A Comprehensive Overview of Security Vulnerability Penetration Methods in Blockchain Cross-Chain Bridges

Qianrui Zhao[1], Yinan Wang[2], Bo Yang[3], Ke Shang[3], Maozeng Sun[3], Haijun Wang[1], Zijiang Yang[1], and Haojie Xin[1]

[1]Xi'an Jiaotong University
[2]Kweichow Moutai Group
[3]Beijing Unionpay Card Technology Co LTD

October 18, 2023

Abstract

Cross-chain bridges are crucial mechanisms for facilitating interoperation between different blockchains, allowing the flow of assets and information across various chains. Their pivotal role and the vast value of assets they handle make them highly attractive to attackers. Major security incidents involving cross-chain bridge projects have been occurring frequently, resulting in losses of several billion due to cyber attacks. The diversity of vulnerability exploitation methods by hackers is vast, but not entirely untraceable. There are scarce research outcomes studying cross-chain bridge cyber incidents, and we have conducted a study based on the most recent cross-chain bridge security incidents. We introduce the working principles, components, and architecture of cross-chain bridges, explain the categorization mechanisms of the trust layer in cross-chain bridges, summarize four categories of hacker vulnerability exploitation techniques from real cases, and propose preventative measures for cross-chain bridge security.

Received: Added at production | Revised: Added at production | Accepted: Added at production

DOI: xxx/xxxx

REVIEW ARTICLE

A Comprehensive Overview of Security Vulnerability Penetration Methods in Blockchain Cross-Chain Bridges

Qianrui Zhao[1] | **Yinan Wang**[2,3] | **Bo Yang**[4,5] | **Ke Shang**[6] | **Maozeng Sun**[6] | **Haijun Wang**[7] | **Zijiang Yang**[1] | **Haojie Xin**[1]

[1]Computer Science and Technology, Xi'an Jiaotong University, Xi'an, China

[2]Digital and Information Management Department, Kweichow Moutai Group, Renhuai, China

[3]Digital and Information Management Center, Kweichow Moutai Co., LTD., Renhuai, China

[4] R&D Center, Beijing Unionpay Card Technology Co., LTD., Beijing, China

[5]R&D Center, National Fintech Evaluation Center, Beijing, China

[6]High-end IoT Department, Beijing Unionpay Card Technology Co., LTD., Beijing, China

[7]Cyber Science and Engineering, Xi'an Jiaotong University, Xi'an, China

Correspondence
Haijun Wang,Cyber Science and Engineering,Xi'an Jiaotong University,Xi'an,China.
Email: haijunwang@xjtu.edu.cn

Abstract

Cross-chain bridges are crucial mechanisms for facilitating interoperation between different blockchains, allowing the flow of assets and information across various chains. Their pivotal role and the vast value of assets they handle make them highly attractive to attackers. Major security incidents involving cross-chain bridge projects have been occurring frequently, resulting in losses of several billion due to cyber attacks. The diversity of vulnerability exploitation methods by hackers is vast, but not entirely untraceable. There are scarce research outcomes studying cross-chain bridge cyber incidents, and we have conducted a study based on the most recent cross-chain bridge security incidents. We introduce the working principles, components, and architecture of cross-chain bridges, explain the categorization mechanisms of the trust layer in cross-chain bridges, summarize four categories of hacker vulnerability exploitation techniques from real cases, and propose preventative measures for cross-chain bridge security.

KEYWORDS

Cross-Chain Bridge, Security Analysis, Attack Cases

1 | INTRODUCTION

Blockchain is a decentralized distributed ledger technology, maintained and verified collaboratively by multiple nodes within a network[1]. Every participating node has a complete copy of the ledger, and a consensus algorithm ensures the consistency of the data. This decentralized nature grants blockchain with extreme reliability and tamper-resistance. Blockchain technology has demonstrated tremendous potential in various fields such as finance, supply chain management, and digital assets, ensuring data transparency, security, and traceability, thereby improving efficiency and reducing intermediaries.

With the rise of fintech and the enormous potential of blockchain technology across various industries, it has attracted the attention of major companies and organizations, leading to the continuous development of new blockchain and cryptocurrency projects. These projects have different purposes and features, so they need their own blockchain networks to work properly. For enterprises and organizations, they may prefer to establish private or consortium blockchains, separate from public blockchains, to meet specific business requirements. Reports[2] indicate that there are now at least 1, 000 blockchain networks in existence today. The number of blockchains keeps growing. Because each blockchain forms a distinct digital environment with its own communication protocols, consensus rules, governance models, and unique blockchain characteristics, this "independence" makes it challenging for blockchains to communicate effectively with each other. They become isolated "islands" and struggle to facilitate value transfer between them. For instance, users cannot directly use Bitcoin on the Ethereum. To achieve the integration

Abbreviations: ANA, anti-nuclear antibodies; APC, antigen-presenting cells; IRF, interferon regulatory factor.

of blockchain projects across industries and realize the interconnection and interoperability of blockchains, thereby achieving the purpose of inter-chain flow of business and value, blockchain has an urgent practical need for cross-chain interaction technology, hence the emergence of cross-chain technology.

The inception of cross-chain technology can be traced back to Ripple's introduction of the Interledger Protocol (ILP) in 2012. In 2013, Schwartz et al. proposed atomic swap technology, which ensures the atomicity of cross-chain transactions while enabling cross-chain asset exchanges[3 4]. In 2014, the BlockStream team disclosed the Pegged Sidechains cross-chain interaction protocol in the white paper[5]. This protocol introduced the concept of Pegged Sidechains, which involves the creation of a sidechain anchored bi-directionally to the main chain through a two-way peg mechanism, enabling cross-chain asset transfers. In 2015, the Bitcoin Lightning Network adopted the Hashed Timelock mechanism, enabling rapid transaction channels on the Bitcoin blockchain[6]. The concept of hash locking is primarily applied in the payment domain and is currently more suitable for scenarios involving partial assets or critical data exchange rather than cross-chain asset transfers. Examples of its application include the Lightning Network, Raiden Network, and cross-chain asset transfer protocols like Interledger. In the same year, the Ripple team released the Interledger Protocol white paper, which is built upon the cross-ledger interoperability protocol. They used a notary mechanism to make it possible for assets to be converted between different ledgers. In 2016, Vitalik Buterin published "Chain Interoperability, " which provided a comprehensive analysis of blockchain interoperability issues[7]. In this work, they summarized previous interoperability efforts and proposed three main categories of cross-chain protocols as solutions: notary schemes, sidechains & relays, and hash-locking. In 2017, Fusion proposed utilizing a distributed private key control model for cross-chain transaction processing, achieving a separation of ownership and usage rights, enabling all tokens to have interoperability on the same blockchain.

Most existing cross-chain bridge projects are based on the four mainstream technologies mentioned earlier. In May 2016, ConsenSys launched the BTC Relay project[8], which was touted as the world's first sidechain. It used Ethereum smart contracts to allow users to verify Bitcoin transactions. Kwon et al. introduced Cosmos to support interoperability between heterogeneous blockchains[9]. In 2018, Polkadot[10] released the first version of the Polkadot Testnet, supporting a multi-chain architecture with different consensus systems. Cosmos and Polkadot are both classic examples of sidechain/relay chain architectures. Cosmos achieves cross-chain connectivity through the IBC protocol, while Polkadot implements cross-chain functionality through slots based on the XCMP protocol. In 2021, THORChain[11] unveiled the THORChain cross-chain interaction protocol, enabling permissionless cross-chain transactions for various cryptocurrencies.THORChain is a cross-chain bridge project based on a notary mechanism, and there's another well-known distributed ledger technology, Ripple[12], that employs a similar notary mechanism to facilitate asset exchanges among global financial institutions. In the same year that FUSION introduced distributed key control technology, Wanchain[13] utilized this technology along with threshold key sharing schemes to facilitate cross-chain transactions among various public blockchains. The field of cross-chain bridge projects is continuously evolving, and in 2022, Mutichain[14] (formerly known as Anyswap) introduced a universal message-based cross-chain protocol called anyCall in the data exchange layer. It has been continuously updated and has reached version 7 as of March 2023, integrating top applications like Curve Finance. In 2023, LayerZero introduced the first cross-chain bridge, Stargate Finance[15], designed to solve the "bridging trilemma. " This protocol functions across multiple chains, establishing a unified liquidity pool for seamless communication between different blockchains.

Today, cross-chain bridges have become a new source of interest for hackers. Cross-chain bridges facilitate the transfer of funds across two separate blockchain platforms, enabling token swaps, smart contract execution, data transmission, and other instructions between them. According to data from Dune Analytics[16], the total value locked (TVL) in major cross-chain bridges on Ethereum is approximately $5. 56 billion. Currently, Polygon Bridges has the highest TVL at $2. 949 billion, followed closely by Aritrum Bridge at $1. 206 billion, and Optimism Bridges ranking third at $834 million. However, despite the enormous potential of cross-chain bridges, they face security challenges. The cross-chain bridge ecosystem is relatively new, with its development starting in early 2021. Additionally, cross-chain protocols often involve interactions with multiple blockchains and contracts, making the processes complex and introducing various potential points of risk. According to the report[17], in 2022, cross-chain bridges experienced 15 cyber attacks, resulting in losses of up to $1. 92 billion, which accounted for 59% of the total stolen funds in the DeFi ecosystem in 2022.

These cyber attacks have raised concerns about the security of cross-chain bridges and related projects. Large-scale security breaches not only damage the reputation of the entire blockchain and cryptocurrency industry but also have a negative impact on the cross-chain ecosystem. To prevent similar issues from occurring again, both the academic and industrial communities have conducted focused research efforts. Zhang[18] summarized the operational procedures of existing cross-chain bridges and developed a tool called XScope. This tool can identify malicious transactions involving three types of vulnerabilities:

Unrestricted Deposit Emitting(UDE), Inconsistent Event Parsing (EIP), and Unauthorized Unlocking (UU). Quantstamp Lab[19] summarized cross-chain bridge attack incidents from the past two years and categorized the attack surface into Custodian Attacks, Communicator Attacks, Debt Issue Attacks, and Token Interface Attacks. Several blockchain security companies such as SlowMist Technology[20], CertiK[21], and others are actively monitoring cross-chain bridge attack incidents and promptly responding by providing comprehensive and detailed traceability reports. In the NONEAGE[22] company's research report, cross-chain bridge attacks are categorized into cyber attacks, private key theft, asset theft, information leakage, and incorrect permissions. Recommendations for addressing multiple attack incidents include contract auditing, interface call inspection, and security reevaluation after updates. PeckShield[17] company classified security events based on hacker attack principles, including Lack of Input Validation, Business Logic, Signature Validation, Compromised Key, and Front-end Exploit. In our article, we conducted a more detailed classification of cyber attacks on cross-chain bridges based on the attack principles observed in real-world events, aiming to cover as many attack scenarios as possible.

