3-2020

PrivateEx: Privacy Preserving Exchange of Crypto-assets on Blockchain

Lei Xu
The University of Texas Rio Grande Valley, lei.xu@utrgv.edu

Lin Chen

Zhimin Gao

Keshav Kasichainula

Miguel Fernandez
The University of Texas Rio Grande Valley

See next page for additional authors

Recommended Citation

Xu, Lei, Lin Chen, Zhimin Gao, Keshav Kasichainula, Miguel Fernandez, Bogdan Carbunar, and Weidong Shi. 2020. "PrivateEx: Privacy Preserving Exchange of Crypto-Assets on Blockchain." In Proceedings of the 35th Annual ACM Symposium on Applied Computing, 316–323. SAC '20. New York, NY, USA: Association for Computing Machinery. https://doi.org/10.1145/3341105.3373901.

PrivateEx: Privacy Preserving Exchange of Crypto-assets on Blockchain

Lei Xu
University of Texas Rio Grande Valley
xuleimath@gmail.com

Lin Chen
Texas Tech University
chenlin198662@gmail.com

Zhimin Gao
Auburn University at Montgomery
mtion@msn.com

Keshav Kasichainula
University of Houston
kkasicha@central.uh.edu

Miguel Fernandez
University of Texas Rio Grande Valley
miguel.fernandez02@utrgv.edu

Bogdan Carbunar
Florida International University
carbunar@cs.fiu.edu

Weidong Shi
University of Houston
wshi3@central.uh.edu

ABSTRACT

Bitcoin introduces a new type of cryptocurrency that does not rely on a central system to maintain transactions. Inspired by the success of Bitcoin, all types of alt cryptocurrencies were invented in recent years. Some of the new cryptocurrencies focus on privacy enhancement, where transaction information such as value and sender/receiver identity can be hidden, such as Zcash and Monero. However, there are few schemes to support multiple types of cryptocurrencies/assets and offer privacy enhancement at the same time. The major challenge for a multiple asset system is that it needs to support two-way assets exchange between participants besides one-way asset transfer. Thus, we propose a privacy-preserving exchange scheme, PrivateEx, which preserves the privacy of the exchange of different assets. PrivateEx utilizes zero-knowledge proof and a novel way to "lock" assets involved in the exchange to guarantee the correctness, fairness, and privacy of exchange of assets in the system. We also implement a prototype of PrivateEx and evaluate its performance to show that it is practical with modern computers.

CCS CONCEPTS

• **Security and privacy** → **Privacy-preserving protocols**; *Domain-specific security and privacy architectures*;

KEYWORDS

Crypto assets, exchange, privacy

1 INTRODUCTION

Since the invention of Bitcoin [11], a variety of blockchain based cryptocurrency systems have been developed. By requiring each participant to keep a local copy of the transactions history, which are organized as blocks and linked one by one, such a system can prevent double-spending without relying on a centralized party. Although these schemes usually allow a user to create multiple pseudonyms by him/herself, existing works have demonstrated that one can establish the relationships between the pseudonyms and even identify the user behind a pseudonym [14]. Several efforts have been made to enhance the privacy of blockchain based cryptocurrency system, such as Zerocash [16], Monero [12], Dash [7] and ValueShuffle on Bitcoin [15]. These schemes utilize different cryptography tools to prevent a node disclosing payment related information, e.g., recipient/sender identities and amount of cryptocurrency transferred. These blockchain based cryptocurrency systems only support one type of asset, and it is a natural idea to extend the concept to build multi-asset systems. A multi-asset system has two basic transaction types: one-way transfer and two-way exchange. While the one-way transfer is the same as those classical cryptocurrency systems, the two-way exchange is more interesting and plays an important role to help thrive the whole cryptocurrency ecosystem: (i) It helps to break down the silos and enable more convenient value flow; and (ii) It facilitates the creation of new cryptocurrency systems as investigators can easily convert existing cryptocurrencies to the new one. Currently, two-way exchange is mainly done through following ways: centralized exchange platforms and decentralized platforms. Centralized

platforms such as Coinbase requires the users to fully trust the platforms and disclose all exchange information to the centralized platforms. A decentralized platform can utilize smart contract to support two-way exchange. Although the users don't need to fully trust any single node, they still need to expose all information about an exchange to the public for them to verify and to guarantee the correctness and fairness of the operation. For both cases, existing privacy enhancement mechanisms for one-way transfer cannot be applied directly.

To mitigate the privacy concern and fully unleash the potential of blockchain based cryptocurrencies, we propose a privacy-preserving exchange scheme, PrivateEx. The proposed scheme would support privacy protection for exchange operations in a multi-asset system. Under the PrivateEx framework, different types of assets are first converted to notes for exchange operations. Conversion to note would guarantee the fairness of exchange, PrivateEx introduces a locking structure on notes to be exchanged. The locking mechanism allows any participant to take correct actions to move forward and does not have the problem of deadlock. PrivateEx also leverages zero-knowledge proof to allow the blockchain participants to verify the correctness of transactions without disclosing information about the exchange. In summary, our contributions of the paper include: (i) We formalize the privacy-preserving exchange and describe the requirements of such a scheme; (ii) We develop a concrete privacy-preserving exchange scheme for blockchain based multi-asset schemes and demonstrate that it meets all the requirements given in the formal framework; (iii) We discuss implementation details of the designed scheme and conduct experiments to show its practicability.

