



Anonymous Transaction System in Public Blockchain with Proof of Funds Origin Trustworthiness

M.D. Vasilyev, ORCID: 0009-0002-9786-515X <vslymax@itmo.ru>
I.V. Struchkov, ORCID: 0000-0001-9142-9440 <ivstruchkov@itmo.ru>

*Institute of Applied Computer Science, ITMO University,
49A (liter A), Kronverkskiy prospekt, St. Petersburg, 197101, Russia.*

Abstract. The development is described of decentralized financial systems based on distributed ledger technologies and blockchain technologies allows creating trusted systems that are resistant to external attacks and manipulation. In public blockchain networks, each user's transactions are publicly available, which limits the possibility of their use for financial settlements. The solution to this problem is to use decentralized protocols for anonymous transactions. The disadvantage of existing protocols is that honest users do not have the opportunity to dissociate themselves from criminal financial activities. One of the ways to solve this problem is to use association sets, the implementation of which is discussed in this article. Zero-knowledge proofs, association sets based on Merkle trees with Poseidon hash function. A system of anonymous transactions in a public blockchain has been designed and implemented with the ability to prove the good faith of the origin of funds. The system is based on the concept of association sets and uses zero-knowledge proofs to provide cryptographic guarantees. Arithmetic circuits in the Circom 2.0 language are implemented to generate proofs of membership in an association set, along with smart contracts in the Solidity language. A decentralized governance system based on participant voting has been developed. A comparative analysis with the existing solution showed that the cost of deposit transactions in the developed protocol is lower than in the original one, which is explained by the use of a more optimal Poseidon hashing function. When making withdrawals, the cost of transactions increases due to the additional verification of the association set root signature and of a zero-knowledge proof covering two Merkle trees. The developed system can be used to comply with regulatory requirements while maintaining the anonymity of users of distributed ledgers.

Keywords: blockchain; transaction anonymization; association sets; anonymity; Merkle tree; zero-knowledge proofs; smart contract.

For citation: Vasilyev M.D., Struchkov I.V. Anonymous Transaction System in Public Blockchain with Proof of Funds Origin Trustworthiness. Trudy ISP RAN/Proc. ISP RAS, 2026, vol. 38, issue 6, part 1, pp. 299-310. DOI: 10.15514/ISPRAS-2026-38(6)-19.

Система для проведения анонимных транзакций в публичной блокчейн-сети с доказательством благонадежности происхождения средств

М.Д. Васильев, ORCID: 0009-0002-9786-515X <vslymax@itmo.ru>
И.В. Стручков, ORCID: 0000-0001-9142-9440 <ivstruchkov@itmo.ru>

*Институт прикладных компьютерных наук, Университет ИТМО,
Россия, 197101, Санкт-Петербург, Кронверкский пр., д. 49, лит. А.*

Аннотация. Описывается разработка децентрализованных финансовых систем на базе технологий распределенных реестров и блокчейн-технологий позволяет создавать доверенные системы, устойчивые ко внешним атакам и манипуляциям. В публичных блокчейн-сетях транзакции каждого пользователя общедоступны, что ограничивает возможность их применения для финансовых взаиморасчетов. Решением этой проблемы является использование децентрализованных протоколов для анонимных транзакций. Недостатком существующих протоколов является тот факт, что у честных пользователей нет возможности отмежеваться от преступной финансовой деятельности. Одним из способов решения этой проблемы является использование ассоциативных множеств, реализация которого рассматривается в данной статье. Приводятся доказательства с нулевым разглашением, ассоциативные множества на базе деревьев Меркла с хэш-функцией Poseidon. Спроектирована и реализована система анонимных транзакций в публичном блокчейне с возможностью доказательства добросовестности происхождения средств. Система основана на концепции ассоциативных множеств и использует доказательства с нулевым разглашением для обеспечения криптографических гарантий. Реализованы арифметические схемы на языке Circom 2.0 для генерации доказательств принадлежности ассоциативному множеству и смарт-контракты на языке Solidity. Разработана система децентрализованного управления на основе голосования участников. Сравнительный анализ с существующим решением показал, что стоимость транзакций внесения депозита в разработанном протоколе меньше, чем в оригинальном, что объясняется применением более оптимальной функции хэширования Poseidon. При выполнении вывода средств стоимость транзакций увеличивается из-за дополнительных проверок: подписи корня ассоциативного множества и доказательства с нулевым разглашением сразу для двух деревьев Меркла. Разработанная система может применяться для соблюдения регуляторных требований при сохранении анонимности пользователей распределенных реестров.