We will conduct an in-depth exploration of the principles underlying cross-chain bridges. We will categorize these bridges based on the various validation methods employed within their trust layers. Drawing insights from a multitude of real-world security incidents, we have identified four categories of vulnerabilities associated with cross-chain bridges. These include verification bypass vulnerabilities, asymmetric processing mechanisim vulnerabilities, key theft vulnerabilities, and diverse vulnerabilities. We will create specific vulnerability models for each category and provide security recommendations aimed at assisting blockchain users and companies in enhancing their security awareness.

2 | BRIDGE ARCHITECTURE

The architecture of a blockchain cross-chain bridge typically includes three main parts: the source chain, the target chain, and the cross-chain component. Figure 1 shows the cross-chain bridge architecture model between blockchain A and blockchain B, where the user intend to exchange information and transfer data between two independent blockchain systems, including well-known actions like asset transfers and the representation of changes in account balances through data. The cross-chain contracts in blockchain A and B mainly handle the cross-chain business logic. The cross-chain contract on source chain A performs the asset locking and transaction initialization functions, while the one on chain B processes the cross-chain information from the source chain, conducting asset minting or the specified operation. The cross-chain component is the core part, including a transmission module and a trust module. The transmission module's duty is to transmit cross-chain messages from the source chain to the target chain. This process might involve relayer nodes (helping users transmit cross-chain messages, passing the source chain's block headers/latest block headers to the target chain, to establish light node contracts)[23], etc. The trust module's role is to verify the authenticity of cross-chain messages, ensuring that only validated messages are transmitted to the target chain.

Suppose there's a cross-chain DeFi application, and a user wants to transfer a token (like usdt) from Ethereum to Polygon. In this case,the cross-chain business contract locks the user's USDT on the source chain (Ethereum) and specifies the receiving address (Addr_B) for USDT on the Polygon chain (B chain). After the instructions and data information are determined by the transmission module and verified, an equivalent amount of Polygon assets in the form of USDT tokens is unlocked on the Polygon chain at Addr_B. The transmission module would transmit information about the user locking USDT and their target address on Polygon from the Ethereum chain to the Polygon chain. The validator in the trust module needs to confirm whether the USDT on the source chain has been locked by reading the block header information on the source chain and verifying the proof of the cross-chain transaction (like a Merkle proof). Once the verification is successful, the message will be passed on to the target chain, and the business contract on the target chain will perform the corresponding operation according to the message content. The transmission module and trust module are not entirely independent. For example, an oracle has both verification and transmission responsibilities. It verifies data from outside the blockchain and then provides a service to transmit off-chain data onto the blockchain. In addition, different cross-chain validation bridges employ varying verification mechanisms, allowing blockchains to perceive and trust each other in different ways.

Arjun Chand[24] and Andre Cronj[25] have categorized data transfer methods in cross-chain bridges differently. While we primarily discussed the classic lock/burn and mint methods when introducing cross-chain bridge architecture, it's important to note that there are also two other types: atomic swaps and liquidity swaps.Atomic swaps are based on Hash Time-Locked Contracts (HTLC). HTLC is a one-way and low-collision technology that relies on a hash function. In the context of atomic cross-chain bridges, assets can be accessed using private keys. In the event of malicious transactions, these assets can be retrieved through a time lock, without the need for a trusted third party. Among the three types of cross-chain bridges, this is the most

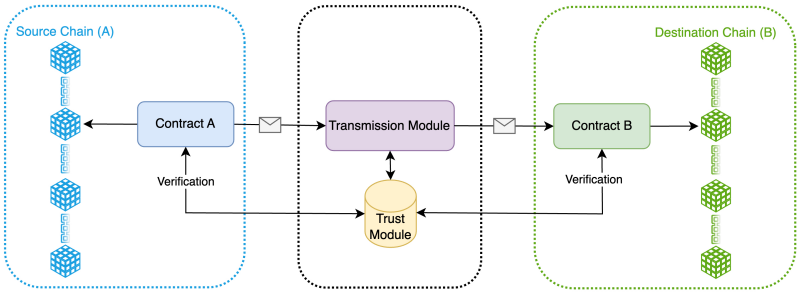

FIGURE 1 Cross-chain Bridge Architecture Model

secure method. However, it requires that the source chain and the target chain use the same algorithm and are compatible with HTLC. The liquidity pool-based cross-chain bridge, built on the foundation of lock-and-mint/burn cross-chain bridges, incorporates liquidity pools to enhance cross-chain speed. In the original example, when a user transfers USDT from Ethereum to Polygon, the USDT is placed in the liquidity pool on the source chain. Then users on the target chain can exchange assets (USDT or other tokens) from the liquidity pool. The exchange rate is obtained through an oracle. To incentivize users to deposit their assets into the liquidity pool, cross-chain bridges often provide staking rewards. However, this also exposes LPs to significant risks. In the case of the Wormhole cross-chain bridge, which employs a lock-and-burn/mint mechanism and deploys liquidity pools, a security incident demonstrated that if a hacker attack occurs, resulting in the theft of the liquidity pool on the source chain, the liquidity pool becomes unbalanced, causing the value of wrapped tokens on the target chain to evaporate suddenly. To enhance liquidity and facilitate asset trading, liquidity pool-type cross-chain bridges often adopt another, more widespread operation: that is, the cross-chain bridge itself issues a specific type of cross-chain asset, serving as a trading medium between assets on the source chain and target chain.

The data transfer methods in cross-chain bridges are diverse, and the validation mechanisms vary between different types of cross-chain bridges. The trust module of a cross-chain bridge primarily focuses on verifying the authenticity and reliability of communication information. For natively verified bridges, the trust module consists of on-chain light client programs and the Head Relayer responsible for transmitting block headers. For externally verified bridges, the trust module is a Proof-of-authority(PoA)/Proof of Stake(PoS) network or an external Oracle network. For locally verified bridges, the trust module is the transaction process itself. For optimistic verification bridges, the trust module comprises PoS validators and observers responsible for reporting validator fraudulent behavior.

By deeply analyzing the architecture and workflow of cross-chain bridges, we reveal the close connections between various components and their important roles in transferring assets and data between blockchain networks. In the next section, we will further study the verification methods of cross-chain bridges, their security, and potential attack threats.

3 | CROSS-CHAIN VERIFICATION METHOD

Regardless of how cross-chain technology evolves, it cannot bypass the role of the "middleman" because blockchains are closed and independent from each other. Each chain is a "Walled Garden", unable to directly perceive the transactions and states of others. For one chain, another chain is considered an off-chain system. Thus, one chain's perception of another is essentially an oracle problem. During cross-chain transactions, before the target chain issues mapped assets, the "middleman" needs to undertake the function of passing and verifying cross-chain messages. The core is the verification mechanism to ensure the credibility of the messages.

We introduce a cross-chain analysis framework proposed by Arjun Bhuptani[26] classified bridges into three types based on how they are verified: natively verified, externally verified and locally verified. We will compare and analyze the three types of cross-chain verification bridge projects mentioned above, along with the optimized version of external verification known as Optimistic Verification.

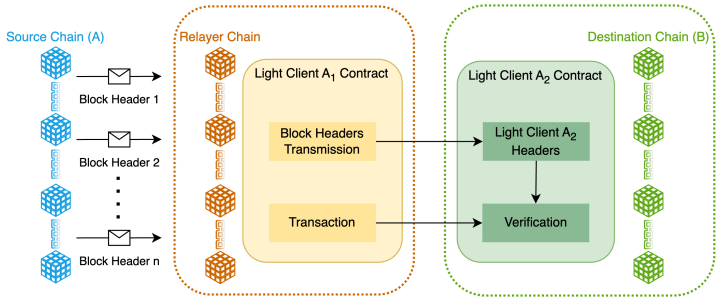

FIGURE 2 Natively Verified Bridges Architecture

3.1 | Natively Verified Bridges

Native Verified Bridge requires deploying lightweight nodes from both the source and target chains on each other's chains. They perform Simple Payment Verification(SPV) to compare hash values, thereby confirming the authenticity of the transaction. SPV verification is a method to verify payments without running a full node. When transferring messages from Blockchain A to Blockchain B, a lightweight node from Blockchain A is deployed on Blockchain B to verify the messages passed from the source chain. A lightweight node stores only the sequence of block headers containing encrypted digests of complete data in the blocks. As shown in Figure 3, when initiating an information transfer request, the relayer carries the block height from Chain A, SPV proof, and other information into Chain B. The deployed lightweight node contract on Chain B, through SPV verification, recalculates the block header hash value and compares it with the lightweight node's hash value. This is done without downloading the entire historical record of the blockchain. If they match, it confirms that the transaction indeed occurred in that block. Furthermore, Benedikt Bunz introduced Flyclient[18], which is an ultra-light client capable of synchronizing new block headers or continuously pruning old ones. In an ultra-light client, the latest block header can include commitments for all historical block headers and can verify the finality of block headers through a set of probabilistic sampling algorithms. However, in cross-chain bridges, Flyclient ultra-light clients require high gas for continuous synchronization of the latest block headers and substantial computing resources for the sampling algorithm, making it unsuitable for use in cross-chain bridges.

Waterloo Bridge, developed by the Kyber Network, is the first cross-chain bridge to implement bidirectional natively verified bridges. It establishes bidirectional bridging between Ethereum and EOS, creating light nodes through their respective smart contracts. Given the block production speed and resource quantity of EOS and Ethereum, the Ethereum light node contract built on EOS adopts the design of the SPV light node (similar to the working principle of BTC Relay), synchronizing the Ethereum block headers one by one into the light nodes. On the other hand, the EOS light node contract designed on Ethereum only synchronizes blocks where BP changes occur. When there is a message on EOS that needs to be transmitted cross-chain to Ethereum, Waterloo Bridge will first check whether the block header of the block where the message resides already exists in the light nodes. If not, it retrieves the block header from a full node, then the light node verifies it according to the latest BP set. If it already exists, the message is executed for SPV verification using the verified block header.

Rainbow Bridge, developed by Near Protocol, is a cross-chain bridge to connect the Near and Ethereum ecosystems. Its infrastructure includes two on-chain contracts and three off-chain relays. The on-chain contracts are lightweight nodes deployed by both parties, while the off-chain relays are responsible for passing block headers from their respective chains to the corresponding clients and issuing challenges for invalid block headers. In addition to native verification, the lightweight nodes of Near clients deployed on Ethereum also use an optimistic verification solution. When a relay submits a block header to the Near client deployed on Ethereum, it requires collateral from the Near chain, and then the Watchdog challenges it within a specified window. If there are no challenges within the window, the block header is accepted. Otherwise, the Near2EthRelay that submitted the invalid block header will face economic penalties. The Rainbow Bridge effectively reduces the risk of service interruptions, allows multiple relayers to operate simultaneously, and compete and back up each other.