2 MODEL AND REQUIREMENTS

Let Alice and Bob be two blockchain users who own different types of assets, and want to exchange the ownership of their assets. We define the security model as follows: (i) The exchange participants do not trust each other. We assume that each of them can attempt to take advantage of other or the system, to maximize their benefit, e.g., receive the other's asset without giving their own asset. (ii) The blockchain used for exchange is trusted. We assume the underlying blockchain as a whole is trusted, i.e., it will follow the predefined protocols and guarantee the computation correctness of all accepted transactions. Depending on the consensus protocol used by the blockchain, this assumption has different requirements on the blockchain maintainers. For example, when PoW and longest chain principle is used to determine the legitimate blockchain, we require more than half of the miners to be honest. The blockchain can be either a dedicated system for the exchange or an existing blockchain with multiple types of assets and supports smart contract such as Ethereum.

Under the security model, an adversary can: (i) Observe the blockchain. An adversary is able to access all transactions stored on the blockchain to review and analyze the contents of these transactions. (ii) Interact with the blockchain. An adversary can actively interact with the blockchain by submitting new transactions as long as the transactions themselves are valid. This also includes cases where one party of an exchange operation tries to cheat, e.g., stopping in the middle of an exchange process, submitting a transaction that does not match with the initial exchange purpose, and conducting exchange operations with multiple parties at the same time. (iii) Control some blockchain maintainers. An adversary can take over a certain percentage of blockchain maintainers and control their behaviors in block construction, e.g., selecting transactions that are included in a block and ignoring received blocks. An adversary can also combine above activities. For example, Alice as an adversary can also try to take advantage of Bob by manipulating some blockchain maintainers, or try to find out whether Bob is doing exchange with others.

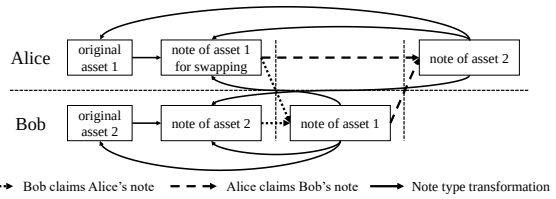

Figure 1: Workflow of the privacy-preserving asset exchange. Alice initiates the exchange. After Bob claims Alice's note, Alice can claim Bob's note to complete the exchange from her side. If Bob does not respond, Alice can cancel the exchange and get her note back. The exchanged notes can be converted to other forms and the owners can use them for another round of exchange with others.

We now define the objectives of the privacy-preserving exchange on multi-asset blockchain under the given security model: (i) Correctness. The exchange operation should only allow participants to exchange their own existing assets, and will not create new assets. It also requires that one cannot cheat in an exchange using assets that are different from promised. (ii) Fairness. The exchange operation can only have one of two outcomes: either the exchange succeeds and both participants receive the other's asset, or the exchange fails and both participants receive their asset back. (iii) Privacy. One can only learn information of exchanges that he/she is involved, but cannot cannot figure out who and what are involved in other exchange operations.

When privacy is not a concern, one can easily implement the exchange operation using a smart contract, where the blockchain guarantees the fairness feature of the operation, i.e., either the exchange succeeds that Alice and Bob get each other's asset, or the exchange fails that each one still keeps his/her own assets. However, this approach does not provide any privacy protection. Every node in the blockchain network can see the exchange information including the owner identities and the type and value of exchanged assets.

3 OVERVIEW OF PRIVATEEX

We now provide an overview of PrivateEx, the proposed privacy-preserving exchange of crypto assets on the blockchain. PrivateEx implements the following steps to preserve the privacy of an exchange operation: (i) Alice and Bob communicate off-chain and agree to exchange their assets with each other. Without loss of generality, we assume Alice initiates the exchange operation. (ii) Alice first converts her asset into a note. This note has two parts: one part stands for her asset with a positive value, the other part stands for Bob's asset which she wants to receive. Alice also creates a secret value that one can use to spend her newly created note. All blockchain nodes verify the new note and store it on the blockchain. (iii) Bob also converts his asset into a note, which only includes the value and type information of his own asset. (iv) Alice and Bob check that both notes are correctly created and accepted by the blockchain. (v) Alice shares the claiming secret of the note she created with Bob, so Bob can claim the ownership of Alice's asset. This operation also freezes his own note, which is guaranteed by the blockchain. (vi) After Bob claims Alice's asset, Alice can claim the ownership of Bob's note.

For each of the above steps, Alice and Bob utilize publicly verifiable zero-knowledge proofs to allow the public to verify the validity of the corresponding transaction submitted to the blockchain. Figure 1 demonstrates the sequence of steps of exchange operation.

4 THE DETAILED DESIGN OF PRIVATEEX

In this section, we provide the details of the construction of PrivateEx. Without loss of generality, we assume that each asset has a unique type identifier and a value, which are positive integers. Therefore, an asset is represented as a pair (t, v), where $t, v \in \mathbb{Z}^+$. In the following, we first consider the case where both Alice and Bob cooperate to finish the swap operation. We then separately consider other special cases, e.g., where Alice and/or Bob may want to terminate the operation.

4.1 System Setup

To setup PrivateEx, several algorithms and corresponding parameters are determined: (i) Consensus protocol, such as PoW and PoS. The consensus protocol does not affect the design of PrivateEx. (ii) Original asset creation. The asset can be created on the blockchain itself or ported from an external blockchain. This does not affect the operation of PrivateEx. (iii) Cryptography algorithms. PrivateEx utilizes several cryptography primitives, including collision resistant hash function `CRH()`, commitment scheme `COMMIT()`, a non-interactive zero-knowledge proof system, and a one-time signature scheme. Parameters of these primitives are also initialized in this step.

