Tornado Cash 2.0: Native EVM Privacy-Preserving Lending Platform

Alexander John Lee
November 17, 2024

1 Introduction

Privacy within the Ethereum Virtual Machine (EVM) is critical for fostering a secure, decentralized, and equitable blockchain ecosystem. As decentralized applications (dApps) and decentralized finance (DeFi) protocols grow, protecting user privacy becomes increasingly important. Privacy is a fundamental human right that safeguards individual autonomy and freedom.

Tornado Cash 2.0 is a next-generation Ethereum privacy solution designed to address these concerns. Building upon the original Tornado Cash protocol, it introduces significant enhancements to improve security, efficiency, and user experience. Key innovations include adopting the Poseidon hash function, utilizing the Noir language and PlonK proofs, integrating with Aave for liquidity provision, and enforcing a verifiable one-day liquidity lock.

This white paper presents the design, implementation, and security considerations of Tornado Cash 2.0, highlighting how it advances privacy within the EVM and contributes to a more secure blockchain environment.

2 Protocol Description

Tornado Cash 2.0 implements a privacy-preserving protocol on Ethereum, allowing users to deposit and withdraw Ether (and ERC-20 tokens in the future) without linking the transactions. The protocol enhances privacy by breaking the on-chain link between source and destination addresses using zero-knowledge proofs.

2.1 Key Features and Improvements

- **Poseidon Hash Function**: Replaces MiMC with the Poseidon hash function, offering better performance, transparency, and security optimized for zero-knowledge proofs.
- **Noir and PlonK Proofs**: Utilizes the Noir language for circuit definitions and employs PlonK proofs instead of Groth16, enabling universal and updatable trusted setups.
- **Verifiable One-Day Liquidity Lock**: Ensures user liquidity is locked for a verifiable period of one day, enhancing the anonymity set and temporal unlinkability.

- **Proof Submission on Deposit**: Requires users to submit a proof during deposit, ensuring only valid commitments are added to the Merkle tree.
- **Liquidity Provision to Aave**: Supplies user liquidity to Aave, earning interest over time and providing an incentive mechanism for withdrawals.
- **Incentivized Withdrawals**: Allows anyone to pay for withdrawal transactions and earn a reward from the accrued interest, improving efficiency and user experience.
- **Enhanced Privacy via Aave Integration**: Uses aave.withdraw() to transfer funds, making withdrawals appear as if they originate from Aave, thereby increasing transaction privacy.

2.2 Protocol Functionality

The protocol comprises two main operations:

1. **Deposit**: Users deposit a fixed amount of Ether into the Tornado Cash 2.0 smart contract, submit a zero-knowledge proof, and their funds are supplied to Aave to earn interest.
2. **Withdraw**: After at least one day, users (or any participant) can withdraw the deposited funds by submitting a valid zero-knowledge proof. Withdrawals are funded through Aave, and the transaction fee is incentivized by the interest accrued.

2.3 Setup and Cryptographic Primitives

- **Field Definitions**: All cryptographic operations are performed over a prime field F (BN254).
- **Poseidon Hash Function**: A hash function optimized for zero-knowledge proofs, used for commitments and Merkle tree hashing.
- **Noir Language**: A domain-specific language for writing arithmetic circuits compatible with zero-knowledge proofs.
- **PlonK Proof System**: A universal and efficient zk-SNARK proof system enabling succinct, non-interactive zero-knowledge proofs with updatable trusted setups.
- **Merkle Tree**: A fixed-height Merkle tree (e.g., height 32) stores commitments (leaves) generated during deposits.

2.4 Deposit Process

To deposit funds, a user follows these steps:

1. **Generate Secrets**: Randomly generate a secret s and nullifier k, both 248-bit values.
2. **Compute Leaf Commitment**: Use the Poseidon hash function to compute the leaf commitment incorporating the secret, nullifier, asset type, liquidity amount, and timestamp.

3. **Generate Proof**: Create a zero-knowledge proof using the Noir language and PlonK proofs, proving knowledge of the secrets without revealing them.
4. **Submit Deposit Transaction**: Send a transaction to the Tornado Cash 2.0 contract with the proof and commitment. The contract verifies the proof and updates the Merkle tree.
5. **Supply to Aave**: The contract supplies the deposited funds to Aave, initiating interest accrual.

