The Formal Proof of BlockMaze

Zhangshuang Guan, Zhiguo Wan, Yang Yang, Yan Zhou, and Butian Huang

APPENDIX A
SUMMARY OF PRIVACY-PRESERVING CRYPTOCURRENCIES

Quite a few privacy-preserving cryptocurrencies have been proposed in the literature, and they allow a user to hide transaction amounts and/or obscure the linkage between a transaction and its sender and recipient. We now survey typical privacy-preserving cryptocurrencies and compare them in Table I.

As shown in Table I, we can see that only Verge [1] supports hiding IP address using Tor and I2P [2], and others obscure the linkage between a transaction and the public wallet addresses of the involved parties. Zcash [3], Komodo [4] and ZenCash [5] offer privacy guarantees to hide the sender address, recipient address, and transaction amounts using zk-SNARK. Monero [6] achieves the same goal using CryptoNote, which is a protocol based on ring signatures. Grin [7] also solves it using MimbleWimble, which is based on elliptic curve cryptography and derived from confidential transactions, while NavCoin [8] achieves it using ZeroCT based on Zerocoin protocol and confidential transactions.

Moreover, Zcoin [9] utilizes Zerocoin, a protocol based on zero- knowledge proof, to achieve unlinkability and untraceability. As a Dash [10] fork, PIVX [11] disconnects senders and recipients using the zPIV protocol, which is based on Zerocoin protocol with custom Proof of Stake (PoS). Dash [10] and CoinShuffle [12] employ CoinJoin, a method based on a mixer idea, to obscure the linkage between a transaction and its sender and recipient.

APPENDIX B
SECURITY OF BLOCKMAZE SCHEMES

A BlockMaze scheme Π = (**Setup**, **CreateAccount**, **Mint**, **Redeem**, **Send**, **Deposit**, **VerTx**) is *secure* if it satisfies ledger indistinguishability, transaction unlinkability, and balance. We now formally define them here.

Following a similar model defined in Zerocash [3], we design an experiment as a game between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$. Each experiment is employed depending on a stateful BlockMaze oracle $\mathcal{O}^{\text{BM}}$, which provides an interface for executing the algorithms defined in a BlockMaze scheme Π. The oracle $\mathcal{O}^{\text{BM}}$ stores and maintains a ledger L, a set of accounts ACCOUNT, a set of plaintext balance PT_BALANCE, and a set of zero-knowledge balance ZK_BALANCE, where the sets are initialized to be empty at the beginning. The oracle $\mathcal{O}^{\text{BM}}$ responses to queries for different algorithms.

- $Query$(***CreateAccount***). Receiving ***CreateAccount*** query, $\mathcal{C}$ computes an account address $addr$ and a key pair (sk, pk) by calling **CreateAccount** algorithm, which are added to ACCOUNT. Then, $\mathcal{C}$ outputs $(addr, pk)$.
- $Query(\textbf{\textit{Mint}}, addr_A, v)$. Receiving ***Mint*** query, $\mathcal{C}$ computes $(\texttt{zk\_balance}^*_A, \texttt{tx}_{\texttt{Mint}})$ for user A by calling **Mint** algorithm. Add $\texttt{zk\_balance}^*_A$ to ZK_BALANCE, $\texttt{pt\_balance}^*_A$ to PT_BALANCE, and $\texttt{tx}_{\texttt{Mint}}$ to L, where $\texttt{zk\_balance}^*_A.value = \texttt{zk\_balance}_A.value + v$ and $\texttt{pt\_balance}^*_A = \texttt{pt\_balance}_A - v$.
- $Query(\textbf{\textit{Redeem}}, addr_A, v)$. Receiving ***Redeem*** query, $\mathcal{C}$ computes $(\texttt{zk\_balance}^*_A, \texttt{tx}_{\texttt{Redeem}})$ for user A by calling **Redeem** algorithm. Add $\texttt{zk\_balance}^*_A$ to ZK_BALANCE, $\texttt{pt\_balance}^*_A$ to PT_BALANCE, and $\texttt{tx}_{\texttt{Redeem}}$ to L, where $\texttt{zk\_balance}^*_A.value = \texttt{zk\_balance}_A.value - v$ and $\texttt{pt\_balance}^*_A = \texttt{pt\_balance}_A + v$.
- $Query(\textbf{\textit{Send}}, addr_A, addr_B, v)$. Receiving ***Send*** query, $\mathcal{C}$ computes $(\texttt{zk\_balance}^*_A, \texttt{tx}_{\texttt{Send}})$ by calling **Send** algorithm. Then, add $\texttt{zk\_balance}^*_A$ to ZK_BALANCE, and $\texttt{tx}_{\texttt{Send}}$ to L, where $\texttt{zk\_balance}^*_A.value$ is equal to $(\texttt{zk\_balance}_A.value - v)$.
- $Query(\textbf{\textit{Deposit}}, addr_B, h_{\texttt{tx}_{\texttt{Send}}})$. Receiving ***Deposit*** query, $\mathcal{C}$ computes $\texttt{zk\_balance}^*_B$ and generates $\texttt{tx}_{\texttt{Deposit}}$ for user B by executing **Deposit** algorithm. Then, add $\texttt{tx}_{\texttt{Deposit}}$ to L, and $\texttt{zk\_balance}^*_B$ to ZK_BALANCE, where $\texttt{zk\_balance}^*_B.value$ is equal to $(\texttt{zk\_balance}_B.value+v)$.
- $Query(\textbf{\textit{Insert}}, \texttt{tx})$. Receiving ***Insert*** query, $\mathcal{C}$ verifies the output of **VerTx** algorithm: if the output is 1, add the Mint/Redeem/Send/Deposit transaction tx to L; otherwise, it aborts.

B.1 Ledger Indistinguishability

We utilize the method employed in the extended version of [3] to describe ledger indistinguishability. It can be represented by an experiment L-IND, which involves a probabilistic polynomial-time adversary $\mathcal{A}$ trying to attack a BlockMaze scheme. Before giving a formal experiment, we first define public consistency for a pair of queries.

Definition 1 (Public consistency)**.** A pair of queries (Q, Q') is *publicly consistent* if both queries are of the *same* type and consistent in $\mathcal{A}$'s view. The public information contained in (Q, Q') must be equal including: (i) the value to be transformed; (ii) the account address; (iii) the balance commitment; (iv) the transfer commitment; and (v) published serial number. Moreover, both queries must satisfy the following restrictions for different query types:

Corresponding authors: Zhiguo Wan and Yang Yang.

Z. Guan, Z. Wan, and Y. Zhou are with the School of Computer Science and Technology, Shandong University, Qingdao, 266237, China (e-mail:{guanzs@mail.,wanzhiguo@,sduzhouyan@mail.}sdu.edu.cn).

Y. Yang is with the School of Mathematics and Computer Science, Fuzhou University, Fuzhou, 350116, China (e-mail:yang.yang.research@gmail.com).

B. Huang is with Hangzhou Yunphant Network Technology Co. Ltd., Hangzhou, 311121, China (e-mail: hbt@yunphant.com).

TABLE I: Comparison of privacy-preserving blockchain systems based on UTXO-model

Cryptocurrencies	Consensus	Privacy guarantees IP	Sender	Recipient	Amount	Techniques	Pros/Cons
Zcash [3]	PoW (Equihash)	×	√	√	√	zk-SNARK	Provide privacy protection of transaction amount and sender/recipient, but require a trust setup.
Monero [6]	PoW (CryptoNight)	×	√	√	√	Ring Signature	Achieve unlinkability of transactions but with limited privacy protection and large transaction size.
Zcoin [9]	PoW	×	√	√	×	Zero Knowledge Proof	Provide privacy protection of sender/recipient but with large proof size and high verification latency.
Dash [10]	PoW (X11)	×	√	√	×	Mix	Achieve untraceability of transactions, but mixing process is slow.
CoinShuffle [12]	PoW	×	√	√	×	Mix	Allow users to utilize Bitcoin in a truly anonymous manner without any trusted third party.
PIVX [11]	PoS	×	√	√	×	Zero knowledge proof	Be forked from Dash, and provide private instant verified transactions.
Komodo [4]	dPoW	×	√	√	√	zk-SNARK	Be forked from Zcash, and focus on security, scalability, interoperability, and adaptability.
NavCoin [8]	PoW/PoS (X13)	×	√	√	√	ZeroCT [13]	Be forked from Bitcoin, and provide affordable and fast digital payments focused on privacy and simplicity.
ZenCash [5]	PoW	×	√	√	√	zk-SNARK	Be forked from Zcash, and focus on usability of the end-to-end encrypted system.
Verge [1]	PoW	√	×	×	×	Tor and I2P	Provide end-user identity obfuscation, but repeatedly suffer from 51% attacks.
Grin [7]	PoW (Cuckoo Cycle)	×	√	√	√	MimbleWimble	Focus on privacy, fungibility, and scalability, but require a secure communication channel.

For ***CreateAccount*** type, (Q, Q') are always publicly consistent since that the ledgers remain unchanged during the queries. Moreover, we require that both oracles generate the same account to reply to both queries.

For ***Mint*** and ***Redeem*** type, (Q, Q') must be mutually independent and meet the following restrictions:

- the balance commitment cmt_A in Q must correspond to $\texttt{zk\_balance}_A$ that appears in $\texttt{ZK\_BALANCE}$;
- the zero-knowledge balance $\texttt{zk\_balance}_A$ must be valid, i.e., its serial number must never be published before;
- the account address $addr_A$ contained in Q must match with that in $\texttt{zk\_balance}_A$ and $\texttt{zk\_balance}^*_A$;
- $(\texttt{zk\_balance}_A.value + \texttt{pt\_balance}_A)$ is equal to $(\texttt{zk\_balance}^*_A.value + \texttt{pt\_balance}^*_A)$.

For ***Send*** type, (Q, Q') must be mutually independent and meet the following restrictions:

- the balance commitment cmt_A in Q must correspond to $\texttt{zk\_balance}_A$ that appears in $\texttt{ZK\_BALANCE}$;
- the zero-knowledge balance $\texttt{zk\_balance}_A$ must be valid, i.e., its serial number must never be published before;
- the account address $addr_A$ specified in Q must match with that in $\texttt{zk\_balance}_A$, $\texttt{zk\_balance}^*_A$ and cmt_v;
- $\texttt{zk\_balance}_A.value - v = \texttt{zk\_balance}^*_A.value$.