Ключевые слова: блокчейн; анонимизация транзакций; ассоциативные множества; анонимность; дерево Меркла; доказательства с нулевым разглашением; смарт-контракт.

Для цитирования: Васильев М.Д., Стручков И.В. Система для проведения анонимных транзакций в публичной блокчейн-сети с доказательством благонадежности происхождения средств. Труды ИСП РАН, 2026, том 38, вып. 6, часть 1, стр. 299–310 (на английском языке). DOI: 10.15514/ISPRAS-2026-38(6)-19.

1. Introduction

In recent decades, distributed ledger technologies have become an important part of the global financial system and represent a point of growth for digital transformation across various sectors of the economy. Due to their fundamental properties – decentralization, immutability, and transparency – distributed ledgers [1, 2] provide the ability to create trusted systems that are resistant to external attacks and manipulation. This opens broad prospects for their application in financial technologies, logistics, government services, and other industries where security and reliability are of paramount importance.

At the same time, in public blockchain networks, the existence of a publicly available database containing every transaction of every user creates the possibility of deanonymizing users through transaction history analysis, violating the requirement of operational privacy, which limits the applicability of such networks in financial settlement systems.

These problems have led to the emergence of decentralized protocols that enhance user anonymity. Such protocols allow users to deposit digital assets from one user address and then withdraw them from another address that is in no way linked to the first, thereby breaking the connection between the source and recipient of assets. Although all transactions are still public and visible to all network participants, it becomes impossible to directly establish a link between a deposit and a withdrawal. The most popular protocol implementing this idea is Tornado Cash [3]. Nevertheless, in addition to honest users trying to protect their anonymity on the network, malicious actors who obtained their funds through criminal means also began using this protocol [4]. This led to the smart contract address of the protocol being added to the “Specially Designated Nationals and Blocked Persons List” of the U.S. government (also known as the SDN list). Although Tornado Cash was removed from the SDN list in March 2025, after a federal appellate court ruled that OFAC had exceeded its statutory authority in sanctioning the protocol [5], the episode clearly demonstrated the regulatory risks that such protocols pose to their honest users.

In essence, the critical problem of Tornado Cash was that honest users had limited ability to dissociate themselves from the criminal activity that the protocol attracted.

There are several approaches to solving this problem. Some of them include:

1. **Deposit limit.** A standard approach used by many applications is to set a limit on the amount that can be deposited in a single operation or over a given time period. This allows restricting the flow of assets from unreliable sources. However, some amount of unverified assets can still be deposited. Moreover, this approach significantly restricts honest users who use the protocol as a tool for enhancing anonymity on the network.
2. **Address blacklist.** Companies involved in blockchain transaction analysis, such as Chainalysis [6] and TRM Labs [7], provide tools and services that allow tracking the movement of criminally obtained funds. These tools can help regulators, such as OFAC, maintain blacklists of addresses from which deposits should not be accepted. However, circumventing this precaution would not be a difficult task, as these lists are not updated in real time. Furthermore, it is unclear what the mechanism should be for removing erroneously blacklisted addresses.
3. **View-only access for regulators.** Some user anonymity enhancement protocols employ an approach in which any network participant who has deposited and withdrawn funds from the system can provide the regulator with a so-called view key, which allows deanonymizing the data stored on the network (including verifying that a specific withdrawal is linked to a specific deposit). Although this approach allows honest users to prove to the regulator that they are not involved in illegal activity, it does not prevent the protocol from being used by users who obtained funds through criminal means.
4. **Selective deanonymization.** This approach [8] is based on the idea that the regulator has some mechanism that allows them to request a key with which they can analyze the movement of funds within the system. The adequacy of the regulator’s actions is monitored by a special committee, which determines through quorum-based voting whether to provide the regulator with the requested key for a specific transaction. This approach has the same problem – it allows deanonymizing malicious transactions but does not prevent them. In addition, at the current stage of development of decentralized financial systems, the practical implementation of the aforementioned committee is challenging.
5. **Association sets [9].** This approach gives the regulator full control over anonymous withdrawals from the system – in order for a user to withdraw their deposit to an arbitrary address, it must be pre-approved by the regulator (or one of the regulators). Unlike the approaches described above, it prevents the anonymous withdrawal of illicit funds rather than merely enabling their post-hoc deanonymization, while the force quit mechanism protects honest users from losing their deposits.

