# Private Bridging

Union
0xc0dejug, 0xkaiserkarel & 0x_Robbie

**Abstract**

We present a private bridging solution enabled by Zero-Knowledge Proofs (ZKPs) that allows users to prove ERC20 token transfers across multiple EVM-compatible blockchains. A user transfers (virtually burns) tokens to an unspendable address derived from a generated secret $s$ on a source chain. The user later redeems (mints) equivalent tokens by demonstrating knowledge of the secret s and additional information on the destination chain. Additionally, we introduce the Z-asset standard which allows any ERC20 token to be promoted to support both standard (transparent) and confidential bridging modes while maintaining full fungibility. Furthermore, we introduce an attestation layer where a pre-defined trusted attestor, Union [1], maintains complete visibility of both source and destination addresses, creating an auditable trail. This design enables regulatory compliance and institutional accountability while preserving transaction confidentiality from public observers. Our proposed design is inspired by EIP-7503 [2] but significantly extends it.

## 1 Introduction

Privacy in blockchain remains an unsolved problem. In traditional finance, transactions are private to the public but transparent to the institutions and its regulators. When corporations move treasury funds or high-net-worth individuals transfer assets, these activities remain shielded from competitors and markets, accessible only to counterparties and authorized bodies. This structure preserves strategic privacy, prevents front-running, and safeguards competitive advantage. Regulators and law enforcement have the ability to audit transactions. Crypto, in contrast, exposes every transaction publicly. Amounts, balances, and relationships are visible to all, making it fundamentally incompatible with institutional confidentiality standards and limiting adoption of stablecoins for applications such as (neo)banking and settlement. Financial service providers must protect client data and strategic information [3] while meeting regulatory obligations. The challenge grows in multichain environments where, even if a single chain has privacy features, bridging assets still reveals user addresses and asset flows. Enabling institutions to participate fully and compliantly in DeFi therefore requires a privacy model that protects transactions onchain and across chains [4].

We propose a model based on EIP-7503 [2] and Union's aggregated light clients to allow for a new category of assets that guarantee confidentiality for both regular transfers and crosschain transfers. The protocol allows for extensions to ensure transfers are auditable only by designated parties, ensuring asset issuers remain compliant.

## 2 Protocol Design

### 2.1 Unspendable Address

Following EIP-7503 [2], the protocol employs deterministic address derivation to create unique EVM addresses from user secrets. The protocol pairs this with Union's light clients to access state roots of counterparty chains. On the destination chain we prove using the state root of the source chain that funds were sent to an unspendable address on the source chain. By proving this, we allow for redemption of funds on the destination chain. Given a secret $s$ and destination chain identifier $c_{\text{dest}}$, the system derives an address through:

$$\text{deposit\_domain} = \text{0xDEAF}$$

$$\text{addr} = \text{Poseidon2}(\text{deposit\_domain} \parallel c_{\text{dest}} \parallel s \parallel \boldsymbol{b})_{160}$$

where Poseidon2 represents a zero-knowledge-friendly hash function [5], $\boldsymbol{b}$ is a vector of $\text{B}_{\text{max}}$ beneficiary addresses, and we extract the bottom 160 bits (20 bytes) to form a valid EVM address. The incorporation of the destination chain ID prevents crosschain double-spending attacks by crypto-

graphically binding each deposit to a specific destination chain. Without this binding, an attacker could use the same deposit proof to claim funds on multiple chains, effectively multiplying their withdrawal from a single deposit.

#### 2.1.1 Bounded and Unbounded Modes

The protocol supports two operational modes determined by the beneficiary vector $\boldsymbol{b}$:

**Bounded mode** ($1 \leq |\boldsymbol{b}| \leq \mathrm{B}_{\max}$): When one or more beneficiary addresses are specified, only those addresses can redeem the deposited tokens. The remaining slots in $\boldsymbol{b}$ are zero-padded to $\mathrm{B}_{\max}$. During redemption, the circuit verifies that the selected beneficiary matches one of the non-zero entries in $\boldsymbol{b}$. In bounded mode, the secret $s$ can be safely shared with other parties since funds can only flow to the pre-specified beneficiaries. For example, when a multisig wants to deposit to an unspendable address, the secret can be shared with all signers so they can independently verify the address construction before approving the deposit transaction.