For ***Deposit*** type, (Q, Q') must be mutually independent and meet the following restrictions:

- the balance commitment cmt_B in Q must correspond to $\texttt{zk\_balance}_B$ that appears in $\texttt{ZK\_BALANCE}$;
- the zero-knowledge balance $\texttt{zk\_balance}_B$ must be valid, i.e., its serial number must never be published before;
- the transfer commitment cmt_v must be valid, i.e., it must appear in a valid transaction $\texttt{tx}_{\texttt{Send}}$ on the corresponding oracle's ledger, and its serial number must never be published before;
- the account address $addr_B$ in Q must match with that in $\texttt{zk\_balance}_B$ and $\texttt{zk\_balance}^*_B$;
- $\texttt{zk\_balance}_B.value + v = \texttt{zk\_balance}^*_B.value$.

Formally, let Π = (**Setup**, **CreateAccount**, **Mint**, **Redeem**, **Send**, **Deposit**, **VerTx**) be a BlockMaze scheme. Let $\mathcal{A}$ be an adversary, which is formally just a (stateful) algorithm. Let λ be a security parameter. We define the ledger indistinguishability experiment $\mathsf{BlockMaze}^{\texttt{L-IND}}_{\Pi,\mathcal{A}}(\lambda)$ as follows:

1) The public parameters $\mathsf{pp} := \mathbf{Setup}(1^\lambda)$ is computed and given to $\mathcal{A}$. Two *independent* BlockMaze oracles $\mathcal{O}^{\mathsf{BM}}_0$ and $\mathcal{O}^{\mathsf{BM}}_1$ are initialized. A uniform bit $b \in \{0, 1\}$ is chosen.
2) Whenever $\mathcal{A}$ sends a pair of *publicly consistent* queries (Q, Q'), answer the queries in the following way:
   a) Provide two separate ledgers (L_b, L_{1-b}) to $\mathcal{A}$ in each step. L_b is the current ledger in $\mathcal{O}^{\mathsf{BM}}_b$, and L_{1-b} is the one in $\mathcal{O}^{\mathsf{BM}}_{1-b}$.
   b) Send Q to $\mathcal{O}^{\mathsf{BM}}_b$ and Q' to $\mathcal{O}^{\mathsf{BM}}_{1-b}$ to obtain two oracle answers (a_b, a_{1-b}).
   c) Return (a_b, a_{1-b}) to $\mathcal{A}$.
3) Continue answering *publicly consistent* queries of $\mathcal{A}$ until $\mathcal{A}$ outputs a bit b'.
4) The game outputs 1 if $b' = b$, and 0 otherwise. If $\mathsf{BlockMaze}^{\texttt{L-IND}}_{\Pi,\mathcal{A}}(\lambda) = 1$, we say that $\mathcal{A}$ succeeds.

Definition 2 ($\texttt{L-IND}$ Security)**.** A BlockMaze scheme Π = (**Setup**, **CreateAccount**, **Mint**, **Redeem**, **Send**, **Deposit**, **VerTx**) is $\texttt{L-IND}$ secure, if for all probabilistic polynomial-time adversaries $\mathcal{A}$ there is a negligible function negl such that, for security parameter λ,

$$\Pr\left[\mathsf{BlockMaze}^{\texttt{L-IND}}_{\Pi,\mathcal{A}}(\lambda)=1\right] \le \frac{1}{2} + \mathsf{negl}(\lambda).$$

B.2 Transaction Unlinkability

Formally, let Π = (**Setup**, **CreateAccount**, **Mint**, **Redeem**, **Send**, **Deposit**, **VerTx**) be a BlockMaze scheme, $\mathcal{A}$ an adversary, and λ the security parameter. Let $\mathcal{T}$ be the table of zero-knowledge transactions (i.e., $\texttt{tx}_{\texttt{Send}}$ and $\texttt{tx}_{\texttt{Depoist}}$) generated by $\mathcal{O}^{\text{BM}}$ in response to ***Send*** and ***Deposit*** queries. We define the transaction unlinkability experiment $\mathsf{BlockMaze}^{\texttt{TR-UL}}_{\Pi,\mathcal{A}}(\lambda)$ as follows:

1) The public parameters $\mathsf{pp} := \mathbf{Setup}(1^\lambda)$ is computed and given to $\mathcal{A}$. A BlockMaze oracle $\mathcal{O}^{\text{BM}}$ is initialized.
2) Whenever $\mathcal{A}$ queries $\mathcal{O}^{\text{BM}}$, answer this query along with the ledger L at each step.
3) Continue answering queries of $\mathcal{A}$ until $\mathcal{A}$ sends a pair of zero-knowledge transactions $(\texttt{tx}, \texttt{tx}')$ with the requirements: (i) $(\texttt{tx}, \texttt{tx}') \in \mathcal{T}$ are of the *same* type; (ii) $\texttt{tx} \neq \texttt{tx}'$; (iii) the senders of $(\texttt{tx}, \texttt{tx}')$ are not $\mathcal{A}$ if $\texttt{tx} = \texttt{tx}_{\texttt{Send}}$; (iv) the recipients of $(\texttt{tx}, \texttt{tx}')$ are not $\mathcal{A}$ if $\texttt{tx} = \texttt{tx}_{\texttt{Deposit}}$.
4) The experiment outputs 1 (indicating $\mathcal{A}$ wins the game) if one of the following conditions holds: (i) the recipients of payments contained in $(\texttt{tx}, \texttt{tx}')$ are the same if $\texttt{tx} = \texttt{tx}_{\texttt{Send}}$; and (ii) the senders of payments contained in $(\texttt{tx}, \texttt{tx}')$ are the same if $\texttt{tx} = \texttt{tx}_{\texttt{Deposit}}$. Otherwise, it outputs 0.

Definition 3 ($\texttt{TR-UL}$ Security)**.** A BlockMaze scheme Π = (**Setup**, **CreateAccount**, **Mint**, **Redeem**, **Send**, **Deposit**, **VerTx**) is $\texttt{TR-UL}$ secure, if for all probabilistic polynomial-time adversaries $\mathcal{A}$ there is a negligible function negl such that, for security parameter λ,

$$\Pr\left[\mathsf{BlockMaze}^{\texttt{TR-UL}}_{\Pi,\mathcal{A}}(\lambda)=1\right] \le \mathsf{negl}(\lambda).$$

B.3 Balance

We employ an experiment $\texttt{BAL}$, which involves a probabilistic polynomial-time adversary $\mathcal{A}$ trying to attack a given BlockMaze scheme, where a similar definition is given in [3]. Firstly, we define eight variables for the security model of balance.

- $v_{\texttt{zk\_unspent}}$, the spendable amount in $\texttt{zk\_balance}^*$, i.e., $\texttt{zk\_balance}^*.value$. The challenger $\mathcal{C}$ can check that $\texttt{zk\_balance}^*$ is valid by accessing to $\mathcal{A}$'s balance commitment recorded on MPT on L.
- $v_{\texttt{pt\_unspent}}$, the spendable amount of plaintext balance $\texttt{pt\_balance}^*$. The challenger $\mathcal{C}$ can check that $\texttt{pt\_balance}^*$ is valid by accessing to $\mathcal{A}$'s account plaintext balance recorded on MPT on L.
- $v_{\texttt{Mint}}$, the total value of all plaintext amount minted by $\mathcal{A}$. To compute $v_{\texttt{Mint}}$, the challenger $\mathcal{C}$ looks up all $\texttt{Mint}$ transactions placed on L via ***Mint*** queries and sums up the values that were transformed to $\mathcal{A}$.
- $v_{\texttt{Redeem}}$, the total value of all zero-knowledge amount redeemed by $\mathcal{A}$. To compute $v_{\texttt{Redeem}}$, the challenger $\mathcal{C}$ looks up all $\texttt{Redeem}$ transactions placed on L via ***Redeem*** queries and sums up the values that were transformed to $\mathcal{A}$.
- $v_{\mathsf{zk}:\texttt{ACCOUNT}\to\mathcal{A}}$, the total value of payments received by $\mathcal{A}$ from account addresses in $\texttt{ACCOUNT}$. To compute $v_{\mathsf{zk}:\texttt{ACCOUNT}\to\mathcal{A}}$, the challenger $\mathcal{C}$ looks up all $\texttt{Deposit}$ transactions placed on L via ***Deposit*** queries and sums up the values in S_{cmt_v} whose recipient is $\mathcal{A}$.
- $v_{\mathsf{pt}:\texttt{ACCOUNT}\to\mathcal{A}}$, the total value of payments received by $\mathcal{A}$ from account addresses in $\texttt{ACCOUNT}$. To compute $v_{\mathsf{pt}:\texttt{ACCOUNT}\to\mathcal{A}}$, the challenger $\mathcal{C}$ looks up all plaintext transactions placed on L and sums up the values that were transferred to $\mathcal{A}$.
- $v_{\mathsf{zk}:\mathcal{A}\to\texttt{ACCOUNT}}$, the total value of payments sent by $\mathcal{A}$ to account addresses in $\texttt{ACCOUNT}$. To compute $v_{\mathsf{zk}:\mathcal{A}\to\texttt{ACCOUNT}}$, the challenger $\mathcal{C}$ looks up all $\texttt{Send}$ transactions placed on L via ***Send*** queries and sums up the values in S_{cmt_v} whose sender is $\mathcal{A}$.
- $v_{\mathsf{pt}:\mathcal{A}\to\texttt{ACCOUNT}}$, the total value of payments sent by $\mathcal{A}$ to account addresses in $\texttt{ACCOUNT}$. To compute $v_{\mathsf{pt}:\mathcal{A}\to\texttt{ACCOUNT}}$, the challenger $\mathcal{C}$ looks up all plaintext transactions placed on L and sums up the values whose sender is $\mathcal{A}$.