In this paper, we propose an implementation of a decentralized protocol based on the use of association sets as described by Buterin et al. [9]. Additionally, we propose an implementation of a decentralized governance system that allows the community of protocol users to make management decisions regarding it. The developed protocol is compared with the Tornado Cash protocol in terms of the cost of deposit and withdrawal operations to verify its suitability for use in real-world blockchain networks.

The source code of the developed protocol is available on GitHub [10].

2. Method for Solving the Problem Using Association Sets

This section will describe a method for increasing user anonymity, allowing users to avoid unnecessary attention from the regulator using Association Sets.

2.1 Two Merkle trees

An association set in the context of this article is a collection of transactions whose legitimacy has been verified by an external observer, referred to in this work as an Association Set Provider (ASP). Only for transactions from this set can assets be withdrawn from the system to an address different from the one from which the deposit was made. If a transaction has not been approved, the assets can only be withdrawn back to the same address from which the transaction was initiated (this operation is called “Force Quit”).

This approach gives the regulator full control over anonymous withdrawals from the system, while the force quit mechanism ensures that users do not risk losing their funds in the absence of regulatory approval.

Using zero-knowledge proofs, even given that the regulator does not know which deposit is linked to which withdrawal, after the deposit is approved, the user gains the ability to withdraw funds to an arbitrary address.

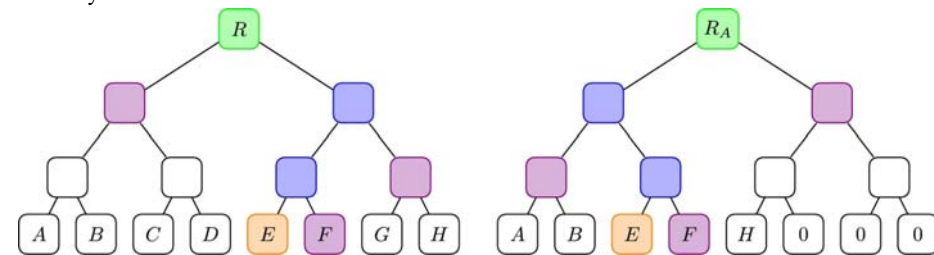


Fig. 1. The user proves the presence of their deposit in the commitment tree (left) and the same deposit (with the same hash) in the tree provided by the provider (right).

To withdraw funds from the protocol, the user creates a zk-SNARK proof [11, 12] of two facts:

1. The deposit is in the set of all deposits made (the tree with root R in Figure 1).
2. The deposit is in the set of deposits approved by the provider (the tree with root R_A in Figure 1).

The smart contract deployed on the blockchain network verifies this proof, and if it is correct, allows the user to withdraw their deposit to an arbitrary address.

As part of the security audit of the arithmetic circuit used in Tornado Cash, certain recommendations were made regarding the hash functions used and some other technical details [13]. Specifically:

1. It is recommended to use the Poseidon hash function [14] as the hash function used in the Merkle tree.

- For computing the commitment and nullifierHash, it is recommended to use the following formulas:

$$\text{Commitment} = \text{PoseidonHash}(\text{nullifier}, 0)$$

$$\text{nullifierHash} = \text{PoseidonHash}(\text{nullifier}, 1, \text{leafIndex})$$

In this work, these formulas are applied, and the Merkle tree [15] is built using the Poseidon hash function.

