Security Model for Privacy-Preserving Blockchain-Based Cryptocurrency Systems

Mayank Raikwar[1(✉)], Shuang Wu[2], and Kristian Gjøsteen[3]

[1] University of Oslo, Oslo, Norway
mayankr@ifi.uio.no
[2] DNV, Trondheim, Norway
Shuang.Wu@dnv.no
[3] Norwegian University of Science and Technology (NTNU), Trondheim, Norway
kristian.gjosteen@ntnu.no

Abstract. Privacy-preserving blockchain-based cryptocurrency systems have become quite popular as a way to provide confidential payments. These cryptocurrency systems differ in their designs, underlying cryptography, and confidentiality level. Some of these systems provide confidentiality for their users or transactions or both. There has been a thriving interest in constructing different privacy-preserving cryptocurrency systems with improved security and additional features. Nevertheless, many of these available systems lack security models which makes it hard to prove the security properties of these systems.

Despite the differences in the privacy notions of existing privacy-preserving cryptocurrency systems, in this paper, we present a first attempt to create a general framework for a privacy-preserving blockchain-based bank PBB. We present the security properties of this system and model the security experiments for each of the properties. Our PBB model can also work for bank-less cryptocurrency systems. Henceforth, we present a brief security analysis for one of the most notable privacy-preserving cryptocurrencies, Monero, using the security model of the PBB system. Our analysis proves that our PBB system can be easily used to formalise the security of other available privacy-preserving cryptocurrencies.

1 Introduction

Privacy-preserving blockchain-based cryptocurrency systems are financial systems that allow users to conduct cryptocurrency transactions. Similar to online banks, users can store their funds in cryptocurrencies, transfer cryptocurrencies between each other and withdraw their funds to fiat money. The privacy-preserving feature is that these financial activities are not completely transparent to the outsiders, in contrast to Bitcoin [16] and Ethereum [22] where

M. Raikwar and S. Wu—This work was done in part while the author was at NTNU, Norway.

S. Li et al. (Eds.): NSS 2023, LNCS 13983, pp. 137–152, 2023.
https://doi.org/10.1007/978-3-031-39828-5_8

transaction information is public. There are many constructions of privacy-preserving blockchain-based cryptocurrency systems [2,3,5,9–11,18,20], both in the literature and deployed. Even though these systems offer the same functionalities/services to the end-users, each of them has its own definitions and security claims, and some of them even lack security proofs despite being deployed. The diversity of the definitions and security models makes it hard to analyse and compare these protocols in general, resulting in less convincing security claims. There is therefore a need for a unified formal description and security model for these systems which we attempt to provide in this paper. The security model presented in the paper can be further matured by formal verification.

Giving a general definition and security model for privacy-preserving cryptocurrency systems is not a simple task since the systems vary in many aspects. Depending on the amount of information released to the public, these systems provide different levels of privacy which is captured by the security model. Some designs [3,7] provide confidential transactions where transfer values are hidden, but the identities of the senders and recipients are public. In anonymous cryptocurrencies such as Monero [20] and Zcash [18], the identities that are relevant to a certain transaction are hidden, but they still have a somewhat different form of anonymity. In addition to the differences in privacy levels, these systems also differ structurally from each other. They can generally be divided as: Layer 1 blockchain-based anonymous cryptocurrency like Monero, Zcash and Zether [3] and Layer 2 cryptocurrency like Pribank [7]. Pribank introduces a semi-trusted centralized bank operator above the blockchain which hides the sensitive information while relaying the users' transactions to the blockchain periodically.

Despite the above differentiation among the systems, we attempt to build a general model for privacy-preserving cryptocurrency systems. Firstly, we rename these systems as blockchain-based online banks, as the functionality and the services these systems provide are mostly identical to the traditional online bank system. Our model is capable of expressing the differences of privacy, and it captures the most essential functionality and properties of an online financial system (online bank) regardless of how they use the blockchain. We provide a security model that has two essential security properties: *Transaction Indistinguishability* (T-IND) and *Overdraft Safety*. Transaction indistinguishability captures the privacy features of an online bank, informally speaking, a transaction record on the blockchain does not reveal any information about the user's activity. Overdraft safety claims an honest user is always able to withdraw his/her full balance. This implies that no adversary can steal others' funds or withdraw more than it has.