This information is embedded into the genesis block so every participant of the system uses the same algorithms and parameters for operations. The setup process also initializes two empty Merkle trees T_1 and T_2 with a fixed height, which determines how many transactions PrivateEx can handle. To prevent an adversary from tampering the two Merkle trees, the roots of the trees are included in the blockchain when there is an update of the tree.

PrivateEx can also be integrated with existing blockchain based multi-asset system by creating a specific block that includes all PrivateEx specific information.

4.2 Notes Initialization

Initialization actions of Alice. Before Alice can exchange her asset (t_1, v_1) with Bob for (t_2, v_2) in a privacy-preserving way, she needs to initialize the exchange by converting the asset to a note, the structure of which is discussed below. Alice does the following steps for the initial conversion:

(1) Serial numbers generation. Alice selects random numbers sn_1 and r_0, and calculates $sn_2 \leftarrow \texttt{COMMIT}(sn_1, r_0)$. The constructed note has (sn_1, sn_2) as its serial numbers. The serial numbers are used to prevent double spending, and we explain in more details the use of these two-serial-number structure later.

(2) Quid pro quo determination. Alice specifies the asset she wants to get from Bob, which is also identified as a pair (t_2, v_2). Note that the pair just indicates the type and value of the asset Alice is interested and does not need to be bound with anything specific that Bob has.

(3) Note construction. Alice selects four random numbers r_1, r_2, r_3, r_4, and calculates a sequence of commitments:

$$ct_1 \leftarrow \texttt{COMMIT}(sn_1, r_1),\ ct_2 \leftarrow \texttt{COMMIT}(sn_2||ct_1, r_2)$$
$$ct_3 \leftarrow \texttt{COMMIT}(t_2||v_2||ct_2, r_3),\ ct_4 \leftarrow \texttt{COMMIT}(t_1||v_1||ct_3, r_4)$$

The final note created by Alice is in the form

$$nt_{01} \leftarrow (t_1, v_1, t_2, v_2, sn_1, sn_2, r_1, r_2, r_3, r_4).$$

(4) Transaction construction. Alice submits the transaction

$$tx_{01} \leftarrow (t_1, v_1, r_4, ct_3, ct_4)$$

to the blockchain, which stands for the note nt_{01}.

The purpose of this transaction is to convert Alice's asset to a note, so the public only needs to verify whether Alice owns the asset to create such a note, and do not need to check what Alice asks for exchange. Specifically, each peer of the blockchain processes the received transaction tx_{01} as follows:

(1) The peer checks whether the transaction is well formed by checking

$$ct_4 \stackrel{?}{=} \texttt{COMMIT}(t_1||v_1||ct_3, r_4).$$

(2) The peer checks whether Alice has enough balance in her account and reduces the balance of type t_1 asset by v_1.

(3) The peer then works with other blockchain peers to include tx_{01} in the blockchain using the consensus protocol.

(4) The peer also adds ct_4 as a new leaf of the Merkle tree T_1, and updates the value of the root rt_1.

Alice also shares the information of nt_{01} with Bob except the values sn_1 and r_1 to allow Bob to claim the ownership of note nt_{01} for exchange.

Initialization actions of Bob. Bob also needs to convert his asset to a note for the exchange operation. Bob constructs a note for his asset (t_2, v_2) as follows:

(1) Note construction. Bob selects a random serial number sn_3, random numbers r_5, r_6, and calculates two commitments:

$$ct_5 \leftarrow \texttt{COMMIT}(sn_3, r_5),\ ct_6 \leftarrow \texttt{COMMIT}(t_2||v_2||ct_5, r_6)$$

The final note is in the form

$$nt_{02} \leftarrow (t_2, v_2, sn_3, r_5, r_6).$$

(2) Transaction construction. Bob submits the transaction

$$tx_{02} \leftarrow (t_2, v_2, r_6, ct_5, ct_6)$$

to the blockchain, which stands for the note nt_{02}.

Similar to the situation of initialization actions of Alice, each peer of the blockchain processes the received transaction tx_{02} as follows:

(1) The peer checks whether the transaction is well formed by checking

$$ct_6 \stackrel{?}{=} \texttt{COMMIT}(t_2||v_2||ct_5, r_6).$$

(2) The peer checks whether Bob has enough balance in his account and reduces the balance of type t_2 asset by v_2.
(3) The peer includes tx_{02} in the blockchain.
(4) The peer also adds ct_6 as a new leaf of the Merkle tree T_1, and updates the value of the root rt_1.

Bob does not need to share any information of this note with Alice.

Information exchange between Alice and Bob. After Alice finishes her initialization operation, she needs to send part of information of nt_{01} to Bob. The sent information will allow bob to carry forward the exchange operation. This activity cannot be disclosed to the public, otherwise they will learn that Alice and Bob are trying to exchange their assets. To protect this information, there are two ways for Alice to send information to Bob and vice versa: (i) Using off-chain channel. Alice can share information with Bob directly without using the blockchain. (ii) Using key privacy encryption [2]. Alice can encrypt the message she wants to share using key privacy encryption with Bob's public key. The key privacy feature guarantees that an adversary cannot link the cipher-text to Bob.

4.3 First Claim Operation

After Alice initializes the exchange and Bob creates his own note, Bob can carry forward the exchange by consuming his own note (which allows Alice to claim the ownership later) and transferring Alice's note to his account in a single transaction.

