Cardano Mixer: a Private Transactions Protocol for the Cardano Blockchain

Version 0.1

Vladimir Sinyakov, Elena Ivanova

September 24, 2021

1 Introduction

1.1 Motivation

Every year more and more personal data is getting collected by corporations and governments all around the world. Storage of real identity-wallet pairs in centralized databases poses considerable risks for crypto assets owners. Personal data collection happens, for example, when a user sends crypto assets from a centralized exchange to a personal wallet. We believe that users of a public blockchain have the right to remain pseudonymous or to disconnect their real-world identities from their assets while remaining compliant with the laws of their respective jurisdictions.

Cardano Mixer is a privacy solution for the Cardano blockchain. It is a protocol of a mixer type that breaks the public link between the sender and the receiver of assets on the Cardano blockchain, enabling private transactions. Several existing solutions tackle the private transaction problem [1, 2, 6] in the context of a decentralized ledger. One that is more relevant to this work is the Tornado Cash protocol for the Ethereum blockchain [7]. Cardano Mixer is an adaptation and evolution of that design. Notable differences in our protocol include off-chain proof verification and a completely different economic model.

1.2 On terminology

In this paper, we use the following terminology. A decentralized application or a *dApp* is a collection of software programs that work in a peer-to-peer manner and realize a certain function: in our case, enable private transactions on Cardano. A *protocol* is the description of dApp's logic. A *mixer* is a private transaction protocol where equal deposits are first collected in a common "pot". Then, at any time, users can withdraw a deposit if they possess the knowledge of certain secret parameters for a particular deposit. The term *mixing amount* denotes the type and quantity of a digital asset in a single deposit for a particular mixer protocol. *Anonymity mining* refers to a practice of depositing assets into the mixing protocol for a prolonged time in order to capture the so-called anonymity mining *rewards*. The miners are awarded with *anonymity points* based on the number of deposits they waited before withdrawing. These anonymity points can be then exchanged for the anonymity mining rewards.

1.3 Zero-knowledge proofs and setup ceremony

In order to break the link between the sending and the receiving wallets, we use zero-knowledge proofs of knowledge. In particular, we chose the Groth16 algorithm from [4] for the proof construction and verification. Our choice is motivated primarily by the proof size. Algorithms without trusted setup (e.g., [5, 6, 8]) have the proof bit length significantly higher than the Groth16 algorithm (the difference is 20 times or more). With shorter proofs, we have smaller transaction fees than otherwise, which is especially important for mixers with smaller mixing amounts.

Thus, we will have a trusted setup ceremony with users participating in the creation of a *common reference string* (CRS), a public bytestring that is produced using inputs from all contributors. For the protocol's security, it is sufficient that at least one contributor destroyed the random seed that was used to produce their input. To encourage participation, a significant portion of tokens will be distributed between the contributors (see below).

2 Mixer design

In this section, we describe the algorithms behind the protocol and the components that constitute our dApp.

2.1 Notations

Let $\mathbb{Z}_p$ be a field of integers modulo p. Let H: $(Z_p, Z_p) \to Z_p$ be a hash function. Let $\mathcal{T}$ be a Merkle tree of height d, where each non-leaf node hashes its 2 children nodes with H. Given a leaf value $L \in Z_p$, a position $l \in 1, \dots, 2^d$ in a tree, and a co-path $O \in Z_p^d$, let $\mathcal{R}(L, l, O)$ be the corresponding root value.

2.2 The model

The core protocol functionality includes four functions: deposit, withdraw, collect rewards, and claim rewards.

The mixer is a state machine with the state consisting of the following components:

- Merkle trees $\mathcal{T}$ and $\mathcal{T}'$ with all its leaves initially set to 0 values;
- current indices $l_c, l'_c \in 2^d$ (initially set to 0);
- current co-paths O_c, O'_c;

- current reward shares number S (see Section 3.5).

Leaves of $\mathcal{T}$ are gradually filled with public parameters L associated with each deposit. Similarly, upon withdrawal, the reward's public parameter L' is placed into another Merkle tree $\mathcal{T}'$. It can be then withdrawn into a shielded account using *collect rewards* function. Finally, the rewards can be withdrawn from the protocol using *claim rewards* function.

When $\mathcal{T}$ is completely filled, a new Merkle tree is created for future deposits. We call this a *mixer cycle*.

Currently, all functions barring deposit are performed with the help of relayers: a randomly chosen relayer checks the cryptographic proof and submits the corresponding transaction on behalf of the user.

