IBIMA Publishing
Communications of International Proceedings
https://ibimapublishing.com/p-articles/44SE/2024/4433824/
Vol. 2024 (24), Article ID 4433824

Security Analysis of Miksi – Ethereum Zero Knowledge Transaction Mixer*

Kamil KACZYŃSKI

Military University of Technology, Faculty of Cybernetics,
Institute of Mathematics and Cryptology, Poland

Correspondence should be addressed to: Kamil KACZYŃSKI, kamil.kaczynski@wat.edu.pl

* Presented at the 44th IBIMA International Conference, 27-28 November 2024 Granada, Spain

Abstract

Miksi is a decentralized blockchain protocol that uses zero knowledge proofs (ZKP) and cryptographic commitments to enable anonymous transactions within a secure decentralized network. The system consists of three components: the main application (*miksi.ts*), the ZKP circuits (*deposit* and *withdraw.circom*), and the smart contracts (*Miksi.sol*, *deposit-verifier.sol*, and *withdraw-verification.sol*). Each component plays a pivotal role in the overall security of the system, and we evaluate them for common vulnerabilities such as improper input validation, reentry attacks, and weaknesses in proof verification. Through this analysis, we highlight both the strengths and potential areas for improvement in ensuring secure, anonymous cryptocurrency transactions.

Keywords: Ethereum, ZKP, blockchain, DeFi, anonymity, mixer

Introduction

As privacy-centric cryptocurrencies continue to gain traction, the need for secure and anonymous transaction mechanisms becomes increasingly critical. Miksi [1] addresses this requirement by integrating Zero-Knowledge Proof (ZKP) technology within its framework, enabling users to conduct transactions without disclosing sensitive details, such as transaction amounts and wallet addresses. This solution guarantees a high degree of privacy using cryptographic techniques and decentralized smart contracts to securely process and verify deposits and withdrawals.

This paper provides a comprehensive evaluation of the Miksi security framework, focusing on three pivotal components: the main application (miksi.ts), the ZKP circuits (deposit.circom and withdraw.circom), and the smart contracts (Miksi.sol, deposit-verifier.sol, and withdraw-verifier.sol).). Each component plays a crucial role in securing the system overall, and we evaluate them for common vulnerabilities, including improper input validation, reentry attacks, and deficiencies in proof verification. In the following sections, we systematically analyze the Miksi architecture, identifying potential security weaknesses, and proposing actionable enhancements.

Miksi Architecture

Miksi is a cryptocurrency protocol that leverages zero knowledge proofs and cryptographic commitments to allow anonymous transactions within a decentralized network. Its architecture is centered on the concept of ensuring that both deposits and withdrawals are cryptographically verifiable without revealing sensitive information about participants and transaction amounts. The main building elements of Miksi are:

Cite this Article as: Kamil KACZYŃSKI, Vol. 2024 (24) "Security Analysis of Miksi – Ethereum Zero Knowledge Transaction Mixer " Communications of International Proceedings, Vol. 2024 (24), Article ID 4433824, https://doi.org/10.5171/2024.4433824

- Commitment and nullifier
- Zero-knowledge proof circuits
- Merkle tree for commitment storage
- ZKP verifier contracts

All of those are being used to build the interaction workflow for deposits and withdrawals of funds. The deposit workflow follows the following steps:

1. The user generates a $secret$ and computes $commitment$=$H(coinCode,amount,secret,nullifier)$.
2. User submits zero-knowledge proof $\Pi_{deposit}$ Proving the correctness of the commitment and sending the corresponding amount of cryptocurrency to the contract.
3. The smart contract verifies that ZKP updates Merkle root and stores the commitment.

In the event of withdrawal, the workflow consists of the following steps:

- The user proves ownership of a commitment using zero-knowledge proof $\Pi_{withdraw}$. This ZKP proves that the user knows the value of the key and $secret$ correspothe commitment.$ment$.
- The contract verifies the proof, checks the uniqueness of the nullifier and transfers the corresponding amount of cryptocurrency to the user.

Miksi authors state that commitment-nullifier scheme ensures privacy while preventing double-spending. The core variables used in that process are:

- $key \in F_{p^*}$ - the user's private key or identifier.
- $secret \in F_{p^*}$ - Randomly generated secret associated with the deposit.
- $commitment \in F_{p^*}$ - a cryptographic value representing the deposit. It is generated using the Poseidon [2] hash function H. The value is calculated as $commitment$=$H(coinCode,amount,secret,nullifier)$. This value is stored in the Merkle tree and used later for proving ownership without revealing underlying $secret$ or key.
- $nullifier \in F_{p^*}$ - a cryptographic value that prevents double spending. It is calculated as $nullifier$=$H(key,secret)$. This value is stored in a contract to ensure it is not reused for double spending.