Natively verified bridges offer high security, but they come with high costs due to the need for running lightweight clients of all interacting chains on a single chain and performing SPV verification for events, which results in substantial gas consumption. The development cost of supporting new chains is also substantial. To be compatible with a blockchain, it is necessary to build at least one pair of lightweight clients, which makes scalability challenging.

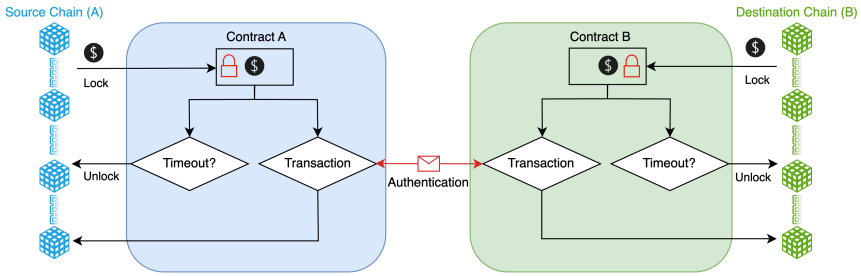

FIGURE 3 Externally Verified Bridges Architecture

3.2 | Externally Verified Bridges

As shown in Figure 4, externally verified bridges introduce one or a group of external validators, or Third-Party Exchanges, to verify cross-chain information. In cross-chain transactions, when users send assets to a specific address on the source chain, the assets are temporarily locked. Third-party validators verify it and reach a consensus internally through some mechanism, and the target chain generates the corresponding assets. External validators take various forms, such as PoA-based Multi-Party Computation(MPC) networks (Multichain, Wormhole), PoS-based MPC networks (Axelar, Hyperlane), Oracles (LayerZero, CCIP), and TEE networks (pNetwork, Bool Network). Regardless of the form, their main tasks are to verify cross-chain messages and confirm their authenticity. The security of externally verified bridges is limited by the security of the source chain, target chain, and the set of external validators. In most cases, the security of the external validation set is the weakest link among these three factors. That is, the security of cross-chain message transmission is primarily constrained by the security of the external validation set.

Multichain Bridge, originally Anyswap, adopts a multi-chain interaction overall architecture called MBI (Multiple Blockchain Interaction). MBI is divided into three layers from top to bottom. The first layer is the application layer, such as commonly used asset cross-chain bridges. The second layer is the data layer, to which the cross-chain messaging protocol anyCall belongs; the third layer is the trust layer, which includes the multi-party trust mechanism fastMPC Network (based on Multi-Party Computation (MPC) technology, which allows multiple groups of users to jointly compute a function with their private data as inputs, and all users can only get the output of the function without any other information) and zkRouter (a chain inter-trust mechanism built using Zero-Knowledge Proof (ZKP) technology). anyCall is a decentralized cross-chain message passing protocol on multichain that can send any data, such as smart contracts, NFTs, messages, tokens, etc. , to any other blockchain. Anycall is based on SMPC Secure Multi-Party Computing + TSS Threshold Signature technology, consisting of a node network of Distributed Control Rights Management (DCRM) and cross-chain bridge smart contracts, and adopts the AMM automated pricing and liquidity mechanism. The SMPC network consists of 24 nodes, these nodes are responsible for listening to pending messages in the source chain [anycall] contract, signing and relaying to the target chain's [anyExec] contract. Dapp realizes the sending and receiving of cross-chain messages by interacting with the anycall/anyExec contract. SMPC node members do not need to pledge, and are relatively fixed, AnyCall's security is based on the trust assumption of SMPC nodes.

Wormhole Bridge,[27] is a versatile AMB bridge that supports the transmission of any message between 14 heterogeneous public chains. It utilizes the PoA (Proof of Authority) guardian network as an oracle and employs a permissionless relay network to transmit messages cross-chain. These guardians are specific entities with capital and reputation endorsements, all running full nodes for each chain supported by Wormhole and listening to messages sent out by Wormhole core contracts (such as Core Bridge Contract) on each chain. The cross-chain message format on this bridge is called VAA (Verifiable Action Approval), the guardians verify and sign these messages, and then pass them to each other on the P2P network. Once a message receives signatures from more than 2/3 of the guardians (at least 13), it is forwarded to the target chain. The members of the guardians are relatively fixed, if a replacement is needed, it will be completed through governance voting. In the Wormhole architecture, a completely trustless relay network is allowed to publish messages on the target chain because these messages are signed by the guardians, so the message content cannot be changed or censored, as anyone can run a relay to submit any information. As of July 2023, this cross-chain bridge has experienced three large-scale hacking incidents, with losses as high as $650M[?] .

Externally verified bridges rely on external validators of the two interacting chains to verify and supervise the interaction content. Although they have many advantages such as efficiency and high scalability, their security is limited by external

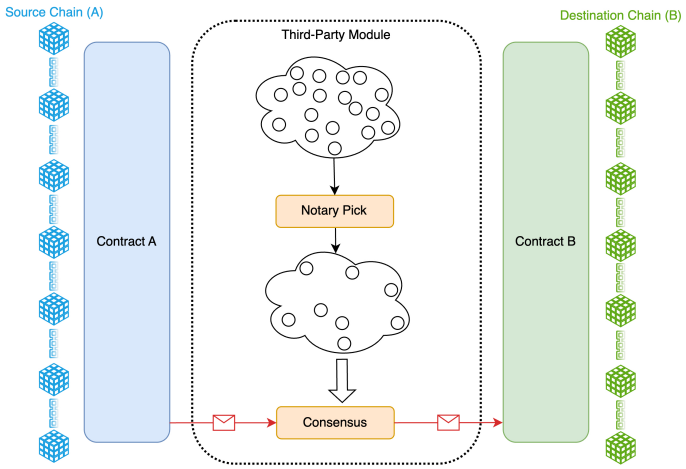

FIGURE 4 Locally Verified Bridges Architecture

validators, and there is indeed a risk of validators colluding in their security assumptions. Both of the aforementioned PoA bridges received severe hacker attacks in 2022, resulting in huge losses.

3.3 | Locally Verified Bridges

Locally verified bridges use a peer-to-peer liquidity network model, as shown in Figure 5, where the source chain Block Chain A and target chain Block Chain B trade original assets and directly verify. The transaction verification speed is fast, the cost is low, and the capital efficiency is higher than externally verified bridges, but lower than natively verified bridges. The classic paradigm is atomic swap based on hash time lock. Models of locally verified type include Hop, Connext, Celer, Liquality, etc.

cBridge, a high-speed, low-cost cross-chain payment network, allows users to transfer value within or between any Ethereum Layer 2 system. It seamlessly connects the Ethereum mainnet, along with other Layer 1 or Layer 2 systems, eliminating the need for Ethereum transit cross-chain. This, in turn, significantly shortens the cross-layer time from Layer 2 to Ethereum, enabling users to complete cross-layer interactions quickly. The entire cross-chain process of cBridge is safeguarded by Celer's SGN (State Guardian Network), a decentralized PoS blockchain, ensuring that all transfers are securely carried out. In contrast to common multi-signature cross-chains, cBridge uses a special Hash Time Lock Contract (HTLC) for its funds transactions, making its security completely independent of third parties, and it does not suffer from security issues such as multi-signature validator authorization misconfigurations and private key leaks. Nevertheless, in 2022, cBridge's underlying infrastructure was hacked, resulting in a loss of $190M[28].

Despite the dual advantages of trustlessness and easy multi-chain adaptation, local validation cannot achieve universality, and it only supports Swap bridges for asset cross-chain transmission and cannot support Wrap bridges and AMB bridges for generic information and data inter-chain transmission.

3.4 | Optimistic Verification Bridges

As shown in Figure 6, optimistic verification bridges are a new choice that improves trust assumption mechanisms based on externally verified bridges and break the cross-chain impossible triangle[29]. Arjun Bhuptanix proposed the cross-chain bridge interoperability trilemma, which includes: Extensibility (supports arbitrary message passing), Trustlessness (does not introduce new trust assumptions), and Generalizability (can easily adapt to more blockchains). No matter whether it is locally verified bridges, externally verified bridges, or natively verified bridges, it cannot fully meet the above three requirements. Optimistic verification bridges set up updater proxies that have pledged some assets to sign a merkle root for information posted to contract functions, allowing any relay system to read this root. They set up challengers and a continuous 30-minute anti-fraud window. The Watcher is responsible for monitoring activities on the source chain and the target chain to ensure that all transactions are

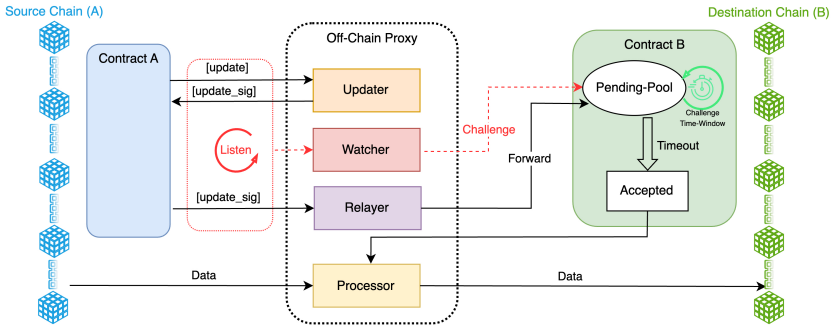

FIGURE 5 Optimistic Verification Bridges Architecture

processed correctly, not overlooked or misprocessed. If an incorrect validation is observed, it can be challenged and a fraud proof provided. If the challenge is successful, the updater's collateral will become a bounty for the challenger.

Nomad Bridge, a typical optimistic verification bridge model, employs an optimistic-rollup mechanism. It enables users to safely send messages and bridge assets.[30]. Its on-chain smart contracts include two parts, Home and Replica, which implement the logic of message sending and receiving for Nomad. Off-chain agents include four roles: Updater, Watcher, Relayer, and Processor. The Updater mainly listens to the Home contract on the original chain, signs the new root value generated by the Home contract, generates the corresponding proof (including the proof of the previous root and the new root), and sends it back. The Watcher ensures the security of the Updater by listening to the interaction between the Updater and the Home contract, submitting the Updater's malicious or incorrect verification. The Relayer is responsible for forwarding the update messages sent by the Home contract to many Replica contracts. The Processor is responsible for verifying the validity of the messages to be processed and sending them to the final recipient. Applications queue messages T waiting to be cross-chained in the source chain (Home) contract. The source chain contract calculates the new message root, _newRoot, which includes the new message T, and saves the previous message root_oldRoot. The Updater calls the update function of the source chain contract, signs the new and old message roots, and returns them to the source chain contract. This is known as the update operation, which may be carried out individually or in batches to save costs. The Relayer passes the Updater-signed update to the target chain, creates a pending update in the target chain's Replica contract, and begins the dispute window period. During the dispute window, the Watcher checks for fraudulent behavior, ensuring that the updates on the source chain and the target chain are consistent. If no fraudulent behavior is reported, the Processor gets message T and its Merkle path from the source chain, passes it to the target chain, and verifies the new message root's inclusiveness for message T in the target chain's Replica contract. After verification, the Processor removes the message from the pending pool and sends it to the target application. If fraudulent behavior occurs (for example, updates on the source chain and the target chain are inconsistent), the Watcher needs to submit a fraud proof to the target chain's Replica contract and to the source chain's Home contract within 30 minutes, triggering penalties for the dishonest Updater, including slashing the collateral and suspending the processing of new pending messages. Despite the fact that Nomad Bridge is a new choice to overcome the cross-chain impossible triangle, it was still hacked due to contract configuration errors in 2022, resulting in a loss of $190M.

