The Integrated Architecture, Economic Imperatives, and Security Profile of Liquidity-Based Cross-Chain Bridges

I. Foundational Principles and Architectural Mandate of Cross-Chain Liquidity Bridges (CLBs)

1.1 CLB Taxonomy: Architectural Differentiators

Liquidity-based cross-chain bridges (CLBs) represent a significant architectural evolution beyond earlier methods, primarily characterized by their use of pooled liquidity on both the source and destination chains.[1] These bridges fulfill transfer requests by tapping into existing liquidity reserves, fundamentally shifting the paradigm away from older "lock-and-mint" mechanisms.[1]

The defining innovation of the CLB architecture is the elimination of wrapped tokens.[1] Unlike lock-and-mint models, which require collateralized, synthetic representations of assets on foreign chains, CLBs facilitate direct swaps of native assets. This approach dramatically enhances capital efficiency, simplifies the user journey, and reduces the inherent complexity associated with managing wrapped token risk.[2]

The standard operational flow, exemplified by protocols such as Allbridge Core, involves several distinct steps: when a user initiates a transfer, the incoming tokens on the source chain are first exchanged into their internal dollar value within the local liquidity pool.[4] This dollar value is then securely transmitted via a cross-chain messaging protocol (XCMP) to the destination chain. The destination bridge contract receives the value data, converts it back into the native tokens available in the destination pool, and finally executes the withdrawal to the recipient.[4]

This architectural choice changes the primary security concentration. In lock-and-mint systems, the major risk centers on the custodial solvency and security of the validator set responsible for minting the wrapped tokens. Conversely, CLBs, by circumventing wrapped assets, shift the core security emphasis from *custodial solvency* to the *integrity of the Automated Market Maker (AMM) mathematical model* and the operational capacity to manage destination liquidity. The latter is encapsulated as Inventory Risk, a critical economic threat explored in Section IV.[3] This structural change mandates a greater investment in complex economic modeling (e.g., StableSwap) and the rigor of fixed-point mathematical implementations.

1.2 Interoperability Primitives: The Role of Cross-Chain Messaging Protocols (XCMPs)

Blockchains are inherently siloed economic environments, lacking native capability to monitor or understand events occurring on other networks.[5] Cross-Chain Messaging Protocols (XCMPs) are the foundational technological primitive that solves this isolation, enabling smart contracts to read external data and securely convey the verified *value* of the token deposit from the source chain to trigger a withdrawal on the destination chain.[4]

Modern CLBs integrate a diversity of XCMPs to optimize for specific assets and security tolerances. Examples include:

- **Wormhole:** A decentralized protocol frequently used as the default routing mechanism due to its strong trust model.[6]
- **Allbridge Custom Protocol:** This approach utilizes a minimal validation set (just two validators), sacrificing decentralization for significant gains in transaction speed and reduced cost, making it effective for certain chains like Tron and Stellar.[6]
- **Circle's Cross-Chain Transfer Protocol (CCTP):** This protocol is purpose-built for USDC transfers, leveraging a permissionless utility that natively burns the stablecoin on the source chain and mints it on the destination chain. This ensures exceptional capital efficiency and maintains minimal slippage, even for high-volume stablecoin transfers.[2]

The choice between these protocols highlights a fundamental trade-off in bridge design: maximized trust versus user experience (speed and cost). The Wormhole and CCTP routes offer enhanced trust through decentralization, while custom, lightweight messaging protocols provide lower latency and fees.[6] A sophisticated CLB architecture must incorporate dynamic routing logic, prioritizing the most decentralized and trust-minimized routes (Wormhole/CCTP) for high-value transactions, while selectively utilizing faster, lower-cost routes for smaller or specific asset transfers (e.g., USDT via OFT/LayerZero).[6]

The table below contrasts the CLB architecture with its predecessor.

Table 1: Comparison of Cross-Chain Bridge Architectures

Feature	**Lock-and-Mint (Wrapped Token)**	**Liquidity-Based Bridge (CLB)**	**Primary Security Risk**
Native Asset Usage	No (uses synthetic/wrapped tokens)	Yes (direct swap of native assets) [1]	Custodian/Minting centralization
Capital Efficiency	Low (capital locked on source chain)	High (optimized via pooled liquidity) [7]	Pool solvency / Rebalancing failure
Fee Model	Fixed minting/burning fees, gas.	Dynamic fees, LP fees, gas (0.3% LP fee example) [3]	
Message Verification	Proof of Authority/Consensus	Verified Cross-Chain Messaging (Wormhole, CCTP) [5]	

II. Smart Contract Engineering and Cryptographic Verification Mechanisms

The successful implementation of a CLB requires specialized smart contract design tailored to the disparate environments of the Ethereum Virtual Machine (EVM) and the Solana runtime, particularly concerning state management and the verification of external messages.