The definition of transaction indistinguishability was first proposed by Zether without formalisation. There is a corresponding security definition in Zcash which is called *Ledger Indistinguishability* (L-IND). We prove that these two concepts are equivalent, but we believe that transaction indistinguishably is intuitively easier to use. In addition, we add a leakage function to the security experiment which controls the amount of information that is leaked to the adversary. By this leakage function, the model is able to express different privacy levels, i.e., from confidential transactions to completely anonymous transactions.

The proposed security model is built by following and modifying the security model of PriBank [7]. Our proposed model can also be employed to assess the security of bank-less cryptocurrency systems. We demonstrate the applicability and usability of our model by applying it to Monero as a case study.

1.1 Contribution

This paper presents a general definition and security model for the privacy-preserving blockchain-based online bank for cryptocurrencies. Our model can express different levels of privacy regardless of the structure of the systems. We prove that two privacy-related security definitions in the literature, *Transaction indistinguishability* and *Ledger Indistinguishability*, are equivalent. We also discuss the relationship among the definitions that are related to the integrity of the protocol, namely, *Balance* and *Overdraft Safety*. We give a general security model for a blockchain-based online bank system that has two essential properties: transaction indistinguishability and overdraft safety. We believe the proposed security notions are helpful to simplify and formalise the security analysis for privacy-preserving blockchain-based cryptocurrencies. We further analyse the security properties of anonymous cryptocurrency system Monero.

1.2 Related Work

There are many constructions of privacy-preserving blockchain based online banks for cryptocurrencies [2,3,5,9–11,18–20]. These systems are either built on top of an underlying blockchain or on top of a smart contract. The main goal of these systems are to provide privacy for their users and transactions. A detailed overview of can be found here [1]. These systems employ cryptographic primitives such as zero-knowledge proofs, ring signatures, or mixing techniques, and achieve a variety of different trade-offs with respect to privacy.

After the proposal of the Zcash [18] system, many follow-up [2,8,9,12,15,23] ledger-based constructions were proposed offering different meaningful privacy notions for cryptocurrency systems. Furthermore, the security model of these systems is based on the security model of Zcash. Systems such as Hawk [12], Zexe [2] and Z-Channel [23] provide on-chain privacy by performing high-frequency transactions (computation) off-chain. Hawk and Zexe employ zk-SNARK proof systems to generate privacy-preserving transactions. But neither system defines its security model and underlying properties. Z-Channel [23] is a micropayment system that adds multi-signature and time-lock functionalities to the Zcash. Z-Channel follows the security model of Zcash to prove its security guarantees.

Zcash inspired systems such as Lelantus [8], Spark [9] and a scheme by Mitani and Otsuka [15] provide on-chain privacy with additional usability features. Lelantus [8] is an anonymous payment system that ensures confidentiality, small proof size, and fast transaction verification. Spark [9] is a modification to the Lelantus that provides recipient privacy and opt-in transaction visibility to third parties. Mitani and Otsuka [15] constructed a Confidential and Auditable Payment (CAP) scheme. A CAP scheme uploads encrypted transactions to the

ledger and also allows an authority to audit the transactions while keeping the transactions confidential. The security of all these on-chain privacy-enabled payment schemes is defined by following the security model of the Zcash system.

Privacy-preserving systems can also be built on top of smart contracts. Zether [3], zKay [19] and Kachina [10] are such systems built on Ethereum. These systems establish privacy-preserving smart contracts to provide confidential payment or confidential data. Zether [3] defines its security properties but does not provide formal security proofs which are essential to prove the security of an anonymous payment system. zKay [19] defines its privacy model by defining a language zKay for writing smart contracts with private data. Further, these contracts are transformed to be executable on Ethereum. zKay presents different security notions related to indistinguishability and privacy of data. However, the security proofs in zKay are informal and require more formalization.

Kachina [10] provides a unified security model based on the Universal Composition (UC) model to deploy privacy-preserving, general-purpose smart contracts. The security model of Kachina consists of complex proofs and does not capture properties, e.g., liveness. Kachina gives a proof sketch of Zcash but it is hard to justify whether Kachina actually manages to capture all the security properties through the UC-emulated Zcash contract. Hence, Kachina's security model cannot be used to assess the security of other privacy-preserving systems.

Another class of privacy-enabled systems is based on ring signature. The most notable one is Monero [20]. Using ring signatures Monero achieves anonymity of transacting parties. To achieve confidentiality, Monero uses confidential transactions together with a ring signature, referred to as Ring Confidential Transaction (RingCT). Although there has been a decent amount of work to explore the attacks on Monero [21], to the best of our knowledge only a recent work [4] formalises the security of Monero (published after the submission of this work).