Now, optimistic verification bridges are actively improving the defects caused by the delay of the challenge window period, by combining with other verification methods, such as the on-chain contract NearOnEthClient of the Rainbow Bridge mentioned earlier, which combines natively verified bridges, and Celer IM which combines externally verified bridges, giving users the balance between security and efficiency.

We compared the four types of cross-chain verification bridges across nine aspects in table 1 and teble 2. In summary, native verification has the smallest trust assumption but the highest cost for multi-chain adaptation. External verification has advantages and disadvantages opposite to native verification. It has lower costs for multi-chain adaptation but introduces new trust assumptions. To reduce these trust assumptions, it requires increasing cross-chain costs and having validators stake tokens. Local verification, despite having the advantages of no trust assumptions and easy multi-chain adaptation, is limited in its applicability as it can only support Swap bridges. Optimistic verification bridges are often combined with other verification methods, allowing users to independently balance security and efficiency.

TABLE 1 Composition of Chains(not combine).

	Native	**Locally**	**External**	**Optimistic**
Third-Party Witness	Not required	Not required	Required	Required
Light Nodes	**Required**	Not required	Not necessarily	Not necessarily
Applicability	Wrap&Swap	**Swap**	Wrap&Swap	Wrap&Swap
Trust Components	Light Clients and Head Relayer	Transaction itself	PoA/PoS Networks/Oracle	Watcher
Incentive Targets	Relayer	Public Counterparties	Validator	Watcher
Transport Layer	On-chain Smart Contracts & Message State Updates	On-chain Smart Contracts & Message State Updates	On-chain Smart Contracts & Message State Updates	On-chain Smart Contracts & Message State Updates

TABLE 2 Performance Comparison.

	Native	**Locally**	**External**	**Optimistic**
Development Cost	**Highest**	High	Low	High
Generality	Poor	Poor	**Good**	**Good**
Trustworthiness	**Best**	Good	Affected by Third-Party Witnesses	Fairly good

4 | VERIFICATION BYPASS VULNERABILITY

This section will introduce the *Verification Bypass Vulnerability* that occur on the source/target chain of the cross-chain bridge. Verification Bypass refers to vulnerabilities caused by insufficient return value checks, verification flaws, and inadequate validations in the process of asset transactions or information transmission on the cross-chain bridge. This section will summarize the Verification Bypass model from the perspective of hackers exploiting vulnerabilities, based on 7 real cross-chain bridge attack incidents.

4.1 | Over-Refund Vulnerability

The *over-refund vulnerability* is a type of security issue where hackers carefully manipulate attack contracts to disrupt regular transactions, causing the automatic unlocking of funds for service providers, liquidity pools, and the source chain. However, attackers falsify the data in the funds unlocking request or replay refund vouchers, resulting in a refund amount that exceeds the actual amount.

The basic architecture of cross-chain bridge asset transactions involves a user submitting a transaction request on the source chain. This is then validated and processed via smart contracts, transmission modules, and trust modules, generating corresponding assets on the target chain. During this process, certain special situations may arise, such as transactions failing due to not meeting certain conditions of the smart contract, transactions timing out because the user's submitted transaction hasn't been processed within a certain time frame, errors in the contract, network faults, etc. , that lead to the transaction entering the refund logic.

Different types of cross-chain bridges have different refund logics. In sidechain/childchain type with unilateral or bilateral locking + minting mechanism, the refund is automatically unlocked by the original chain's smart contract. In federated bridge type, "federation members" or "intermediaries" unlock assets on the source chain. In centralized cross-chain bridges, a centralized entity or service provider operates it, and if the asset issuance fails, the service provider will refund the user's original assets. All of these refund logics can present various security issues. Atomic swap requires users to be online and actively participate, otherwise, assets may be lost. Sidechain/childchain type, federated bridge type, and centralized cross-chain bridges may face trust and security issues with federation members, sidechain/childchain networks, and service providers. Besides, attackers can exploit *Over-Refund Vulnerability* or use the form of a refund to steal assets.

Exploiting Over-Refund Vulnerability. Hackers meticulously construct attack contracts, causing the cross-chain node program to be unable to process, thereby entering the refund process. In 2021, THOR Chain was attacked three times in a row, and in the third attack, the hacker exploited a *Over-Refund Vulnerability*, causing $8M in losses. The attacker disguised the

attack contract as an address similar to the real Asgard address. Asgard is a special multisig wallet address used for storing and managing user assets. It serves as a node in the node network, handling asset transfers and transactions. When the THORChain Router contract sends Ether to this address, it triggers an event called "deposit". Attackers can exploit this incident to create arbitrary assets and amounts and construct inappropriate "memos" (additional information or messages attached to a transaction that allows for extra interaction or communication between the sender and receiver). This can lead to THORChain node programs mistakenly processing the transaction, triggering the refund logic, and refunding the fabricated amount of funds created by the attacker.

Polygon Plasma Bridge Incident: Acquiring illegal funds through the form of a refund. The general process for a cross-chain bridge transaction refund is as follows: The user initiates a Withdraw transaction to destroy tokens. After the checkpoint interval, this transaction is included in the checkpoint. After a majority of validators sign, this transaction is submitted to the network. After system validation, an NFT refund voucher is minted for the user. After waiting for a period, the user destroys this NFT and completes the refund. NFTs uniquely identify a Withdraw transaction's refund voucher. However, due to the flaw in the NFT ID generation method, an attacker can generate multiple NFTs with different IDs for the same Withdraw transaction, then use these NFTs for multiple refunds, realizing a double-spending attack. In 2021, Polygon Plasma Bridge suffered from such an attack, causing $850M in losses[31]. Polygon Plasma Bridge adopts a combination of optimistic validation bridge and external validation bridge, which provides higher transaction efficiency and security level, but still manages to steal assets by bypassing the transaction validation process. The main source of this vulnerability lies in the addExitToQueue() method, which mints an NFT, its ID generated by the Plasma Bridge's age priority. When initiating a refund transaction, it relies on the parameter data to decode branchMaskBytes. One function transcode branceMaskBytes for validating the refund transaction, generating the corresponding age. However, this transcoding operation has a flaw. It cannot ensure the complete difference of the transcoding values, thus leading to the attacker being able to construct parameters and successfully execute a double-spending attack.

In *Over-Refund Vulnerability*, attackers deliberately construct transactions that do not meet the conditions of smart contracts, thereby triggering processing exceptions and refund logic, and with the help of other attack vulnerabilities, they use the form of a refund to steal assets.

4.2 | Phantom Function Vulnerability.

Attackers design attack contracts that call a non-existent function in the cross-chain bridge, triggering a fallback function. Through this special route, they bypass legitimate validation and conduct asset transfers. As the technical standards and protocols in the blockchain field are constantly changing, new chains and new standards are constantly emerging, which requires cross-chain bridges to be updated in a timely manner. Otherwise, they may not be able to support the latest chains and standards, and thus lose market competitiveness. This includes Ethereum's Polygon, Binance Smart Chain's PancakeSwap, Polkadot's Darwinia, etc. , all of which are constantly iterating and optimizing their products. They hope to attract more users by improving transaction efficiency, reducing handling fees, and increasing the number of supported chains. However, this rapid iteration and update requirement also brings some security issues. The most intuitive one is the cross-chain bridge *phantom function vulnerability*, which originates from the interaction effect of the execution environment of smart contracts and the characteristics of object-oriented programming (OOP). In smart contracts, when a non-existent method is called, the contract's fallback function is triggered. If developers do not fully consider this when designing smart contracts, they may introduce security risks, allowing attackers to exploit this feature for attacks.

Multichain Security Incident: Multichain suffered a *phantom function vulnerability* type of hacker attack in 2022, losing $1. 8M. The attacker forged an invalid token address and exploited the feature of the fallback function, bypassing the legitimate verification of the incoming parameter address, and thus implementing illegal asset transfer. First, the attacker constructed a malicious ERC20Token tokenX and set its underlying address to the contract address of WETH held by legitimate users. Then, the attacker set the token parameter to the malicious tokenX address, and set the amount parameter to the balance of the legitimate user's WETH. Because the function did not effectively verify the incoming token address during execution (i. e. , it defaulted to a legal AnySwapERC20 type of token contract address), all subsequent verifications such as "underlying" defaulted the legality of the token address. If an attacker exploits this, they can bypass the permit logic, causing the function to transfer the WETH authorized by the legitimate user to the router contract into the maliciously constructed tokenX address. Because the function does not check the return value of the permit function call, the attacker can use the feature of the fallback function to completely skip the verification process, eventually achieving illegal asset transfer. The core of this vulnerability attack is that

the underlying token contract of the protocol call does not implement the permit method, but contains a fallback function, so the contract that calls the permit method runs normally and successfully bypasses the check.

Nomad Bridge Security Incident: In the Nomad Bridge attack case, the core of the vulnerability exploit lies in the incorrect deployment of function parameters, and the attacker uses the special nature of the 0x00 address to bypass the verification mechanism. The attacker mainly exploited a bug, that is, during the initialization process, the developer set the "A" parameter to 0x00. The process() function calls "function B" to check the submitted root and whether time lock has expired. When executing the process() method, the value of "A" is still zero, and due to the incorrect parameter deployment, 0x0000 is set to true during initialization, and the return value of acceptableRoot is True, indicating that this message has been approved, making all messages default to valid. The attacker can then transfer the funds of the Nomad bridge by sending constructed messages, which will not be intercepted by other verification mechanisms of the smart contract. In the design and deployment of smart contracts, each function and parameter needs to be fully reviewed and verified to prevent phantom function vulnerabilities in cross-chain bridges caused by not being updated and iterated in time, to prevent attackers from exploiting these vulnerabilities for attacks, and to ensure the safety and reliability of smart contracts.

