GhostPad: Anonymous Token Launching

Simon Judd
simon@psychovirtual.io

February 25, 2025

Abstract

We present a generalized framework for anonymous token launching. This prototol can be used to launch and mint tokens anonymously with no reference to the liquidity provider.

Contents

1 Introduction

Over the last year, platforms like pump.fun demonstrated high demand for token launching and minting. However, users still face significant friction points that hinder seamless token creation and distribution.

Traceable Creation

Current token launch platforms expose the creators wallet address, eliminating privacy in the token creation process. This transparency makes it impossible to launch tokens anonymously.

Complex Privacy Tools

Users who want privacy need to rely on complex technical solutions or centralized exchanges. These tools typically require:

- Understanding of advanced cryptographic concepts
- Bridging to a centralized exchange and then bridging back to a new wallet
- Using a mixer protocol

Identity and Proof Fragmentation

Current token launch mechanisms lack privacy-preserving identity proofs, creating significant operational friction:

- Wallet addresses permanently link creator wallets to launches, compromising privacy and security
- Token locking requires manual verification through third-party platforms and public social media posts
- No standardized way to cryptographically prove social media ownership without exposing wallet addresses

These limitations force token creators to either give up privacy or take multiple awkward steps which increase operational overhead and security risks.

2 Notation

Let $\mathbb{B} = \{0, 1\}$. Let $\mathcal{T}$ be a sparse Merkle tree of height 24, where each non-leaf node computes a Poseidon hash of its two children. The tree is initialized with zero values. Let $H_1 : \mathbb{B}^* \to \mathbb{Z}_p$ be a Poseidon-based commitment scheme, and $H_2 : (\mathbb{Z}_p, \mathbb{Z}_p) \to \mathbb{Z}_p$ be a MiMC permutation in sponge mode for nullifier derivation.

3 Protocol Description

3.1 Setup

A trusted setup ceremony generates a Groth16 key pair $(\mathsf{pk}, \mathsf{vk})$ for the circuit $\mathcal{S}$, proving knowledge of:

- A secret nullifier $k \in \mathbb{B}^{256}$
- A secret randomness $r \in \mathbb{B}^{256}$
- A leaf index $l \in \mathbb{B}^{24}$
- A valid Merkle path O for commitment $C = H_1(k \parallel r)$ at position l
- A hash of token metadata $D = H_1(\mathsf{name} \parallel \mathsf{supply})$ (optional)

The smart contract stores:

- The current Merkle root R and a history of the last 128 roots
- A nullifier set $\mathcal{N}$ to prevent double-minting
- A whitelist of relayers and fee parameters

3.2 Deposit

To deposit liquidity for anonymous minting:

1. The user generates $k, r \xleftarrow{\$} \mathbb{B}^{256}$ and computes $C = H_1(k \parallel r)$.
2. The user sends 1 SOL to the protocol contract, attaching C as a public commitment.
3. The contract appends C to $\mathcal{T}$, updates the Merkle root, and stores the new root in its history.

3.3 Mint

To mint tokens via a relayer:

1. The user selects a fresh Solana address A (generated anonymously) and token metadata name, supply.
2. They compute:
   - Nullifier hash $h = H_2(k)$
   - Token metadata hash $D = H_1(\mathsf{name} \parallel \mathsf{supply})$ (if applicable)
   - Merkle proof O for C under root R
3. Using pk, they generate a Groth16 proof π attesting to:

$$\mathcal{S} = \{\text{I KNOW } k, r, l, O \text{ SUCH THAT } C = H_1(k \parallel r), \\ O \text{ validates } C \text{ in } R, \\ h \notin \mathcal{N}\}$$

4. The user submits (π, R, h, A, D) to a relayer. The relayer:
   - Pays gas fees to call the protocol contract
   - Executes `mint_token` on pump.fun with metadata D
   - Sends 97% of minted tokens to A and retains 3% as fee
5. The contract verifies π with vk, checks $h \notin \mathcal{N}$, adds h to $\mathcal{N}$, and authorizes the mint.

Key Improvements over Tornado Cash(WIP):
Our current implemtation relies on Groth16 commitments and employs the same basic structure as Tornado.Cash, however future implementations will leverage the privacy preserving properties of private blockchains.

- Cross-Chain Privacy
- Chain A $->$ Privacy Chain $->$ Chain A
- Sparse Merkle trees for efficient Solana state management

4 Cross-Chain Privacy Architecture

We present a streamlined implementation of GhostPad that leverages the privacy guarantees of Zcash while maintaining a simple user experience entirely on Ethereum. This approach abstracts away the complexity of cross-chain interactions, requiring users to execute only two Ethereum transactions for the complete token launch process.