**Unbounded mode** ($|\boldsymbol{b}| = 0$): When no beneficiaries are specified, all entries in $\boldsymbol{b}$ are zero addresses. In this mode, any non-zero address can be specified as the beneficiary at redemption time. The circuit detects unbounded mode by checking if all $\mathrm{B}_{\max}$ beneficiaries are zero, and if so, allows any non-zero beneficiary. In unbounded mode, the secret $s$ must remain strictly private, as anyone possessing it can redeem to any address of their choosing.

### 2.2 Double-Spend Prevention

To prevent secret reuse per destination chain, the protocol computes a public nullifier that commits to both the secret and the destination chain:

$$\text{nullifier\_domain} = \text{0xDEAD}$$

$$\nu = \text{Poseidon2}(\text{nullifier\_domain} \parallel c_{\text{dest}} \parallel s)$$

where $c_{\text{dest}}$ represents the destination chain ID (withdrawal chain) and $s$ denotes the user's secret.

### 2.3 Attestation

Each asset is instantiated with an attestor. For a given transfer, the attestor signs a commitment to both the unspendable source address and the beneficiary destination address:

$$\text{attested\_message\_domain} = \text{0xBEEF}$$

$$m = \text{Poseidon2}(\text{attested\_message\_domain} \parallel \text{addr} \parallel \beta)$$

where addr represents the source address where tokens are locked and $\beta$ denotes the selected beneficiary address that will receive tokens on the withdrawal chain. The attestor public key is known to all verifiers, enabling anyone to verify the authenticity of attestations while the addresses themselves remain hidden within the commitment.

The signature generation occurs off-circuit and only after signature verification succeeds does the contract proceed to verify the ZKP. Privacy is maintained because only the commitment $m$ becomes public, not the underlying addr. Accountability exists because the attestor maintains complete knowledge of both source and destination addresses for audit purposes. An attestor could be implemented as a centralized registry, or a verifier for Proof of Innocence [6].

## 2.4 Confidential Set

Users randomly sample a subset of enabled chains as vectors $\boldsymbol{r}, \boldsymbol{a}, \boldsymbol{m}$ of length $< \mathrm{C}_{\max}$, where:
- $\boldsymbol{r}$ denotes the state roots,
- $\boldsymbol{a}$ denotes the keccak256 hashes of token contract addresses,
- $\boldsymbol{m}$ denotes the balance mapping slots,
- $\mathbf{C_{max}}$ denotes the max size of the subset *(not a limit of connected chain count)*.

The prover selects one configuration at private index i where they hold funds at the unspendable address. Since the selected $\{\boldsymbol{r}_\mathrm{i}, \boldsymbol{a}_\mathrm{i}, \boldsymbol{m}_\mathrm{i}\}$ values remain private within the ZKP, external observers cannot determine which source chain was used of the provided subset.

This design creates a confidential subset of source chains. A user redeeming a token $T$ could have sourced it from Ethereum, Base, Arbitrum, Optimism, or any supported chain, with the actual source remaining cryptographically hidden. Even the attestor, despite knowing the unspendable address, must scan multiple chains to determine the source.

Furthermore, the protocol uses standard ERC20 transfers to unspendable addresses rather than explicit burn transactions. These transfers are indistinguishable from regular transactions, making it impossible for observers to identify which transfers are intended for bridging versus ordinary token movements.

## 2.5 Merkle Proofs

The protocol leverages Ethereum's Merkle Patricia Trie (MPT) [7] structure to prove token ownership without revealing the source chain. Each EVM chain organizes account and storage data in MPTs.

To prove a balance $v$ exists at an unspendable address addr, the prover provides two nested MPT proofs:

1. **Account Proof** ($\pi_{\text{account}}$): Demonstrates that the token contract with key $\boldsymbol{a}_\mathrm{i}$ (keccak256 hash of the contract address) exists in the state tree with root $\boldsymbol{r}_\mathrm{i}$, from which we extract the contract's storage root $r_{\text{storage}}$,
2. **Storage Proof** ($\pi_{\text{storage}}$): Proves the balance $v$ exists at the storage slot $k_{\text{storage}} = \text{keccak256(addr)} \parallel \boldsymbol{m}_\mathrm{i}$ within the token contract's storage root $r_{\text{storage}}$.