Thus, the complete algorithm of the protocol is as follows in the next section.

2.2 Deposit algorithm

- The user generates a random number that will serve as the nullifier. The nullifier must be kept secret by the user, as it can be used to withdraw a previously made deposit.
- The user computes the commitment as $\text{PoseidonHash}(\text{nullifier}, 0)$.
- The user calls the smart contract's deposit function and passes the computed commitment along with a certain amount of ether. If this deposit has not been made before, there is free space in the Merkle tree storing commitments, and the user has sufficient ether balance, the protocol approves the deposit and adds the next commitment to the tree. After making the deposit, the user receives a Note – a string containing information about the deposit. The user must keep it secret, as anyone who has it can withdraw the deposit. The Note structure includes:
 - tokenSymb (deposit token identifier),
 - denomination (number of tokens deposited),
 - nullifier (the number generated in the first step),
 - leafIndex (the commitment's index in the Merkle tree stored in the smart contract),
 - chainId (the blockchain ID where the smart contract is deployed),
 - tornadoAddress (the smart contract address).

2.3 Withdrawal algorithm

- The user computes the nullifierHash using the formula above. For this, they use the leafIndex from the received Note.
- The user computes a proof (zk-SNARK Proof) that their deposit is among all previously made deposits (namely, that they know a nullifier from which the commitment was derived through hashing, as well as a Merkle path starting from one of the historical Merkle tree roots and ending at the leaf with the specified commitment), and also among deposits approved by the ASP (Association Set Provider). More details on this step are provided below.
- The user calls the smart contract's withdraw function, passing the nullifierHash, the zk-SNARK proof, and the address to which the funds should be withdrawn.

The smart contract verifies the proof, and if it is correct and the nullifierHash is not in the list of spent nullifiers, the funds are withdrawn to the specified address, and the nullifierHash is added to the list of used ones.

3. Architecture of the Developed System

The developed system consists of three main components (see Fig. 2): cryptographic circuits for generating and verifying proofs, smart contracts for managing deposits and withdrawals for ETH and ERC-20 tokens, and smart contracts for the governance system.

The cryptographic circuits are implemented in the Circom 2.0 language [16] and include: a deposit circuit for generating commitments, a withdrawal circuit with proof of Merkle tree membership, and an association set verification circuit. The Poseidon hash function [14] is used for hashing, optimized for arithmetic circuits due to the minimal number of multiplication operations.

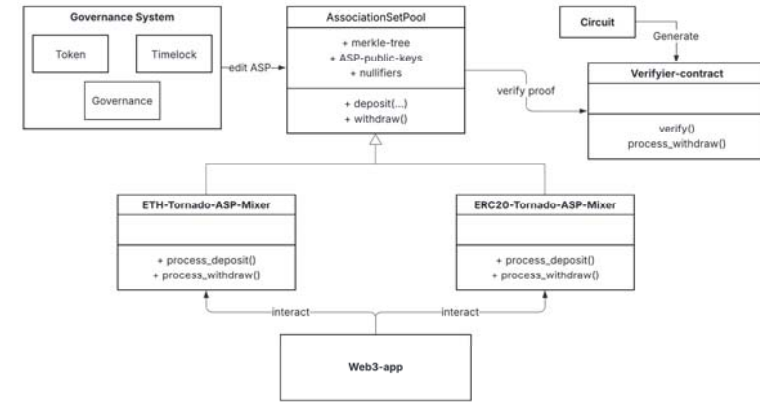


Рис. 2. Архитектура разработанной системы.
Fig. 2. Architecture of the developed system.

The smart contracts are developed in the Solidity language [17] and deployed on an EVM-compatible blockchain [18]. The main contract, AssociationSetPool, implements the logic for deposits and withdrawals with zk-SNARK proof verification.

4. Development of the Arithmetic Circuit

The arithmetic circuits for zk-SNARK are implemented in the Circom 2.0 language. The deposit circuit computes the commitment as the Poseidon hash of a secret nullifier value (generated by the user immediately before making the deposit) and the number 0: $\text{commitment} = \text{Poseidon}(\text{nullifier}, 0)$. Upon withdrawal, the user reveals $\text{nullifierHash} = \text{Poseidon}(\text{nullifier}, 1, \text{leafIndex})$, which prevents reuse of the deposit. leafIndex is the leaf number in the tree.