Recall that Bob has information of the note nt_{01} that Alice created except serial number sn_1 and corresponding commitment random number r_1, i.e., Bob knows

$$(t_1, v_1, t_2, v_2, sn_2, r_2, r_3, r_4, ct_1).$$

To transfer Alice's asset to himself, Bob creates a new note in the form of

$$nt_1 \leftarrow (sn_4, t_1, v_1),$$

where sn_4 is a newly selected random serial number. The corresponding transaction is

$$tx_1 \leftarrow (sn_2, sn_3, ct_8),$$

where ct_8 is created through two steps:

$$ct_7 \leftarrow \texttt{COMMIT}(sn_4, r_7),\ ct_8 \leftarrow \texttt{COMMIT}(t_1||v_1||ct_7, r_8).$$

Here r_7, r_8 are two random values selected by Bob. This transaction means two old notes with serial numbers sn_2 and sn_3 are consumed, and a new commitment value ct_8 is created, which represents the note nt_1.

To prove to the public that tx_1 is valid, Bob constructs a zero-knowledge proof π_1 on the following statement:

Given the Merkle tree T_1's root rt_1, serial number sn_2 (representing Alice's note for exchange) and serial number sn_3 (representing Bob's note for exchange), and commitment ct_8 (representing the new note Bob wants to create), I know existing notes nt_{01} and nt_{02} in the system such that: (i) The notes nt_{01}, nt_{02}, and nt_1 are well formed. (ii) The serial numbers sn_2 and sn_3 are computed correctly. (iii) The note commitments for nt_{01} and nt_{02} appear in the Merkel tree T_1 with root rt_1. (iv) The value and type of these three notes match.

For each peer of the blockchain receiving the transaction tx_1 and corresponding zero knowledge proof π_1, he/she does the following steps to process:

(1) The peer checks that the two serial numbers sn_2 and sn_3 have not been used in the system before so it is not double spending.
(2) The peer verifies the zero-knowledge proof π_1 to accept the new transaction tx_1 to the blockchain.
(3) The peer adds commitment ct_8 as a new leaf to the Merkle tree T_1, and updates the root rt_1.
(4) The peer adds the two disclosed serial numbers sn_2, sn_3 as new leaves to the Merkle tree T_2 and updates the root rt_2.

4.4 Second Claim Operation

After Bob finishes the first claim operation, i.e., he has consumed the notes that two parties generated in the initialization phase, Alice can start to claim the ownership of the note Bob created in the initialization phase.

Alice creates a new note in the form of

$$nt_2 \leftarrow (sn_5, t_2, v_2),$$

where sn_5 is a new random serial number selected by Alice. The corresponding transaction is

$$tx_2 \leftarrow (sn_1, ct_{10}),$$

where sn_1 is the other secret serial number Alice created for note nt_{01}, and ct_{10} is created through two steps:

$$ct_9 \leftarrow \texttt{COMMIT}(sn_5, r_9),\ ct_{10} \leftarrow \texttt{COMMIT}(t_2||v_2||ct_9, r_{10}).$$

Here r_9, r_{10} are two random values selected by Alice.

To prove to the peers of the blockchain that tx_2 is valid, Alice constructs a zero-knowledge proof π_2 on the following statement:

Given the Merkle tree T_1's root rt_1 and T_2's root rt_2, serial number sn_1, and the commitment value ct_{10}, I know the existing note nt_{01} and random value r_0 such that: (i) Note nt_{01} is well formed. (ii) The serial number sn_1 is computed correctly. (iii) The commitment for note nt_{01} appears in the Merkle tree T_1 with root rt_1. (iv) The other serial number of note nt_{01}, sn_2, appears in the Merkle tree T_2 with root rt_2, and sn_2 is derived from sn_1 correctly with r_0. (v) The exchange target value and type of note nt_{01} match the new note nt_2 with commitment value ct_{10}.

For each peer of the blockchain receiving the transaction tx_2 and corresponding zero-knowledge proof π_2, he/she does the following steps to process:

(1) The peer checks that the serial number sn_1 has not been used in the system before so it is not double spending.
(2) The peer verifies the proof π_2 to accept the new transaction tx_2 to the blockchain.
(3) The peer adds commitment ct_{10} as a new leaf to the Merkle tree T_1, and updates the root rt_1.

After this step is finished, both serial numbers sn_1 and sn_2 of note nt_{01} are disclosed to the public. But since a random value r_0 is involved to the derivation of sn_2 from sn_1, the public cannot link these two values as long as there are multiple exchanges in the system.

4.5 Use of Notes

After Alice and Bob finish the exchange operation, they can convert their new notes back to assets, or use them for another round of exchange with others.

Converting notes back to assets. If Bob wants to convert back the note nt_1 that he gets from the exchange with Alice to a normal asset, he creates a transaction in the form of

$$tx_4 \leftarrow (sn_4, t_1, v_1, Bob),$$

and generates a zero-knowledge proof π_3 on the following statement:

Given the Merkle tree T_1's root rt_1, serial number sn_4, and the type/value information (t_1, v_1), I know the commitment value ct_8 is a leaf of the tree T_1 and two random values r_7, r_8 such that the following equations hold:

$$ct_7 \leftarrow \textit{COMMIT}(sn_4, r_7),\ ct_8 \leftarrow \textit{COMMIT}(t_1||v_1||ct_7, r_8).$$

Here sn_4 is the serial number of note nt_1 is associated with *Bob* in the system. Each peer of the blockchain checks that sn_4 never appears in the system before and verifies the proof π_3. Then sn_4 is added to the blockchain, and (t_1, v_1) is deposited to Bob's account.

The note nt_2 that Alice gets from the exchange operation has the same structure as nt_1, and Alice can use the same way to convert the note to an asset in plaintext that belongs to her.

Converting notes for another exchange operation. Bob can also start another exchange operation using the note nt_1 directly. This is further divided into two cases: (i) preparing to spend first (Bob needs to specify what asset he wants to exchange); and (ii) preparing to spend secondly (Bob does not need to specify what assets he wants to exchange).