For an honest account u, the following equations hold:

$$v_{\texttt{zk\_unspent}} + v_{\texttt{Redeem}} + v_{\mathsf{zk}:u\to\texttt{ACCOUNT}} = v_{\texttt{Mint}} + v_{\mathsf{zk}:\texttt{ACCOUNT}\to u},$$

$$v_{\texttt{pt\_unspent}} + v_{\texttt{Mint}} + v_{\mathsf{pt}:u\to\texttt{ACCOUNT}} = v_{\texttt{Redeem}} + v_{\mathsf{pt}:\texttt{ACCOUNT}\to u}.$$

Add these two equations together, and we can obtain that

$$v_{\texttt{zk\_unspent}} + v_{\texttt{pt\_unspent}} + v_{\mathsf{zk}:u\to\texttt{ACCOUNT}} + v_{\mathsf{pt}:u\to\texttt{ACCOUNT}} = v_{\mathsf{zk}:\texttt{ACCOUNT}\to u} + v_{\mathsf{pt}:\texttt{ACCOUNT}\to u}.$$

Formally, let Π = (**Setup**, **CreateAccount**, **Mint**, **Redeem**, **Send**, **Deposit**, **VerTx**) be a BlockMaze scheme, $\mathcal{A}$ an adversary, and λ the security parameter. We define the balance experiment $\mathsf{BlockMaze}^{\texttt{BAL}}_{\Pi,\mathcal{A}}(\lambda)$ as follows:

1) The public parameters $\mathsf{pp} := \mathbf{Setup}(1^\lambda)$ is computed and given to $\mathcal{A}$. A BlockMaze oracle $\mathcal{O}^{\text{BM}}$ is initialized.
2) Whenever $\mathcal{A}$ queries $\mathcal{O}^{\text{BM}}$, answer this query along with the ledger L in each step.
3) Continue answering queries of $\mathcal{A}$ until $\mathcal{A}$ sends a table of transfer commitments S_{cmt_v}, the new account balance $\texttt{zk\_balance}^*$ and $\texttt{pt\_balance}^*$.
4) Compute the eight variables mentioned above.
5) The experiment outputs 1 if $(v_{\texttt{zk\_unspent}} + v_{\texttt{pt\_unspent}} + v_{\mathsf{zk}:\mathcal{A}\to\texttt{ACCOUNT}} + v_{\mathsf{pt}:\mathcal{A}\to\texttt{ACCOUNT}})$ is greater than $(v_{\mathsf{zk}:\texttt{ACCOUNT}\to\mathcal{A}} + v_{\mathsf{pt}:\texttt{ACCOUNT}\to\mathcal{A}})$. Otherwise, it outputs 0.

Definition 4 ($\texttt{BAL}$ Security)**.** A BlockMaze scheme Π = (**Setup**, **CreateAccount**, **Mint**, **Redeem**, **Send**, **Deposit**, **VerTx**) is $\texttt{BAL}$ secure, if for all probabilistic polynomial-time adversaries $\mathcal{A}$ there is a negligible function negl such that, for security parameter λ,

$$\Pr\left[\mathsf{BlockMaze}^{\texttt{BAL}}_{\Pi,\mathcal{A}}(\lambda)=1\right] \le \mathsf{negl}(\lambda).$$

Appendix C
Proof of Security

A BlockMaze scheme Π = (**Setup**, **CreateAccount**, **Mint**, **Redeem**, **Send**, **Deposit**, **VerTx**) is *secure* if it satisfies ledger indistinguishability, transaction unlinkability, and balance.

C.1 Proof of Ledger Indistinguishability

We now give a formal proof to prove Theorem 1, which is proved using game-based frameworks. The notations used in this proof are listed below. The adversary $\mathcal{A}$ interacts with a challenger $\mathcal{C}$ as in the L-IND experiment. After receiving a pair of *publicly consistent* queries (Q, Q') from $\mathcal{A}$, $\mathcal{C}$ answers (Q, Q') as in the simulation ∂_{sim}. Thus, $\mathcal{A}$'s advantage in ∂_{sim} (represented by $\mathsf{Adv}^{\partial_{\mathsf{sim}}}$) is 0. We now prove that $\mathsf{Adv}^{\mathtt{L\text{-}IND}}_{\Pi,\mathcal{A}}(\lambda)$ (i.e., $\mathcal{A}$'s advantage in the L-IND experiment) is at most negligibly different than $\mathsf{Adv}^{\partial_{\mathsf{sim}}}$.

TABLE II: Notations

∂_{real}	The original L-IND experiment
∂_i	A hybrid game with a modification of the ∂_{real}
q_{CA}	The total number of ***CreateAccount*** queries issued by $\mathcal{A}$
q_{M}	The total number of ***Mint*** queries issued by $\mathcal{A}$
q_{R}	The total number of ***Redeem*** queries issued by $\mathcal{A}$
q_{S}	The total number of ***Send*** queries issued by $\mathcal{A}$
q_{D}	The total number of ***Deposit*** queries issued by $\mathcal{A}$
Adv^{∂} [1]	$\mathcal{A}$'s advantage in game ∂
$\mathsf{Adv}^{\Pi_{\mathcal{E}}}$	$\mathcal{A}$'s advantage in $\Pi_{\mathcal{E}}$'s IND-CCA and IK-CCA experiments
$\mathsf{Adv}^{\mathtt{PRF}}$	$\mathcal{A}$'s advantage in distinguishing PRF from a random one
$\mathsf{Adv}^{\mathtt{COMM}}$	$\mathcal{A}$'s advantage against the hiding property of COMM

Firstly, we describe a simulation ∂_{sim} in which the adversary $\mathcal{A}$ interacts with a challenger $\mathcal{C}$ as in the queries defined in Appendix B, with the following modification: for each $i \in \{\mathtt{Mint}, \mathtt{Redeem}, \mathtt{Send}, \mathtt{Deposit}\}$, the zk-SNARK keys are generated as $(\mathsf{pk}_{\mathcal{Z}_i}, \mathsf{vk}_{\mathcal{Z}_i}, \mathtt{trap}_i) := \mathcal{S}(C_i)$, to obtain the zero-knowledge trapdoor $\mathtt{trap}_i$. After checking each query from $\mathcal{A}$, the challenger $\mathcal{C}$ answers the queries as below.

- To answer ***CreateAccount*** queries, $\mathcal{C}$ does the same as in $Query($***CreateAccount***$)$ with the following differences: $\mathcal{C}$ computes $(sk, pk) := \Pi_{\mathcal{E}}.\mathrm{KeyGen}(\mathsf{pp})$; then, $\mathcal{C}$ utilizes a random string to replace pk, and computes an account address $addr := \mathtt{CRH}(pk)$; finally, $\mathcal{C}$ stores (sk, pk) in a table and returns $(addr, pk)$ to $\mathcal{A}$.
- To answer ***Mint*** queries, $\mathcal{C}$ makes the following modifications in $Query($***Mint***$, addr_A, v)$: $\mathcal{C}$ samples a uniformly random sn_A; if $addr_A$ is an account address created by a previous query to ***CreateAccount***, then $\mathcal{C}$ samples a balance commitment cmt^*_A on a random input, otherwise, $\mathcal{C}$ computes cmt^*_A as in the **Mint** algorithm; moreover, $\mathcal{C}$ computes and obtains all remaining values as in the **Mint** algorithm; finally, $\mathcal{C}$ constructs a statement $\vec{x_1}$ and computes the Mint proof $prf_m := \mathcal{S}(\mathsf{pk}_{\mathcal{Z}_1}, \vec{x_1}, \mathtt{trap}_{\mathtt{Mint}})$.
- To answer ***Redeem*** queries, $\mathcal{C}$ makes the modifications in $Query($***Redeem***$, addr_A, v)$ as answering ***Mint*** queries, except for the following modification: $\mathcal{C}$ computes the Redeem proof $prf_r := \mathcal{S}(\mathsf{pk}_{\mathcal{Z}_2}, \vec{x_2}, \mathtt{trap}_{\mathtt{Redeem}})$, where $\vec{x_2}$ is a statement.
- To answer ***Send*** queries, $\mathcal{C}$ makes the following modifications in $Query($***Send***$, addr_A, addr_B, v)$: $\mathcal{C}$ first samples a uniformly random sn_A; if $addr_B$ is an account address created by a previous query to ***CreateAccount***, then $\mathcal{C}$ does as follows: (i) sample a transfer commitment cmt_v and a balance commitment cmt^*_A on random inputs, (ii) compute $(sk'_B, pk'_B) := \Pi_{\mathcal{E}}.\mathrm{KeyGen}(\mathsf{pp}_{\mathcal{E}})$; (iii) encrypt $aux_A := \Pi_{\mathcal{E}}.\mathrm{Enc}_{pk'_B}(r)$, where r is a random string of appropriate length in the plaintext space of the encryption scheme; otherwise, $\mathcal{C}$ computes (cmt_v, cmt^*_A, aux_A) as in the **Send** algorithm; moreover, $\mathcal{C}$ computes and obtains all remaining values as in the **Send** algorithm; finally, $\mathcal{C}$ constructs a statement $\vec{x_3}$ and computes the Send proof $prf_s := \mathcal{S}(\mathsf{pk}_{\mathcal{Z}_3}, \vec{x_3}, \mathtt{trap}_{\mathtt{Send}})$.
- To answer ***Deposit*** queries, $\mathcal{C}$ makes the following modifications in $Query($***Deposit***$, addr_B, h_{\mathtt{tx}_{\mathtt{Send}}})$: $\mathcal{C}$ first samples a uniformly random sn_B; if $addr_B$ is an account address created by a previous query to ***CreateAccount***, then $\mathcal{C}$ samples a balance commitment cmt^*_B on a random input, otherwise, $\mathcal{C}$ computes cmt^*_B as in the **Deposit** algorithm; moreover, $\mathcal{C}$ computes and obtains all remaining values as in the **Deposit** algorithm; finally, $\mathcal{C}$ computes the Deposit proof $prf_d := \mathcal{S}(\mathsf{pk}_{\mathcal{Z}_4}, \vec{x_4}, \mathtt{trap}_{\mathtt{Deposit}})$, where $\vec{x_4}$ is a statement.
- To answer ***Insert*** queries, $\mathcal{C}$ does the same as in $Query($***Insert***$, \mathsf{tx})$.

Note that the answer to $\mathcal{A}$ is computed independently of the bit b for each of the above cases. Thus, when $\mathcal{A}$ outputs a guess b', it must be the case that $\Pr[b = b'] = 1/2$, i.e., $\mathcal{A}$'s advantage in ∂_{sim} is 0.