As mentioned earlier, the protocol at the withdrawal stage must verify that the withdrawer knows the nullifier that produces a commitment present in two Merkle trees – the deposit tree and the ASP tree. This logic can be achieved using the following Circom template (for brevity, the listing omits the public signals recipient, relayer, and fee, which in the actual implementation [10] are bound into the proof statement to prevent the substitution of withdrawal parameters by third parties):

```

1. template Withdraw(levels) {
2.   signal input root;
3.   signal input nullifierHash;
4.   signal input nullifier;
5.   signal input pathElements[levels];
6.   signal input pathIndices[levels];
7.   signal input assocSetRoot;
8.   signal input pathElementsAssoc[levels];
9.   signal input pathIndicesAssoc[levels];
10.
11.   component leafIndexNum = Bits2Num(levels);
12.   for (var i = 0; i < levels; i++) {
13.     leafIndexNum.in[i] <== pathIndices[i];
14.   }
15.
16.   component nullifierHasher = Poseidon(3);
17.   nullifierHasher.inputs[0] <== nullifier;

```

```

18.    nullifierHasher.inputs[1] <== 1;
19.    nullifierHasher.inputs[2] <== leafIndexNum.out;
20.    nullifierHasher.out === nullifierHash;
21.
22.    component commitmentHasher = Poseidon(2);
23.    commitmentHasher.inputs[0] <== nullifier;
24.    commitmentHasher.inputs[1] <== 0;
25.
26.    component tree = MerkleTreeChecker(levels);
27.    tree.leaf <== commitmentHasher.out;
28.    tree.root <== root;
29.    for (var i = 0; i < levels; i++) {
30.        tree.pathElements[i] <== pathElements[i];
31.        tree.pathIndices[i] <== pathIndices[i];
32.    }
33.
34.    component treeAssoc = MerkleTreeChecker(levels);
35.    treeAssoc.leaf <== commitmentHasher.out;
36.    treeAssoc.root <== assocSetRoot;
37.    for (var i = 0; i < levels; i++) {
38.        treeAssoc.pathElements[i] <== pathElementsAssoc[i];
39.        treeAssoc.pathIndices[i] <== pathIndicesAssoc[i];
40.    }
41. }
42.
43. component main {public [root, nullifierHash, assocSetRoot]} = Withdraw(20);

```

Justification of the correctness of this template:

- Lines 16–20 verify the consistency of the provided **nullifier**, **leafIndexNum**, and **nullifierHash**. Thus, satisfying these constraints proves that the user possesses a **nullifier** that, together with the **leafIndex**, yields the **nullifierHash**.
- Next, there are two Merkle proofs: lines 26–32 and 34–40 for trees with roots **root** and **assocSetRoot**, respectively. In each proof, it is verified (lines 27 and 35) that from the nullifier possessed by the user, it is possible to recover not only the **nullifierHash** (as described above), but also the **commitment** whose path is passed into the circuit.
- The subsequent path verification (lines 29–32 and 37–40) shows that the path passed into the circuit is indeed a correct Merkle path from the given leaf to the root.
- Thus, we can conclude that if the roots **root** and **assocSetRoot** are historical roots of the corresponding Merkle trees, then from a correct proof of the Circom component, one can assert that the user's **commitment** is indeed present in the corresponding trees, since the user possesses a **nullifier** and **leafIndex** from which the **commitment** can be recovered through hashing, and then through pairwise hashing along the Merkle path, one can reach the publicly known root.

5. Development of the Main Protocol Smart Contract

The main operations that the protocol under development must support are the deposit and withdrawal of a previously made deposit. This section describes how they work.