2.1 Multi-Chain Contract Design and State Synchronization

On EVM chains, Solidity smart contracts manage liquidity pools and execute swaps by integrating with composable communication layers such as Wormhole or LayerZero.[8] Security protocols for EVM development focus on mitigating classical vulnerabilities, including access control issues and reentrancy attacks, which exploit the sequence of operations within contract execution.[9]

The architecture on Solana, typically implemented in Rust, follows a distinctly different structure. The core logic resides in the program, but the state—the pool balances, fee accruals, and AMM invariants (D)—is externalized. This state is stored in separate accounts known as Program Derived Addresses (PDAs).[10] PDAs are crucial because they are deterministically generated from a program ID and optional seeds, effectively allowing the Solana program itself to "sign" for and control the pool assets without requiring a corresponding private key.[11] The program manages assets using Cross-Program Invocations (CPIs), executing calls to lower-level programs like the SPL Token Program.[12]

This architectural decoupling on Solana introduces specialized engineering challenges concerning state management. Developers must employ serialization libraries (such as borsh in Rust) to convert data structures into byte arrays for storage in the PDA accounts.[10] Furthermore, the creation of new accounts requires precise calculation of the necessary space and corresponding rent (lamports).[10] Failure to accurately calculate space or perform proper data serialization during CPIs can lead to data corruption or unexpected program failures.

2.2 Message Passing Security and On-chain Cryptographic Verification

The security of a CLB hinges on the destination contract's ability to reliably verify that the source chain transaction (the deposit into the pool) actually occurred. This requires integrating "trust-minimized designs," such as threshold signatures or optimistic confirmations, to validate the integrity of the data provided by external validator sets or decentralized relayers.[8]

Solana leverages specific system programs to manage cryptographic verification efficiently. The Secp256k1 Program (KeccakSecp256k1) is employed to handle the computational complexity of signature verification for external messages, such as those originating from an EVM chain.[13] This system verifies that the transaction was signed by the expected private key, recovering the sender’s Ethereum address for validation.[9] To execute the withdrawal instruction on the destination chain, the transaction must include this signature verification instruction, leveraging its index within the transaction bundle.[13]

Mitigating risks such as those observed in historical exploits (e.g., the Horizon Bridge compromise) requires moving away from over-centralized validation.[8] Modern security strategies demand the introduction of multi-signature controls and continuous, real-time on-chain event verification to ensure robustness against attack vectors.[8]

However, the pursuit of performance and trust-minimization introduces operational costs. Solana's Proof-of-History (PoH) mechanism facilitates parallelized verification of transaction ordering, which improves throughput.[14] To achieve this, the proof must contain every intermediate hash, leading to very large proof sizes.[14] This requirement directly translates into high hardware demands for validators and larger transaction data payloads, a crucial operational cost embedded in high-performance CLB infrastructure.[14]

III. Core Mathematical and Computational Integrity (AMM Deep Dive)

The foundational stability of a liquidity-based bridge—its ability to ensure minimal slippage and consistent exchange rates—is dictated entirely by the precision and robustness of its Automated Market Maker (AMM) mathematical implementation.

3.1 Automated Market Maker (AMM) Models in Cross-Chain Contexts

Effective liquidity is paramount for CLBs; deep liquidity allows a decentralized exchange (DEX) or bridge pool to absorb large trades with minimal price impact, whereas low liquidity results in high slippage.[7] Given that CLBs often specialize in stable asset transfers (dollar-pegged tokens) [2], they typically utilize specialized AMM models that outperform the standard constant product model ($x \cdot y = k$).