Wormhole Security Incident: Wormhole was also subjected to a phantom/ghost function vulnerability attack, where a deprecated function was used in the transaction process, which led to the inability to correctly verify all input accounts, which could deceive the guardian to get a signature. There is a role similar to Keeper in the cross-chain bridge, called the guardian, who is responsible for verifying cross-chain transactions. In each cross-chain operation, the guardian will sign the operation, and only when the majority (2/3+) of the guardians sign the same, will the operation be executed. However, there was a problem with the smart contract of Wormhole in the process of verifying the signature. It used a function called verify_signatures to verify the signatures of the guardians, but this function did not really check the signature; instead, it outsourced this task to another function responsible for checking if the Secp256k1 function was called first. However, this function was deprecated in January 2022 because it didn't check if the signature verification was performed by a whitelisted address, also known as a "system address. " In the end, the attacker replaced the "Sysvar: instruction" address with his own provided address, bypassed the verify_signatures signature check, and made the false transactions executed.

The phantom function is a common vulnerability in smart contracts, including the logic of using fallback functions, functions not implemented as expected, not deleted in time after deprecation, not checking parameters, failing to detect signatures, returning useless information, etc. , which leads to attackers being able to bypass the condition detection in the contract in some way to achieve their goals.

4.3 | Impersonation of Validator's Identity Loophole

In cross-chain transactions, to ensure the security and correctness of the transaction, a group of validators are usually needed to verify the transaction. Validators are a group of nodes trusted by the community, with the rights to sign and verify cross-chain transactions. The *impersonation of validator's identity loophole*, in simple terms, is when the attacker falsifies or tampers with the identity of the validator by some means, obtaining the same rights as the original validator, such as signing and verifying transactions. This type of vulnerability often stems from flaws or oversights in system design, such as lax verification of validator identities, or ease of identity tampering. Once the attacker successfully impersonates the validator, they can carry out malicious operations in the name of the validator without detection, such as executing illegal transactions or stealing assets. For example, if a system relies solely on a public key as the unique identity of a validator, once an attacker gets hold of this public key, they can sign and verify transactions as the validator. Worse still, if the system does not rigorously verify signatures, or allows validators to change their own public keys, then attackers can potentially replace the validator's public key, and thereby completely take over the identity of the validator.

ChainSwap Security Incident: The ChainSwap cross-chain[32] bridge attack led to an $8M loss, with the core of the vulnerability being that the receive function did not check for the legality of the incoming parameter Signatory, nor did it verify the value of the mapped index, allowing any user to impersonate the validator and provide signatures. ChainSwap cross-chain bridge sets a quota for each validator (Signatory), and as events accumulate, the amount of transactions each validator verifies over a period of time is limited. During this attack, ChainSwap did not perform legality verification for the validator during transaction verification. In the receive function, ChainSwap verifies each incoming signature, including checking whether the signature array is duplicated and verifying the effectiveness of the signature through ecrecover. However, in the implementation code for reducing the authorization quota, it simply decreased the quota value for validators in the _authQuotas (Authorization Quotas) mapping. This allowed the attacker to randomly generate an address and corresponding signature, and deceive ChainSwap. Even

more strikingly, because there was a computational logic error in calculating the new quota of the validator in authQuotaOf, it returned a very large value when verification was not valid, leading to the breach of the transaction limit, and the attacker could continuously steal funds.

Poly Network Security Incident: Poly Network[33] suffered a similar type of attack, where the Keeper role on the target chain was impersonated and modified, and the attacker used the modified keeper signature to transfer assets and successfully pass verification. Ideally, the Keeper should be a group of trustworthy roles with appropriate authority to verify cross-chain transactions. In the attack on Poly Network, the attacker first constructed a special cross-chain transaction on the source chain. The purpose of this transaction was to call a specific contract function to modify the Keeper's public key. The attacker used the official Relayer to submit data on the target chain and executed the operation to replace the Keeper's public key. The attacker disguised this operation as a normal cross-chain transaction, since cross-chain transactions can call any target contract and execute any function, and therefore passed the original Keeper's check and Merkle root check. Once the Keeper was replaced, the attacker could use the new Keeper address to sign any cross-chain transaction and submit it to EthCrossChainManager for verification. Since the EthCrossChainManager only checks if the signature is provided by the Keeper and doesn't care whether the Keeper's identity has been tampered with, these tampered cross-chain transactions could all pass verification smoothly. This enabled the attacker to control all cross-chain transactions and move a large amount of assets to their own address.

The main manifestations of the impersonation of the validator's identity loophole are identity representation issues, verification shortcomings, and logical errors. Because the identity representation of the validator is too simple, and the validation function was not fully considered during setup, there was a lack of checks for the legality of the validator's identity or input parameters. Furthermore, there were errors in the computational logic when dealing with the validator's quota or other important data. This allowed the attacker to impersonate the validator, continuously bypass signature verification, and achieve their goal.

4.4 | Modeling and Solutions

Verification bypass vulnerabilities, including refund logic loopholes, phantom/ghost function loopholes, and validator identity impersonation vulnerabilities, all stem from irregularities and security deficiencies in the writing and execution process of smart contracts. As shown in Figure 7, verification bypass vulnerabilities often occur in the validator's smart contracts on the middleman or target chain. The smart contract of the middleman/target chain receives cross-chain transaction requests and verifies them. If a transaction information that does not meet the requirements appears in the request, it bypasses the verification of the refund amount and directly jumps into the error handling section, where incorrect amount information is passed into the refund module. If the transaction request verification is passed, it will enter the verification function call and execution module. This is a disaster area for *phantom function vulnerability* attacks. Attackers can bypass parameter legality verification and special 0 address bypass by calling deprecated functions and taking advantage of the characteristics of the fallback function, causing this module and subsequent verification processes to be completely skipped. In the function execution module, attackers will exploit the *impersonation of validator's identity loophole* more. Due to code loopholes in the function, there is a lack of legality verification for signatures, allowing attackers to provide signatures by impersonating validators or blatantly initiate requests to replace the Keeper's public key and have them successfully executed. The *Over-Refund Vulnerability* not only appears in the transaction request verification, but also poses a threat to replay attacks after entering the refund logic. Due to defects in the NFT ID transcoding operation, attackers have the opportunity to bypass ID verification and conduct double spending attacks.

Conditional verification vulnerabilities are mainly security issues arising from Verification deficiencies in code. When the program cannot comprehensively and correctly verify input conditions, or when there are errors in condition judgments, such vulnerabilities may occur. Attackers can bypass Verification by providing special inputs or exploiting logical errors in the program, thereby performing unintended operations or accessing unauthorized resources.

Solutions: We proposes the utilization of data flow analysis and symbolic execution techniques to identify Verification Bypass Vulnerabilities. Data flow analysis[34] is employed to understand how variable values propagate within a program during its execution, thereby establishing dependencies among variables[35]. Data flow analysis techniques aid in discovering information such as parameter input validation, authentication, and authorization within a program. This information contributes to constructing comprehensive verification conditions within the program's execution flow, thus assisting in the detection of *Verification Bypass Vulnerability*. Representative works utilizing data flow analysis for vulnerability detection include FlowDroid[36], developed by William Enck and others. FlowDroid is a tool designed for analyzing Android programs, capable of performing data flow and taint analysis to identify issues like sensitive data leakage, code injection, and abuse of permissions within Android applications. Symbolic execution[37] is also a crucial technique for vulnerability detection, as it involves analyzing

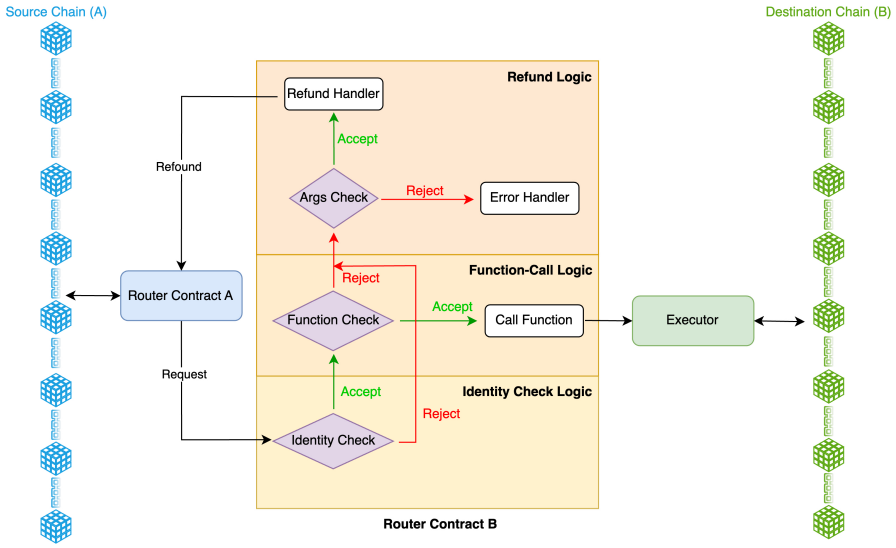

FIGURE 6 Verification Bypass Vulnerability Model

symbolic variables to explore the program's execution paths. Jiang present WANA [38], a smart contract vulnerability detection tool designed for cross-platform use. It operates by employing symbolic execution on WebAssembly bytecode. Lei Bu introduced MLBSE [39], which combines machine learning with symbolic execution. It enables symbolic representation of simple linear path conditions and non-linear arithmetic operations, significantly enhancing the capability of program analysis.

5 | ASYMMETRIC PROCESSING MECHANISMS

This section discusses the asymmetric processing mechanism attack type that occurs in cross-chain bridges between source and target chains. *Asymmetric Processing Mechanisms* refer to inconsistencies between the transaction content on the source chain and the resulting output on the target chain during cross-chain bridge transactions. The attacker can achieve this through replay attacks, *false top-up vulnerability* exploitation through parameter override, and token swapping vulnerability exploitation. This section will analyze 5 real cases of asymmetric processing mechanism vulnerabilities to reveal the general features of such vulnerabilities, as well as the common strategies and methods of attackers.

5.1 | Replay Attack Vulnerability

In a cross-chain bridge environment, a replay attack manifests as an attacker obtaining a valid transaction on a chain (source chain), intercepting and recording the transaction data, replaying the transaction on another chain (target chain), and gaining additional benefits. The risk of replay attacks in cross-chain bridges greatly increases when a blockchain hard forks[?] and produces two chains with the same transaction history. In the event of a hard fork, attackers can replay the transaction information on the original chain, causing transactions that originally only occurred once on the original chain to be repeated on the new forked chain, leading to inconsistencies in asset states between the original chain and the forked chain.