There have been several attempts to analyze the security of RingCT, but due to the complexity of RingCT, most of them either provide informal proofs or miss the fundamental functionality. Moreover, Monero limits the ring size to only a few accounts. Therefore, to solve these issues, a new scheme, Omniring [13] was constructed that not only solves the above issues but also provides improved privacy without sacrificing performance. Omniring does provide a security model followed by rigorous proofs for its scheme but its model is quite complex and cannot be generalized to assess the security of other privacy-preserving systems.

Other works providing provable security notions are Quisquis [6] and an anonymous mixing scheme [14]. The security proofs in these schemes provide targeted security notions but it is not clear how these notions can be generalized.

The above described privacy-preserving systems define a number of security properties. A few main security properties can be (informally) described as:

- *Transaction Indistinguishability* Given two different transactions $\mathsf{tx}_0, \mathsf{tx}_1$ from an adversary $\mathcal{A}$, the ledger L records only one transaction tx_i where $i \in \{0, 1\}$, the adversary $\mathcal{A}$ cannot distinguish which transaction was recorded.
- *Ledger Indistinguishability* Given two different ledgers $\mathsf{L}_0, \mathsf{L}_1$ constructed by an adversary $\mathcal{A}$ using queries to two privacy-preserving system oracles, the adversary $\mathcal{A}$ cannot distinguish between L_0 and L_1.

- *Overdraft Safety* Given an adversary $\mathcal{A}$, an honest user can always spend (or withdraw) the funds that he rightfully owns.
- *Balance* No bounded adversary $\mathcal{A}$ can control more money than he minted or received.

Although there have been many constructions of privacy-preserving systems, as discussed above only a few systems define their security. This demonstrates a need to define a generic privacy-preserving blockchain-based cryptocurrency system and formalize its security. Therefore, we present a formal definition of a privacy-preserving blockchain-based bank system (PBB), followed by an analysis of essential security properties. We analyze the security of Monero system using our framework. Due to the expressiveness of our framework, our analysis can be further used to model the security of other structurally different systems having different levels of privacy such as Zcash, Dash [5], zKay [19], and Spark [9].

2 Privacy-Preserving Blockchain-Based Bank

Privacy-preserving cryptocurrency systems, despite different structures, aim to provide end-users services similar to an online bank. Based on this fact, we call these systems privacy-preserving blockchain-based online banks. In this section, we formally define this notion. In blockchain-based online banks, an entity always keeps two types of states: permanent state and temporary state. Permanent states are states that are already recorded on the blockchain, temporary states are states not recorded on the blockchain yet. We denote $\mathsf{TempSt}_{\mathsf{x}}$ as temporary state and St_{x} as permanent state of an entity x (bank or a user).

We adapt the blockchain-based bank (BBank) defined in [7] and modify it to formally define a unified privacy-preserving blockchain-based bank (PBB). The BBank is a Layer 2 account-based ledger that works periodically in terms of epochs. Hence, the PBB is also based on an account-based model. However, our PBB system can also be used for the security analysis of UTXO-based systems.

Note: Depending on the concrete structure of a cryptocurrency system and its underlying ledger model, some algorithms in this definition can be eliminated.

Definition 1 (PBB). *A Privacy-preserving Blockchain-based Bank* ***PBB*** *is a tuple of algorithms* $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{EstablishBank}, \mathsf{NewUser}, \mathsf{Deposit}, \mathsf{Withdraw}, \mathsf{Pay}, \mathsf{Commit}, \mathsf{Contract})$ *with the following syntax and semantics*

- *The algorithm* Setup *takes input* 1^{λ}*, where* λ *is the security parameter and generates the public parameters* pp*, to be distributed off-chain. It also initialises a set* $\mathsf{KeyList} \leftarrow \emptyset$.
- *The algorithm* KeyGen *takes the public parameters* pp *as input, and generates a key pair for a user* $(\mathsf{pk}_{\mathsf{u}}, \mathsf{sk}_{\mathsf{u}})$ *or a bank* $(\mathsf{pk}_{\mathsf{b}}, \mathsf{sk}_{\mathsf{b}})$.
- *The algorithm* $\mathsf{EstablishBank}$ *runs only once and establishes a bank. It takes the public parameters* pp *and a key pair* $(\mathsf{pk}_{\mathsf{b}}, \mathsf{sk}_{\mathsf{b}})$ *of a bank as inputs, generates an initial state of the bank* $\mathsf{TempSt}_{\mathsf{b}}$ *and a transaction* trans*. Once the transaction* trans *is accepted by the blockchain, the smart contract is launched.* $(\mathsf{trans}, \mathsf{TempSt}_{\mathsf{b}}) \leftarrow \mathsf{EstablishBank}(\mathsf{pk}_{\mathsf{b}}, \mathsf{sk}_{\mathsf{b}}, \mathsf{pp})$