Game ∂_1**.** This is the same as ∂_{real}, except for one modification: $\mathcal{C}$ simulates each zk-SNARK proof. For each $i \in \{\mathtt{Mint}, \mathtt{Redeem}, \mathtt{Send}, \mathtt{Deposit}\}$, the zk-SNARK keys are generated as $(\mathsf{pk}_{\mathcal{Z}_i}, \mathsf{vk}_{\mathcal{Z}_i}, \mathtt{trap}_i) := \mathcal{S}(C_i)$ instead of $\Pi_{\mathcal{Z}}.\mathrm{KeyGen}$, to obtain the zero-knowledge trapdoor $\mathtt{trap}_i$. Then $\mathcal{C}$ computes $prf_i := \mathcal{S}(\mathsf{pk}_{\mathcal{Z}_i}, \vec{x_i}, \mathtt{trap_i})$, without using any witnesses $\vec{a_i}$, instead of using $\Pi_{\mathcal{Z}}.\mathrm{GenProof}$ in the **Mint**, **Redeem**, **Send**, **Deposit** algorithms. Since the zk-SNARK is perfect zero-knowledge, the distribution of the simulated prf_i is identical to that of the proofs computed in ∂_{real}. Moreover, we also modify ∂_{real} such that: each time $\mathcal{A}$ issues a ***CreateAccount*** query, the value pk associated with the returned $addr$ is substituted with a random string of the same length. Since the $(sk, pk) := \Pi_{\mathcal{E}}.\mathrm{KeyGen}(\mathsf{pp})$, the distribution of the simulated (sk, pk) is identical to that of the key pairs computed in ∂_{real}. Hence $\mathsf{Adv}^{\partial_1} = 0$.

Game ∂_2**.** ∂_2 is the same as ∂_1 with one modification: $\mathcal{C}$ utilizes a random string of the suitable length to replace the ciphertext in a Send transaction. If $\mathcal{A}$ sends a ***Send*** query where the address $addr_B$ is an account address created by a previous query to ***CreateAccount***, then $\mathcal{C}$ invokes $\Pi_{\mathcal{E}}.\mathrm{KeyGen}(\mathsf{pp}_{\mathcal{E}})$ to compute (sk'_B, pk'_B) and obtains $aux_A := \Pi_{\mathcal{E}}.\mathrm{Enc}_{pk'_B}(r)$ for a random string r of suitable length; otherwise, $\mathcal{C}$ computes aux_A as in the **Send** algorithm. By Lemma 1 (see below), $|\mathsf{Adv}^{\partial_2} - \mathsf{Adv}^{\partial_1}| \leq 2 \cdot q_{\mathsf{S}} \cdot \mathsf{Adv}^{\Pi_{\mathcal{E}}}$.

Game ∂_3**.** ∂_3 is the same as ∂_2 with one modification: $\mathcal{C}$ utilizes random strings of the suitable length to replace all serial numbers generated by PRF. As the subsequent results of the ***Mint***, ***Redeem***, ***Send***, ***Deposit*** queries, these serial numbers (e.g., sn_A and sn_v) are respectively placed in $\mathtt{tx}_{\mathtt{Mint}}$, $\mathtt{tx}_{\mathtt{Redeem}}$, $\mathtt{tx}_{\mathtt{Send}}$, $\mathtt{tx}_{\mathtt{Deposit}}$. By Lemma 2 (see below), $|\mathsf{Adv}^{\partial_3} - \mathsf{Adv}^{\partial_2}| \leq (q_{\mathsf{M}} + q_{\mathsf{R}} + q_{\mathsf{S}} + q_{\mathsf{D}}) \cdot \mathsf{Adv}^{\mathtt{PRF}}$.

[1] We abuse Adv^{∂} to denote the absolute value of the difference between (i) the L-IND advantage of $\mathcal{A}$ in ∂ and (ii) the L-IND advantage of $\mathcal{A}$ in ∂_{real}.

Game ∂_{sim}. ∂_{sim} defined above is the same as ∂_3 with one modification: $\mathcal{C}$ replaces all balance commitments generated by COMM with commitments to random inputs. As the subsequent results of the ***Mint***, ***Redeem***, ***Send***, ***Deposit*** queries, these balance commitments (e.g., cmt_A, cmt^*_A and cmt_v) are respectively placed in $\mathtt{tx}_{\mathtt{Mint}}$, $\mathtt{tx}_{\mathtt{Redeem}}$, $\mathtt{tx}_{\mathtt{Send}}$, $\mathtt{tx}_{\mathtt{Deposit}}$. By Lemma 3 (see below), $|\mathsf{Adv}^{\partial_{\mathsf{sim}}} - \mathsf{Adv}^{\partial_3}| \leq (q_{\mathsf{M}} + q_{\mathsf{R}} + 2 \cdot q_{\mathsf{S}} + q_{\mathsf{D}}) \cdot \mathsf{Adv}^{\mathtt{COMM}}$.

As mentioned above, we can obtain $\mathcal{A}$'s advantage in the L-IND experiment (i.e., ∂_{real}) by summing over $\mathcal{A}$'s advantages in all games as follows:

$$\mathsf{Adv}^{\mathtt{L\text{-}IND}}_{\Pi,\mathcal{A}}(\lambda) \leq 2 \cdot q_{\mathsf{S}} \cdot \mathsf{Adv}^{\Pi_{\mathcal{E}}} + (q_{\mathsf{M}} + q_{\mathsf{R}} + 2 \cdot q_{\mathsf{S}} + q_{\mathsf{D}}) \cdot (\mathsf{Adv}^{\mathtt{PRF}} + \mathsf{Adv}^{\mathtt{COMM}}).$$

Since $\mathsf{Adv}^{\mathtt{L\text{-}IND}}_{\Pi,\mathcal{A}}(\lambda) := 2 \cdot \Pr\left[\mathsf{BlockMaze}^{\mathtt{L\text{-}IND}}_{\Pi,\mathcal{A}}(\lambda) = 1\right] - 1$ and $\mathcal{A}$'s advantage in the L-IND experiment is negligible in λ, we can conclude that the proof of ledger indistinguishability.

Lemma 1. Let $\mathsf{Adv}^{\Pi_{\mathcal{E}}}$ be $\mathcal{A}$'s advantage in $\Pi_{\mathcal{E}}$'s IND-CCA and IK-CCA experiments. If $\mathcal{A}$ issues q_{S} ***Send*** queries, then $|\mathsf{Adv}^{\partial_2} - \mathsf{Adv}^{\partial_1}| \leq 2 \cdot q_{\mathsf{S}} \cdot \mathsf{Adv}^{\Pi_{\mathcal{E}}}$.

Proof sketch. We utilize a hybrid ∂_{H} as an intermediation between ∂_1 and ∂_2 to prove that $\mathsf{Adv}^{\partial_2}$ is at most negligibly different than $\mathsf{Adv}^{\partial_1}$.

More precisely, ∂_{H} is the same as ∂_1 with one modification: $\mathcal{C}$ utilizes a new public key generated by the $\Pi_{\mathcal{E}}.\mathrm{KeyGen}(\mathsf{pp}_{\mathcal{E}})$, instead of the public key created by a previous query to ***CreateAccount***, to encrypt the same plaintext. After q_{CA} ***CreateAccount*** queries, $\mathcal{A}$ queries the IK-CCA challenger to obtain $pk_{\mathcal{E}} := pk_{\mathcal{E},0}$ where $pk_{\mathcal{E},0}$ is the public key in $(pk_{\mathcal{E},0}, pk_{\mathcal{E},1})$ provided by the IK-CCA challenger. At each ***Send*** query, the IK-CCA challenger encrypts the corresponding plaintext pt as $\mathsf{ct}^* := \Pi_{\mathcal{E}}.\mathrm{Enc}_{pk_{\mathcal{E},\bar{b}}}(\mathsf{pt})$, where $\bar{b}$ is the bit chosen by the IK-CCA challenger, in response to $\mathcal{A}$. Then $\mathcal{C}$ sets $\mathsf{ct} := \mathsf{ct}^*$ and adds $\mathtt{tx}_{\mathtt{Send}}$ (whose aux_A is set by ct) to the ledger L. Finally, $\mathcal{A}$ outputs a guess bit b', which is regarded as the guess in the IK-CCA experiment. Thus, if $\bar{b} = 0$ then $\mathcal{A}$'s view represents ∂_1, while $\mathcal{A}$'s view represents ∂_{H} if $\bar{b} = 1$. Note that $\mathcal{A}$ issues q_{S} ***Send*** queries, then he obtains the q_{S} ciphertexts at most. If the maximum adversarial advantage against the IK-CCA experiment is $\mathsf{Adv}^{\Pi_{\mathcal{E}}}$, then we can conclude that $|\mathsf{Adv}^{\partial_{\mathsf{H}}} - \mathsf{Adv}^{\partial_1}| \leq q_{\mathsf{S}} \cdot \mathsf{Adv}^{\Pi_{\mathcal{E}}}$.

In a similar vein, ∂_2 is the same as ∂_{H} with one modification: $\mathcal{C}$ utilizes a random string of the appropriate length in the plaintext space to replace the plaintext computed in the ***Send*** query. For simplicity, we omit the formal description for IND-CCA experiment, which has a similar pattern above. If the maximum adversarial advantage against the IND-CCA experiment is $\mathsf{Adv}^{\Pi_{\mathcal{E}}}$, then we can conclude that $|\mathsf{Adv}^{\partial_2} - \mathsf{Adv}^{\partial_{\mathsf{H}}}| \leq q_{\mathsf{S}} \cdot \mathsf{Adv}^{\Pi_{\mathcal{E}}}$. Thus, we can sum the above $\mathcal{A}$'s advantages to obtain $|\mathsf{Adv}^{\partial_2} - \mathsf{Adv}^{\partial_1}| \leq 2 \cdot q_{\mathsf{S}} \cdot \mathsf{Adv}^{\Pi_{\mathcal{E}}}$. □

Lemma 2. Let $\mathsf{Adv}^{\mathtt{PRF}}$ be $\mathcal{A}$'s advantage in distinguishing PRF from a random function. If $\mathcal{A}$ issues q_{M} ***Mint*** queries, q_{R} ***Redeem*** queries, q_{S} ***Send*** queries and q_{D} ***Deposit*** queries, then $|\mathsf{Adv}^{\partial_3} - \mathsf{Adv}^{\partial_2}| \leq (q_{\mathsf{M}} + q_{\mathsf{R}} + 2 \cdot q_{\mathsf{S}} + q_{\mathsf{D}}) \cdot \mathsf{Adv}^{\mathtt{PRF}}$.