Converting nt_1 for spending first. Note that in the transaction tx_{01}, t_1, v_1 appear in plain text only because the system needs to make sure the creator has enough balance in his/her account. Once the balance verification is done and the note is created the values are no longer in plain text. Therefore, Bob can consume nt_1 and generates a transaction tx'_{01} for a new note nt'_{01} in the same way as described in Section 4.2 except that the newly created transaction tx'_{01} is in the form

$$tx'_{01} \leftarrow (sn_4, ct'_4).$$

To convince the peers of blockchain that ct'_4 is a valid note, Bob discloses the serial number sn_4 of note nt_1 and builds a zero-knowledge proof π_4 on the following statement:

Given the Merkle tree T_1's root rt_1, the serial number sn_4, and the commitment ct'_4, I know existing note nt_1 in the system such that: (i) The note nt_1 and nt'_{01} are well formed. (ii) The serial number sn_4 is computed correctly. (iii) The note commitment for nt_1 appears in the Merkle tree T_1 with root rt_1. (iv) The note commitment ct'_4 is computed correctly. (v) Value and type information of nt_1 and nt'_{01} match.

A blockchain peer who receives the transaction (sn_4, ct'_4) and corresponding zero-knowledge proof π_4 do the following steps to process:

(1) The peer checks whether the serial number sn_4 has been disclosed in the system before to prevent double spending.
(2) The peer verifies the proof π_4 to accept the new transaction tx'_{01} to the blockchain.
(3) The peer adds the commitment ct'_4 as a new leaf of the Merkle tree T_1, and updates the root rt_1.

Converting nt_1 for spending second. Transaction tx_{02} has the same structure of tx_{01}, and Bob can convert his note nt_1 to tx'_{02} in the same way as tx'_{01}, except that the zero-knowledge proof proves that the commitment ct'_6 is constructed in a different way. Alice can convert the note nt_2 that she owns for another exchange directly in a similar way.

4.6 Termination of Exchange Operation

It is possible that Alice and/or Bob want to stop at a certain point before the exchange operation succeeds. There are two possibilities that the exchange operation terminates in the middle of the process:

- Case 1: Alice wants to terminate the process after she initializes the exchange operation and before Bob claims the ownership of her note; and

- Case 2: Bob wants to terminate the process after he finishes the preparation of his note for the exchange operation and before he consumes it for Alice's note.

Alice has multiple choices when she terminates the exchange: (i) Converting the note nt_{01} back to the asset in her account; (ii) Converting the note nt_{01} for another note for exchange operation with the specified target; and (iii) Converting the note nt_{01} for another note for exchange operation without a specified target. Since Alice knows how the note nt_{01} is constructed and the corresponding commitment ct_4 in Merkle tree T_1, she can create a new note in the desired format and a zero-knowledge proof of the ownership of the old note and the correctness of the new note.

The situation for Bob is similar, and he can convert the note nt_{02} to a selected form (asset, note for exchange with the specified target, or note for exchange without specified target) using corresponding zero-knowledge proof.

4.7 One-way Transfer

PrivateEx can support one-way transfer using a special type of exchange: Alice sets the target note she wants to receive for exchange as 0, and Bob can create a null note to claim Alice's note without paying anything. Compared with existing transfer-only privacy cryptocurrency systems like Zerocash, the major disadvantage of transfer in PrivateEx is that the sender knows when the receiver claims the note, and the sender can cancel the transfer before the receiver claims the new ownership. The benefit of this transfer method is that it is consistent with exchange operation, and the adversary cannot distinguish a one-way transfer and a two-way exchange.

4.8 Non-malleability

Different types of transactions described in this section have multiple parts, and it is necessary to prevent an adversary from taking parts from different transactions to build a new one. PrivateEx utilizes one-time signature [10] to prevent such attacks. Specifically, the creator of a transaction does the following steps:

(1) The creator randomly selects a key pair (pk_s, sk_s) for the selected one-time signature scheme.
(2) The creator binds the public key pk_s with the transaction.
   - If the transaction creates a new note with one serial number $sn^{(1)}$, the creator uses this serial number to generate a tag for the public key using a pseudo-random function as $t \leftarrow \mathtt{PRF}(pk_s, sn^{(1)})$, and modifies corresponding zero-knowledge proof to demonstrate t and pk_s are connected. Since the serial number is selected by the creator him/herself, an adversary cannot produce a valid proof.
   - If the transaction creates a note with two serial numbers (e.g., note nt_{01} created by Alice in initialization), the creator uses the second serial number that is not shared with the other party in the exchange to bind the public key.

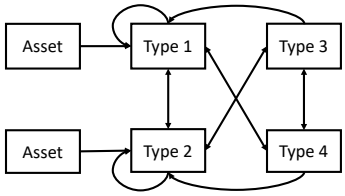

Figure 2: Conversions between different types of notes. An asset is first converted to Type 1 or Type 2 note to enter PrivateEx, and then it can convert to another type of note. A note of any type can also be converted back to an asset. While transactions that only involve notes do not leak type/value/ownership information, converting to/from assets will disclose certain information to the public.

The creator also adds a corresponding statement to the zero-knowledge proof accompanied the transaction:

Given the tag t and the public key pk_s, I know a secret value $sn^{(1)}$ such that $t = PRF(pk_s, sn^{(1)})$.

(3) The creator generates a signature on the transaction using sk_s, and attaches the signature with the transaction.

Correspondingly, a peer of the blockchain receiving a transaction uses the public key to verify the signature on the received transaction besides all the verifications described before, which guarantees non-malleability feature.