- *The algorithm* NewUser *is performed by a user to register on the smart contract but without any deposit for her account yet. It takes the public parameters* pp*, public key* $\mathsf{pk_b}$ *of the bank and the key pair* $(\mathsf{pk_u}, \mathsf{sk_u})$ *of a user as inputs, and outputs an initial state of the user* $\mathsf{TempSt_u}$ *and a transaction* trans.
  $(\mathsf{trans}, \mathsf{TempSt_u}) \leftarrow \mathsf{NewUser}(\mathsf{pk_b}, \mathsf{pk_u}, \mathsf{sk_u}, \mathsf{pp})$
- *The protocol* Deposit *is run by the bank operator and a user to deposit money on the smart contract. It takes the public parameters* pp*, the key pairs and states of a user* $\mathsf{pk_u}, \mathsf{sk_u}, \mathsf{St_u}$ *and of a bank* $\mathsf{pk_b}, \mathsf{sk_b}, \mathsf{St_b}$*, epoch counter* epoch*, deposit value* v *as inputs, outputs a transaction* trans *and temporary states of user and bank. Once the transaction* trans *get accepted by the smart contract, the user gets a commitment for her initial balance.*

$$(\mathsf{trans}, \mathsf{TempSt_b}, \mathsf{TempSt_u}) \leftarrow \mathsf{Deposit}(\mathsf{pk_b}, \mathsf{sk_b}, \mathsf{pk_u}, \mathsf{sk_u}, \mathsf{St_b}, \mathsf{St_u}, \mathsf{pp}, \mathsf{v}, \mathsf{epoch})$$

- *The algorithm* Withdraw *is performed by a registered user who wants to exit the bank. It takes the public parameter* pp*, the key pair* $(\mathsf{pk_u}, \mathsf{sk_u})$ *of a user, states of the user and the bank* $\mathsf{St_u}, \mathsf{St_b}$*, and the amount* v *of the user the bank holds at the epoch counter* epoch*, generates a transaction* trans *and updates the temporary states of this user and the bank.*
  $(\mathsf{trans}, \mathsf{TempSt_b}, \mathsf{TempSt_u}) \leftarrow \mathsf{Withdraw}(\mathsf{pk_u}, \mathsf{sk_u}, \mathsf{St_b}, \mathsf{St_u}, \mathsf{pp}, \mathsf{v}, \mathsf{epoch})$
- *The protocol* Pay *is run by the bank and a user (payer* p*) to send funds to other users. It takes the public key* $\mathsf{pk_r}$ *of the receiver* r*, the key pair* $(\mathsf{pk_p}, \mathsf{sk_p})$ *of the payer and* $(\mathsf{pk_b}, \mathsf{sk_b})$ *of the bank, the temporary states* $\mathsf{TempSt_p}, \mathsf{TempSt_b}$ *of the payer and the bank, the epoch counter* epoch *and the transferred amount* v' *as inputs, and then it updates the temporary states of the payer and the bank.*

$$(\mathsf{TempSt'_p}, \mathsf{TempSt'_b}) \leftarrow \mathsf{Pay}(\mathsf{pk_b}, \mathsf{sk_b}, \mathsf{pk_p}, \mathsf{sk_p}, \mathsf{pk_r}, \mathsf{TempSt_p}, \mathsf{TempSt_b}, \mathsf{pp}, \mathsf{v}', \mathsf{epoch})$$

- *The algorithm* Commit *is performed by the bank at the end of each epoch* epoch*. It takes the key pair* $(\mathsf{pk_b}, \mathsf{sk_b})$*, and the states* $\mathsf{St_b}, \mathsf{TempSt_b}$ *of the bank as inputs and generates a transaction* trans *and updates the temporary state.*

$$(\mathsf{trans}, \mathsf{TempSt'_b}) \leftarrow \mathsf{Commit}(\mathsf{pk_b}, \mathsf{sk_b}, \mathsf{St_b}, \mathsf{TempSt_b}, \mathsf{epoch})$$

