Clean.ton - Scalable Trustless Privacy Protocol

Oleg Tsenilov, Tigran Kashapov

cleanton.org

Abstract

This article presents a novel architecture for blockchain privacy solutions that addresses scalability issues, eliminates the need for trusted setup, and avoids weak cryptographic assumptions inherent in current zkSNARK-based solutions. It introduces ring signatures as a field-specific alternative to zkSNARK, avoiding its complexities and issues. The proposed protocol makes use of Linkable Ring Signature's efficiency, scalability, and confidentiality, along with TON blockchain's performance and scalability, to provide privacy protection in a user-friendly, fast, and accessible manner. Features such as Proof of Exclusion and Dispersion provide foundation for social association mechanism and enhance protocol censorship resistance. The protocol offers broad application possibilities for private payments, exchanges, and services.

Contents

1 Introduction

All blockchain privacy solutions generally follow a similar structure and functionality, most of which incorporate zkSNARK technology at their core. This technology plays a crucial role in confirming certain claims without revealing the supporting arguments.

In spite of the chosen core technology, all transaction anonymizers share one thing in common. It's the zero-knowledge proof of membership in a certain set. Every time user withdraws their funds from one of these services, they are challenged to prove the following statement: given a set of public keys $\mathcal{R} = \{K_1, K_2, K_3, ..., K_n\}$, prove that you possess K_i such that $K_i \in \mathcal{R}$, without revealing either i, or K_i. The last, "hiding the i and K_i" part is essential for preserving the anonymity for the the individual withdrawing funds.

Conventional way to fulfill this statement was through the zkSNARK proof. In this case, the above mentioned statement must be transformed into the following: prove that you possess a valid Merkle-proof for a Merkle-tree of commitments $\mathcal{A} = \{C_1, C_2, C_3, ..., C_n\}$, without revealing neither your commitment C_i, nor it's position i. Despite being extremely complicated in implementation, other issues with this architecture arise.

Two of them are the scalability problems on different levels. The one which appears off-chain is that the prover (i.e the one who withdraws) has to go through all the leaves of the Merkle tree in order to construct a zero-knowledge proof, meaning that it has to perform N operations, where N is the quantity of all ever hidden transactions. The prover also has to download, store and use a large compiled circuits and proving keys (anywhere between 10MB to 1GB) in combination with their secret key, which is an essential part of a predictable and succinct zero-knowledge proof.

The second issue is on the side of the blockchain, that is, the smart-contract has to keep all the N commitments in memory in order to sustain the Merkle structure. It means that the storage fees will grow linearly with the amount of users.

The third, final, and probably the most serious issue, is the presence of the trusted setup and weak cryptographic assumptions.

Although the decentralized multiparty computation ceremony helps distribute trust during setup, it does not entirely eliminate it. Users are still required to place their trust in the centralized organizer who gathers all the shares of a generated private key. If the organizer acts in self-interest, without being detected by the ceremony participants, or if the participants collude, it is possible for the organizer or the participants to generate an unlimited number of false-positive proofs in the future.

Furthermore, zkSNARKs rely on the newer, weak cryptographic assumptions, such as Knowledge of Exponent, q-Power Diffie-Hellman, q-Strong Diffie-Hellman and q-Power Knowledge of Exponent

2 The new architecture

In this section we will briefly discuss the novel way of building the coin anonymization system. While preserving the properties of a conventional architecture, the new one brings scalability and eliminates trust altogether.

2.1 Giving up zkSNARK

Despite being a useful tool, zkSNARK is too general to be used in private payments. In fact, it's probably the most inefficient way to construct such a privacy solution. There is a more field-specific technology which does not introduce any of the mentioned difficulties in the protocol, i.e no trusted setup, weak cryptographic assumptions and scalability issues. Introducing ring signatures.

2.2 Ring signatures

A generalisation of a digital signature algorithm was introduced in 2001 by Ron Rivest, Adi Shamir and Yael Tauman. In a conventional Schnorr signature scheme, for the valid signed message, the verifier can state the following: the message was signed with some unknown private key, from which the presented public key can be derived.

Generalising the notion of Schnorr signature for the valid message and a group of signers with a set of public keys, the verifier can state: The message was signed with some unknown private key, from which one of the public keys in the given set can be derived. See how this statement coincides with the initial statement? That is exactly what we need!

As the reader approached this part, it's worth mentioning that the initial statement was intentionally simplified. One thing that is crucial for every such anonymity tool is the double spending prevention guarantees.

2.3 Linkable ring signatures

It is worth mentioning that the linkability property of a regular ring signature is crucial for privacy solutions in order to prevent double-spending. Note that "linkability" does not disclose any information about the signer. The only thing it discloses is whether the given cryptographic image of the private key was used in order to perform the withdrawal.