5 SECURITY ANALYSIS OF PRIVATEEX

In this section, we analyze the key features of the proposed PrivateEx scheme and demonstrate that it satisfies all the objectives. For all the analysis, we assume the blockchain as a whole is trusted.

Quick review of the design of PrivateEx. In PrivateEx, all assets are created in plaintext and stored on the blockchain. If the owner wants to protect his/her privacy, he/she needs to convert the asset to a note. The PrivateEx scheme involves four types of notes, which we summarize as follows: (i) Type 1. This type of notes are generated for exchange operation and includes type/value information of the target note. (ii) Type 2. This type of notes are generated for exchange operation without information of the target note. (iii) Type 3. This type of notes are generated by first spending operation in exchange. (iv) Type 4. This type of notes are generated by second spending operation in exchange.

Assets are first converted to Type 1 or Type 2 notes, and further conversions between different types of notes are summarized in Figure 2. Converting from one note to another note is similar to a shield-to-shield transaction in Zcash that does not disclose the owner's information. However, if one converts a note back to an asset, it is like a shield-to-non-shield transaction, and the type, value, and owner information is disclosed to the public.

Correctness of PrivateEx. The correctness of PrivateEx is further divided into three requirements and we demonstrate that the proposed design meets all of them: (i) New assets cannot be created through exchange operations. The conversion from an asset to note is done without hiding any information except the serial number, so the public can verify that no new assets is created from scratch. For following steps in the exchange, a zero knowledge proof is always required for a transaction, which allows the public to verify the consistency of the consumed note and the newly created note, and new asset cannot be created either. (ii) Only matched exchange requests can be accepted. The matching condition is embedded in every Type 1 note. If one wants to spend a Type 1 note, he/she must freeze his/her own note that matches the condition, otherwise it is impossible to build a corresponding zero-knowledge proof that can be verified by the public. (iii) Double spending can be detected and prevented. Double spending in conversion from an asset to note is prevented by checking the balance information stored on the blockchain. For most note types, each note has a unique serial number and double spending is avoided by checking whether a serial number has been disclosed in the past. For a Type 1 note with two serial numbers, only one of them can be used to create a new note and double spending does not exist.

Fairness of PrivateEx. Intuitively, fairness means that a participant of the exchange cannot take advantage of the other one. This is equivalent to that the exchange ends up in two and only two cases: (i) Case 1. The exchange succeeds and both parties get the asset of the other party; or (ii) Case 2. The exchange terminates in the middle and each party gets his/her asset back. Here we use the term asset but it can be in the form a note in the system.

The system fails to meet this feature only if one party can get the assets of both sides. If Alice achieves this goal, she needs to produce a valid zero-knowledge proof shows that she knows the corresponding serial number of the note Bob created, which contradicts with the assumption that Bob keeps this serial number secret until he spends his note by himself. If Bob achieves this goal, he needs to produce a valid zero-knowledge proof on the second serial number of the note Alice created, which Alice keeps as a secret. Therefore, PrivateEx guarantees the feature of fairness.

Privacy of PrivateEx Privacy is the core feature of PrivateEx. In general, privacy means that by observing the blockchain transactions and interacting with the blockchain, an adversary cannot figure out who are involved in an exchange and the type/value information of the assets that are exchanged. This feature is captured by blockchain indistinguishability, which was first proposed in Zerocash [16]. The idea of blockchain indistinguishability is that given two blockchain based multi-assets system running PrivateEx BC_0 and BC_1, even if an adversary can control a pair of honest users to submit transactions to these two blockchains (under certain restrictions), he/she cannot distinguish BC_0 and BC_1. Since the adversary cannot distinguish these two blockchains, we can draw the conclusion that transactions stored on the blockchain do not disclose any information. More concretely, the blockchain indistinguishability is defined as a game in the following way: (i) A challenger $\mathcal{C}$ sets up two PrivateEx instances BC_0 and BC_1, and randomly selects a bit b. (ii) $\mathcal{C}$ also initializes two oracles $\mathcal{O}_0$ and $\mathcal{O}_1$, through which the adversary $\mathcal{A}$ can control the two blockchain instances. $\mathcal{C}$ gives $\mathcal{A}$ the view of BC_L and BC_R, where $BC_L = BC_b$ and $BC_R = BC_{1-b}$. (iii) At each time, $\mathcal{A}$ generates a pair of instructions to generate transactions Q and Q', which are forwarded to BC_0 and BC_1 respectively, and processed by corresponding oracles. (iv) Besides submitting transactions through oracles, $\mathcal{A}$ is also allowed to set up his/her own accounts and submit transactions. In this case, Q and Q' are submitted to BC_L and BC_R respectively. (v) At the end of the game, $\mathcal{A}$ outputs a guess b' and wins if $b = b'$. If the probability that $\mathcal{A}$ wins is at most negligible greater than 1/2. To avoid the trivial cases, there are several restrictions on the queries Q and Q': (i) Q and Q' must have the same type; (ii) Q and Q must include the same public information. For example, if they are converting a note back to plaintext assets, these assets must have the same type and value; (iii) Q and Q' must be valid and consistent with existing transactions on the blockchains.

The proof of blockchain indistinguishability feature for PrivateEx follows the proof given in Appendix D of [16] and is done through a sequence of simulation experiments.

6 IMPLEMENTATION AND EVALUATION

In this section, we discuss the implementation of PrivateEx and evaluate its performance and cost.

6.1 Circuit Design

We implement PrivateEx using the *libsnark* library [17], which is a C++ based library for zk-SNARK proof system [3].