Miksi uses a Sparse Merkle Tree (SMT) [3] to efficiently store cryptographic commitments on the chain. The SMT is initialized with an empty root and is updated as new deposits are added. Each deposit corresponds to a leaf in the Merkle tree, where the path to the leaf is determined by the user's unique key. Given that commitments are leaves in the tree, a Merkle inclusion proof is generated for each withdrawal to prove that the corresponding commitment is part of the tree.

The Merkle tree root is updated with each deposit or withdrawal, maintaining consistency in the chain state. The calculation of the root follows the standard recursive hash calculation used in Merkle trees:

$$root = H(commitment, sibling)$$

where the sibling refers to the adjacent nodes at each level of the tree along the path from the leaf to the root.

Miksi uses two ZKP verifier contracts, one for deposits and one for withdrawals, to verify Groth16 proofs [4] on-chain. These verifier contracts are generated using specialized tools that compile the off-chain ZKP circuits into solidity contracts. Each verifier contract checks the cryptographic validity of the proof based on the verification key generated during the ZKP setup phase.

For deposit verification, the on-chain verifier checks the following:

- The proof is valid under the following constraint:
$$H(key, secret) = nullifier$$
- The commitment is correctly added to the Merkle tree.
- The new Merkle root is consistent with the previous state.

For withdrawals, the verifier ensures:

- The nullifier has not been used before.
- The Merkle proof shows a valid inclusion of the $commitment$ in the tree.

- The proof of knowledge of the key and $secret$ is correct.

In the next section we will analyze if Miksi protocol is vulnerable to attacks, which may lead to deanonymization of user transaction or allowing malicious parties to disable the service and block suer funds.

Security Analysis

Miksi was designed in a way that prevents attacks such as reentry attacks [5] and double spending [6]. However, some assumptions made by the author can cause deanonymization of transactions or perform attacks that disable the service. In this section, we will focus on describing a subset of possible attacks which should be addressed before using Miksi for effective transaction anonymization.

All deposits and withdrawals are for an exact amount of 1 ETH. This is a by-design limitation introduced by the authors to decrease the possibility of tracking and correlating the transaction. If the number of pool users is low, or if deposits and withdrawals happen in close timeframes, an observer could correlate a withdrawal with a previous deposit, breaking the anonymity. To mitigate it, authors should implement variable deposit and withdrawal amounts, this preventing exact matching of deposit and withdrawal amounts. It would also be beneficial if some delays or batching could be implemented in withdrawals, so withdrawals occur at unpredictable times.

The attacker could also perform a Sybil attack [7], where fake accounts are created making deposits in the Miksi pool. By depositing and withdrawing multiple times, the attacker can observe the system and potentially infer who the real users are by monitoring their fake accounts to interact with the contract or overwhelm the anonymity pool, thus making it easier to trace the withdrawals of real users based on the remaining participants. To mitigate that type of attack, Miksi should implement deposit limits such as the minimum time between deposits or withdrawals or use anti-Sybil mechanisms requiring users to lock up tokens for some time before they can make multiple deposits.

We identified that Miksi may be vulnerable to front-running [8] and withdrawal censorship attacks [9]. The smart contract utilizes the $send$ metto sendding to the users, which is a choice made to mitigreentryancy attacks [10], but it introduces new vulnerabilities. An attacker, miner, or someone with access to the Ethereum mempool could front-run withdrawal transactions by submitting their own transactions with higher gas prices, potentially allowing them to alter the state of the contract before the legitimate user's transaction is executed. An attacker could also censor withdrawal transactions by ensuring that the legitimate transaction never gets processed, especially if the withdrawal pool is congested. Those attacks can be mitigated by switching to the $call$ method, which provides more flexibility and better error handling. Another way to mitigate this type of attack is to use gas-efficient batching mechanisms for withdrawals, making it more difficult to predict and start-up individual withdrawals. A good form of mitigation would also be to introduce a random delay before withdrawals, thus making timing attacks more complicated.

Even if the contract is designed to work with Ethereum network, we cannot ignore the possibility of creating a successful replay attack. A replay attack involves taking a valid transaction from one blockchain (or network fork) and replaying it on another blockchain, which could cause unwanted or malicious withdrawals. For example, if Miksi is deployed on both Ethereum and a sidechain or fork, an attacker might be able to replay valid deposit or withdrawal proofs on a different chain or fork, causing duplicate transactions. This kind of attack could be mitigated by including chain-specific data (such as the Ethereum chain ID or network ID) in the zero-knowledge proof, $nullifier$, or $commitment$ to ensure that proofs are only valid on a specific chain.