Let C be the mixing amount of a mixer. Let f_d be the total fees paid by the depositor. It is assumed that asset values and wallet addresses were *safely converted* to $\mathbb{Z}_p$ numbers.

2.2.1 Deposit

Let A be a public address of the withdrawing wallet. Deposit function does the following:

1. generates private parameters $(k, r) \in \mathbb{Z}_p^2$;
2. computes the new leaf $L = H(H(A, k), r)$ and the new co-path O;
3. sends a transaction with value $C + f_d$ and data (L, O).

The data (L, O) acts as input and updates the state machine accordingly.

2.2.2 Withdraw

Assume the user knows private parameters (k, r) associated with the deposit for wallet address A. Then they may compute leaf value L, its index l, and its co-path O. Let A' be a reward withdrawal address. The number of anonymity points awarded is $p(l) = l_c - l - 1$. Consider now the following arithmetic circuit.

$$\begin{array}{rcl} R & = & \mathcal{R}(L, l, O), \\ L & = & H(H(A, k), r), \\ L' & = & H(H(H(p(l), A'), k'), r'), \\ h & = & H(0, k), \\ A & \neq & A'. \end{array} \tag{1}$$

where (R, A, h, L') are *public* inputs and (O, l, k, r, A', k', r') are *private* inputs. Withdraw function does the following:

1. generates private parameters $(k', r') \in \mathbb{Z}_p^2$ for the rewards mixer;
2. computes the new leaf $L' = H(H(H(p(l), A'), k'), r')$ and the new co-path O' for the rewards tree $\mathcal{T}'$;

3. computes the cryptographic proof of knowledge of a valid assignment for arithmetic circuit (1);
4. sends the reward data (L', O'), the public parameters (R, A, h, L'), and the proof to one of the relayers.

2.2.3 Collect rewards

Assume the user knows private parameters (k, r) associated with the uncollected reward of p anonymity points for wallet address A. Then they may compute leaf value L, its index l, and its co-path O. Suppose they also have a shielded account with the balance v_1 and the associated private parameter v_2. Let $V(p)$ be the current mining reward for p anonymity points (see Section 3.5). Then they may compute the old hash h_v and the new hash h'_v (using a new private parameter v_3). Consider now the following arithmetic circuit.

$$\begin{array}{rcl} R & = & \mathcal{R}(L, l, O), \\ L & = & H(H(H(p, A), k), r), \\ h & = & H(0, k), \\ h_v & = & H(v_1, v_2), \\ h'_v & = & H(v_1 + V(p), v_3). \end{array} \tag{2}$$

where (R, A, h, h_v, h'_v) are *public* inputs and (p, v_1, v_2, v_3, r) are *private* inputs. Collect rewards function does the following.

1. generates a private parameter $v_3 \in \mathbb{Z}_p$;
2. computes the cryptographic proof of knowledge of a valid assignment for arithmetic circuit (2);
3. sends the public parameters (R, A, h, h_v, h'_v) and the proof to one of the relayers.

2.2.4 Claim rewards

Assume the user knows their shielded account balance v_1 and the associated private parameter v_2. Let w be a valid reward withdrawal amount (see below). Consider now the following arithmetic circuit.

$$\begin{array}{rcl} h_v & = & H(v_1, v_2), \\ h'_v & = & H(v_1 - w, r), \\ w & \leq & v_1. \end{array} \tag{3}$$

where (w, h_v, h'_v) are *public* inputs and (v_1, v_2, v_3) are *private* inputs.
Claim rewards function does the following.

1. generates a private parameter v_3;
2. computes the cryptographic proof of knowledge of a valid assignment for arithmetic circuit (3);
3. sends the public parameters (w, h_v, h'_v) and the proof to one of the relayers.

2.3 Client app

In order to use the protocol, the user connects their wallet and chooses the desired mixing amount. Three mixers will be available at launch with mixing amounts equal to 200 ADA, 1000 ADA, and 10000 ADA, respectively. As the transaction volume increases, additional mixers for different assets and mixing amounts may be added.

The client app supports *deposit*, *withdraw*, *collect rewards*, and *claim rewards* functions described above. The reward withdrawal amount should be equal to $5m\%$ of the mixing amount where $m \in \{1, \ldots, 20\}$.

The client also has a *generate report* function that lists all interactions with the protocol according to the data stored on the user's machine. This report may serve as proof of compliance with laws of the user's jurisdiction.