Protoboard. A zk-SNARK proof system requires a circuit which takes both public and auxiliary inputs and produces an output. The output is a boolean value and shows whether the inputs satisfy the preset constraints (in the form of the circuit) or not. A virtual protoboard is used to attach the circuit and its necessary components. In the *libsnark* toolset, a Rank One Constraint System (R1CS) is the basic component to verify satisfactory [4].

Gadget. When we design a complex zk-SNARK proof system, it is not efficient to build it from R1CSes directly. *libsnark* provides several common gadgets to build a customized protoboard. In the PrivateEX proof system, two major gadgets are applied to the protoboard.

Figure 3 indicates that our proof system is verified if and only if the input satisfies all gadgets on the protoboard. PrivateEX uses SHA256 to compute note commitment. Thus, the note commitment related verification should be performed by the SHA256 gadget. A SHA256 gadget checks whether the given input can reproduce a public known output hash. For instance, in the case of "Initialization actions of Alice",

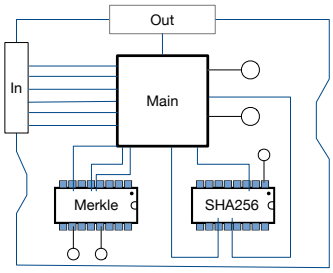

Figure 3: SHA256 gadget is for cryptography-related validation, e.g. commitment. Merkle tree gadget verifies whether the giving note is a leaf in a Merkle tree with root *rt*.

the SHA256 gadget verifies whether input $t_1||v_1||ct_3$ and $r4$ can reproduce ct_4. The Merkle tree gadget checks the commitment for a note appears in the Merkle tree with the given root. We define both gadgets as follows.

```
sha256_two_to_one_hash_gadget<FieldT> f;
/*HashT is the C++ template of hash algorithm*/
merkle_tree_check_read_gadget<FieldT, HashT> ml;
```

And, generate the constraints.

```
f->generate_r1cs_constraints();
ml->generate_r1cs_constraints();
```

Finally, the system takes inputs and produces a witness.

```
left->generate_r1cs_witness(left_bv);
right->generate_r1cs_witness(right_bv);
output->generate_r1cs_witness(hash_bv);
f->generate_r1cs_witness();

leaf_digest->generate_r1cs_witness(leaf);
root_digest->generate_r1cs_witness(root);
ml->generate_r1cs_witness();
```

Variables like *left*, *right* and *output* are the inputs for SHA256 gadget *f*. Typically, *left* is concatenation of several parameters, e.g. t_1, v_1 and ct_3. Input *right* is a random number. Variable *output* is the given commitment value for validator to check whether it matches the result from SHA256 gadget. On the other side, *leaf* and *root* are mandatory inputs for the Merkle tree gadget *ml*.

Other constraints are checked by the main gadget. For instance, to check whether input note and output note have the same type of assets, we can add a basic R1CS constraint to the main gadget as follow.

```
//t1 is the type of old note asset
//t2 is the type of new note asset
this->pb.add_r1cs_constraint(
    r1cs_constraint<FieldT>(
            1, t1, t2
));
//R1CS : 1*t1=t2
```

Proof-key and Verification-key. Before generating proof, zk-SNARK requires a security setup to produce a pair of proof-key (PK) and verification-key (VK). Users use PK and necessary transaction information to generate proofs, and validators use VK and auxiliary inputs/witness to verify the transaction without knowing the details.

Table 1: Circuit Evaluation Results

Circuit	Constraints	PK	VK	Time
Commitment	27,280	6.14 MB	511.8 B	10 sec
Merkle tree∗	448,774	99.2 MB	511.8 B	106 sec
First Claim∗	476,571	103.66 MB	511.8 B	131 sec
Second Claim∗	476,571	115.84 MB	511.8 B	130 sec
Merkle tree†	1,822,358	447.39 MB	511.8 B	464 sec
First Claim†	1,822,875	447.48 MB	511.8 B	466 sec
Second Claim†	1,822,875	447.48 MB	511.8 B	484 sec

∗ Merkle tree depth = 16. † Merkle tree depth = 64.

6.2 Circuit Evaluation

In this section, we evaluate our zk-SNARK circuit. We measure the number of R1CS constraints, size of PK and VK, and the time consumption to generate a proof. The benchmark is performed on an Amazon Web Service (AWS) Elastic Computing Cloud (EC2) instance *t2.large*. This type of instance involves 2 vCPUs and 8 GB memory.

Table 1 illustrates the comparison between different circuit designs. Usually, a circuit with more R1CS constraints may require more time to generate a proof. The verification time is constant regardless of the number of constraints. We observe that the Merkle tree gadget takes more time to execute. The overall transaction time for a integrated circuit is 497 seconds approximately, which include the time for proof-generation and transaction verification (Merkle tree depth = 64). Note that the integrated circuit is directly produced from *libsnark*. This can be further optimized by improving the underlying elliptic curve algorithms [1].

7 RELATED WORKS

In this section, we briefly review works related to the exchange of cryptocurrencies/assets.

Privacy protection for one-way transfer. Several approaches have been developed to protect the privacy of one-way transfer in a single-asset blockchain system, including zero-knowledge [16], ring signature [12], and mixnet [7, 15]. Most of these techniques cannot be extended directly to protect two-way exchange as they cannot guarantee the fairness feature. While it is possible for one to build a mixnet structure to achieve both fairness and privacy protection, the effectiveness of this approach relies on the trustworthy of the mixing nodes, which is not desirable.

