inclusion checks. By forging this cryptographic proof, the attacker was able to inject a malicious block update that instructed the bridge to mint new BNB to their address. It was a complex exploit at the cryptography/verification layer, essentially an attack on the consensus verification between the two Binance chains. In response, Binance halted the entire BSC chain for 8 hours to prevent the

attacker from moving more funds. A large portion of the illicit BNB never left Binance-controlled wallets, limiting the realized theft to around $100M. This incident revealed that even a seemingly trustless bridge (a light-client-based one) can have implementation flaws in its verification logic. The exploit structure is depicted in Figure 10.

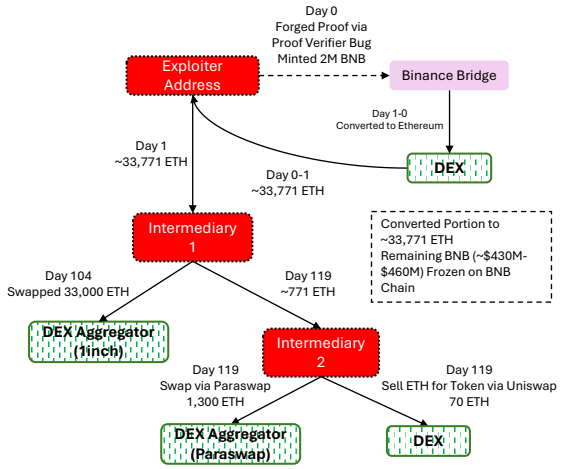

Figure 10: Attack flow of the Binance exploit.

In mid-2023, the *Multichain* bridge (formerly Anyswap), which linked over a dozen EVM chains, experienced a sudden breach resulting in approximately $126 million in assets withdrawn. Subsequent investigation suggested a compromise of the project's private keys or server infrastructure. It came to light that all of *Multichain*'s critical MPC key shares were under the control of its CEO, who had been detained by authorities in China. The attackers (or insiders) managed to use these keys to authorize massive transfers from the bridge's liquidity pools on multiple chains. Essentially, this was an insider compromise. The fallout also disrupted many linked chains' assets and led to the project's collapse. The fund outflow pattern across chains is shown in Figure 11.

Following Ethereum's transition from proof-of-work to proof-of-stake, the OmniBridge was exploited in September 2022 using a replay attack. Initially, the attacker sent 200 WETH using the OmniBridge on the Ethereum PoS chain. Since transaction format was identical did not have chain ID verification, the attacker replayed the same transaction on the EthereumPoW fork. OmniBridge failed to distinguish between the two chains, therefore it processed the transaction and then released 200 ETHW to the attacker again. This exploit highlights the need for implementing strict chain ID validation during hard forks or network upgrades due to the possibility of having identical transaction histories across chains.

Smaller but instructive bridge attacks continued through 2024. For example, *Orbit Chain* (Jan 2024) lost $10M when 7 of its 10 bridge validators were compromised, another case of federated signer failure. An exploit in *ALEX Bridge* (May 2024) (connecting Stacks to other chains) also occurred, reportedly due to a logic bug in its code. These ongoing incidents show that despite industry awareness, bridge security remains challenging.

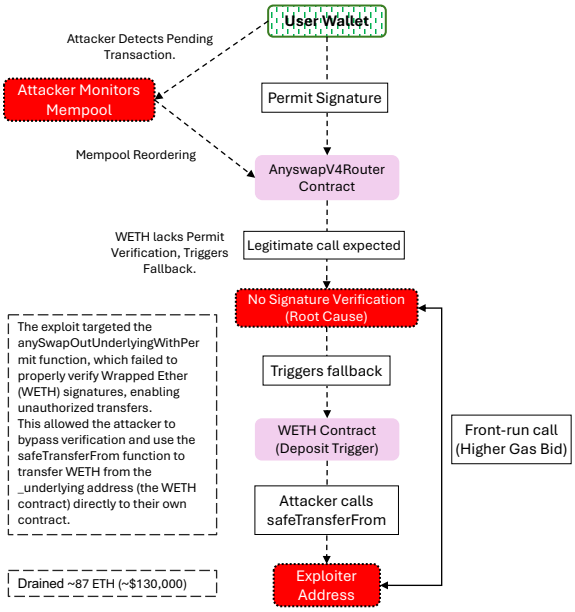

Figure 11: Attack flow of the *Multichain* exploit.