The StableSwap invariant is the preferred model for stablecoin CLBs.[15] This model functions as a dynamic hybrid curve, closely resembling a Constant Sum curve near the asset peg to minimize slippage, but dynamically shifting to behave like a Constant Product curve when one asset's reserves become shallow.[15] This feature prevents a liquidity pool from being completely drained, which is a fatal flaw of Constant Sum models.[15]

The functionality of the StableSwap pool is governed by key computational primitives:

- The get_D() function calculates the measure of total liquidity, D, which is the invariant value derived from the current pool reserves (x_i) and the amplification factor (A).[16] D changes only when liquidity is added or removed.[16]
- The get_y() function is used during the swap itself. Given an input amount of token X, it computes the corresponding output amount of token Y required to keep the invariant D constant.[16]

3.2 Numerical Approximation and Invariant Calculation

The calculation of the invariant D (and consequently the output amount Y) is computationally intensive because D is derived from the Stable Swap Polynomial ($SSP(D)$), which is often a higher-degree polynomial.[15] Since algebraic solutions for such polynomials are impractical, AMM implementations rely on iterative numerical approximation methods.

The **Newton-Raphson method** is commonly used to approximate the true value of D, specifically by finding the positive zero of the $SSP(D)$ equation.[15] This technique requires a suitable initial guess (often the total sum of token reserves) and the ability to compute the derivative of the Stable Swap Polynomial, $SSP'(D)$, to refine the approximation in subsequent iterations.[15]

To ensure the integrity and security of the pool against minute arbitrage attacks, strict rules regarding reserve rounding are implemented. All rounding operations of the pool reserves must consistently be performed **in favor of the pool**, slightly adjusting the balance to mitigate any computational exploits or infinitesimal value leakage.[15]

3.3 Fixed-Point Arithmetic and Precision in DLT Environments

A critical requirement for any decentralized ledger technology (DLT) platform dealing with currency is the strict prohibition of floating-point arithmetic (e.g., float or double), as its inherent rounding errors can create exploitable vulnerabilities and loss of precision.[12]

High-performance CLBs built on Rust-based platforms like Solana must rely entirely on fixed-point math libraries and ensure rigorous integer arithmetic checks within their programs.[12] Projects like Orca Whirlpools demonstrate this specialization by isolating utility, math, and quoting functions into dedicated core packages (orca_whirlpools_core) to maintain the fixed-point implementation's isolation and reliability.[18]

The implementation of complex numerical methods like Newton-Raphson, even when using fixed-point math, relies on iterative approximations.[15] This reliance introduces a highly technical risk: if the fixed-point implementation of the derivative calculation or the iterative function contains a subtle flaw, an attacker could repeatedly exploit the minuscule numerical difference between the system's computed invariant and the true invariant. This potential vulnerability is high-impact and low-visibility, meaning security audits must extend beyond traditional smart contract reviews to include formal verification of the numerical stability and convergence properties of the fixed-point algorithms used for D and Y calculation.

Table 3: Technical Integrity: Fixed-Point Math & Verification

Requirement Area	**Key Technical Challenge**	**Implementation Approach (Rust/Solidity)**	**Security Rationale**
Numerical Precision	Preventing overflow/underflow and float calculation errors.[12]	Fixed-point math libraries (spl-math), rigorous integer arithmetic [12]	Eliminates currency vulnerability exploits.[12]
Invariant Calculation	Solving higher-degree polynomials (StableSwap invariant D).[15]	Iterative approximation methods (e.g., Newton-Raphson) [15]	Ensures D is accurately maintained during liquidity events.
Verification	Validating external messages reliably.	On-chain cryptographic programs (e.g., Solana Secp256k1 Program) [13]	Trust-minimized execution of withdrawal instructions based on external proof.

IV. Economic Architecture, Risk Quantification, and Rebalancing Protocols

The sustainability of a CLB is primarily an economic challenge centered on managing liquidity imbalances and ensuring adequate compensation and protection for Liquidity Providers (LPs).

4.1 Liquidity Provider Risk Profile

LPs contribute capital to the pools and are compensated via earned fees, but they must accept several risks. While the StableSwap model minimizes divergence risk for pegged assets, LPs still face Impermanent Loss (IL) if asset prices within the pool shift relative to simply holding the assets separately.[19]

However, the foremost economic threat unique to CLBs is **Inventory Risk**. This occurs when transfer flows become persistently one-sided. Heavy, long-term demand for an asset on the destination chain drains the associated pool, leaving LPs exposed, or "stuck with the less desirable asset" on the source chain.[3] This imbalance can cause transfers to fail or execute at highly unfavorable slippage rates.[3]