2.4 Relay server app

A relay server is run by users who want to provide verification and relay services to other mixer users.

- *Stake* function sends the required funds to the staking script corresponding to a particular mixer. After that, the relayer becomes eligible to accept the client app requests.
- *Start* function activates the listening mode.
- *Stop* function deactivates the listening mode.
- *Unstake* function retrieve the staked funds from the staking script. It is only available at least 10 minutes after the last request has been processed.

It should be noted that withdrawal and reward collection transactions include posting the public parameter h on the blockchain to avoid the double spending.

3 Protocol token, economics and governance

3.1 MIX token

The protocol comes with a governance token with the symbol MIX. The token is used for two purposes.

First, it facilitates voting on protocol improvement proposals and usage of the project's treasury. Once the governance mechanisms are implemented, token holders will control the parameters of the protocol, vote on improvement proposals, and determine the direction of the project's future development. They will also vote on the allocation of treasury funds as they are unlocked.

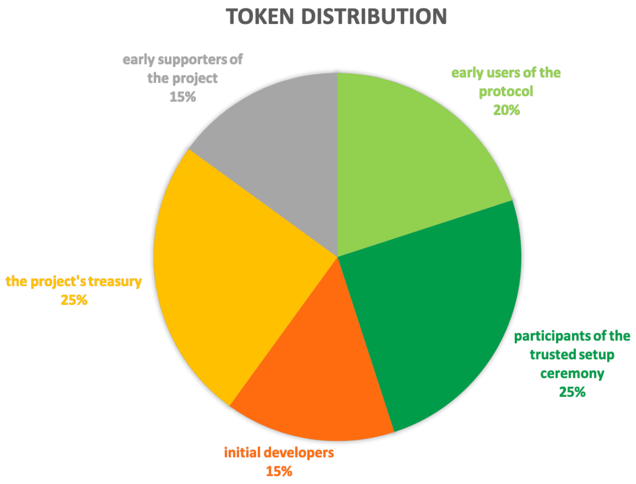
Second, MIX tokens serve as collateral that relayers must stake to become a verifying and relaying service. As it was previously mentioned, proof verification is normally done off-chain by one of these relayers. Relayers submit transactions on the user's behalf, allowing empty wallets to withdraw funds and anonymity mining rewards. The details on staking are given in Section 3.4.

3.2 Initial token distribution

The MIX token has a fixed supply of 1 000 000 tokens, minted at launch. The initial distribution of the token is summarized in the following pie chart.

Below we give the description of each category.

- 25 % of the tokens are airdropped to the participants of the trusted setup ceremony.
- 20 % of the tokens are airdropped to the early users of the protocol.
- Another 15 % of the tokens are airdropped to the early supporters of the project. This includes challenges, lottery, and other activities that drive adoption and help the project succeed. The full details will be revealed at a later date.
- Next, 25 % of the tokens will constitute the project's treasury. This treasury funds will be initially locked. Every month, 1 % of the token supply is unlocked. The first unlock will happen when the governance protocol is set up and fully operational.
- The final 15 % will be given to the initial developers of the protocol and the dApp.



3.3 Fees

Our protocol includes two distinct services:

1. anonymity mining;
2. proof verification and relaying.

Initially, we set the fees for both services at 0.1 % of the deposit. These values and many other protocol parameters could be adjusted through governance if needs be. Here is the complete fee structure paid by a user initiating a deposit:

- 0.1 % fee to the relayer;
- 0.1 % fee to the anonymity mining pool;
- the Cardano network fee on withdrawal transaction;
- the Cardano network fee on deposit transaction.

The user withdrawing the funds does not have to pay anything. The relayer gets a network fee reimbursement plus their 0.1 % upon the confirmation of the withdrawal transaction. This way, a withdrawal can be initiated from a freshly created wallet.

When a user collects or claims the anonymity mining rewards, the fee is paid to the relayer. This fee is initially set to 1% of the reward plus the Cardano network fee on the transaction. For mixing assets other than ADA, we plan to use Babel fees when they are implemented [3].

3.4 Token staking

In order to capture fees, relayers must stake MIX tokens. Every mixer requires a certain amount of tokens (plus the mixing amount) to be staked to become a relayer for that particular mixer. This value depends on the mixing amount and is initially set according to Table 1.

Mixing amount	200 ADA	1000 ADA	10000 ADA
MIX stake	0.04 %	0.2 %	2 %

Table 1: MIX staking requirements.

Every relayer is restricted to one submitted transaction per minute (withdraw, collect rewards, and claim rewards).