Proof sketch. We utilize a hybrid ∂_{H} as an intermediation between ∂_2 and ∂_3 to prove that $\mathsf{Adv}^{\partial_3}$ is at most negligibly different than $\mathsf{Adv}^{\partial_2}$.

More precisely, ∂_{H} is the same as ∂_2 with one modification: $\mathcal{C}$ utilizes a random string of the appropriate length to replace the public key pk associated with the returned $addr$ for $\mathcal{A}$'s first ***CreateAccount*** query; then, on each subsequent ***Mint***, ***Redeem***, ***Send***, ***Deposit*** queries, $\mathcal{C}$ replaces sn_A respectively with a random string of appropriate length and simulates the respective zk-SNARK proof (e.g., $prf_m, prf_r, prf_s, prf_d$) with the help of a trapdoor by the simulator $\mathcal{S}$ for $\mathtt{tx}_{\mathtt{Mint}}$, $\mathtt{tx}_{\mathtt{Redeem}}$, $\mathtt{tx}_{\mathtt{Send}}$, $\mathtt{tx}_{\mathtt{Deposit}}$.

Let sk be the random, secret key created by the first ***CreateAccount*** query and employed in PRF to compute $sn^*_A := \mathtt{PRF}(sk_A, r^*_A)$ and $sn_v := \mathtt{PRF}(sk_A, r_v)$ in **Mint**, **Redeem**, **Send**, **Deposit** algorithm. Note that PRF takes different random number r to publish old serial number sn for different transactions (generated by the same account). Moreover, let $\mathcal{O}$ be an oracle that implements either PRF or a random function. Then, we utilize $\mathcal{O}$ to generate all random strings (i.e., sn) for the two cases of $\mathcal{O}$ in response to a distinguisher (as an experiment). If $\mathcal{O}$ implements a random function, then it represents ∂_{H}, while the experiment represents ∂_2 if $\mathcal{O}$ implements PRF. Thus, $\mathcal{A}$'s advantage in distinguishing PRF from a random one is at most $\mathsf{Adv}^{\mathtt{PRF}}$.

In a similar vein, we extend the above pattern to all $q_{\mathsf{M}} + q_{\mathsf{R}} + 2 \cdot q_{\mathsf{S}} + q_{\mathsf{D}}$ oracle-generated serial numbers (corresponding to what happens in ∂_3). We can obtain that $\mathcal{A}$'s advantage in distinguishing PRF from a random one is at most $(q_{\mathsf{M}} + q_{\mathsf{R}} + 2 \cdot q_{\mathsf{S}} + q_{\mathsf{D}}) \cdot \mathsf{Adv}^{\mathtt{PRF}}$. Finally, we deduce that $|\mathsf{Adv}^{\partial_3} - \mathsf{Adv}^{\partial_2}| \leq (q_{\mathsf{M}} + q_{\mathsf{R}} + 2 \cdot q_{\mathsf{S}} + q_{\mathsf{D}}) \cdot \mathsf{Adv}^{\mathtt{PRF}}$. □

Lemma 3. Let $\mathsf{Adv}^{\mathtt{COMM}}$ be $\mathcal{A}$'s advantage against the hiding property of COMM. If $\mathcal{A}$ issues q_{M} ***Mint*** queries, q_{R} ***Redeem*** queries, q_{S} ***Send*** queries and q_{D} ***Deposit*** queries, then $|\mathsf{Adv}^{\partial_{\mathsf{sim}}} - \mathsf{Adv}^{\partial_3}| \leq (q_{\mathsf{M}} + q_{\mathsf{R}} + 2 \cdot q_{\mathsf{S}} + q_{\mathsf{D}}) \cdot \mathsf{Adv}^{\mathtt{COMM}}$.

Proof sketch. This proof can be proved with the similar pattern used in Lemma 2 above. On each ***Mint***, ***Redeem***, ***Send***, ***Deposit*** query, $\mathcal{C}$ replaces the balance commitment $cmt^* := \mathtt{COMM}_{\mathtt{bc}}(addr, value, sn^*_A, r^*)$ and the transfer commitment $cmt_v := \mathtt{COMM}_{\mathtt{tc}}(addr, v, pk, sn_v, r_v, sn_A)$ with random strings of the appropriate length. Thus $\mathcal{A}$'s advantage in distinguishing this modified experiment from ∂_3 is at most $\mathsf{Adv}^{\mathtt{COMM}}$. If we extend it to all q_{M} ***Mint*** queries, all q_{R} ***Redeem*** queries, all q_{S} ***Send*** queries and all q_{D} ***Deposit*** queries, and utilize random strings of the suitable length to replace $q_{\mathsf{M}} + q_{\mathsf{R}} + 2 \cdot q_{\mathsf{S}} + q_{\mathsf{D}}$ commitments (i.e., cmt_A and cmt_v), then we obtain ∂_{sim}, and conclude that $|\mathsf{Adv}^{\partial_{\mathsf{sim}}} - \mathsf{Adv}^{\partial_3}| \leq (q_{\mathsf{M}} + q_{\mathsf{R}} + 2 \cdot q_{\mathsf{S}} + q_{\mathsf{D}}) \cdot \mathsf{Adv}^{\mathtt{COMM}}$. □

C.2 Proof of Transaction Unlinkability

Letting $\mathcal{T}$ be the table of zero-knowledge transactions (i.e., $\mathtt{tx}_{\mathtt{Send}}$ and $\mathtt{tx}_{\mathtt{Depoist}}$) generated by $\mathcal{O}^{\mathrm{BM}}$ in response to ***Send*** and ***Deposit*** queries. $\mathcal{A}$ wins the TR-UL experiment whenever it outputs a pair of zero-knowledge transactions $(\mathtt{tx}, \mathtt{tx}')$ if one of the following conditions holds: (i) the recipients of payments contained in $(\mathtt{tx}, \mathtt{tx}')$ are the same if $\mathtt{tx} = \mathtt{tx}_{\mathtt{Send}}$; and (ii) the senders of payments contained in $(\mathtt{tx}, \mathtt{tx}')$ are the same if $\mathtt{tx} = \mathtt{tx}_{\mathtt{Deposit}}$.

Suppose $\mathcal{A}$ outputs a pair of Send transactions $(\texttt{tx}_{\texttt{Send}}, \texttt{tx}'_{\texttt{Send}})$, where $\texttt{tx}_{\texttt{Send}}$ satisfies the following equations:

$$\texttt{tx}_{\texttt{Send}} := (addr_A, sn_A, cmt^*_A, cmt_v, aux_A, auth_{enc}, prf_s)$$
$$cmt_v = \texttt{COMM}_{\texttt{tc}}(addr_A, v, pk_B, sn_v, r_v, sn_A)$$
$$aux_A := \Pi_{\mathcal{E}}.\text{Enc}_{pk_B}(\{v, sn_v, r_v, sn_A\}),$$

and $\texttt{tx}'_{\texttt{Send}}$ satisfies the following equations:

$$\texttt{tx}'_{\texttt{Send}} := (addr'_A, sn'_A, {cmt^*_A}', cmt'_v, aux'_A, auth'_{enc}, prf'_s)$$
$$cmt'_v = \texttt{COMM}_{\texttt{tc}}(addr'_A, v', pk'_B, sn'_v, r'_v, sn'_A)$$
$$aux'_A := \Pi_{\mathcal{E}}.\text{Enc}_{pk'_B}(\{v', sn'_v, r'_v, sn'_A\}).$$

$\mathcal{A}$ wins the TR-UL experiment if the recipients of payments contained in $(\texttt{tx}_{\texttt{Send}}, \texttt{tx}'_{\texttt{Send}})$ are the same, i.e., $pk_B = pk'_B$. There are three ways for $\mathcal{A}$ to distinguish whether $pk_B \stackrel{?}{=} pk'_B$: (i) distinguish the public keys from the ciphertexts; (ii) distinguish the public keys from the transfer commitments; (iii) distinguish the public keys from the zero-knowledge proofs.

For condition (i), $\mathcal{A}$ must distinguish (pk_B, pk'_B) based on the different ciphertexts (aux_A, aux'_A), which indicates that $\mathcal{A}$ should win the IK-CCA experiment in Lemma 1. For condition (ii), $\mathcal{A}$ must distinguish (pk_B, pk'_B) from the different commitments (cmt_v, cmt'_v) without knowing other secret values (i.e., hidden variables), which indicates that $\mathcal{A}$ should break the *hiding* property of COMM in Lemma 3. For condition (iii), $\mathcal{A}$ must distinguish (pk_B, pk'_B) from different zero-knowledge proof (prf_s, prf'_s), which indicates that $\mathcal{A}$ should break the *proof of knowledge* property of the zk-SNARK. However, since the security of Lemma 1, Lemma 3 and zk-SNARK, $\mathcal{A}$ cannot distinguish the two recipients (i.e., public keys) from (aux_A, aux'_A) and (cmt_v, cmt'_v).

Suppose $\mathcal{A}$ outputs a pair of Deposit transactions $(\texttt{tx}_{\texttt{Deposit}}, \texttt{tx}'_{\texttt{Deposit}})$, where $\texttt{tx}_{\texttt{Deposit}}$ satisfies the following equations:

$$\texttt{tx}_{\texttt{Deposit}} := (seq, rt_{cmt}, sn_B, cmt^*_B, sn_v, pk_B, prf_d)$$
$$cmt_v = \texttt{COMM}_{\texttt{tc}}(addr_A, v, pk_B, sn_v, r_v, sn_A),$$

and $\texttt{tx}'_{\texttt{Deposit}}$ satisfies the following equations:

$$\texttt{tx}'_{\texttt{Deposit}} := (seq', rt'_{cmt}, sn'_B, {cmt^*_B}', sn'_v, pk'_B, prf'_d)$$
$$cmt'_v = \texttt{COMM}_{\texttt{tc}}(addr'_A, v', pk'_B, sn'_v, r'_v, sn'_A)$$

$\mathcal{A}$ wins the TR-UL experiment if the senders of payments contained in $(\texttt{tx}_{\texttt{Deposit}}, \texttt{tx}'_{\texttt{Deposit}})$ are the same, i.e., $addr_A = addr'_A$. There are two ways for $\mathcal{A}$ to distinguish whether $addr_A \stackrel{?}{=} addr'_A$: (i) distinguish the address of senders from the zero-knowledge proof; (ii) obtain the transfer commitments used in Deposit transactions from $\mathcal{A}$'s view, and distinguish the address of senders combining with previous Send transactions.