2.4 Metadata Integrity

Arbitrary message can be bind to ring signature. During ring signature generation process, one can "sign" data with it. Anyone who verifies this signature can only check whether this signature is valid (and bind to this data), or not. Third parties, including relayers and verifiers cannot change the signing message and thus forge it.

2.5 Proof of Exclusion

Most conflicts around blockchain privacy solutions are based on users' inability to dissociate with certain anonymity set members. In our architecture, users can efficiently prove that their public key is not in some given set of public keys. This could be used by third parties to prove that user's funds are not coming from the unwanted sources. This allows for a completely voluntaristic social association mechanism.

2.6 Dispersion

Another important matter to address is the potential discrimination of the contract users and censorship. We propose a novel solution to this problem - Dispersion mechanism. Every n-th pool, during the withdrawal process, the Dispersion makes indistinguishable transactions to the random blockchain network participants, in turn making it impossible to differentiate users of the contract from the other network users.

3 Closer look at the new architecture

In this section we will discuss the Clean.ton privacy protocol in more detail. The following sections describe the process of the deposit and the withdrawal. In order to have a deep comprehension of the deposit and the withdrawal processes you should be familiar with basics of elliptic curve cryptography.

Deposit is done in a single internal transaction with a constant (N further) amount of TON. The information of the particular deposit is called note

Withdraw is done in a single internal transaction. The contract receives requests for withdrawal through the trustless relayer.

Proof of Exclusion is done in a single internal transaction. The contract receives requests for the proof of exclusion from the user directly.

3.1 Pools

The storage architecture of the smart contract is organized as a hashmap, divided into pools, to streamline access to any desired pool. Each pool itself functions as a separate hashmap. This hierarchical structure enhances efficiency by reducing the number of operations required to verify the presence of a specific public key. Let $\mathcal{M}$ be the pool consisting of at most $\mathcal{L}$ participants. Participants are represented by the set of their public keys. Let $\mathcal{R} = \{K_1, K_2, ..., K_n\}$ be the set of the public keys in the pool $\mathcal{M}$.

To further increase the scalability of the system, pools should be distributed into distinct smart-contracts. This guarantees distribution of the storage fees between network participants and allows TON blockchain to efficiently scale contract through sharding.

3.2 Deposit

The following are the steps performed by the smart contract and user:

1. User generates a random private key k and derives public key K according to the formula $K = G \cdot k$. Knowing public key, it's infeasible to derive private key, according to the elliptic curve discrete logarithm problem
2. User sends $\mathcal{N}$ amount of TON to the contract and specifies K in the transaction
3. User saves their private key k and the ID of the pool (denoted as e further) in which the K is added. k and e are required for withdrawal and called "note" later on. Note must be thoroughly kept in secrecy by user.
4. The smart contract checks whether the K is already present in the pool (if it is, then the transaction is bounced back to the sender)
5. The smart contract stores K in the pool with ID e

3.3 Withdrawal

Ring signature is a generalisation of a traditional digital signature, meaning that signature is done on behalf of a ring (essentially a set of addresses). In the default ring signature algorithm, the signer is exactly one member of that ring, yet it's impossible to differ which one. (More formally, it's impossible to distinguish which public key was used for signing the given message).

Recall that $\mathcal{R}$ is a set of distinct public keys. The signer's private key $k_\pi = k$, extracted from their private note, corresponds to their public key $K_\pi \in \mathcal{R}$, where π is a secret index. Assume the existence of a hash $\mathcal{H}_p$ function, that uniformally maps arbitrary bitstring to one point on an elliptic curve.

The following is the description of the process of creation of linkable ring signature, performed by user:

1. Calculate key image $\widetilde{K} = k_\pi \cdot \mathcal{H}_p(K_\pi)$
2. Generate random number α and random numbers r_i for $i \in \{1, 2, ..., n\}$ but excluding $i = \pi$
3. Compute $c_{\pi+1} = \mathcal{H}_n([\alpha \cdot G], [\alpha \cdot \mathcal{H}_p(K_\pi)])$
4. For $i = \pi + 1, \pi + 2, ..., n, 1, 2, ..., \pi - 1$, calculate, replacing $n + 1 \to 1$,

$$c_{i+1} = \mathcal{H}_n([r_i \cdot G + c_i \cdot K_i], [r_i \cdot \mathcal{H}_p(K_i) + c_i \cdot \widetilde{K}])$$

5. Define $r_\pi \equiv \alpha - c_\pi \cdot k_\pi \pmod{n}$

The signature will be $\sigma = (c_1, r_1, ..., r_n)$, with key image $\widetilde{K}$ and ring $\mathcal{R}$.