There will be a separate scanner tool that searches for invalid transactions submitted by dishonest relayers. A transaction is deemed invalid if the verification algorithm run on-chain does not accept the proof posted by a relayer. In that case, the staked mixing amount is returned to the mixer, 75 % of staked MIX tokens are returned to the project's treasury, and 25 % of staked MIX tokens are given away as a bounty for exposing a dishonest relayer.

3.5 Anonymity mining

When users collect anonymity mining rewards, they send them to a shielded account. Once a certain minimal sum is collected on a shielded account, the miner may withdraw it. The mining reward is computed as follows. Let d be the height of the Merkle tree, f be the mining fee for a single deposit. Let K_j be the number of deposits in the tree at the time of j-th withdrawal. Then, after j withdrawals, the current shares number S_j is calculated as follows:

$$S_j = S_{j-1} - (M - K_j), \quad S_0 = \frac{1}{2}M(M-1)$$

where $M = 2^d$ is the maximal number of deposits in one cycle.

Suppose a miner makes a deposit, waits for p deposits to receive p anonymity points, and makes a withdrawal. The current mining reward is determined as follows

$$V = \frac{p}{S_j}F$$

where $F = Mf$ is the total fees collected in one mixer cycle.

Let us comment on this reward scheme.

- The user's reward is proportional to the number of anonymity points p and to the total fees collected in one mixer cycle F;
- At any point in time, the rewards are fully covered by the collected fees;
- The sum of all anonymity points in one mixer cycle is always equal to S_M;
- If everyone waits until the end of the cycle to collect rewards, all collected anonymity mining fees will be distributed. Otherwise, the leftover fees are sent to the project's treasury.

For the first cycle, we set $d = 10$, resulting in $M = 1024$ total deposits.

4 Mixer properties

Here we summarize the key protocol properties:

- At any time, the number of deposits in the mixer is always greater or equal than the number of withdrawals made (barring a dishonest relayer case);
- A dishonest relayer cannot cause damage to the protocol if there is at least one network node running the scanner app;
- Any deposit can be withdrawn once if the private parameters (k, r) and the withdrawal address A are known;
- A deposit can be withdrawn only to the address selected during the deposit (no transaction front-running);

- For each withdrawal, any deposit since the last moment when the number of deposits was equal to the number of withdrawals could act as the potential corresponding deposit.

References

[1] Kurt M. Alonso and Jordi Herrera Joancomartí. Monero: Privacy in the blockchain, 2018. https://eprint.iacr.org/2018/535.pdf.

[2] Eli Ben-Sasson, Alessandro Chiesa, Christina Garman, Matthew Green, Ian Miers, Eran Tromer, and Madars Virza. Zerocash: Decentralized anonymous payments from bitcoin, 2014. http://zerocash-project.org/media/pdf/zerocash-extended-20140518.pdf.

[3] Manuel M. T. Chakravarty, Nikos Karayannidis, Aggelos Kiayias, Michael Peyton Jones, and Polina Vinogradova. Babel fees via limited liabilities. *CoRR*, abs/2106.01161, 2021. https://arxiv.org/abs/2106.01161.

[4] Jens Groth. On the size of pairing-based non-interactive arguments. In Marc Fischlin and Jean-Sébastien Coron, editors, *Advances in Cryptology – EUROCRYPT 2016*, pages 305–326, Berlin, Heidelberg, 2016. Springer Berlin Heidelberg.

[5] Jens Groth and Markulf Kohlweiss. One-out-of-many proofs: Or how to leak a secret and spend a coin. In Elisabeth Oswald and Marc Fischlin, editors, *Advances in Cryptology - EUROCRYPT 2015*, pages 253–280, Berlin, Heidelberg, 2015. Springer Berlin Heidelberg.

[6] Aram Jivanyan. Lelantus: A new design for anonymous and confidential cryptocurrencies. Cryptology ePrint Archive, Report 2019/373, 2019. https://ia.cr/2019/373.

[7] Alexey Pertsev, Roman Semenov, and Roman Storm. Tornado cash privacy solution, 2019. https://tornado.cash/Tornado.cash_whitepaper_v1.4.pdf.

[8] Srinath Setty. Spartan: Efficient and general-purpose zksnarks without trusted setup. In Daniele Micciancio and Thomas Ristenpart, editors, *Advances in Cryptology – CRYPTO 2020*, pages 704–737, Cham, 2020. Springer International Publishing.