- *The algorithm* Contract *takes the public parameters* pp*, a transaction* trans*, all users' states* $\{\mathsf{St_u}\}, \{\mathsf{TempSt_u}\}$ *and bank states* $\mathsf{St_b}, \mathsf{TempSt_b}$ *as inputs and then updates all of them.*

$$(\mathsf{St'_b}, \{\mathsf{St'_u}\}, \mathsf{TempSt'_b}, \{\mathsf{TempSt'_u}\}) \leftarrow \mathsf{Contract}(\{\mathsf{St_u}\}, \{\mathsf{TempSt_u}\}, \mathsf{St_b}, \mathsf{TempSt_b}, \mathsf{trans}, \mathsf{pp})$$

3 Security Properties

In this section, we provide a brief description of security properties associated with the privacy-preserving blockchain-based online bank for cryptocurrencies. To define these properties, first, we define a notion of *Public Consistent* that

$\mathbf{Q} = (\mathbf{KeyGen})$
1. Compute $(\mathsf{pk}, \mathsf{sk}) := \mathsf{KeyGen}(\mathsf{pp})$
2. Add $(\mathsf{pk}, \mathsf{sk})$ to the key list $\mathsf{KeyList}$
3. Output the public key pk

$\mathbf{Q} = (\mathbf{EstablishBank}, \mathsf{pk_b}, \mathsf{sk_b})$
1. If $(\mathsf{pk_b}, \mathsf{sk_b})$ is not in $\mathsf{KeyList}$, output $\perp$
2. Start a bank instance $(\mathsf{trans}, \mathsf{TempSt_b}) \leftarrow \mathsf{EstablishBank}(\mathsf{pk_b}, \mathsf{sk_b}, \mathsf{pp})$
3. Store the key pair $(\mathsf{pk_b}, \mathsf{sk_b})$ and the temporary state $\mathsf{TempSt_b}$ of the bank, initialise the epoch counter as 1
4. Output trans

$\mathbf{Q} = (\mathbf{NewUser}, \mathsf{pk_u}, \mathsf{sk_u})$
1. If $(\mathsf{pk_u}, \mathsf{sk_u})$ is not in $\mathsf{KeyList}$, output $\perp$
2. Compute $(\mathsf{trans}, \mathsf{Tempst_u}) \leftarrow \mathsf{NewUser}(\mathsf{pk_b}, \mathsf{pk_u}, \mathsf{sk_u}, \mathsf{pp})$
3. Store the key pair $(\mathsf{pk_u}, \mathsf{sk_u})$ and the temporary state $\mathsf{TempSt_u}$ of the user
4. Output trans

$\mathbf{Q} = (\mathbf{Deposit}, \mathsf{pk_u}, \mathsf{v}, \mathsf{epoch})$
1. If $(\mathsf{pk_u}, \mathsf{sk_u}, \mathsf{TempSt_u})$ is not recorded, output $\perp$
2. Execute this instance of deposit protocol $(\mathsf{trans}, \mathsf{TempSt_b}, \mathsf{TempSt_u}) \leftarrow \mathsf{Deposit}(\mathsf{pk_b}, \mathsf{sk_b}, \mathsf{pk_u}, \mathsf{sk_u}, \mathsf{St_b}, \mathsf{St_u}, \mathsf{pp}, \mathsf{v}, \mathsf{epoch})$
3. Record the temporary states along with the transaction $(\mathsf{trans}, \mathsf{TempSt_b}, \mathsf{TempSt_u})$ if already not recorded
4. Output trans

$\mathbf{Q} = (\mathbf{Pay}, \mathsf{pk_p}, \mathsf{pk_r}, \mathsf{v}')$
1. If $\mathsf{pk_p}$ or $\mathsf{pk_r}$ is not in the $\mathsf{KeyList}$, output $\perp$
2.Execute this instance of the pay protocol for payer p and receiver r. $(\mathsf{TempSt}'_\mathsf{p}, \mathsf{TempSt}'_\mathsf{b}) \leftarrow \mathsf{Pay}(\mathsf{pk_b}, \mathsf{sk_b}, \mathsf{pk_p}, \mathsf{sk_p}, \mathsf{pk_r}, \mathsf{TempSt_p}, \mathsf{TempSt_b}, \mathsf{pp}, \mathsf{v}', \mathsf{epoch})$
3. Record and output the updated temporary states of the user and the bank $(\mathsf{TempSt}'_\mathsf{b}, \mathsf{TempSt}'_\mathsf{u})$