5.1 Deposit operation

The deposit function code is as follows:

```

1. function deposit(bytes32 _commitment) external payable nonReentrant {
2.     uint32 insertedIndex = _insert(_commitment);
3.     _processDeposit();
4.     emit Deposit(_commitment, insertedIndex, block.timestamp);
5. }

```

Description of this code:

- Line 2: The deposit is added to the Merkle tree.
- Line 3: In the processDeposit function for the ETH contract, only a check is performed to ensure the amount sent in the transaction matches the amount specified in the contract itself (the deposit amount is fixed). In the ERC20 token contract, an explicit transfer of tokens from the token smart contract to the main protocol contract address is performed, and it is verified that no ETH was sent in the transaction itself.
- Line 4: A deposit event is added to the blockchain. Using these events, the ASP can track occurring deposits, and a Web3 application can obtain the leafIndex for made commitments.

5.2 Withdrawal operation

The signature of the withdrawal function is as follows:

```

1. function withdraw(
2.     Proof calldata proof,
3.     bytes32 root,
4.     bytes32 nullifierHash,
5.     address payable recipient,
6.     address payable relayer,
7.     uint256 fee,
8.     RootWithSignature calldata assocSetRootWithSignature,
9.     address providerPublicKey
10. ) external payable nonReentrant

```

The function operates according to the following algorithm:

- Checks are performed:
 - the fee amount is less than the withdrawal amount;
 - the nullifierHash is not in the list of spent nullifiers, meaning the deposit has not been previously withdrawn;
 - the provided root parameter is a historical root of the Merkle tree;
 - the provider's public key is known to the contract, i.e., it was previously added through voting or is the primary key (set during contract deployment).
- Verification of the signature validity of the Merkle tree root received from the ASP. The ECDSA algorithm [19] is used for signing.
- The zk-SNARK proof is verified using the Verifier contract, which was automatically generated from the arithmetic circuit using the Circom library.
- The nullifierHash is added to the list of spent nullifiers.
- The transfer of funds from the contract address to the recipient address is executed, and the fee is transferred to the relayer address.
- A Withdrawal event is emitted containing the recipient, nullifierHash, relayer, and fee.

5.3 Deposit and Withdrawal for ERC20 Tokens

A separate smart contract was developed to support ERC-20 tokens. It overrides the internal functions for directly executing token transfers from one address to another (processDeposit() and processWithdraw(...)). The core logic for deposits and withdrawals remained unchanged.

6. Development of the Decentralized Governance System

It is necessary to provide logic for adding new ASPs. For this purpose, a smart contract for decentralized governance (a so-called Governance contract) was developed, the essence of which is as follows: a user who has sufficient Governance tokens in their account can initiate a vote on calling

a certain function of the main protocol smart contract. If the number of “for” votes exceeds a certain minimum threshold, a decision is made to call the corresponding function with the corresponding arguments through an intermediary smart contract called Timelock – it serves to impose a delay before executing the voting decision.

The OpenZeppelin library [20] was chosen to implement the described functionality, as the current industry standard. Three base contracts were implemented: GovernanceToken, MyGovernor, and MyTimelock.

The process of adding new ASPs proceeds as follows:

1. A user (referred to as the Proposer), who has enough Governance tokens in their balance, forms a proposal (a pair consisting of a method of the main protocol contract and its input data) and submits this proposal to the blockchain by calling the propose function of the Governor contract. Immediately after this, the proposal will be in the Pending state.
2. After a predetermined number of blocks, the voting process begins. After this, the proposal transitions to the Active state.
3. Participants delegate votes to each other (or to themselves) in accordance with how many Governance tokens they held at the time voting began (it is not possible to first delegate votes, then transfer tokens to another address and delegate them again – the OpenZeppelin library provides protection against this).
4. After a specified number of blocks allocated for voting, a verdict is reached – to accept or reject the proposal. If the number of “for” votes exceeds the threshold value set in the Governance contract, the verdict is “Accept” and the proposal transitions to the Succeeded state; otherwise, the verdict is “Reject” and the proposal transitions to the Defeated state.
5. If the proposal was accepted, then immediately afterward (or after some time), any user can add the proposal to the queue by calling its queue function. After this, the proposal transitions to the Queued state.
6. After a specified number of blocks (determined in the Timelock contract), any user can call the execute function of the Governance contract and pass the approved proposal. After this, the proposal transitions to the Executed state and is immediately executed. The actual call to the function of the main protocol contract will be made through the Timelock contract.