Fair information exchange. Fair information exchange is the process for two parties to exchange their own secrets. At the end of the process, they either learn each other's secret at the same time, or nothing is leaked to the other party. It has been proved that it is impossible to guarantee the fairness without a trusted third party [13], but several approaches are

developed to achieve the goal with the help of a blockchain, such as FairSwap [8]. Two-way asset exchange is more than fair information exchange as exchanging the secrets does not guarantee the change of ownership.

Multi-asset exchange. There are a large number of centralized exchange platforms for crypto asset exchange, e.g., Coinbase, Bitbuy, and Coinsquqre. These platforms require participants' fully trust and work as proxies for exchange operations. From another point of view, these centralized platforms can also be treated as a mixing service provider. Multi-asset exchange on blockchain has also been studied. Bentov et.al. proposed a real-time cryptocurrency exchange protocol using trusted hardware [5]. 0x is built on top of Ethereum and supports exchange of ERC2.0 tokens [18]. TEX is another decentralized exchange scheme that operates in two layers [9]. However, these schemes only focus on the exchange operation itself but do not provide any protection on the privacy of exchange.

8 CONCLUSION

Exchange of different types of blockchain based crypto-assets has become a significant business, and it is a natural extension of privacy-preserving cryptocurrency systems to build a privacy-preserving, decentralized exchange system. In response to this demand, we develop PrivateEx, a novel multi-asset exchange platform on blockchain, and applies zero-knowledge proof to protect the exchange information and a two-serial-number structure to guarantee the fairness of the exchange operation. Also, we discuss the implementation of the zero-knowledge proof system used in PrivateEx and formalize the security requirements. We implement the key sub-circuits for different types of operations in PrivateEx. Though the cost is not cheap, it is acceptable for modern computers. The performance can be further improved using other ZK-SNARK primitives such as Bulletproof [6].

ACKNOWLEDGMENTS

The research was supported in part by NSF 1756014.

REFERENCES

[1] Stephanie Bayer and Jens Groth. 2012. Efficient zero-knowledge argument for correctness of a shuffle. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 263–280.

[2] Mihir Bellare, Alexandra Boldyreva, Anand Desai, and David Pointcheval. 2001. Key-privacy in public-key encryption. In *International Conference on the Theory and Application of Cryptology and Information Security*. Springer, 566–582.

[3] Eli Ben-Sasson, Alessandro Chiesa, Daniel Genkin, Eran Tromer, and Madars Virza. 2013. SNARKs for C: Verifying program executions succinctly and in zero knowledge. In *Advances in Cryptology–CRYPTO 2013*. Springer, 90–108.

[4] Eli Ben-Sasson, Alessandro Chiesa, Michael Riabzev, Nicholas Spooner, Madars Virza, and Nicholas P Ward. 2019. Aurora: Transparent succinct arguments for R1CS. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 103–128.

[5] Iddo Bentov, Yan Ji, Fan Zhang, Yunqi Li, Xueyuan Zhao, Lorenz Breidenbach, Philip Daian, and Ari Juels. 2017. Tesseract: Real-Time Cryptocurrency Exchange using Trusted Hardware. *IACR Cryptology ePrint Archive* 2017 (2017), 1153.

[6] Benedikt Bünz, Jonathan Bootle, Dan Boneh, Andrew Poelstra, Pieter Wuille, and Greg Maxwell. 2018. Bulletproofs: Short proofs for confidential transactions and more. In *2018 IEEE Symposium on Security and Privacy (SP)*. IEEE, 315–334.

[7] Evan Duffield and Daniel Diaz. 2015. Dash: A privacy centric cryptocurrency. (2015). https://www.dash.org

[8] Stefan Dziembowski, Lisa Eckey, and Sebastian Faust. 2018. Fairswap: How to fairly exchange digital goods. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 967–984.

[9] Rami Khalil, Arthur Gervais, and Guillaume Felley. 2019. TEX–A Securely Scalable Trustless Exchange. *IACR Cryptology ePrint Archive* (2019).

[10] Leslie Lamport. 1979. *Constructing digital signatures from a one-way function*. Technical Report. Technical Report CSL-98, SRI International Palo Alto.

[11] Satoshi Nakamoto. 2008. Bitcoin: A peer-to-peer electronic cash system.

[12] Shen Noether. 2015. Ring Signature Confidential Transactions for Monero. *IACR Cryptology ePrint Archive* 2015 (2015), 1098.

[13] Henning Pagnia and Felix C Gärtner. 1999. *On the impossibility of fair exchange without a trusted third party*. Technical Report. Technical Report TUD-BS-1999-02, Darmstadt University of Technology .

[14] Dorit Ron and Adi Shamir. 2013. Quantitative analysis of the full bitcoin transaction graph. In *International Conference on Financial Cryptography and Data Security*. Springer, 6–24.

[15] Tim Ruffing and Pedro Moreno-Sanchez. 2017. ValueShuffle: Mixing confidential transactions for comprehensive transaction privacy in bitcoin. In *International Conference on Financial Cryptography and Data Security*. Springer, 133–154.

[16] Eli Ben Sasson, Alessandro Chiesa, Christina Garman, Matthew Green, Ian Miers, Eran Tromer, and Madars Virza. 2014. Zerocash: Decentralized anonymous payments from bitcoin. In *2014 IEEE Symposium on Security and Privacy*. IEEE, 459–474.

[17] SCIPR-Lab. 2017. libsnark: a C++ library for zkSNARK proofs. https://github.com/scipr-lab/libsnark

[18] Will Warren and Amir Bandeali. 2017. 0x: An open protocol for decentralized exchange on the Ethereum blockchain. *URl: https://github. com/0xProject/whitepaper* (2017).