TornadoX Cash

TornadoX.Cash Team

April 2021

Abstract

Blockchains are amazing system technologies that allow trustless, secure and efficient data transfer across countless nodes. This is all possible because the data is distributed, making it verifiable. However, while this transparency empowers the entire system, it comes at a huge cost to the individual user - a lack of anonymity. Anyone can look up any transaction at any address on Etherscan, which makes it very easy to trace and investigate personal assets.

Applications such as Tornado.Cash[4] offer a solution. By using Tornado.Cash it is possible to transfer money anonymously. But it has a fatal drawback: in times of rising prices for digital currencies such as ETH, each deposit requires a high GAS fee to be consumed. The cost of implementing an anonymous transfer is constantly increasing.

Therefore, based on the Tornado.Cash protocol, we have introduced TornadoX, which uses a faster and cheaper HASH algorithm with the ability to merge multiple deposits, while reducing the GAS fee.

1 Implemented Solution

Using the above approach, we provide an easy to use solution where the client generates a random note with no user intervention and provides an option to save it to a file. The option is to save it to a file. When confirmed, a provider prompt will validate the sending of funds and transaction data to the chain.

The note is then given to the user who intends to receive the funds.

When user 2 withdraws, the client transparently fetches the deposit information from the TornadoX smart contract. Deposit information is fetched from the TornadoX smart contract, proofs are generated and the provider is used to request the release of funds. Requesting the release of funds from the smart contract. If the user's request can be confirmed, the funds are transferred to the target address.

All information interacting with the contract (including the private key) is public and can be used without a network.

It can be used publicly and does not require a web interface.

2 Technical Overview

Based on the tornado protocol, we have N pools with fixed denominations of ether. In specific smart contract with a fixed amount of M ether, the deposit method allows you to deposit the amount of T times (1-9) of M ether at one time, and you will get T notes at the same time. Each note can be used for a single withdrawal. Thanks to zkSNARK technology, there is no connection between deposit and withdrawal, every transaction is secure and private.

2.1 Hash function

We change non leaf node hash function of Merkle tree from MiMC[1] to poseidon[2] which is more snark-friendly.

Compare with MiMC, poseidon is more efficient which significantly reduces the number of circuit constraints and proof generation time and require less gas on chain.

2.2 Deposit

Generate two random number nullifier and secret, compute preimage = nullifier || secret, commitment = Pedersen(preimage). User can call deposit function with parameter commitment and fixed denomination of ether. If using TornadoX's UI, user will get a note consist of commitment, "TornadoX", network id and related pool name.

2.3 Bulk deposit

We have an innovative change to tornado: when depositing, user can choose the number of deposits. When depositing multiple deposits at the same time, the gas fee will not increase linearly but only slightly.

We test deposits of different lengths on the goerli testnet and compare the results with the gas used of deposits of the same length on tornado:

deposit	Tornado		TornadoX	
length	total	one deposit	total	one deposit
1	979456	979456	823670	823670
2	1958912	979456	847101	423551
3	2938368	979456	908004	302668
4	3917824	979456	991861	247965
5	4897280	979456	1151415	230283
6	5876736	979456	1082749	180458
7	6856192	979456	1200909	171558
8	7835648	979456	1220830	152603
9	8815104	979456	1309542	145504

Table 1: Gas used on different length deposits on goerli testnet[1]

Compare with deposit on Tornado, Tornado X require 84.09% gas on single deposit, 14.86% gas on 9 deposits, 21.64% for average one time deposit.

2.4 Withdrawal

Parse note to restore nullifier and secret. Generate public parameters:

- root: catch deposit events of specific pool as leaves to generate Merkle tree and compute root
- nullifierHash: Pedersen(nullifier)
- recipient: recipient's address
- relayer: relayer address[2]
- fee: relayer's fee

Generate private parameters:

- nullifier: nullifier
- secret: secret
- pathElements: generated by pathIndices and Merkle tree
- pathIndices: find commitment of deposit events fetched on public parameters which is equal to commitment concatenated by nullifier and secret and use its leaf index as parameter

Compute proof using Groth16[3] by public and private parameters. Call withdraw by proof and public parameters and recipient's address will receive (M - fee) ether.

3 Token Overview

Usually the developers of the project will hold a part of the tokens, which is why the developers can get huge benefits after the tokens enter the secondary market. The TornadoX developer team will not participate in the distribution of tokens, and any tokens should be obtained from events that are beneficial to increase anonymity, such as mining or airdrops.

The token has the following, but not limited to, use-cases:

1. **Governance** - We want TornadoX to evolve over time and adapt to user needs. To achieve this, the token will be used to draft proposals and vote on how the service should change. The direction of the product will be driven by the community.

[1]Original data with transaction hash: `https://github.com/TornadoXCash/TornadoX/blob/main/original_data`

[2]Set relayer and fee to 0 will not make extra cost, but a little less anonymity.

2. **Increasing Anonymity** - The token is to be used as rewards for users providing funds to the TornadoX contracts. Provided funds actively help with increasing the anonymity set and securing the usage of other users. To encourage this, the token will be attributed to those users.

References

[1] Martin Albrecht et al. "MiMC: Efficient Encryption and Cryptographic Hashing with Minimal Multiplicative Complexity". In: *Cheon J., Takagi T. (eds) Advances in Cryptology – ASIACRYPT 2016.* Vol. 10031. Springer, Berlin, Heidelberg, 2016, pp. 191–219.

[2] Lorenzo Grassi et al. *Poseidon: A New Hash Function for Zero-Knowledge Proof Systems*. Cryptology ePrint Archive, Report 2019/458. `https://eprint.iacr.org/2019/458`. 2019.

[3] Jens Groth. *On the Size of Pairing-based Non-interactive Arguments*. Cryptology ePrint Archive, Report 2016/260. `https://eprint.iacr.org/2016/260`. 2016.

[4] Tornado Cash Team. *Tornado Cash*. `https://tornado-cash.medium.com/introducing-private-transactions-on-ethereum-now-42ee915babe0`. Apr. 2020.