Replay attacks can be divided into two types: transaction replay and signature replay. Transaction replay refers to the attacker obtaining a valid transaction on the source chain and executing this transaction again on another chain (target chain) with the goal of gaining additional benefits. However, with the implementation of EIP 155, this attack method cannot be successfully executed on chains with different chainId. Signature replay refers to the attacker copying a message signed with a private key on the source chain and replaying it on the target chain. Methods to prevent this type of attack include adding specific chain identifiers such as chainId to the message content, ensuring that the content and signatures of messages on different chains are unique and cannot be directly replayed and reused. Having implemented ChainId-specific chain identifiers does not mean that it is immune to the threat of replay attacks from hackers. If a transaction or operation does not correctly use chainId for signature,

the chainId verification process is not strictly checked, or if the hacker can obtain the original message without chainId and its corresponding signature on the source chain, there is still a threat of replay attacks. After ETH upgraded to the PoS consensus system, the original PoW mechanism ETH chain hard forked with support from parts of the community and formed ETHW. During this hard fork, security incidents were triggered by replay attacks on OmniBridge.

Omni Bridge Security Incident: The Omni Bridge[19] attack event lost 200 ETHW. The core of the vulnerability exploitation is that the historical problem caused by the failure to handle chainId in time. OmniBridge, serving as the bridge for asset transfer between xDAI and the ETH mainnet, mainly relies on specified validators to submit cross-chain messages to complete asset transfer. The decoding of data messages includes chainId. The vulnerability lies in the fact that when checking the legitimacy of the chainId, it directly uses the value stored in the variable instead of using the EVM-native chainId opcode to obtain the chain's own chainId. Furthermore, if a hard fork occurs and chainId is not reset or set incorrectly, the actual chainId of the cross-chain message cannot be correctly verified. The OmniBridge in use at the time of the attack event was using the solidity version 0. 4. 24 and did not obtain the actual chainId through the CHAINID (0x46) operation code specified in EIP-1344, but manually stored and updated chainId. Because all states will be preserved unchanged on both chains during the hard fork process, attackers can exploit signature replay to extract assets from the same contract on ETHW.

To prevent this type of vulnerability, Solidity developers on the bridge must correctly verify the actual ChainID of cross-chain messages. On the other hand, you can also add the parameter nonce to record the number of transactions in the signature information.

5.2 | False Top-up Vulnerability

The operation of cross-chain bridge transactions is based on a 1:1 mapping relationship. As described in Chapter 3, when a user destroys/locks a certain number of tokens on one network, they can generate the same number of tokens on another network. False top-up vulnerabilities disrupt this 1:1 mapping relationship, causing an asymmetry in the message processing mechanism between the source chain and target chain, and it is not possible to ensure the consistency of assets on the two chains.

Thorchain Security Incident: Thorchain exploited the Bifrost vulnerability when making cross-chain bridge transactions. Bifrost is a bridge between THORChain and the Ethereum network, responsible for processing cross-chain transactions. The amount value in the Deposit event was replaced, resulting in a loss of 4. 9ETH. Thorchain nodes (also known as THORNode) need to collectively record cross-chain transactions. The Bifrost protocol is the connection layer between the Thorchain network and other blockchains, responsible for finding and processing transactions. Once a new transaction is confirmed, the Thorchain protocol begins the exchange process, returning Ether to the user. In this attack event, the attacker first called the deposit method of the THORChain Router contract in the attack contract, with the amount parameter set to 0. This operation in itself is not abnormal, but the attacker then initiated a transaction calling the attack contract, setting the transaction value(msg. value) to a non-zero value. Due to a defect in THORChain code, the system mistakenly replaced the correct amount value in the Deposit event with the msg. value value in the transaction when obtaining the user's deposit amount. As a result, even though the attacker did not actually make a deposit, the system mistook it for a valid deposit. The hacker repeatedly performed the above attack behavior, successfully extracting tokens from various liquidity pools.

In essence, this type of attack is due to code defects, leading to inconsistencies between the implementation of code syntax and the expected semantic behavior, giving attackers an opportunity. Formal verification and automated testing can be considered to avoid the emergence of such vulnerability exploitation.

5.3 | Token Identity Swap Vulnerability

Given the variety of tokens supported by different blockchain networks, each with different values and transaction rules, accurate identification and verification of token types are necessary when conducting cross-chain transactions. Every token has a unique identifier, such as the token name, token symbol, and token contract address. The token name and token symbol are typically easily identifiable by people, while the token contract address serves as a unique identifier recognized by machines. During transactions, the cross-chain bridge program checks the type of tokens involved in the transaction. This usually involves the use of smart contracts or other on-chain logic, referring to a token registry that lists all types of tokens available for cross-chain exchange. If the type of token in the transaction is not in this registry, the transaction will be rejected. For example, for Ethereum-based ERC20 tokens, the smart contract can identify the type of token by invoking the token contract's symbol() function. If the

token type verification is successful, the cross-chain bridge program will further validate the transaction. Token swap exploits involve attackers disguising tokens of lower value or ones that are not accepted by the system as higher-value or system-accepted tokens, thereby bypassing system verification and earning additional benefits. The following will discuss real cases that occurred on three cross-chain bridges: Meter Bridge, THOR Chain, and Qubit Bridge.

Qubit Bridge Security Incident: In the Qubit Bridge, methods for depositing standard ETH and ERC20 tokens were implemented separately. The depositETH method was used to record the source and quantity of ETH. Since ERC20 tokens require the contract's transfer or transferFrom function to transfer funds, when designing the cross-chain bridge, the deposit method calls the ERC20 token's transferFrom function to transfer the user's ERC20 tokens to the cross-chain bridge contract. In Ethereum, the execution of smart contracts triggers specific events that are recorded on the blockchain. These events are primarily used to notify listeners of changes in the state of the smart contract. However, deposits of ETH and ERC-20 share the same event proof. The event proofs generated by deposits of ETH and ERC-20 are identical, which is key to the success of the attack event. Furthermore, when the attacker calls the transfer function of the ERC20 Token, they set the ERC20 Token's address to the zero address. In the QBridge [19] contract, the zero address is also on the whitelist, meaning that deposits to this address will be accepted by the QBridge contract. This is because when depositing Eth, the default Eth address is the zero address. In the end, the attacker minted a large amount of xETH Tokens out of thin air and borrowed a substantial amount of funds from the Qubit contract.

Meter Bridge Security Incident: The attack process on the Meter Bridge was similar to that on the Qubit Bridge. The Meter.io cross-chain protocol also did not prevent the direct interaction between the wrapped ERC20 tokens and the native tokens, exploiting the deposit (ERC20 token deposit) and depositETH (WETH/WBNB token deposit) functions. When the attacker made a deposit, they passed the WBNB address into the deposit method. The deposit method did not verify the deposit status of the ERC20 tokens, and the system mistakenly believed that assets had been deposited. The attacker bypassed the judgment condition and obtained funds without making a deposit.

THOR Chain Security Incident: THOR Chain bypassed the token's detection condition by forging the token name to the Ethereum symbol ETH. The attacker designed the symbol of the cross-chain recharge ERC20 token as ETH and deployed the attack contract. Due to a logical error in the getAssetFromTokenAddress method, the symbol verification was bypassed, and the recharged token was recognized as the real Ethereum ETH.

5.4 | Modeling and Solutions

Figure 8 illustrates the vulnerability model of the *Asymmetric Processing Mechanisms*. The information transferred by the cross-chain bridge is signed with a private key and includes a chainId identifier during the transmission to the target chain. In this process, the attack contract can conduct a replay attack by replaying both the signature and the chainId. The chainId validator was originally designed to prevent such attacks. However, due to vulnerabilities in its legitimacy checks, the attack contract can successfully bypass the detection and execute an attack. During cross-chain transfers when determining the actual value, the attack contract initializes this value, setting it to 0, and then assigns a "fake value" greater than 0. Due to a coding flaw within the cross-chain contract, this "fake value" replaces the actual value. As a result, what gets transferred to the target chain is a "fake value" that hackers can arbitrarily modify. When transferring tokens across chains, attackers are skilled at disguising low-value tokens as high-value tokens. They exploit vulnerabilities in token type detection, bypassing symbol verification, causing deposited tokens to be identified as genuine higher-value tokens or non-deposited tokens to be identified as already deposited.

Solutions: This paper suggests utilizing fuzz testing techniques to detect vulnerabilities related to handling mechanism imbalances. Fuzz testing [40] is a technique that involves supplying random, invalid, or abnormal data as input to a program in order to test its behavior. It has already been applied to the detection of vulnerabilities in blockchain smart contracts. For instance, Ding proposed "hfcontractfuzzer[41], " a method based on Fuzzing technology to detect vulnerabilities in Hyperledger Fabric smart contracts. This approach combines a Fuzzing tool called "go-fuzz" with smart contracts written in the Go programming language. Therefore, by employing fuzz testing, the program's ability to parse and handle messages can be tested, leading to the discovery of vulnerabilities related to message handling mechanism imbalances.

6 | KEY THEFT VULNERABILITY

The *key theft vulnerability* involves attacks on cross-chain bridge information transmission processes. Key theft refers to the attacker's use of brute force cracking, social engineering methods, and system vulnerability exploitation methods to forge

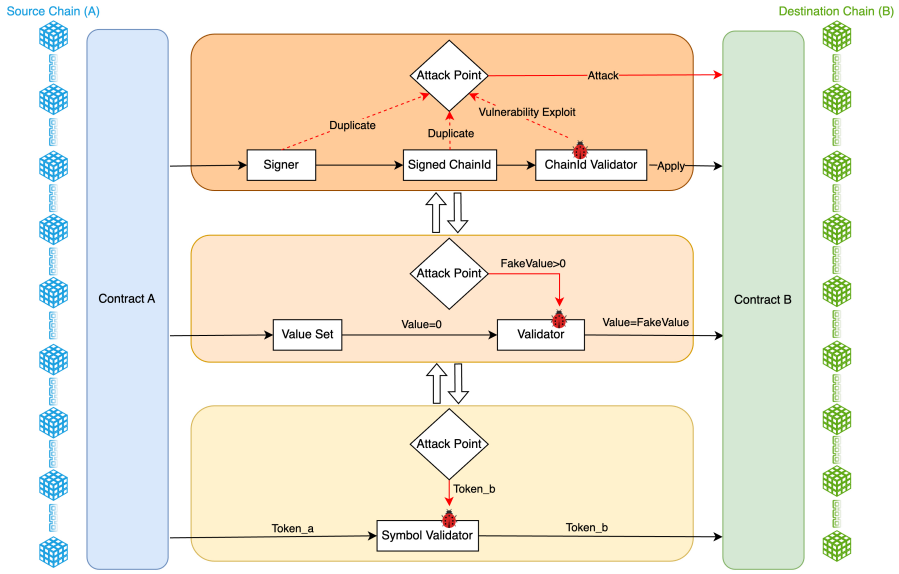

FIGURE 7 Asymmetric Processing Mechanism Vulnerability Model

middlemen/users' signatures and bypass transaction information verification during cross-chain bridge transactions, thereby achieving illegal profits.