For condition (i), $\mathcal{A}$ must distinguish $(addr_A, addr'_A)$ from different zero-knowledge proof (prf_d, prf'_d), which indicates that $\mathcal{A}$ should break the *proof of knowledge* property of the zk-SNARK. For condition (ii), if $\mathcal{A}$ can analyze (cmt_v, cmt'_v) without knowing other secret values (i.e., hidden variables), then, he can distinguish $(addr_A, addr'_A)$ by seeking senders of the previous transactions $(\texttt{tx}_{\texttt{Send}}, \texttt{tx}'_{\texttt{Send}})$ containing (cmt_v, cmt'_v) respectively. Note that (cmt_v, cmt'_v) are not visible for anyone during the generation of $(\texttt{tx}_{\texttt{Deposit}}, \texttt{tx}'_{\texttt{Deposit}})$. There are two ways for $\mathcal{A}$ to obtain different transfer commitments: (a) the Merkle roots; (b) the zero-knowledge proofs. For condition (a), $\mathcal{A}$ must analyze (cmt_v, cmt'_v) from different Merkle roots (rt_{cmt}, rt'_{cmt}), which indicates that $\mathcal{A}$ should break the collision resistance property of CRH. For condition (b), $\mathcal{A}$ must analyze (cmt_v, cmt'_v) from different zero-knowledge proofs (prf_d, prf'_d), which indicates that $\mathcal{A}$ should break the *proof of knowledge* property of the zk-SNARK. However, since the security of zk-SNARK and CRH, $\mathcal{A}$ cannot neither distinguish the two senders from (prf_d, prf'_d) nor analyze (cmt_v, cmt'_v) from (rt_{cmt}, rt'_{cmt}) and (prf_d, prf'_d).

To conclude, due to the security of zk-SNARK, CRH, COMM and encryption schemes, the adversary $\mathcal{A}$ cannot distinguish the unlinkability from $(\texttt{tx}_{\texttt{Send}}, \texttt{tx}'_{\texttt{Send}})$ nor $(\texttt{tx}_{\texttt{Deposit}}, \texttt{tx}'_{\texttt{Deposit}})$.

C.3 Proof of Balance

The BAL experiment is changed without affecting $\mathcal{A}$'s view as follows: for each Deposit transaction $\texttt{tx}_{\texttt{Deposit}}$ on the ledger L, $\mathcal{C}$ computes the zk-SNARK proof $prf_d := \Pi_{\mathcal{Z}}.\text{GenProof}(\mathsf{PK}_{\mathcal{Z}}, \vec{x}_4, \vec{a}_4)$ where $\vec{x}_4$ is a statement corresponding to $\texttt{tx}_{\texttt{Deposit}}$ and $\vec{a}_4$ is a witness for the zk-SNARK instance $\vec{x}_4$, then $\mathcal{C}$ organizes all $(tx, \vec{a})$ as an augmented ledger $(L, \boldsymbol{A})$, which is a list of matched pairs $(\texttt{tx}_{\texttt{Deposit}}, \vec{a_4}_j)$ where $\texttt{tx}_{\texttt{Deposit}}$ is the j-th Deposit transaction in L and $\vec{a_4}_j \in \boldsymbol{A}$ is the witness of $\texttt{tx}_{\texttt{Deposit}}$ for $\vec{x_4}_j$.

Definition 5 (Balanced ledger)**.** An augmented ledger $(L, \boldsymbol{A})$ is balanced if the following conditions hold:

- *Condition I.* Each $(\texttt{tx}_{\texttt{Deposit}}, \vec{a_4}) \in (L, \boldsymbol{A})$ publishes openings (e.g., sn_B) of a unique balance commitment cmt_B, which is the output of a previous zero-knowledge transaction before $\texttt{tx}_{\texttt{Deposit}}$ on L.
- *Condition II.* Each $(\texttt{tx}_{\texttt{Deposit}}, \vec{a_4}) \in (L, \boldsymbol{A})$ publishes openings (e.g., sn_v and pk_B) of a unique transfer commitment cmt_v, which is the output of a previous Send transaction before $\texttt{tx}_{\texttt{Deposit}}$ on L.
- *Condition III.* For all $(\texttt{tx}_{\texttt{Deposit}}, \vec{a_4}) \in (L, \boldsymbol{A})$ publish distinct and unique openings (e.g., sn_v, sn_B and pk_B) of the commitment (i.e., cmt_v and cmt_B).
- *Condition IV.* Each cmt_B, cmt_v, cmt^*_B to values $value_B$, v, $value^*_B$ (respectively) contained in $(\texttt{tx}_{\texttt{Deposit}}, \vec{a_4}) \in (L, \boldsymbol{A})$ satisfies the condition that $value_B + v = value^*_B$.
- *Condition V.* The values (i.e., $addr_A$, v, pk_B, sn_v, r_v, sn_A) used to compute cmt_v are respectively equal to the values for cmt'_v, if $cmt_v = cmt'_v$ where cmt_v is employed in $(\texttt{tx}_{\texttt{Deposit}}, \vec{a_4}) \in (L, \boldsymbol{A})$, and cmt'_v is the output of a previous Send transaction before $\texttt{tx}_{\texttt{Deposit}}$.
- *Condition VI.* If $(\texttt{tx}_{\texttt{Deposit}}, \vec{a_4}) \in (L, \boldsymbol{A})$ was inserted by $\mathcal{A}$, and cmt_v used in $\texttt{tx}_{\texttt{Deposit}}$ is the output of a previous Send transaction $\texttt{tx}'$, then $addr \notin \texttt{ACCOUNT}$ where $addr := \texttt{CRH}(pk_B)$ is the recipient's account address of $\texttt{tx}'$.

One can prove that $(L, \boldsymbol{A})$ is balanced if the following equation holds: $v_{\text{zk\_unspent}} + v_{\text{pt\_unspent}} + v_{\text{zk}:\mathcal{A}\to\texttt{ACCOUNT}} + v_{\text{pt}:\mathcal{A}\to\texttt{ACCOUNT}} = v_{\text{zk}:\texttt{ACCOUNT}\to\mathcal{A}} + v_{\text{pt}:\texttt{ACCOUNT}\to\mathcal{A}}$.

For each of the above conditions, we utilize a contraction to prove that each case is in a negligible probability. Note that we suppose $\Pr\left[\mathcal{A}(\overline{\textit{Con-i}}) = 1\right]$ to denote a non-negligible probability that $\mathcal{A}$ wins but violates *Condition i*.

Violating *Condition I*. Each $(\texttt{tx}_{\texttt{Deposit}}, \vec{a_4}) \in (L, \boldsymbol{A})$, where $\texttt{tx}_{\texttt{Deposit}}$ was not inserted by $\mathcal{A}$, must satisfy *Condition I* in $\mathcal{O}^{\text{BM}}$; thus, $\Pr\left[\mathcal{A}(\overline{\textit{Con-I}}) = 1\right]$ is a probability that $\mathcal{A}$ inserts $\texttt{tx}_{\texttt{Deposit}}$ to construct a pair $(\texttt{tx}_{\texttt{Deposit}}, \vec{a_4}) \in (L, \boldsymbol{A})$ where cmt_B used in $\texttt{tx}_{\texttt{Deposit}}$ is not the output of any previous zero-knowledge transaction before $\texttt{tx}_{\texttt{Deposit}}$. However, each $\texttt{tx}_{\texttt{Deposit}}$ utilizes the witness $\vec{a_4}$, which must contain a balance commitment cmt_B for the serial number sn_B, to compute the proof proving the validity of $\texttt{tx}_{\texttt{Deposit}}$. Obviously, cmt_B is the output of an earlier zero-knowledge transaction and recorded on MPT in L. Therefore, if cmt_B to sn_B does not previously appear in L, then it means that this violation contradicts the binding property of COMM in Lemma 3.

Violating *Condition II*. Each $(\texttt{tx}_{\texttt{Deposit}}, \vec{a_4}) \in (L, \boldsymbol{A})$, where $\texttt{tx}_{\texttt{Deposit}}$ was not inserted by $\mathcal{A}$, must satisfy *Condition II* in $\mathcal{O}^{\text{BM}}$; thus, $\Pr\left[\mathcal{A}(\overline{\textit{Con-II}}) = 1\right]$ is a probability that $\mathcal{A}$ inserts $\texttt{tx}_{\texttt{Deposit}}$ to construct a pair $(\texttt{tx}_{\texttt{Deposit}}, \vec{a_4}) \in (L, \boldsymbol{A})$ where cmt_v used in $\texttt{tx}_{\texttt{Deposit}}$ is not the output of any previous Send transaction before $\texttt{tx}_{\texttt{Deposit}}$. However, each $\texttt{tx}_{\texttt{Deposit}}$ utilizes the witness $\vec{a_4}$, which must contain a authentication path $path$ for a Merkle tree, to compute the proof proving the validity of $\texttt{tx}_{\texttt{Deposit}}$. Obviously, the Merkle tree, whose root rt_{cmt} is published, is constructed using the transfer commitment cmt_v of any previous Send transactions on L. More precisely, if cmt_v does not previously appear in L, then it means that $path$ is invalid but with a valid rt_{cmt}. Therefore, this violation contradicts the collision resistance of CRH.

Violating *Condition III*. Each $(\texttt{tx}_{\texttt{Deposit}}, \vec{a_4}) \in (L, \boldsymbol{A})$ publishes a unique and distinct sn_v, $(sn_B,\ pk_B)$ for cmt_v, cmt_B (respectively). Obviously, $\Pr\left[\mathcal{A}(\overline{\textit{Con-III}}) = 1\right]$ is a probability that $\mathcal{A}$ wins in the following two situations: (i) spending the same balance commitment cmt_B in two zero-knowledge transactions; or (ii) receiving the same transfer commitment cmt_v. In (i), it means that two Deposit transactions $\texttt{tx}_{\texttt{Deposit}}, \texttt{tx}'$ recorded on L spend the same balance commitment cmt_B, but publish two different serial numbers sn_B and sn'_B, furthermore, their corresponding witnesses $\vec{a_4}, \vec{a_4}'$ must contain different openings of cmt_B since both transactions are valid Deposit transactions on L. Therefore, this violation contradicts the binding property of COMM. In a similar vein, for (ii), it means that two Deposit transactions $\texttt{tx}_{\texttt{Deposit}}, \texttt{tx}'$ receive the same transfer commitment cmt_v but publish two different serial numbers sn_v and sn'_v. Therefore, this violation also contradicts the binding property of COMM in Lemma 3.