Verification means proving σ is a valid signature created by a private key corresponding to a public key in $\mathcal{R}$, and is done in the following manner. The following is the description of the process of verification of linkable ring signature, performed by smart contract:

1. Check $l \cdot \widetilde{K} \stackrel{?}{=} 0$.
2. For $i = \pi + 1, \pi + 2, ..., n$, iteratively compute, replacing $n + 1 \to 1$,

$$c_{i+1}^{'} = \mathcal{H}_n([r_i \cdot G + c_i \cdot K_i], [r_i \cdot \mathcal{H}_p(K_i) + c_i \cdot \widetilde{K}])$$

3. If $c_1 = c_1^{'}$ then the signature is valid.

The following are the steps performed by the smart contract and user. Definitions from the Pools section are also present further:

1. User, using their secret note, generates a ring signature T on ring R. User specifies the withdrawal address and sends it to C along with T.
2. Contract checks if IK is already present in M. If it isn't, then user has not spent their assets yet and C will compute the following steps:
3. Contract checks if T is valid on the ring R. If it is, then C withdraws N - f (where f is the amount of fees required for computation) amount of TON to the specified address.
4. Contract adds the image key of user to the pool in order to prevent the same user from spending twice.

3.4 Relayer

An important privacy concern arises from the described protocol - if the user has to pay for the verification, where the anonymous fee money will come from? The contract can't pay for the verification with it's own money, because this will allow malicious actors to drain contract's balance by sending invalid signatures. Thus, we need some middleman to verify the signature and invest their own TON to pay for the contract's fee, providing anonymous verification and taking commission + verification cost for this service. We call this market actor Relayer. Relayers install special open-source software to facilitate this operation. The relayer's address, withdraw amount, user's address and withdraw amount is put together by the user and signed using the ring signature, using it's metadata integrity feature we mentioned above. The ring signature's properties allow the relayer and the contract to validate this data. It's signed with the secret key of the user and validated using the image key. This makes sure that relayers cannot forge this data.

3.5 Proof of Exclusion

Let $\mathcal{E} = \{K_i, K_j\}$ be the subset of the given set $\mathcal{R} = \{K_1, K_2, K_3, ..., K_n\}$. User wants to prove that his public key K_i is not in this subset. In the realm of ring signatures, user can create a new ring $\mathcal{R}' = \mathcal{R} \oplus \mathcal{E}$. If K_i is indeed not in $\mathcal{E}$, then user can generate a valid ring signature on ring $\mathcal{R}'$. User sends this signature to the contract, which checks it's validity. If the signature is valid, the contract sends back a SBT - Soul Bound Token, which contains information about subset $\mathcal{E}$.

3.6 Dispersion

When the first user in the given pool withdraws their funds from the contract, the contract generates 6 cryptographically random bits, which are saved in the pool. The random seed depends on the contract storage fees and first user's withdrawal logical time. Since these 6 bits are determined at the time, when all pool participants commited their public keys to the pool, nobody can brute force secret key, whose image key ends with these 6 bits. $\mathcal{D} = random(storagefees() \oplus logicaltime())$ mod 64. For the remaining pool participants, contract checks whether their image key's last 6 bits coincide with the ones mentioned above.

4 Future applications and Mass adoption

The proposed system makes use of Linkable Ring Signature's properties, such as efficiency, scalability, linkability and confidentiality as well as TON blockchain's efficiency and scalability to offer privacy protection to users in a simple, fast and convenient way. The fundamental operations of this protocol can be executed seamlessly on any user device, the distributed pool structure in different smart contracts and constant computational complexity allows for low and predictable fees.

Proof of Exclusion enables community to protect itself through the market based association, as well as individual member to dissociate with certain users to receive services, enter exchanges and settle conflicts. Finally, Dispersion mechanism protects community from different discrimination acts based on the interaction with the contract, including potential bans, while also contributing to the anonymity of the users and promoting the protocol.

These features of the protocol enable many abstract application possibilities on top of it. Such as private payments, exchanges and services. Our future work will be focused on providing easy-to-use privacy solutions to the daily and professional tasks. Two of these services are planned to be released in the coming months after release:

Clean Pay offers an intuitive confidential payment service that simplifies the complexities of protocol operations. With its user-friendly interface, it brings the convenience of mainstream digital payment apps to private transactions, effectively democratizing privacy.

Clean Exchange is a private exchange platform designed with privacy at its core. It allows users to trade digital assets and wrapped physical assets while preventing common blockchain trading issues like front-running and strategy copying.

References

[1] Ring signatures https://link.springer.com/chapter/10.1007/3-540-45682-1_32

[2] Linkable ring signature https://link.springer.com/chapter/10.1007/11424826_65

[3] zkSNARK Assumptions https://www.di.ens.fr/~nitulesc/files/Survey-SNARKs.pdf

[4] Ristretto https://ristretto.group

[5] Elligator https://eprint.iacr.org/2013/325

[6] Ed25519 http://ed25519.cr.yp.to/ed25519-20110926.pdf

[7] zkSNARK https://eprint.iacr.org/2016/260

[8] bls12-381 https://electriccoin.co/blog/new-snark-curve/