Valid proofs confirm that:

- The prover knows which chain configuration $i$ to select
- The balance existed when the state root $\boldsymbol{r}_\mathrm{i}$ was finalized
- The unspendable address holds at least $v$ tokens

## 2.6 Algorithms

The circuit and prover are implemented using gnark [8] and gnark-crypto [9], using the bn254 curve and Groth16 proving system [10].

### 2.6.1 Offchain: Knowledge of secret transfer

**Algorithm 1: Circuit**

▷ Maximum number of chains and beneficiaries supported by the circuit.
$\mathrm{C}_{\max} \leftarrow 4$
$\mathrm{B}_{\max} \leftarrow 4$

**procedure** CIRCUIT(X, **W**)
  ▷ Extract private witnesses.
  $c_{\text{dest}}$, $s$, $\boldsymbol{b}$, $\beta$, $\mathrm{i} \in \mathbb{N}_{<\mathrm{C}_{\max}}$, $\pi_{\text{account}}$, $\pi_{\text{storage}}$, $\boldsymbol{r}$, $\boldsymbol{a}$, $\boldsymbol{m}$, $v$, $v_r \leftarrow \boldsymbol{W}$

  ▷ Compute address and nullifier using Poseidon2.
  addr $\leftarrow$ **poseidon2**(deposit_domain $\parallel$ $c_{\text{dest}}$ $\parallel$ $s$ $\parallel$ $\boldsymbol{b}$)
  $\nu \leftarrow$ **poseidon2**(nullifier_domain $\parallel$ $c_{\text{dest}}$ $\parallel$ $s$)

  ▷ Verify beneficiary: unbounded mode (all zeros) or bounded mode (matches one).
  **assert**($\beta \neq 0$)
  unbounded $\leftarrow (\boldsymbol{b}_0 = 0) \wedge (\boldsymbol{b}_1 = 0) \wedge (\boldsymbol{b}_2 = 0) \wedge (\boldsymbol{b}_3 = 0)$

matches $\leftarrow (\beta = \boldsymbol{b}_0) \vee (\beta = \boldsymbol{b}_1) \vee (\beta = \boldsymbol{b}_2) \vee (\beta = \boldsymbol{b}_3)$
**assert**(unbounded $\vee$ matches)

▷ Compute attested message with domain separator.
$m \leftarrow$ **poseidon2**(attested_message_domain $\|$ addr $\|$ $\beta$)

▷ Verify membership of the token address against the selected state root.
**mpt_verify**($\boldsymbol{r}_\text{i}$, $\pi_\text{account}$, $\boldsymbol{a}_\text{i}$)

▷ Extract storage root from the token account.
$r_\text{storage} \leftarrow$ **extract_storage_root**($\pi_{\text{account}_\text{leaf}}$)

▷ Verify unspendable address balance against previously verified token storage root.
$k_\text{storage} \leftarrow$ **keccak256**(addr $\|$ $\boldsymbol{m}_\text{i}$)
**mpt_verify**($r_\text{storage}$, $\pi_\text{storage}$, $k_\text{storage}$)

▷ Verify deposit covers total redemption (already redeemed + current).
$v' \leftarrow$ **extract_value**($\pi_{\text{storage}_\text{leaf}}$)
**assert**($v_r + v \leq v'$)

▷ Verify private witnesses commitment.
$X' \leftarrow$ **keccak256**($\boldsymbol{r}$, $\boldsymbol{a}$, $\boldsymbol{m}$, $c_\text{dest}$, $\nu$, $v$, $v_r$, $m$, $\beta$)
▷ We use a single public witness representing the private witnesses hash. Truncated to fit the bn254 scalar field.
**assert**($X = X'_{248}$)
**end**

### 2.6.2 Onchain: Redemption verification

**Algorithm 2: Redeem**

▷ Constant shared with the circuit.
$\text{C}_\text{max} \leftarrow 4$

**procedure** REDEEM(union, $\mathbf{C} \in \mathbb{N}^{\text{C}_\text{max}}$, $\pi$, $\nu$, $v$, $\beta$, $m$)
▷ Valid attestation check prior to ZKP check.
**verify_signature**(attestor, $m$)