Violating *Condition IV*. Each $(\texttt{tx}_{\texttt{Deposit}}, \vec{a_4}) \in (L, \boldsymbol{A})$ contains a proof ensuring that cmt_B, cmt_v, cmt^*_B to values $value_B$, v, $value^*_B$ (respectively) satisfy the equation $value_B + v = value^*_B$. Obviously, $\Pr\left[\mathcal{A}(\overline{\textit{Con-IV}}) = 1\right]$ is a probability that the equation $value_B + v \neq value^*_B$ holds. Therefore, this violation contradicts the *proof of knowledge* property of the zk-SNARK.

Violating *Condition V*. Each $(\texttt{tx}_{\texttt{Deposit}}, \vec{a_4}) \in (L, \boldsymbol{A})$ contains values (i.e., $addr_A$, v, pk_B, sn_v, r_v, sn_A) of cmt_v, and cmt_v is a also transfer commitment to values (i.e., $addr_A$, v', pk_B, sn_v, r_v, sn_A) in a previous Send transaction. Obviously, $\Pr\left[\mathcal{A}(\overline{\textit{Con-V}}) = 1\right]$ is a probability that the equation $v \neq v'$ holds. Therefore, this violation also contradicts the binding property of COMM in Lemma 3.

Violating *Condition VI*. Each $(\texttt{tx}_{\texttt{Deposit}}, \vec{a_4}) \in (L, \boldsymbol{A})$ publishes the recipient's account address $addr := \texttt{CRH}(pk_B)$ of a transfer commitment cmt_v. Obviously, $\Pr\left[\mathcal{A}(\overline{\textit{Con-VI}}) = 1\right]$ is a probability that an inserted Deposit transaction $\texttt{tx}_{\texttt{Deposit}}$ publishes $addr$ of cmt_v which is the output of a previous Send transaction $\texttt{tx}'$ whose recipient's account address $addr$ lies in ACCOUNT; moreover, the witness associated to $\texttt{tx}'$ contains pk such that $addr = \texttt{CRH}(pk)$. Therefore, this violation contradicts the collision resistance of CRH.

Finally, we utilize the similar structure of the argument to prove balance for the other three transactions (i.e., Mint, Redeem and Send) generated by $\mathcal{O}^{\text{BM}}$ in response to ***Mint***, ***Redeem*** and ***Send*** queries, and obtain that $\Pr\left[\mathsf{BlockMaze}^{\texttt{BAL}}_{\Pi,\mathcal{A}}(\lambda) = 1\right]$ is negligible in λ.

APPENDIX D
ALGORITHMS OF BLOCKMAZE SCHEMES

For convenience, we summarize algorithms employed in a BlockMaze scheme Π = (**Setup**, **CreateAccount**, **Mint**, **Redeem**, **Send**, **Deposit**, **VerTx**) in Fig. 1.

REFERENCES

[1] CryptoRekt, "Blackpaper: Verge Currency," https://vergecurrency.com/static/blackpaper/verge-blackpaper-v5.0.pdf, Accessed 05/18/2019.
[2] B. Conrad and F. Shirazi, "A Survey on Tor and I2P," in *Proc. 9th International Conference on Internet Monitoring and Protection*, 2014.
[3] E. Ben-Sasson, A. Chiesa, C. Garman, M. Green, I. Miers, E. Tromer, and M. Virza, "Zerocash: Decentralized anonymous payments from Bitcoin," in *IEEE Symposium on Security and Privacy (SP)*, 2014, pp. 459–474.
[4] Komodo, "Komodo white paper: Advanced Blockchain Technology, Focused On Freedom," https://komodoplatform.com/whitepaper, Accessed 05/19/2019.
[5] R. Viglione, R. Versluis, and J. Lippencott, "Zen white paper," 2017.
[6] N. Nicolas Saberhagen, "Cryptonote v2.0," https://cryptonote.org/whitepaper.pdf, 2013.
[7] I. Peverell, "Introduction to MimbleWimble and Grin," https://github.com/mimblewimble/grin/blob/master/doc/intro.md, Accessed 05/21/2019.
[8] NavCoin, "NavCoin: An Easy To Use Decentralized Cryptocurrency," https://navcoin.org/, Accessed 05/19/2019.
[9] I. Miers, C. Garman, M. Green, and A. D. Rubin, "Zerocoin: Anonymous Distributed E-Cash from Bitcoin," in *IEEE Symposium on Security and Privacy (SP)*, 2013, pp. 397–411.
[10] E. Duffield and D. Diaz, "Dash: A PrivacyCentric CryptoCurrency," https://github.com/dashpay/dash/wiki/Whitepaper, 2015.
[11] PIVX, "PIVX white paper: Private Instant Verified Transaction," https://pivx.org/wp-content/uploads/2018/10/PIVX-White.pdf, Accessed 05/19/2019.
[12] T. Ruffing, P. Moreno-Sanchez, and A. Kate, "Coinshuffle: Practical decentralized coin mixing for bitcoin," in *Proc. European Symposium on Research in Computer Security*. Springer, 2014, pp. 345–364.
[13] A. Vazquez, "ZeroCT: Improving Zerocoin with Confidential Transactions and more," *IACR Cryptology ePrint Archive*, 2019.

Setup
The algorithm generates a list of public parameters.

- inputs: a security parameter λ
- outputs: public parameters pp

1) Compute $\mathsf{pp}_{\mathcal{E}} := \Pi_{\mathcal{E}}.\text{Setup}(1^{\lambda})$.
2) Compute $\mathsf{pp}_{\mathcal{Z}} := \Pi_{\mathcal{Z}}.\text{Setup}(1^{\lambda})$.
3) For each $i \in \{\texttt{Mint}, \texttt{Redeem}, \texttt{Send}, \texttt{Deposit}\}$
   a) Construct a circuit C_i.
   b) Compute $(\mathsf{pk}_{\mathcal{Z}_i}, \mathsf{vk}_{\mathcal{Z}_i}) := \Pi_{\mathcal{Z}}.\text{KeyGen}(C_i)$.
4) Set $\mathsf{PK}_{\mathcal{Z}} := \bigcup \mathsf{pk}_{\mathcal{Z}_i}$ and $\mathsf{VK}_{\mathcal{Z}} := \bigcup \mathsf{vk}_{\mathcal{Z}_i}$.
5) Output $\mathsf{pp} := (\mathsf{pp}_{\mathcal{E}}, \mathsf{pp}_{\mathcal{Z}}, \mathsf{PK}_{\mathcal{Z}}, \mathsf{VK}_{\mathcal{Z}})$.

CreateAccount
The algorithm creates an account address and key pair for a user.

- inputs: public parameters pp
- outputs: $(addr, (sk, pk))$

1) Compute an account key pair $(sk, pk) := \Pi_{\mathcal{E}}.\text{KeyGen}(\mathsf{pp}_{\mathcal{E}})$.
2) Compute an account address $addr := \texttt{CRH}(pk)$.
3) Output the $(addr, (sk, pk))$.

Mint
This algorithm merges a plaintext amount with the current zero-knowledge balance of an account (say A).

- inputs:
  - public parameters pp
  - the current zero-knowledge balance $\texttt{zk\_balance}_A$
  - the current plaintext balance $\texttt{pt\_balance}_A$
  - account private key sk_A
  - a plaintext amount v to be converted into a zero-knowledge amount
- outputs:
  - the new zero-knowledge balance $\texttt{zk\_balance}^*_A$
  - a $\texttt{Mint}$ transaction $\texttt{tx}_{\texttt{Mint}}$

1) Return fail if $\texttt{pt\_balance}_A < v$.
2) Parse $\texttt{zk\_balance}_A$ as $(cmt_A, addr_A, value_A, sn_A, r_A)$.
3) Generate a new random number r^*_A.
4) Sample a new serial number $sn^*_A := \texttt{PRF}(sk_A, r^*_A)$.
5) Compute $cmt^*_A := \texttt{COMM}_{\texttt{bc}}(addr_A, value_A + v, sn^*_A, r^*_A)$.
6) Set $\vec{x}_1 := (cmt_A, addr_A, sn_A, cmt^*_A, v)$.
7) Set $\vec{a}_1 := (value_A, r_A, sk_A, sn^*_A, r^*_A)$.
8) Compute $prf_m := \Pi_{\mathcal{Z}}.\text{GenProof}(\mathsf{PK}_{\mathcal{Z}}, \vec{x}_1, \vec{a}_1)$.
9) Set $\texttt{tx}_{\texttt{Mint}} := (addr_A, v, sn_A, cmt^*_A, prf_m)$, $\texttt{zk\_balance}^*_A := (cmt^*_A, addr_A, value_A + v, sn^*_A, r^*_A)$.
10) Output $\texttt{zk\_balance}^*_A$ and $\texttt{tx}_{\texttt{Mint}}$.

Redeem
This algorithm converts a zero-knowledge amount back into the plaintext balance of an account (say A).

- inputs:
  - public parameters pp
  - the current zero-knowledge balance $\texttt{zk\_balance}_A$
  - account private key sk_A
  - a plaintext amount v to be converted back from zero-knowledge balance
- outputs:
  - the new zero-knowledge balance $\texttt{zk\_balance}^*_A$
  - a $\texttt{Redeem}$ transaction $\texttt{tx}_{\texttt{Redeem}}$

1) Parse $\texttt{zk\_balance}_A$ as $(cmt_A, addr_A, value_A, sn_A, r_A)$.
2) Return fail if $value_A < v$.
3) Generate a new random number r^*_A.
4) Sample a new serial number $sn^*_A := \texttt{PRF}(sk_A, r^*_A)$.
5) Compute $cmt^*_A := \texttt{COMM}_{\texttt{bc}}(addr_A, value_A - v, sn^*_A, r^*_A)$.
6) Set $\vec{x}_2 := (cmt_A, addr_A, sn_A, cmt^*_A, v)$.
7) Set $\vec{a}_2 := (value_A, r_A, sk_A, sn^*_A, r^*_A)$.
8) Compute $prf_r := \Pi_{\mathcal{Z}}.\text{GenProof}(\mathsf{PK}_{\mathcal{Z}}, \vec{x}_2, \vec{a}_2)$.
9) Set $\texttt{tx}_{\texttt{Redeem}} := (addr_A, v, sn_A, cmt^*_A, prf_r)$, $\texttt{zk\_balance}^*_A := (cmt^*_A, addr_A, value_A - v, sn^*_A, r^*_A)$.
10) Output $\texttt{zk\_balance}^*_A$ and $\texttt{tx}_{\texttt{Redeem}}$.