4.2 Advanced Liquidity Management and Incentivization

Maintaining liquidity balance is crucial; without active management, high imbalance requires the node infrastructure to carry the burden of manual rebalancing, often without adequate compensation.[21] Protocols implement specialized self-balancing mechanisms:

- **Incentive Pools (IPs):** A percentage of the transfer fee (F) paid by the user is routed into a dedicated incentive pool state within the liquidity pool.[21] This capital is used to reward external agents—referred to as "solvers"—for undertaking the complex task of rebalancing the funds.
- **Dynamic Fee Adjustments:** The bridge mechanism uses value adjustments that are contingent on the current pool condition. These adjustments dynamically increase transaction costs for swaps that worsen the pool imbalance and conversely decrease

costs for swaps that help restore equilibrium.[4]

Automated Solver Rebalancing Protocols are the operational core of liquidity management. Solvers are intermediary entities specifically tasked with efficiently redistributing liquidity across network boundaries.[22] This process, while complex, is essential for maintaining minimal slippage and promoting efficient price discovery across the interconnected market.[3]

The integration of solvers, who operate based on profit incentives, introduces a sophisticated economic element. Solvers may prioritize Maximal Extractable Value (MEV) opportunities over simply restoring pool balance, potentially leading to unfair transaction sequencing or liquidity manipulation. Therefore, the architecture requires robust mechanism design, implementing safeguards such as restricted transaction visibility or time-locked execution to ensure that the economic benefits derived from rebalancing primarily accrue to the LPs and users, rather than being exploited solely by the external solvers.

Table 2: Economic Risk and Mitigation Matrix for CLBs

Risk Vector	**Description**	**Economic Consequence**	**Mitigation Strategy**
Inventory Risk	Persistent one-sided flow drains destination pool liquidity.[3]	Failed transfers, increased slippage, LP devaluation.	Solver Rebalancing, Incentive Pools (IP), Dynamic Fee adjustment [21]
Impermanent Loss (IL)	Asset price divergence relative to initial contribution.[19]	Reduced LP value upon withdrawal.	StableSwap invariant for pegged assets, Concentrated Liquidity, Fee accrual [7]
Systemic Contagion	Local asset or protocol failure propagates across interconnected pools.[23]	Cascading liquidations, liquidity freezes across chains.	Compartmentalized risk design, strict verification protocols (trust-minimized), Multi-Sig controls [8]

4.3 Strategic Liquidity Models

Some protocols enhance capital efficiency by utilizing single-sided liquidity pools, which can reduce costs and potentially increase LP returns.[24] However, this model heightens the exposure to Inventory Risk, placing a greater strategic reliance on the effectiveness and responsiveness of the automated solver rebalancing system to prevent pool depletion and maintain functionality.[3]

Crucially, the highly interoperable structure of CLBs means they can act as amplifiers for systemic risk.[23] If a significant external event (such as an asset de-peg or major regulatory action) induces rapid, prolonged one-sided flow, the severe resulting imbalance will not remain confined to a single protocol or chain. This failure state would propagate across the CLB network to all connected pools, potentially triggering cascading liquidations or liquidity freezes throughout the broader decentralized finance ecosystem, thus meeting the criteria for systemic risk defined in DeFi.[23]

V. Systemic Vulnerabilities and Trust-Minimized Security Audits

The inherent complexity of bridging infrastructure introduces specialized vulnerabilities that demand rigorous auditing protocols focused on both smart contract execution and centralization vectors.

5.1 Smart Contract Exploits and Application-Specific Risks

CLBs are subject to general risks endemic to DeFi, including smart contract bugs, economic model failures, and governance risks.[25] Specific contract vulnerabilities include Reentrancy attacks, where execution is interrupted before state is fully updated, and Access Control violations.[9]

Cross-chain exploits frequently target the mechanisms used for verification and consensus. The Horizon Bridge exploit, for example, demonstrated the significant risk posed by over-centralized validation combined with inadequate monitoring and insufficient multi-signature security controls.[8] Addressing these requires implementing robust security measures like multi-sig controls and real-time on-chain event verification to detect and prevent malicious activity.[8]

5.2 Centralization Vectors and Trust Mitigation

The security level of centralized components, such as validator committees, relies heavily on aligning economic incentives (e.g., stake-slashing) and securing key management (e.g., using hardware security modules).[3] However, these systems are fundamentally less trust-minimized than designs that rely on proof-based light-client verification.[3]