▷ Query and extract client states identified by their IDs. The clients must be pre-configured by the admin or at contract creation.
$\boldsymbol{r}$, $\boldsymbol{a}$, $\boldsymbol{m} \leftarrow$ **query_extract**(union, $\boldsymbol{C}$)

▷ Load the already redeemed amount.
$v_r \leftarrow$ **load_redeemed**($\nu$)

▷ Verify ZKP.
$X \leftarrow$ **keccak256**($\boldsymbol{r}$, $\boldsymbol{a}$, $\boldsymbol{m}$, $c_\text{dest}$, $\nu$, $v$, $v_r$, $m$, $\beta$)
**verify_groth16**($\pi$, $X_{248}$)

▷ Update redeemed amount for this nullifier.
**store_redeemed**($\nu$, $v_r + v$)

**mint**($\beta$, $v$)
**end**

## 2.7 Z-Asset

Additionally, we introduce the Z-asset standard to enable any ERC20 issuer to promote their existing token into a privacy-preserving crosschain asset. This standard provides a unified framework where tokens can be bridged through both Union's standard transparent bridging and the confidential bridging mechanism described in this paper.

### 2.7.1 Protocol Actors

The Z-asset standard operates through four distinct roles:

The **instantiator** deploys and configures Z-asset contracts for existing ERC20 tokens. They establish the initial parameters including supported chains, mapping slots for balance proofs, and attestation requirements. The instantiator typically represents the token issuer.

The **attestor** maintains compliance oversight across all transfers. This service signs commitments for confidential transfers and maintains the complete audit trail.

The **user** initiates transfers through either standard or confidential bridging paths. For standard transfers, users interact directly with Union's existing infrastructure. For confidential transfers, users generate secrets, transfer tokens to derived unspendable addresses, and later redeem through ZKP verification as described in our protocol.

The **relayer** processes the redemption transaction once the user has generated the necessary ZKP, minting tokens back to the beneficiary.