During cross-chain bridge transactions, private keys and public keys carry out different tasks. The private keys of users and validators are used to sign transaction requests and cross-chain information, while the public keys of users and validators are used to verify signatures. Once a key leakage crack occurs, it will result in information leakage on the cross-chain bridge, asset loss, and even the collapse of the entire system.

This section will analyze four cases of key theft attacks and propose measures to prevent this type of vulnerability exploitation for reference by information security companies and workers.

6.1 | Key Cracking Vulnerability

Key cracking vulnerabilities mainly involve problems in the key generation, use, or management process. Due to defects in the key generation algorithm or design flaws in the encryption protocol, attackers can deduce the actual key value through specific calculations or reasoning. For example, if a password generation algorithm lacks randomness or uses repeated signature values, the key may be cracked.

AnySwap Security Incident: AnySwap cross-chain bridge was attacked by hackers, losing $8M. This incident showed the potential risk of reusing the random number K in transaction signatures. During the transaction process, the user's private key signed a specific subset of hash values. The signature is composed of R, S, V. R and S are the outputs of the ECDSA (Elliptic Curve Digital Signature Algorithm) signature, and V is the Recover Id used to assist in checking the signature. In ECDSA, the R value is obtained by calculating the private key and the random number k, while the S value is the calculation result containing the original message, private key, random number k, and R value. Due to an AnySwap client upgrade, the parameter K was used twice, resulting in two different transactions (different signature information) having the same R value. This allowed the random number K to be deduced, and along with the known public key, the private key could be deduced and finally used for criminal activities.

QAN Platform Security Incident: QAN Platform experienced a similar attack. Analysis of the attack event found that the address initiating the attack was the same as the creation address of the cross-chain bridge contract[42]. It is speculated that the attack was due to the key of the team or organization responsible for maintaining QANplatform being cracked. A common reason for the crack is that the attacked user's cryptocurrency address was created using insecure vanity address creation tools, for example, hackers using vulnerabilities related to vanity addresses created by the Profanity tool to steal cryptocurrency. Wintermute lost $160M due to this issue. Vanity addresses are personalized cryptocurrency addresses created based on a series

of parameters given by the user, often characterized by patterns or text in the address, similar to custom license plates on cars[43]. However, the methods used to create these addresses make them more susceptible to brute force attacks because two major security risks have already been found with Profanity. One is that it only gets 32 bits when obtaining random numbers from hardware or entropy pools; the other is that in subsequent parallel multi-round calculations, its iterative algorithm does not use one-way functions (such as hash functions).

In the implementation and management of blockchain technology, it is essential to avoid reusing random numbers in transaction signatures.

6.2 | Key Leakage Vulnerability

Key crack type vulnerability occurs due to operational errors or inherent system vulnerabilities in the process of key storage, transmission, or management, leading to the direct or indirect exposure of key information to attackers. For instance, the key is not adequately protected during storage or transmission, or there is a defect in permission management, resulting in unauthorized personnel accessing the key information or authorized managers being bribed to actively disclose key information.

Horizon Bridge Security Incident: In the Horizon Bridge attack incident, the attacker gained control of the Horizon Bridge contract project's permissions, minting a large amount of WETH without collateralizing any assets, causing a loss of $97M[44]. The attacker took advantage of the project's permissions, calling the submitTransaction() function to submit transactions, generate transaction IDs, and called the confirmTransaction() and executeTransaction() functions to confirm and execute the transaction. The attacker repeated the above process, generating different transaction IDs to transfer large amounts of assets.

Meter Bridge/Ronin Bridge chain Security Incident: In the Meter Bridge/Ronin Bridge chain, a validator set composed of nine validation nodes is required, with at least five signatures needed to identify deposit or withdrawal events. In this attack incident, the attacker gained control over four Sky Mavis's Ronin validators and one third-party validator run by Axie DAO. Because Axie DAO allowed Sky Mavis to sign transactions on its behalf for a period before the attack, and after this process ended, the whitelist access permissions were not revoked. This behavior led to the attackers being able to obtain the Axie DAO validator's signature through gas-free RPC once they had access to the Sky Mavis system. In the end, the attacker gathered enough signatures to complete large-scale withdrawal operations.

6.3 | Modeling and Solutions

In the *key theft vulnerability* module, the security of keys may be compromised due to users' improper actions or lack of security awareness. Data during key generation or signing processes might be manipulated using techniques such as virtual addresses or fake random numbers, making even robust keys susceptible to brute force attacks by attackers. Additionally, due to users' lack of cybersecurity awareness, social engineering methods can easily enable attackers to obtain users' keys and carry out malicious attacks.

Solutions: Additionally, to protect against the threat of key theft vulnerabilities and ensure user security, it's advisable to consider the following measures: Utilize Keyless Solutions with MPC (Multi-Party Computation): MPC algorithms are increasingly being used in blockchain applications to enhance private key security. They enable universal multisignature solutions, addressing the significant differences in multisignature schemes across various blockchains. Ensure Private Key Concealment: Implement solutions that hide the actual private keys, thereby mitigating single-point risks. In addition to multisignature, consider using SSS (Shamir's Secret Sharing Scheme)[45], which divides a seed into multiple shares. To recover a wallet, a specified number of shares are required. Private Key Storage and Backup: If cloud backup of data is deemed necessary, use encryption methods like GPG (GNU Privacy Guard)[46] to encrypt wallet-related content securely before storing it on cloud servers. However, it's strongly recommended to use hardware wallets or cold storage methods for private key storage. Choose Trusted Tools for Generating Virtual Addresses: Be cautious when selecting tools to generate virtual addresses. Avoid using tools with known vulnerabilities that could be exploited to compromise your secure wallet. Strengthen Awareness Against Social Engineering Attacks: Enhance awareness and vigilance against social engineering attacks. Learn to identify website domains carefully, and steer clear of advertisements on social platforms to prevent falling victim to phishing attacks.

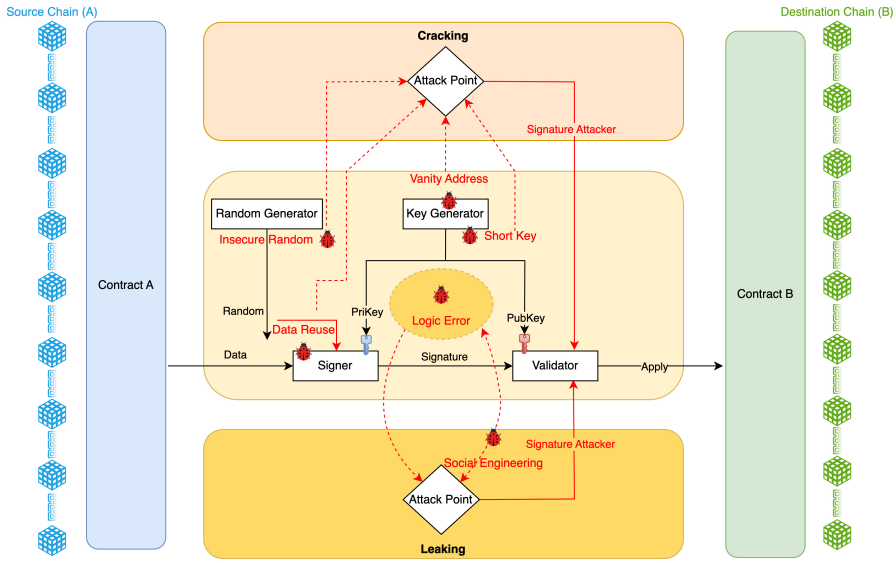

FIGURE 8 Key theft vulnerablities model

7 | DIVERSE VULNERABILITY

Cross-chain bridges, involving large amounts of funds, have been the focus of attackers for a long time. In addition to using verification bypass, processing mechanism imbalance, and key leakage cracking attack methods, there are also some niche but equally high-threat attack patterns, including log utilization, root hash construction, and others. This section will delve into the security risks of cross-chain bridges through the discussion of these special attack methods.

7.1 | Log Utilization

Attackers forge or misuse log events in some way. As the system or contract does not handle these logs correctly, the attackers are able to exploit these logs to achieve their goals. For this type of vulnerability, the key to prevention lies in correctly processing and verifying log events, especially when handling requests, it is necessary to verify the source and legality of the request to prevent illegal requests from being executed.

pNetwork Security Incident: In the attack event of pNetwork, the attacker exploited log utilization vulnerabilities to achieve their attack goals[47]. The attacker first deployed a set of smart contracts for the attack, generating a series of event logs containing legal and illegal peg-out requests. Due to errors in the Rust code of pNetwork when extracting and processing these log events, the legality of the requests was not verified, which allowed the attacker's illegal requests to be processed in error, resulting in the unlocking and acquisition of tokens by the attacker.

The log utilization category lacks sufficient verification when processing event logs, allowing the attacker to use fake request logs to bypass asset locks. Developers should add strict verification mechanisms when bridge contracts handle event logs to ensure that all requests come from trusted sources.

7.2 | Block Swap

Block swap vulnerabilities allow attackers to tamper with or replace block information in the blockchain. A block is the basic component of a blockchain, containing a series of transaction data. If a block is tampered with or replaced, all the transaction information it contains is controlled by the attacker.

Rainbow Bridge Security Incident: An attack related to the Rainbow Bridge occurred, but luckily, due to Rainbow Bridge's watchdog mechanism and the intervention of MEV robots, the attacker's plot was ultimately unsuccessful[48]. In this attack, the attacker first deployed a contract capable of generating fake blocks and used this contract to send blocks with incorrect

timestamps, trying to send transactions ahead of Rainbow Bridge's relay and replace the original block. This fake block with incorrect timestamp information was able to bypass some checks successfully and thus replaced the original block. However, when the Rainbow Bridge's relay received this fake block, it not only was not deceived by it but instead noticed the anomaly. The "watchdog" discovered that the fake block submitted by the attacker did not exist in the NEAR blockchain, so it initiated a challenge transaction, trying to expose this fake block in the Ethereum network. Eventually, this challenge transaction was discovered and executed by a MEV robot (a robot that automatically executes profitable transactions), causing the attacker's fake block to be successfully rolled back, and the real block was retained. Rainbow Bridge is an example of successful defense, by effectively combining the relay and "watchdog" mechanisms, it successfully prevented a block swap attack. Therefore, we believe that a robust security mechanism and efficient emergency response are crucial for preventing blockchain attacks.