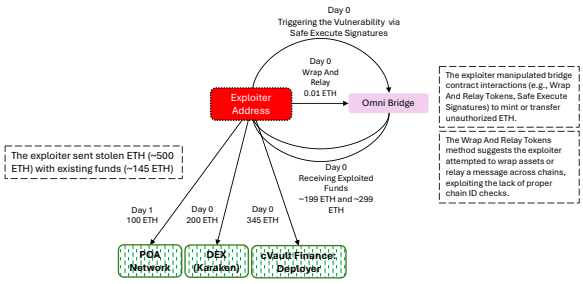

Figure 12: Attack flow of the *Omni* bridge exploit.

STATIC CODE TABLE COLUMN DESCRIPTIONS

Tables 3, 4, and 5 summarize features of bridge smart contracts extracted via static code analysis. Below, we define each column and its relevance.

- **Local vars**: Number of local (function-scoped) variables, indicating complexity of internal logic.
- **Inheritances**: Number of Solidity contracts inherited, reflecting code reuse or modularity.
- **Modifier Count**: Number of `modifier` constructs used to restrict access or enforce invariants.
- **RoleBased**: Indicates whether role-based access control mechanisms (e.g., `AccessControl`) are implemented.
- **Standard Libs**: Count of imported standard libraries such as `SafeMath` or `Ownable`.
- **LOC (Lines of Code)**: Number of non-comment, non-whitespace lines in the source code.
- **Total Lines**: Full line count including comments and whitespace.

Table 10: Blockhain Bridge Exploits

Exploit Address	Attack Name	Attack Date	Bridge Name	Bridge Address
0xc8a...963	Poly Network Trusted State Root Exploit 1	2021-08-10	Poly Network EthCrossChainManager	0x144...88c
0x5dc...214	Poly Network Trusted State Root Exploit 2	2021-08-10	Poly Network EthCrossChainManager	0x144...88c
0x123...678	Multichain Rush Attack 1	2023-02-15	Multichain Router V4	0x6b7...522
0x9d5...b68	Multichain Rush Attack 2	2023-02-15	Multichain Router V4	0x6b7...522
0xefe...c88	Multichain Rush Attack 3	2023-02-15	Multichain Router V4	0x6b7...522
0x418...bb7	Multichain Rush Attack 4	2023-02-15	Multichain Router V4	0x6b7...522
0x622...ba0	Multichain Rush Attack 5	2023-02-15	Multichain Router V4	0x6b7...522
0x48b...537	Multichain Rush Attack 6	2023-02-15	Multichain Router V4	0x6b7...522
0x027...cd8	Multichain Rush Attack 7	2023-02-15	Multichain Router V4	0x6b7...522
0x489...bec	Binance Bridge Proof Verifier Bug	2022-10-06	Binance Bridge Ethereum Contract	0x69F...66D
0x82f...677	Omni Bridge ChainID Vulnerability Exploit	2022-09-18	OmniBridge Multi-Token Mediator	0x88a...671
0xb5c...90e	Nomad Trusted State Root Exploit 1	2022-08-01	Nomad BridgeRouter	0x88a...0a3
0x56D...4e3	Nomad Trusted State Root Exploit 2	2022-08-01	Nomad BridgeRouter	0x88a...0a3
0xBF2...179	Nomad Trusted State Root Exploit 3	2022-08-01	Nomad BridgeRouter	0x88a...0a3
0x0d0...d00	Horizon Bridge Private Key Compromised	2022-06-24	Horizon Bridge	0x2dc...857
0x098...f96	Ronin Private Key Compromised (Social Engineering)	2022-03-23	Ronin Bridge Vault (V1)	0x1A2...4F2
0x629...71a	Wormhole Account Spoofing	2022-02-02	Wormhole Portal Token Bridge	0x3ee...585
0xd01...5c7	Qubit Finance Deposit Function Exploit	2022-01-28	Qubit QBridge (Ethereum)	0xD88...726
0x8c1...d62	Thorchain Private Key Compromised (Phishing Attack)	2021-07-24	THORChain Bifrost ETH Router	0xc14...2ce

- **Public / External / Internal / Private Funcs**: Number of functions by visibility. Public and external functions are externally callable and represent direct attack surfaces.
- **Global vars Declared**: Number of state variables declared at the contract level.
- **Ext Funcs**: Number of externally callable functions.
- **Low-level**: Number of low-level calls (`call`, `delegatecall`, `staticcall`).
- **Untrusted**: Instances where the target of a low-level call is not a hardcoded or trusted address.
- **Reentry Guard**: Presence of reentrancy protection (e.g., via the `nonReentrant` modifier).
- **Require / Assert**: Total number of `require` and `assert` statements used for validation and safety checks.
- **Custom Errors**: Count of Solidity custom error definitions used for gas-efficient error handling.
- **Checks/Fn**: Average number of `require` or `assert` checks per function, used as a proxy for defensive programming density.