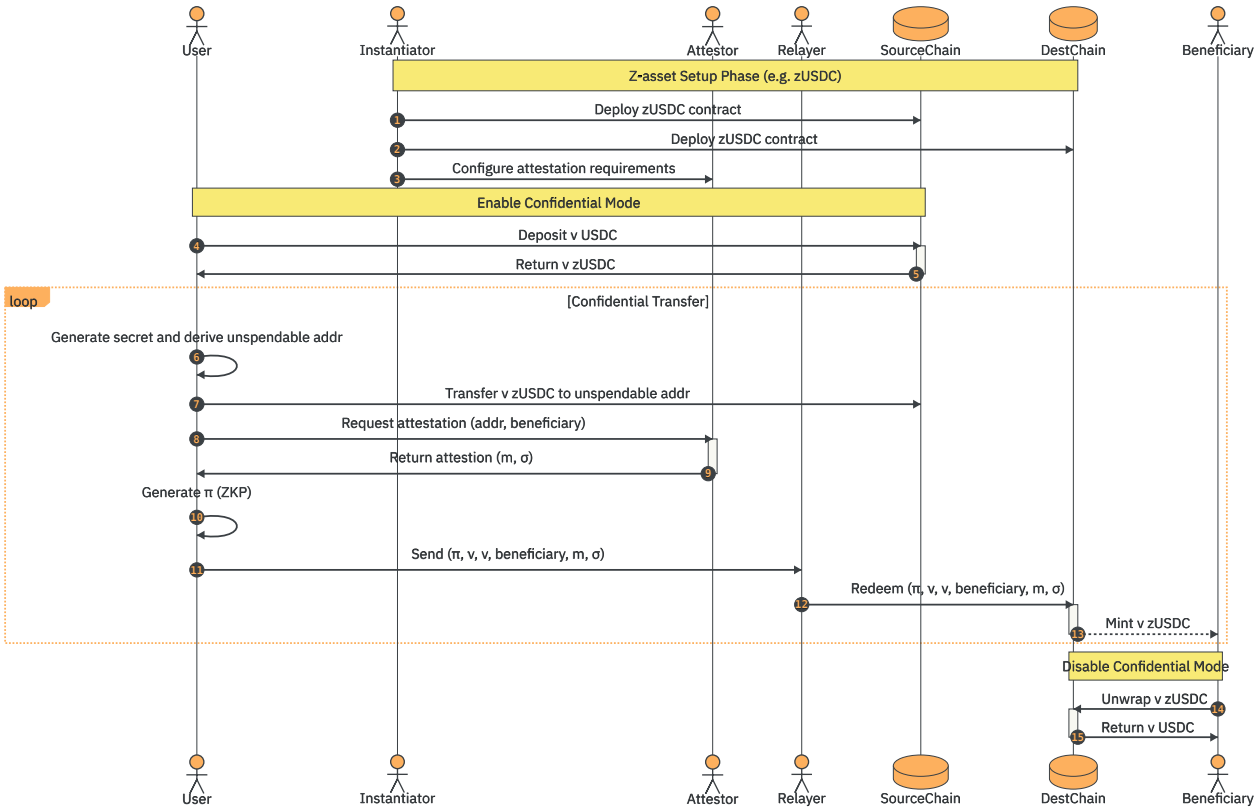
### 2.7.2 Dual-Mode Operation

Z-assets support two bridging modes within a single framework. Standard mode leverages Union's existing infrastructure for transparent transfers, suitable for users prioritizing speed and simplicity without privacy requirements. Confidential mode uses the virtual burning and ZKP redemption mechanism, providing full transaction privacy while maintaining regulatory compliance through attestation.

Furthermore, tokens remain fully fungible regardless of the bridging method used. Tokens that arrive on a destination chain through confidential bridging are indistinguishable from those bridged transparently and can be freely exchanged.

## 2.8 Protocol Flow for Confidential Mode

Our proposed protocol for confidential mode involves coordination between the instantiator who configures Z-assets, users who initiate transfers and relayers who redeem on behalf of the beneficiaries.



## 2.9 Advantages

The protocol extends EIP-7503 [2] from single-chain privacy to full crosschain confidentiality by incorporating multiple state roots from different source chains via Union's infrastructure. This enables users to confidentially bridge assets from any supported chain without revealing the origin, providing a significant privacy improvement over existing bridges.

By reusing Union's existing infrastructure, the protocol avoids re-implementing complex crosschain communication logic. Union's IBC light clients already maintain verified state roots for multiple chains, making this a practical and immediately deployable solution.

The protocol supports partial redemptions, enabling a $1 : N$ relationship between deposits and redemptions. Rather than requiring full withdrawal in a single transaction, users can redeem portions of their deposited value $v'$ across multiple transactions, provided the cumulative redeemed amount does not exceed the original deposit. This flexibility is achieved by tracking the previously redeemed amount $v_r$ and enforcing $v_r + v \leq v'$ within the circuit.

While Union signs the commitment $m = \text{Poseidon2}(\text{attested\_message\_domain} \parallel \text{addr} \parallel \beta)$ it does not inherently know which source chain or token is involved. Even after redemption occurs, determining the source chain requires additional work: the attestor must scan all supported chains to identify where addr holds a balance for the redeemed token. This computational friction provides an additional layer of temporal privacy, as the attestor cannot preemptively analyze transfers and must expend resources to determine origins post-redemption.

The smart contract-only implementation requires no protocol-level changes, enabling deployment on any EVM-compatible chain.

## 2.10 Limitations

State root propagation introduces inherent latency. Users must wait for Union's light clients to update with the latest state roots from source chains before they can complete withdrawals. This delay, typically 1-30 minutes depending on finality requirements, may impact user experience for time-sensitive transfers.

The protocol's virtual burning mechanism creates a supply accounting challenge. Rather than truly burning tokens, users transfer them to unspendable addresses, effectively locking them forever. This virtual burn approach means:

- Any transfer from chain A $\rightarrow$ B of amount $v$ artificially increases the apparent total *(sum of all chains')* token supply by $v$. Notably this includes round-trip transfers (A $\rightarrow$ B $\rightarrow$ A).
- Total supply across all chains exceeds the original minted amount
- Market analytics and tracking tools may misreport circulating supply

However, this supply obfuscation can be addressed by the attestor, which maintains complete visibility of all transfers. Since Union knows exactly how many tokens have been virtually burnt versus minted on destination chains, it can publish accurate supply metrics. The attestor could provide:

- Real-time circulating supply per chain adjusted for locked tokens
- Net token flows between chains
- A public API or dashboard showing true supply distribution

This positions the attestor as both a compliance gateway and a source of truth for supply accounting. The trust assumption in Union as the attestor, while enabling regulatory compliance, introduces a centralization point. A malicious or compromised attestor could potentially censor transactions or correlate addresses across chains, though they cannot steal funds or forge proofs.