When deploying such a pool, authors should keep in mind that the size of the anonymity set is critical to the effectiveness of any mixing service. If the pool only has a few participants, it becomes easier to identify or infer who is depositing and withdrawing, breaking the anonymity. To make deanonymization more complicated, the author should encourage a larger pool size by promoting the service and incentivizing users to participate. It may be worth rewarding users for longer withdrawal delays or using additional privacy-preserving techniques to strengthen anonymity.

To conclude, while the Miksi contract is secure in terms of basic functionality, several theoretical vulnerabilities could compromise its anonymity or lead to denial of service under specific conditions. Implementing flexible

deposit/withdrawal amounts, input validation, and optimized gas usage can address these attack vectors. Additionally, addressing the issues of small anonymity sets and potential replay attacks will help ensure that the Miksi system can operate securely in a variety of environments.

Conclusions

The examination of Miksi, an Ethereum-based cryptocurrency mixer, underscores a robust foundation predicated on zero-knowledge proofs (ZKP) and cryptographic commitments to uphold user privacy. Although Miksi illustrates substantial protection against attacks such as reentry and double spending, numerous prospective vulnerabilities must be addressed to augment its security and anonymity assurances.

Primarily, the use of fixed deposit amounts (1 ETH) introduces risks associated with transaction correlation, potentially compromising user anonymity in smaller pools. The implementation of variable deposit and withdrawal amounts, coupled with random delays, could mitigate such risks, thereby preventing observers from correlating transactions. Moreover, the system is prone to Sybil attacks, where attackers generate multiple fictitious identities to manipulate the anonymity pool and trace legitimate transactions. Mitigation strategies, such as deposit limits, token lock-up periods, and anti-Sybil mechanisms, could reinforce the system against such attacks.

Although Miksi's architecture minimizes reentrancy risks through the use of the send method, it generates new vulnerabilities, particularly front-running and withdrawal censorship. By transitioning to the call method and executing gas-efficient batching and withdrawal randomization, these risks can be alleviated, ensuring smoother and more secure transactions. In addition, the system poses a potential vulnerability to replay attacks, especially across disparate forks or blockchains. Incorporating chain-specific data within the nullifiers or commitments can ensure that proofs are valid solely on the intended chain, thereby mitigating this risk.

In summary, Miksi's use of ZKP technology is promising for anonymous cryptocurrency transactions; however, practical improvements in its input validation, gas efficiency, and anonymity protection will be essential for its long-term security and adoption. Future efforts should focus on realizing these improvements to further strengthen the resilience of the system against sophisticated attack vectors.

References

- Miksi source code repository, https://github.com/arnaucube/miksi-core, accessed 21.10.2024
- Grassi, L., Khovratovich, D., Rechberger, C., Roy, A., & Schofnegger, M. (2021). Poseidon: A new hash function for {Zero-Knowledge} proof systems. In *30th USENIX Security Symposium (USENIX Security 21)* (pp. 519-535).
- Dahlberg, R., Pulls, T., & Peeters, R. (2016). Efficient sparse merkle trees: Caching strategies and secure (non-) membership proofs. In *Secure IT Systems: 21st Nordic Conference, NordSec 2016, Oulu, Finland, November 2-4, 2016. Proceedings 21* (pp. 199-215). Springer International Publishing.
- Groth, J. (2016). On the size of pairing-based non-interactive arguments. In *Advances in Cryptology–EUROCRYPT 2016: 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Vienna, Austria, May 8-12, 2016, Proceedings, Part II 35* (pp. 305-326). Springer Berlin Heidelberg.
- Samreen, N. F., & Alalfi, M. H. (2020, February). Reentrancy vulnerability identification in ethereum smart contracts. In *2020 IEEE International Workshop on Blockchain Oriented Software Engineering (IWBOSE)* (pp. 22-29). IEEE.
- Chohan, U. W. (2021). The double spending problem and cryptocurrencies. *Available at SSRN 3090174*.
- Douceur, J. R. (2002, March). The sybil attack. In *International workshop on peer-to-peer systems* (pp. 251-260). Berlin, Heidelberg: Springer Berlin Heidelberg.
- Eskandari, S., Moosavi, S., & Clark, J. (2020). Sok: Transparent dishonesty: front-running attacks on blockchain. In *Financial Cryptography and Data Security: FC 2019 International Workshops, VOTING and WTSC, St. Kitts, St. Kitts and Nevis, February 18–22, 2019, Revised Selected Papers 23* (pp. 170-189). Springer International Publishing.
- Kim, J. S., Shin, J. M., Choi, S. H., & Choi, Y. H. (2022). A study on prevention and automatic recovery of blockchain networks against persistent censorship attacks. *IEEE Access*, *10*, 110770-110784.
- Samreen, N. F., & Alalfi, M. H. (2020, February). Reentrancy vulnerability identification in ethereum smart contracts. In *2020 IEEE International Workshop on Blockchain Oriented Software Engineering (IWBOSE)* (pp. 22-29). IEEE.