4.1 System Overview

Our cross-chain privacy mechanism follows a circular pattern, starting and ending on Ethereum while utilizing Zcash's shielded pool as a privacy layer. This design allows users to launch tokens with complete anonymity without requiring direct interaction with the privacy chain.

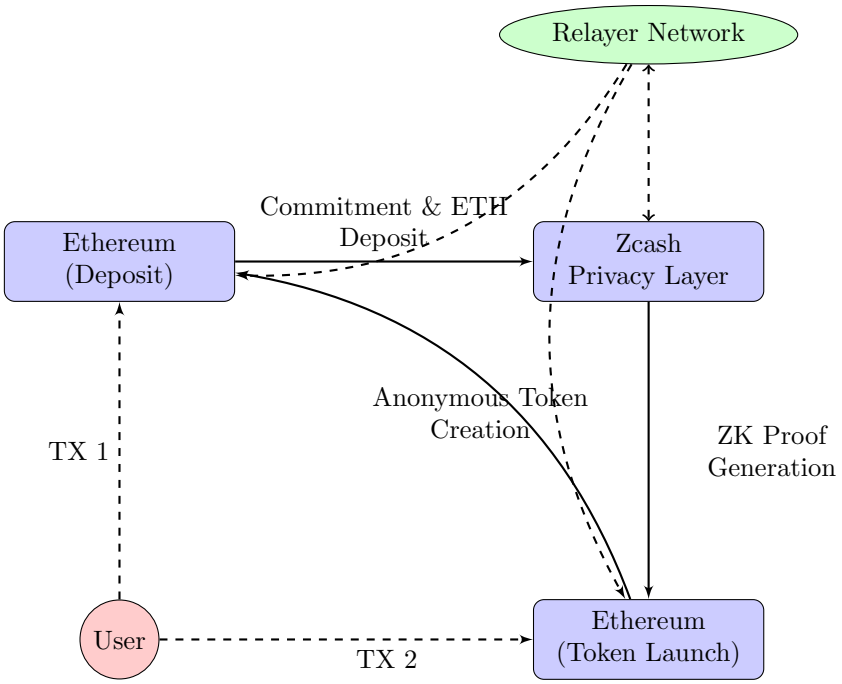

Figure 1: Cross-chain circular privacy flow for anonymous token launching

4.2 User Flow

The user experience is deliberately simplified to maintain Ethereum-native interaction:

1. **Deposit Phase (Ethereum)**
   - User generates a secret key pair in their browser
   - User commits ETH along with a hash commitment $C = H_1(k\|r)$ to Ethereum contract
   - User can include encrypted token metadata if desired
2. **Privacy Processing (Automated)**
   - Relayer service observes the deposit event on Ethereum
   - Corresponding shielded transactions are executed on Zcash
   - Deposit commitment is added to the Merkle tree
   - No user interaction required during this phase
3. **Token Launch Phase (Ethereum)**
   - User connects with a fresh Ethereum wallet (no link to original)
   - User provides secret key to generate nullifier and proof
   - User specifies token parameters (name, supply, etc.)
   - ZK proof π verifies valid deposit without revealing which one
   - Token is created and initial allocation is sent to user's address

4.3 Relayer Service

The relayer service is a critical infrastructure component that abstracts away cross-chain complexity:

- **Architecture**: Decentralized network of operators who monitor both Ethereum and Zcash
- **Responsibilities**:
  - Monitor deposit events on Ethereum
  - Create and submit shielded transactions on Zcash
  - Maintain Merkle tree state
  - Facilitate proof verification across chains
- **Incentives**: Relayers collect a small fee from each token launch
- **Security**: Relayers operate independently and cannot compromise user privacy

4.4 Technical Components

4.4.1 Ethereum Smart Contracts

- **Deposit Contract**: Accepts ETH and commitment, emits event for relayers
- **Launch Contract**: Verifies ZK proofs and creates tokens
- **Nullifier Registry**: Prevents reuse of commitments

4.4.2 Zcash Integration

- **Shielded Pool**: Provides anonymity set for user commitments
- **Zero-Knowledge Circuits**: Extended to support GhostPad-specific operations
- **Merkle Tree**: Maintains provable state of commitments

4.4.3 Privacy Guarantees

The cross-chain approach provides stronger anonymity because:

- Original Ethereum address has no on-chain connection to token launch address
- Zcash shielded pool makes tracking transactions cryptographically impossible
- Even blockchain analysts monitoring both chains cannot link deposits to launches
- User control of secrets ensures only they can authorize the token launch

References

[1] Alexey Pertsev, Roman Semenov, and Roman Storm. "Tornado Cash: Privacy Solution for Ethereum." Whitepaper, Version 1.4, December 2019.

[2] Jens Groth. "On the Size of Pairing-Based Non-interactive Arguments." In *EUROCRYPT 2016*, Vol. 9666 of Lecture Notes in Computer Science, pp. 305-326. Springer, 2016.