7. Transaction Cost Measurements

As part of this work, an experimental study of the implemented system was conducted on the Ethereum Sepolia test network. The developed protocol should not only allow users to increase their anonymity when using the blockchain but should also have comparable features to similar protocols (and to Tornado Cash as the main functional equivalent). Therefore, we conducted a comparison of the amount of gas used for deposit and withdrawal transactions between the developed protocol based on Association Sets and the Tornado Cash protocol.

The comparison was conducted for both ETH and ERC20 tokens. In addition, Tornado Cash was considered in two implementations: the original one and the one using the Poseidon hash function.

Table 1. Comparison of gas consumption for deposit and withdrawal functions of ETH and ERC20 tokens in the Tornado Cash contract and the contract developed in this work, based on association sets.

	ETH deposit	ETH withdraw	ERC20 deposit	ERC20 withdraw
Tornado Cash	1088.8K	301.2K	1128.1K	311.4K
Tornado Cash Poseidon	859.6K	271.7K	900.0K	283.7K
Association Set approach	859.6K	356.6K	900.0K	366.1K

8. Discussion

Analyzing Table 1, it can be seen that the transaction cost in the developed protocol is lower than in the original Tornado Cash (this is due to the use of a more optimal hash function) and coincides with the result for Tornado Cash using the Poseidon hash function (the code for making a deposit is absolutely identical for these two implementations – differences exist only for the withdrawal function).

At the same time, when performing withdrawals, the gas consumption, despite the more optimal hashing function, has increased. This is explained by the additional on-chain checks introduced by the association set mechanism: verification of the ASP’s ECDSA signature over the association set root and verification of a zk-SNARK proof over a larger statement covering membership in two Merkle trees.

The developed system has broad capabilities for integration into the decentralized finance sector as one of the methods for ensuring anonymity of users whose anonymity was previously compromised (or there are suspicions that it may have been compromised). Furthermore, the global trend of moving away from the traditional financial and economic relations system in favor of decentralized services makes such solutions increasingly in demand.

9. Conclusion

In this work, a system of anonymous transactions in a public blockchain has been designed and implemented with the ability to prove the good faith of the origin of funds. The system is based on the concept of association sets and uses zk-SNARK zero-knowledge proofs to provide cryptographic guarantees.

The main results of the work include arithmetic circuits implemented in the Circom 2.0 language for generating zk-SNARK proofs of association set membership; smart contracts developed in the Solidity language encapsulating the core protocol logic; a decentralized governance system; and a comparative analysis with the existing solution.

The practical significance of this work lies in the possibility of using the developed system to comply with regulatory requirements while maintaining the anonymity of users of distributed ledgers.

Directions for further research include optimization of arithmetic circuits to reduce computational complexity, integration with existing decentralized finance protocols, and development of mechanisms for automatic generation of prohibited address lists based on blockchain analytics data. Additionally, the developed protocol can be applied in decentralized platforms for the circulation of Digital Financial Assets to enhance user anonymity without drawing undue attention from regulatory authorities, which constitutes one of the primary objectives in the development of such systems at present.

References

[1]. Zheng Z., Xie S., Dai H.-N., Chen X., Wang H. Blockchain Challenges and Opportunities: A Survey. International Journal of Web and Grid Services, 2018, vol. 14, no. 4, pp. 352-375.

[2]. Nakamoto S. Bitcoin: A Peer-to-Peer Electronic Cash System. 2008. Available at: <https://bitcoin.org/bitcoin.pdf>, accessed: 15.05.2026.

[3]. Pertsev A., Semenov R., Storm R. Tornado Cash Privacy Solution Version 1.4, 2019. Available at: https://tornado.cash/Tornado.cash_whitepaper_v1.4.pdf, accessed: 15.05.2026.

[4]. Youn M., Chin K., Omote K. Empirical Analysis of Cryptocurrency Mixer: Tornado Cash. 2023 Congress in Computer Science, Computer Engineering, & Applied Computing (CSCE), IEEE, 2023, pp. 2324-2331.