Trust-minimized engineering requires adopting protocols that maximize decentralized verification, such as threshold signatures and adaptive verification systems (like LayerZero's Ultra Light Node approach). These systems reduce security reliance on small validator sets and allow the bridge to optimize gas fees without compromising throughput or the trust model.[8]

A significant, yet subtle, threat lies in the integrity of external inputs, specifically the pricing oracles and the messaging protocol itself. A CLB relies on accurate valuation derived from the AMM or external price feeds, combined with verified messages from the XCMP.[4] A failure in the oracle feed or a compromise of the XCMP's decentralized relayer set introduces *external input risk*. Such a failure would cause the bridge to execute swaps based on fraudulent or outdated valuation, potentially leading to immediate and systemic liquidity drain that is exceptionally difficult to stop once initiated. This vulnerability necessitates the deployment of highly robust monitoring systems and automated circuit breakers to isolate and halt compromised operations.

VI. Strategic Synthesis and Recommendations

6.1 Capital Efficiency vs. Security Matrix

Liquidity-based bridges have successfully addressed the challenge of capital inefficiency inherent in older cross-chain transfer methods. By enabling native asset swaps and integrating protocols like Circle's CCTP for native stablecoin burn/mint functionality, CLBs achieve maximized capital efficiency and significantly reduced slippage for users.[2]

However, the cost structure for users is multi-faceted. A typical cross-chain transaction involves a trading fee paid to LPs (e.g., 0.15% on both the source and destination chains), a protocol or relayer fee, and gas costs on both chains.[3] For transparency, the user interface must provide a clear, detailed breakdown of all cost components to allow users to accurately compare routes and protocols before committing to a transfer.[3]

The CLB infrastructure is not merely a tool for asset transfer; it fundamentally acts as a global market price equalizer. Blockchains, being isolated economies, create opportunities for arbitrageurs who exploit price discrepancies across chains.[5] By providing fast, secure, and capital-efficient bridging, CLBs accelerate this arbitrage activity. The resulting rapid restoration of price equilibrium leads to tighter spreads and enables more efficient price discovery across the entire multi-chain ecosystem.[3]

6.2 Operationalizing Rebalancing and Inventory Risk Mitigation

The analysis demonstrates that Inventory Risk is the preeminent economic vulnerability for CLBs. Therefore, the strategic requirement for operational stability is the implementation of an actively managed rebalancing mechanism, driven either by strong economic incentives or external solvers.[3]

Looking ahead, CLBs are expected to adopt more advanced AMM models to enhance capital utilization further. This includes incorporating concepts such as Concentrated Liquidity Pools (CLPs) and potentially models like Readjusting Concentrated Liquidity AMM (ReCLAMM).[25] These innovations aim to allocate liquidity only to the most relevant price ranges, maximizing the effectiveness of every dollar deposited and further decreasing slippage in a cross-chain context.

Conclusions

The liquidity-based cross-chain bridge architecture successfully addresses the limitations of previous lock-and-mint models by eliminating wrapped token dependency, achieving high capital efficiency, and providing a seamless native asset swap experience. The structural security of this architecture is critically reliant on the fidelity of two core components: first, the integrity of cryptographic message verification, which must withstand centralization attacks through trust-minimized designs (e.g., Solana's Secp256k1 Program and threshold signatures); and second, the numerical stability of the AMM, which depends on rigorously audited fixed-point mathematical implementations of complex functions like the StableSwap invariant calculation (Newton-Raphson approximation).

Economically, the primary long-term challenge is mitigating Inventory Risk caused by persistent one-sided capital flows. The sustainability of CLBs mandates the active deployment of economic mechanisms, such as incentive pools and automated solver rebalancing protocols, to ensure pool equilibrium. A failure in either the mathematical integrity of the AMM or the economic efficacy of the rebalancing system could propagate localized distress across the highly interconnected DeFi ecosystem, thereby amplifying systemic risk. Continuous security development must therefore focus on securing external data inputs and formally verifying the numerical precision of the underlying algorithms.