Send
This algorithm sends a zero-knowledge amount from sender A to recipient B.

- inputs:
  - public parameters pp
  - the current zero-knowledge balance $\texttt{zk\_balance}_A$
  - account private key sk_A
  - recipient's public key pk_B
  - a plaintext amount v to be transferred
- outputs:
  - the new zero-knowledge balance $\texttt{zk\_balance}^*_A$
  - a $\texttt{Send}$ transaction $\texttt{tx}_{\texttt{Send}}$

1) Parse $\texttt{zk\_balance}_A$ as $(cmt_A, addr_A, value_A, sn_A, r_A)$.
2) Generate a new random number r_v.
3) Sample a new serial number $sn_v := \texttt{PRF}(sk_A, r_v)$.
4) Compute $cmt_v := \texttt{COMM}_{\texttt{tc}}(addr_A, v, pk_B, sn_v, r_v, sn_A)$.
5) Set $aux_A := \Pi_{\mathcal{E}}.\text{Enc}_{pk_B}(\{v, sn_v, r_v, sn_A\})$.
6) Generate a new random number r^*_A.
7) Sample a new serial number $sn^*_A := \texttt{PRF}(sk_A, r^*_A)$.
8) Compute $cmt^*_A := \texttt{COMM}_{\texttt{bc}}(addr_A, value_A - v, sn^*_A, r^*_A)$.
9) Compute $h_{enc} := \texttt{CRH}(aux_A)$.
10) Compute $auth_{enc} := \texttt{PRF}(sk_A, h_{enc})$.
11) Set $\vec{x}_3 := (cmt_A, addr_A, sn_A, cmt_v, cmt^*_A, h_{enc}, auth_{enc})$.
12) Set $\vec{a}_3 := (value_A, r_A, sk_A, v, pk_B, sn_v, r_v, sn^*_A, r^*_A)$.
13) Compute $prf_s := \Pi_{\mathcal{Z}}.\text{GenProof}(\mathsf{PK}_{\mathcal{Z}}, \vec{x}_3, \vec{a}_3)$.
14) Set $\texttt{tx}_{\texttt{Send}} := (addr_A, sn_A, cmt^*_A, cmt_v, aux_A, auth_{enc}, prf_s)$, $\texttt{zk\_balance}^*_A := (cmt^*_A, addr_A, value_A - v, sn^*_A, r^*_A)$.
15) Output $\texttt{zk\_balance}^*_A$ and $\texttt{tx}_{\texttt{Send}}$.

Deposit
This algorithm makes a recipient (say B) check and deposit a received payment into his account.

- inputs:
  - the current ledger $\texttt{Ledger}_{\texttt{T}}$
  - public parameters pp
  - account key pair (sk_B, pk_B)
  - the hash of a send transaction $h_{\texttt{tx}_{\texttt{Send}}}$
  - the current zero-knowledge balance $\texttt{zk\_balance}_B$
- outputs:
  - the new zero-knowledge balance $\texttt{zk\_balance}^*_B$
  - a $\texttt{Deposit}$ transaction $\texttt{tx}_{\texttt{Deposit}}$

1) Parse $\texttt{zk\_balance}_B$ as $(cmt_B, addr_B, value_B, sn_B, r_B)$.
2) Obtain transaction information of $h_{\texttt{tx}_{\texttt{Send}}}$ from $\texttt{Ledger}_{\texttt{T}}$
   - the send transaction is $\texttt{tx}_{\texttt{Send}}$,
   - the block number of $\texttt{tx}_{\texttt{Send}}$ is N.
3) Parse $\texttt{tx}_{\texttt{Send}}$ as $(addr_A, sn_A, cmt^*_A, cmt_v, aux_A, auth_{enc}, prf_s)$.
4) Decrypt shared parameters $(v, sn_v, r_v, sn_A) := \Pi_{\mathcal{E}}.\text{Dec}_{sk_B}(aux_A)$.
5) Return fail if sn_v appears in $\texttt{SNSet}_{\texttt{T}}$.
6) Return fail if $cmt_v \neq \texttt{COMM}_{\texttt{tc}}(addr_A, v, pk_B, sn_v, r_v, sn_A)$.
7) Randomly select a set $seq := \{n_1, n_2, ..., N, ..., n_9\}$ from existed block numbers.
8) Construct a Merkle tree $\texttt{MT}$ over $\bigcup_{n \in seq} \texttt{TCMSet}_{\texttt{n}}$.
9) Compute $path := \texttt{Path}(cmt_v)$ and rt_{cmt} over $\texttt{MT}$.
10) Generate a new random number r^*_B.
11) Sample a new serial number $sn^*_B := \texttt{PRF}(sk_B, r^*_B)$.
12) Compute $cmt^*_B := \texttt{COMM}_{\texttt{bc}}(addr_B, value_B + v, sn^*_B, r^*_B)$.
13) Set $\vec{x}_4 := (pk_B, sn_v, rt_{cmt}, cmt_B, addr_B, sn_B, cmt^*_B)$, $\vec{a}_4 := (cmt_v, addr_A, v, r_v, sn_A, value_B, r_B, sk_B, sn^*_B, r^*_B, path)$.
14) Compute $prf_d := \Pi_{\mathcal{Z}}.\text{GenProof}(\mathsf{PK}_{\mathcal{Z}}, \vec{x}_4, \vec{a}_4)$.
15) Set $\texttt{tx}_{\texttt{Deposit}} := (seq, rt_{cmt}, sn_B, cmt^*_B, sn_v, pk_B, prf_d)$, $\texttt{zk\_balance}^*_B := (cmt^*_B, addr_B, value_B + v, sn^*_B, r^*_B)$.
16) Output $\texttt{zk\_balance}^*_B$ and $\texttt{tx}_{\texttt{Deposit}}$.

VerTx
This algorithm checks the validity of all zero-knowledge transactions.

- inputs:
  - the current ledger $\texttt{Ledger}_{\texttt{T}}$
  - public parameters pp
  - a zero-knowledge transaction $\texttt{tx}$
- outputs: bit b

1) If given a transaction $\texttt{tx}$ is $\texttt{tx}_{\texttt{Mint}}$
   a) Parse $\texttt{tx}_{\texttt{Mint}}$ as $(addr_A, v, sn_A, cmt^*_A, prf_m)$.
   b) Obtain related information of $addr_A$ from $\texttt{Ledger}_{\texttt{T}}$
      - the current plaintext balance is $\texttt{pt\_balance}_A$,
      - the current balance commitment is cmt_A.
   c) Return 0 if $\texttt{pt\_balance}_A < v$.
   d) Return 0 if sn_A appears in $\texttt{SNSet}_{\texttt{T}}$.
   e) Set $\vec{x}_1 := (cmt_A, addr_A, sn_A, cmt^*_A, v)$.
   f) Output $b := \Pi_{\mathcal{Z}}.\text{VerProof}(\mathsf{VK}_{\mathcal{Z}}, \vec{x}_1, prf_m)$.
2) If given a transaction $\texttt{tx}$ is $\texttt{tx}_{\texttt{Redeem}}$
   a) Parse $\texttt{tx}_{\texttt{Redeem}}$ as $(addr_A, v, sn_A, cmt^*_A, prf_r)$.
   b) Obtain related information of $addr_A$ from $\texttt{Ledger}_{\texttt{T}}$
      - the current balance commitment is cmt_A.
   c) Return 0 if sn_A appears in $\texttt{SNSet}_{\texttt{T}}$.
   d) Set $\vec{x}_2 := (cmt_A, addr_A, sn_A, cmt^*_A, v)$.
   e) Output $b := \Pi_{\mathcal{Z}}.\text{VerProof}(\mathsf{VK}_{\mathcal{Z}}, \vec{x}_2, prf_r)$.
3) If given a transaction $\texttt{tx}$ is $\texttt{tx}_{\texttt{Send}}$
   a) Parse $\texttt{tx}_{\texttt{Send}}$ as $(addr_A, sn_A, cmt^*_A, cmt_v, aux_A, auth_{enc}, prf_s)$.
   b) Obtain related information of $addr_A$ from $\texttt{Ledger}_{\texttt{T}}$
      - the current balance commitment is cmt_A.
   c) Return 0 if sn_A appears in $\texttt{SNSet}_{\texttt{T}}$.
   d) Compute $h_{enc} := \texttt{CRH}(aux_A)$.
   e) Set $\vec{x}_3 := (cmt_A, addr_A, sn_A, cmt_v, cmt^*_A, h_{enc}, auth_{enc})$.
   f) Output $b := \Pi_{\mathcal{Z}}.\text{VerProof}(\mathsf{VK}_{\mathcal{Z}}, \vec{x}_3, prf_s)$.
4) If given a transaction $\texttt{tx}$ is $\texttt{tx}_{\texttt{Deposit}}$
   a) Parse $\texttt{tx}_{\texttt{Deposit}}$ as $(seq, rt_{cmt}, sn_B, cmt^*_B, sn_v, pk_B, prf_d)$.
   b) Compute $addr_B := \texttt{CRH}(pk_B)$.
   c) Obtain related information of $addr_B$ from $\texttt{Ledger}_{\texttt{T}}$
      - the current balance commitment is cmt_B.
   d) Return 0 if sn_B or sn_v appears in $\texttt{SNSet}_{\texttt{T}}$.
   e) Return 0 if rt_{cmt} is not the Merkle root over $\bigcup_{n \in seq} \texttt{TCMSet}_{\texttt{n}}$.
   f) Set $\vec{x}_4 := (pk_B, sn_v, rt_{cmt}, cmt_B, addr_B, sn_B, cmt^*_B)$.
   g) Output $b := \Pi_{\mathcal{Z}}.\text{VerProof}(\mathsf{VK}_{\mathcal{Z}}, \vec{x}_4, prf_d)$.

Fig. 1: BlockMaze main algorithms