[5]. Tornado Cash Delisting. U.S. Department of the Treasury, 21.03.2025. Available at: <https://home.treasury.gov/news/press-releases/sb0057>, accessed: 13.07.2026.

[6]. Chainalysis Website. Available at: <https://www.chainalysis.com/>, accessed: 15.05.2026.

[7]. TRM Labs Website. Available at: <https://www.trmlabs.com/>, accessed: 15.05.2025.

[8]. Chaudhary A., Ivey-Law H. SeDe: Balancing Blockchain Privacy and Regulatory Compliance by Selective De-Anonymization. arXiv preprint, 2023. DOI: 10.48550/arXiv.2311.08167.

- [9]. Buterin V., Illum J., Nadler M., Schär F., Soleimani A. Blockchain privacy and regulatory compliance: Towards a practical equilibrium. *Blockchain: Research and Applications*, 2024, vol. 5, no. 1, article no. 100176. DOI: 10.1016/j.bcr.2023.100176.
- [10]. Vasilyev M.D., Struchkov I.V. asp-poseidon-tornado: Implementation of Tornado Cash with Poseidon hash function and Association Set Provider. 2025. Available at: <https://github.com/sideov/asp-poseidon-tornado>, accessed: 13.07.2026.
- [11]. Petkus M. Why and How zk-SNARK Works. arXiv preprint, 2019. DOI: 10.48550/arXiv.1906.07221.
- [12]. Groth J. On the Size of Pairing-Based Non-Interactive Arguments. *Advances in Cryptology – EUROCRYPT 2016*. Springer, 2016, pp. 305-326.
- [13]. Khovratovich D., Vladimirov M. Tornado Privacy Solution Cryptographic Review. Version 1.1. ABDK Consulting, 2019. Available at: https://tornado.cash/Tornado_cryptographic_review.pdf, accessed: 15.05.2026.
- [14]. Grassi L., Khovratovich D., Rechberger C., Roy A., Schofnegger M. Poseidon: A New Hash Function for Zero-Knowledge Proof Systems. 30th USENIX Security Symposium (USENIX Security 21), 2021, pp. 519-535.
- [15]. Liu H., Luo X., Liu H., Xia X. Merkle Tree: A Fundamental Component of Blockchains. 2021 International Conference on Electronic Information Engineering and Computer Science (EIECS), IEEE, 2021, pp. 556-561.
- [16]. Circom 2.0 Documentation. Available at: <https://docs.circom.io/>, accessed: 15.05.2026.
- [17]. Solidity Documentation. Available at: <https://docs.soliditylang.org/>, accessed: 15.05.2026.
- [18]. Buterin V. Ethereum White Paper. 2013. Available at: <https://ethereum.org/en/whitepaper/>, accessed: 13.07.2026.
- [19]. Johnson D., Menezes A., Vanstone S. The Elliptic Curve Digital Signature Algorithm (ECDSA). *International Journal of Information Security*, 2001, vol. 1, pp. 36-63.
- [20]. OpenZeppelin Official Website. Available at: <https://www.openzeppelin.com/>, accessed: 15.05.2026.

Информация об авторах / Information about authors

Максим Дмитриевич ВАСИЛЬЕВ – аспирант Института прикладных компьютерных наук университета ИТМО. Сфера научных интересов: распределенные системы хранения информации, доказательства с нулевым разглашением, блокчейн.

Maxim Dmitrievich VASILYEV is a postgraduate student at the ITMO University's Institute of Applied Computer Science. His research interests include distributed information storage systems, zero-knowledge proofs, and blockchain technology.

Игорь Вячеславович СТРУЧКОВ – кандидат технических наук, старший преподаватель института прикладных компьютерных наук Университета ИТМО. Сфера научных интересов: распределенные вычислительные системы, децентрализованные системы, блокчейн, корпоративные информационные системы.

Igor Vyacheslavovich STRUCHKOV – Cand. Sci. (Tech.), Senior Lecturer at the Institute of Applied Computer Science, ITMO University. Research interests: distributed computing systems, decentralized systems, blockchain, corporate information systems.