2.5 Withdrawal Process

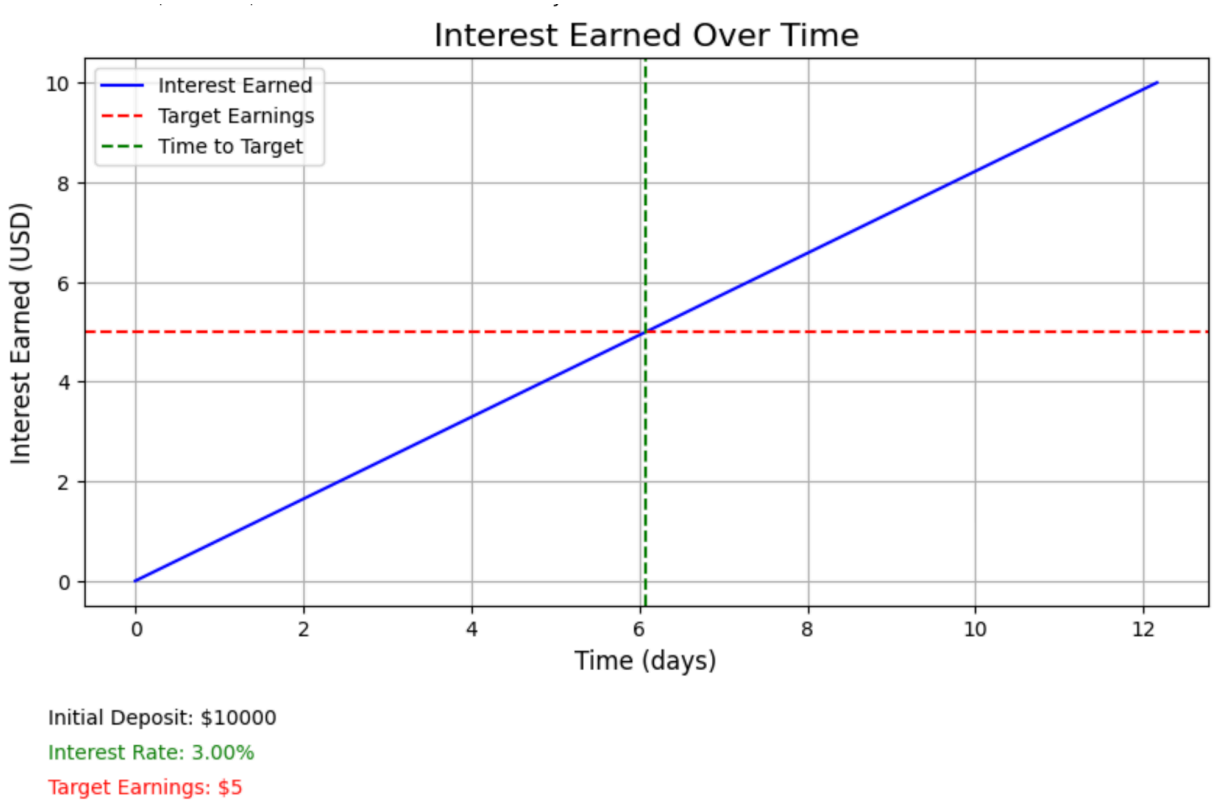
To withdraw funds after the one-day lock period:

1. **Ensure Lock Period Has Passed**: At least one day must have passed since the deposit timestamp.
2. **Generate Proof**: Produce a zero-knowledge proof demonstrating knowledge of a valid commitment in the Merkle tree and that the lock period has elapsed.
3. **Submit Withdrawal Transaction**: Anyone can submit the transaction to the contract with the proof, recipient address, and other required parameters.
4. **Contract Verification**: The contract verifies the proof, ensures the nullifier hasn't been used, and checks the lock period.
5. **Execute Withdrawal via Aave**: Uses aave.withdraw() to transfer funds to the recipient, appearing as if the funds come directly from Aave.
6. **Distribute Rewards**: Allocates a portion of the interest earned to the transaction submitter as an incentive.

3 Incentivized and Decentralized Withdrawals

By allowing anyone to submit the withdrawal transaction and providing a reward sourced from the interest earned by the liquidity position, the protocol ensures withdrawals are processed promptly without requiring the original depositor's involvement. The accrued interest acts as an incentive for participants to pay for the withdrawal proof submission. This mechanism decentralizes the withdrawal process, enhances network participation, and ensures that withdrawals are not dependent on the depositor's actions or availability.

The chart below illustrates the accumulation of interest over time through continuous compounding of an initial deposit in a liquidity pool on Aave. The initial investment generates interest at a specified annual percentage yield (APY), and this earned interest is utilized to pay out rewards and cover gas fees upon withdrawal. The chart demonstrates how the deposited liquidity grows and how the interest can effectively offset transaction costs and offer a reward to users who submit withdrawal proofs to the Tornado Cash 2.0 Vault contract.



4 Enhanced Privacy through Aave Integration

Integrating with Aave adds an additional layer of privacy. Withdrawals executed via aave.withdraw() make the recipient's address appear as if it was funded by Aave, obscuring the origin and enhancing the overall privacy of the transaction. This integration not only improves privacy but also leverages Aave's liquidity and interest-generating capabilities to benefit both the network and its participants.

5 Smart Contract Architecture

The smart contracts are written in Solidity and include:

- **Deposit Contract**: Handles deposits, proof verification, Merkle tree updates, and supplies funds to Aave.
- **Withdrawal Contract**: Verifies withdrawal proofs, interacts with Aave for fund withdrawal, and manages reward distribution.
- **Merkle Tree Management**: Implements the Merkle tree using Poseidon hashes, storing roots and facilitating proof generation.
- Deposit and Withdrawl PlonK Proof Verifiers: Contain the logic for verifying PlonK proofs generated by the Noir circuits.

Integration with Aave

The smart contracts interface with Aave's protocol to supply and withdraw funds:

- **Supply to Aave**: Deposited funds are supplied to Aave, earning interest over time.
- **Withdraw from Aave**: The aave.withdraw() function is used during withdrawals, making the transaction appear as originating from Aave.
- **Interest Accrual**: Interest earned on the deposited funds is used to incentivize withdrawal transaction submitters.

References

1. **Poseidon Hash Function**: Cryptographic hash function optimized for zero-knowledge proofs.
2. **Noir Language**: Domain-specific language for writing circuits compatible with PlonK proofs.
3. **PlonK Proofs**: Universal and efficient zero-knowledge proof system with updatable trusted setups.
4. **Aave Protocol**: Decentralized non-custodial liquidity market protocol.
5. **Tornado Cash**: Original privacy solution for Ethereum transactions.
6. **Ethereum Virtual Machine (EVM)**: Runtime environment for smart contracts on Ethereum.

Note: This white paper provides an overview of Tornado Cash 2.0. For detailed technical specifications and code, please refer to the official GitHub repository and documentation.