# 3 Audit Trail

The protocol achieves a distinctive balance between privacy and auditability through an attestor. This design enables regulatory compliance while maintaining transaction confidentiality from public observers.

Public information visible to all parties includes attestor signature $\sigma$ on the message $m$, the attested message itself $m = \text{Poseidon2}(\text{attested\_message\_domain} \parallel \text{addr} \parallel \beta)$, the nullifier $\nu$ committing to the secret and destination chain, the redemption amount $v$, the already redeemed amount $v_r$, the destination chain ID $c_{\text{dest}}$, the selected beneficiary $\beta$, and the attestor public key.

Private information includes the secret $s$, the beneficiaries vector $\boldsymbol{b}$, the unspendable source address addr where tokens were locked, the exact deposited value (if partial redemption), and the index i indicating which source chain and token was used.

# 4 Conclusion

We present a confidential bridging protocol that achieves crosschain confidentiality through virtual burning and ZKP-based redemption. By transferring tokens to unspendable addresses rather than explicitly burning them, our design creates a dual-layer confidential set: transfers blend with regular ERC20 movements while chain selection remains cryptographically hidden. Additionally, we introduce the Z-asset standard that enables ERC20-issuers to promote their tokens with dual-mode bridging capabilities, supporting both transparent and confidential transfers while maintaining full fungibility.

The integration of MPT proofs within ZKPs enables users to demonstrate token ownership without actual onchain burn events, while the attestation layer provides a controlled transparency mechanism. Union maintains complete visibility for regulatory compliance and accurate supply accounting, yet cannot forge proofs or steal funds. This positions the attestor as both a compliance gateway and a source of truth for addressing the supply obfuscation.

Built entirely at the smart contract layer using Union's existing crosschain infrastructure, the protocol requires no protocol-level modifications and deploys immediately on any EVM-compatible chain.

Future work may explore recursive proof composition for improved efficiency, threshold attestation schemes to reduce trust assumptions, and mechanisms to minimize state propagation latency. The protocol demonstrates that institutional-grade confidentiality is achievable in crosschain DeFi without sacrificing the auditability required for regulatory compliance.

# 5 Bibliography


[1] Union, [Online]. Available: https://union.build/

[2] K. Kambakhsh, H. Bateni, A. Kahoori, N. Labs, and 0xwormhole, "EIP-7503: Zero-Knowledge Wormholes [DRAFT]." [Online]. Available: https://eips.ethereum.org/EIPS/eip-7503

[3] European Parliament Research Service, "Blockchain and the General Data Protection Regulation: Can Distributed Ledgers be Squared with European Data Protection Law?," technical report PE634.445, July 2019. [Online]. Available: https://www.europarl.europa.eu/RegData/etudes/STUD/2019/634445/EPRS_STU(2019)634445_EN.pdf

[4] a16z Crypto, "State of Crypto 2025: The Year Crypto Went Mainstream." [Online]. Available: https://a16zcrypto.com/posts/article/state-of-crypto-report-2025/

[5] L. Grassi, D. Khovratovich, C. Rechberger, A. Roy, and M. Schofnegger, "Poseidon2: A Faster Version of the Poseidon Hash Function," in *Cryptology ePrint Archive, Report 2023/323*, 2023. [Online]. Available: https://eprint.iacr.org/2023/323

[6] Chainway, "Proof of Innocence." Accessed: Oct. 27, 2025. [Online]. Available: https://github.com/chainwayxyz/proof-of-innocence

[7] Ethereum, "Merkle Patricia Trie." [Online]. Available: https://ethereum.org/developers/docs/data-structures-and-encoding/patricia-merkle-trie

[8] G. Botrel *et al.*, "Consensys/gnark: v0.14.0." [Online]. Available: https://doi.org/10.5281/zenodo.16929663

[9] G. Botrel *et al.*, "Consensys/gnark-crypto: v0.19.2." [Online]. Available: https://doi.org/10.5281/zenodo.17464937

[10] J. Groth, "On the Size of Pairing-based Non-interactive Arguments." [Online]. Available: https://eprint.iacr.org/2016/260