Works cited

1. accessed October 29, 2025, https://webisoft.com/articles/types-of-blockchain-bridges/#:~:text=Liquidity%2Dbased%20cross%2Dchain%20bridges%20operate%20by%20using%20pooled%20liquidity,need%20for%20wrapped%20tokens%20altogether.
2. Allbridge on Solana: Project Reviews, Token, Roadmap, Top Strategies + More, accessed October 29, 2025, https://solanacompass.com/projects/allbridge
3. Cross Chain Liquidity Pools Guide to Swapping Assets Across Networks, accessed October 29, 2025, https://blog.bitunix.com/en/cross-chain-liquidity-pools-explained/
4. allbridge-io/allbridge-core-docs - GitHub, accessed October 29, 2025, https://github.com/allbridge-io/allbridge-core-docs
5. What Is A Cross Chain Bridge? | Chainlink, accessed October 29, 2025, https://chain.link/education-hub/cross-chain-bridge
6. Messaging protocols - What is Allbridge Core?, accessed October 29, 2025, https://docs-core.allbridge.io/product/how-does-allbridge-core-work/messaging-protocols
7. DEX Liquidity Pools Management: Key Strategies & Insights - IdeaSoft.io, accessed October 29, 2025, https://ideasoft.io/blog/dex-liquidity-pools-management/
8. Cross-Chain Protocols for Solidity Developers Building Blockchain Bridges | MoldStud, accessed October 29, 2025, https://moldstud.com/articles/p-building-bridges-a-comprehensive-guide-to-cross-chain-protocols-for-solidity-development
9. john-lawniczak/Smart-Contract-Resources: Solidity, Defi, and blockchain security resources. - GitHub, accessed October 29, 2025, https://github.com/john-lawniczak/Smart-Contract-Resources
10. Create a Basic Program, Part 2 - State Management - Solana, accessed October

29, 2025, https://solana.com/developers/courses/native-onchain-development/program-state-management

11. Program Derived Address (PDA) - Solana, accessed October 29, 2025, https://solana.com/docs/core/pda
12. EGORAGANIN/solana-amm: Implementation automated ... - GitHub, accessed October 29, 2025, https://github.com/EGORAGANIN/solana-amm
13. allbridge-io/allbridge-contract-docs - GitHub, accessed October 29, 2025, https://github.com/allbridge-io/allbridge-contract-docs
14. Deploying Solidity Smart Contracts to Solana, accessed October 29, 2025, https://soliditydeveloper.com/solana
15. aldrin-labs/amm - GitHub, accessed October 29, 2025, https://github.com/aldrin-exchange/amm
16. get_D() and get_y() in Curve StableSwap | By RareSkills, accessed October 29, 2025, https://rareskills.io/post/curve-get-d-get-y
17. Concentrated N-dimensional AMM with Polar Coordinates in Rust - arXiv, accessed October 29, 2025, https://arxiv.org/html/2510.05428
18. orca-so/whirlpools: Open source concentrated liquidity AMM contract on Solana - GitHub, accessed October 29, 2025, https://github.com/orca-so/whirlpools
19. Impermanent Loss & Liquidity Provider Risk: A Practical Guide - DcentraLab, accessed October 29, 2025, https://www.dcentralab.com/blog/impermanent-loss-lp-risk
20. Crypto Liquidity Pool Guide: Earn DeFi with Your Tokens - SCAND, accessed October 29, 2025, https://scand.com/company/blog/crypto-liquidity-pool-guide/
21. Self Balancing Cross-Chain Liquidity Pools - Notion, accessed October 29, 2025, https://biconomy.notion.site/Self-Balancing-Cross-Chain-Liquidity-Pools-c19a725673964d5aaec6b16e5c7ce9a5
22. What is Solver Rebalancing and Why It Matters for DeFi Users | Eco Support Center, accessed October 29, 2025, https://eco.com/support/en/articles/12151694-what-is-solver-rebalancing-and-why-it-matters-for-defi-users
23. Mapping Microscopic and Systemic Risks in TradFi and DeFi: a literature review - arXiv, accessed October 29, 2025, https://arxiv.org/html/2508.12007v1
24. 7 Safest Cross Chain Bridges: Real User Tests & Pros/Cons of Blockchain Bridging Protocols - TAS, accessed October 29, 2025, https://tas.co.in/7-safest-cross-chain-bridges/
25. Risks of using Balancer, accessed October 29, 2025, https://balancer.fi/risks
26. Blockchain Bridges: A Deep Dive into Cross-Chain Interoperability - Blaize Tech, accessed October 29, 2025, https://blaize.tech/blog/blockchain-bridging-explained/