7.3 | Root Hash Construction

Root hash construction vulnerabilities mainly refer to vulnerabilities that allow an attacker to carefully construct and adjust data, making the root hash of the hash tree (such as the Merkle tree or IAVL tree) match the target hash. In a blockchain, the root hash of a tree is often used as a key parameter to verify data integrity and consistency. In a hash tree, each leaf node represents a piece of data, and each internal node represents a combination of the hash values of its child nodes. The root node (or "root hash") of the tree represents a summary of the entire dataset. According to the properties of the hash function, any minor changes to the data will cause a significant change in the root hash.

BSC Token Hub Security Incident: The attack on BSC Token Hub resulted in a loss of $143. 57M[49]. The attacker first chose a hash value of a successfully submitted block and constructed an attack payload to add an arbitrary new leaf node on the IAVL tree. At the same time, to satisfy the proof implementation, the attacker also added an empty internal node. The attacker adjusted the data of the leaf node so that the calculated root hash equaled the correctly selected root hash. In the end, the attacker successfully constructed a withdrawal proof that matched a specific block.

The essence of the root hash construction attack method is the reverse engineering of the hash function. The attacker needs to understand and master the working principle of the hash function and design data carefully to successfully construct the required root hash. For cross-chain bridge developers, the prevention method is to patch vulnerabilities in precompiled contracts and enhance the rigor of IAVL tree verification.

In the blockchain field, special attack categories highlight the diversity of vulnerabilities and the innovation of attack methods. For security workers, keeping up with technological advances and learning and analyzing real attack cases from the past can more accurately grasp the characteristics of such attacks and preemptively prevent and respond to possible new types of attacks.

8 | CONCLUSION

In this article, we've collected and analyzed 19 significant cross-chain bridge attack incidents to date, dissecting hacker tactics and categorizing them according to their differences and similarities. We've summarized three large-scale, conventional hacker attacks and vulnerability exploitation methods: Verification Bypass, Imbalance in Handling Mechanism, and Key Leakage Decryption, as well as a Special Attack category with unique exploitation methods.

In the process of writing this article, we have drawn on vulnerability analysis reports from well-known blockchain security companies such as SlowMist, ensuring our understanding of various attack methods is both accurate and in-depth. Here, we extend our appreciation for the outstanding work done by these companies. We hope this article provides a practical reference for blockchain developers and researchers, enabling them to better understand the potential security risks of cross-chain bridges, and hence adopt more effective security measures when designing and developing cross-chain bridges. Simultaneously, we hope that this article will inspire more research on the security of blockchain cross-chain technology, promoting continuous advancement in this emerging technology field.

REFERENCES

1. Guo H, Yu X. A survey on blockchain technology and its security. *Blockchain: research and applications.* 2022;3(2):100067.
2. Indah K. How Many Blockchains are There in 2023?. https://increditools.com/blockchains/#:~:text=According%20to%20research%2C%20there%20are,network%2C%20and%20permissioned%20blockchain%20network.; . Accessed 2023.
3. Hope-Bailie A, Thomas S. Interledger: Creating a standard for payments. In: 2016:281–282.
4. Schwartz E. A payment protocol of the web, for the web: Or, finally enabling web micropayments with the interledger protocol. In: 2016:279–280.

5. Back LDMFGMAMAPJT. Enabling blockchain innovations with pegged sidechains. http://www.opensciencereview.com/papers/123/enablingblockchain-innovations-with-pegged-sidechains; . Accessed 2014.
6. Divakaruni A, Zimmerman P. The Lightning Network: Turning Bitcoin into Money. *Finance Research Letters.* 2023;52:103480.
7. Buterin V. Chain interoperability. *R3 research paper.* 2016;9:1–25.
8. Chow J. BTC Relay. http://btcrelay.org/; . Accessed 2023-08-09.
9. Amoordon A, Rocha H. Presenting tendermint: Idiosyncrasies, weaknesses, and good practices. In: IEEE. 2019:44–49.
10. Abbas H, Caprolu M, Di Pietro R. Analysis of polkadot: Architecture, internals, and contradictions. In: IEEE. 2022:61–70.
11. PUBLISHING B. Tokenomics 101: THORChain. https://banklesspublishing.com/tokenomics-101-thorchain/; . Accessed 2021.
12. Qiu T, Zhang R, Gao Y. Ripple vs. SWIFT: Transforming cross border remittance using blockchain technology. *Procedia computer science.* 2019;147:428–434.
13. Lu J. wanchain. https://www.wanchain.org/; . Accessed 2023-08-15.
14. Xu A. multichain. https://multichain.xyz/; . Accessed 2023.
15. Layerzero . Stargate. https://stargate.finance/; . Accessed 2023.
16. dune. https://dune.com/; . Accessed 2023.
17. PeckShield . Web3 Industry Security Report. https://peckshield.com/#research; . Accessed 2023.
18. Zhang J, Gao J, Li Y, Chen Z, Guan Z, Chen Z. Xscope: Hunting for cross-chain bridge attacks. In: 2022:1–4.
19. Lee SS, Murashkin A, Derka M, Gorzny J. Sok: Not quite water under the bridge: Review of cross-chain bridge hacks. In: IEEE. 2023:1–14.
20. SlowMist. https://www.slowmist.com/; . 2023.
21. CertiK. https://www.certik.com/; . 2023.
22. noneage . 2022 Global Web3 Industry Security Research Report. https://www.noneage.com/; . Accessed 2023.
23. Bünz B, Kiffer L, Luu L, Zamani M. Flyclient: Super-light clients for cryptocurrencies. In: IEEE. 2020:928–946.
24. Chand A. What Are Blockchain Bridges And How Can We Classify Them?. https://blog.li.fi/what-are-blockchain-bridges-and-how-can-we-classify-them-560dc6ec05fa; . Accessed 2023.
25. Cronje A. Multichain dapp guide, standards, and best practices. https://andrecronje.medium.com/multichain-dapp-guide-standards-and-best-practices-8fabe2672c60; . Accessed 2022.
26. Bhuptani A. A New Paradigm for Crosschain Communication. https://blog.connext.network/optimistic-bridges-fb800dc7b0e0; . Accessed 2022.
27. wormhole. https://wormhole.com/; . 2023.
28. Celer Network cBridge was Hacked in DNS Attack. https://hackenproof.com/blog/industry-news/cbridge-was-hacked; . 2023.
29. BlockChannel . The Interoperability Trilemma. https://medium.com/blockchannel/the-interoperability-trilemma-c02ffe27fa6c; . Accessed 2023.
30. NOMAD Bridge. https://app.nomad.xyz/; . 2023.
31. Polygon Swerves $850M Hack on Ethereum Bridge. https://cryptobriefing.com/polygon-swerves-850-million-hack-ethereum-bridge/; . 2023.
32. ChainSwap hackers steal $8m and crash token prices. https://finance.yahoo.com/news/chainswap-hackers-steal-8m-crash-121056965.html; . 2023.
33. The Root Cause Of Poly Network Being Hacked. https://slowmist.medium.com/the-root-cause-of-poly-network-being-hacked-ec2ee1b0c68f; . 2023.
34. Kushwaha SS, Joshi S, Singh D, Kaur M, Lee HN. Ethereum smart contract analysis tools: A systematic review. *IEEE Access.* 2022;10:57037–57062.
35. Ghaleb A, Pattabiraman K. How effective are smart contract analysis tools? evaluating smart contract static analysis tools using bug injection. In: 2020:415–427.
36. Zhang J, Wang Y, Qiu L, Rubin J. Analyzing android taint analysis tools: FlowDroid, Amandroid, and DroidSafe. *IEEE Transactions on Software Engineering.* 2021;48(10):4014–4040.
37. Yang G, Filieri A, Borges M, Clun D, Wen J. Advances in symbolic execution. *Advances in Computers.* 2019;113:225–287.
38. Jiang B, Chen Y, Wang D, Ashraf I, Chan W. WANA: Symbolic execution of wasm bytecode for extensible smart contract vulnerability detection. In: IEEE. 2021:926–937.
39. Bu L, Liang Y, Xie Z, et al. Machine learning steered symbolic execution framework for complex software code. *Formal Aspects of Computing.* 2021;33(3):301–323.
40. Beaman C, Redbourne M, Mummery JD, Hakak S. Fuzzing vulnerability discovery techniques: Survey, challenges and future directions. *Computers & Security.* 2022;120:102813.
41. Ding M, Li P, Li S, Zhang H. Hfcontractfuzzer: Fuzzing hyperledger fabric smart contracts for vulnerability detection. In: , , , 2021:321–328.
42. QANplatform Bridge Exploited $2 Million By Hackers. https://coincu.com/132818-qanplatform-bridge-exploited/; . 2023.
43. Crypto Market Maker Wintermute Hacked for $160M. https://www.investopedia.com/wintermute-got-hacked-6741864; . 2023.
44. EXPLAINED: THE HARMONY HORIZON BRIDGE HACK. https://www.halborn.com/blog/post/explained-the-harmony-horizon-bridge-hack; . 2023.
45. Pang LJ, Wang YM. A new (t, n) multi-secret sharing scheme based on Shamir's secret sharing. *Applied Mathematics and Computation.* 2005;167(2):840–848.
46. Nguyen PQ. Can we trust cryptographic software? Cryptographic flaws in GNU Privacy Guard v1. 2.3. In: Springer. 2004:555–570.
47. EXPLAINED: THE PNETWORK HACK (SEPTEMBER 2021). https://www.halborn.com/blog/post/explained-the-pnetwork-hack-september-2021; . 2023.
48. Hackers Lose 5 Ether While Trying to Attack Near Protocol's Rainbow Bridge. https://www.coindesk.com/tech/2022/08/23/hackers-lose-5-ether-while-trying-to-attack-near-protocols-rainbow-bridge/; . 2023.
49. BNB Hack: IAVL Spoofing Explained. https://sanebow.me/bnb-hack-iavl-explained; . 2023.
50. Noura M, Atiquzzaman M, Gaedke M. Interoperability in internet of things: Taxonomies and open challenges. *Mobile networks and applications.* 2019;24:796–809.
51. Fortunato S. Community detection in graphs. *Phys. Rep.-Rev. Sec. Phys. Lett.*. 2010;486:75-174.
52. CoinTelegraph . DeFi security losses rose 47.4CoinTelegraph. https://cointelegraph.com/news/defi-security-loses-rose-47-4-in-2022-to-hit-3-64b-report; . Accessed 2023.
53. Breidenbach L, Cachin C, Chan B, et al. Chainlink 2.0: Next steps in the evolution of decentralized oracle networks. *Chainlink Labs.* 2021;1:1–136.

54. Wormhole Bridge Exploit Incident Analysis. https://www.certik.com/resources/blog/1kDYgyBcisoD2EqiBpHE5l-wormhole-bridge-exploit-incident-analysis; . 2023.

AUTHOR BIOGRAPHY

empty.pdf

Author Name. Please check with the journal's author guidelines whether author biographies are required. They are usually only included for review-type articles, and typically require photos and brief biographies for each author.