$\mathbf{Q} = (\mathbf{Withdraw}, \mathsf{pk_u}, \mathsf{v})$
1. Compute $(\mathsf{trans}, \mathsf{TempSt_b}, \mathsf{TempSt_u}) \leftarrow \mathsf{Withdraw}(\mathsf{pk_u}, \mathsf{sk_u}, \mathsf{St_b}, \mathsf{St_u}, \mathsf{pp}, \mathsf{v}, \mathsf{epoch})$
2. Output trans

$\mathbf{Q} = (\mathbf{Commit}, \mathsf{epoch})$
1. Compute $(\mathsf{trans}, \mathsf{TempSt}'_\mathsf{b}) \leftarrow \mathsf{Commit}(\mathsf{pk_b}, \mathsf{sk_b}, \mathsf{St_b}, \mathsf{TempSt_b}, \mathsf{epoch})$
2. Output trans

$\mathbf{Q} = (\mathbf{Contract}, \mathsf{trans})$
1. Compute $(\mathsf{St}'_\mathsf{b}, \{\mathsf{St}'_\mathsf{u}\}, \mathsf{TempSt}'_\mathsf{b}, \{\mathsf{TempSt}'_\mathsf{u}\}) \leftarrow \mathsf{Contract}(\{\mathsf{St_u}\}, \{\mathsf{TempSt_u}\}, \mathsf{St_b}, \mathsf{TempSt_b}, \mathsf{trans}, \mathsf{pp})$
2. Output $\{\mathsf{St}'_\mathsf{u}\}, \mathsf{St}'_\mathsf{b}$

$\mathbf{Q} = (\mathbf{Reveal}, \mathsf{pk})$
Output the secret key and the state of the user/bank who owns pk, i.e., output $\mathsf{sk_u}, \mathsf{St_u}$

Fig. 1. Query Description in Blockchain-based Bank Experiment

will be used in security proofs. Further, we give formal descriptions of the security properties in different ledger models. For the game-based security proofs, we describe the queries in Fig. 1 that an adversary can make in the security experiment. Our security definitions are based on this experiment.

Public Consistent: In our model, an adversary $\mathcal{A}$ will submit queries containing pairs of transactions. Public consistent applies restrictions for the queries that the adversary can make, with the goal being to prevent adversary from winning trivially. Informally, public consistent requires that the queries sent by an adversary $\mathcal{A}$ have transactions of matching type that are identical in terms of publicly-revealed information and the information related to addresses controlled by the adversary $\mathcal{A}$. Apart from public consistent, we also define a leakage function that captures the amount of information that leaked to the adversary for every query that the adversary makes.

Definition 2 (Leakage Function). *A leakage function* $\mathsf{Leakage}$ *maps a transaction to the query information leaked:* $\eta \leftarrow \mathsf{Leakage}(\mathbf{Q})$

Definition 3 (Public Consistent). *We require the query pairs* $(\mathbf{Q}_0, \mathbf{Q}_1)$ *for* **Commit** *and* **Pay** *must be jointly consistent with respect to public information and* $\mathcal{A}$*'s view, namely:*

- *For all the users that the adversary controls (adversary has asked* **Reveal** *query for them), their states in the two banks should be consistent.*
- *If one of the queries* $\mathbf{Q}_0$ *and* $\mathbf{Q}_1$ *is not legitimate, the other query will not be proceeded by the experiment as well.*
- *The leaked information of* $\mathbf{Q}_0$ *and* $\mathbf{Q}_1$ *should be the same, i.e.,* $\mathsf{Leakage}(\mathbf{Q_0}) = \mathsf{Leakage}(\mathbf{Q_1})$.

3.1 Privacy

Privacy in a privacy-preserving blockchain-based online bank can refer to different meanings, ranging from the privacy of balance or identities of transacting parties to the privacy of transacting amount. Therefore, to capture these privacy requirements, two main security notions have been defined, transaction indistinguishability and ledger indistinguishability. Our preferred notion of transaction indistinguishability captures privacy at a basic level, where each transaction reveals no new information about transacting parties' activity. Ledger indistinguishability meanwhile captures the big picture where the ledger reveals no new information to an adversary beyond the publicly revealed information.

Ledger Indistinguishability. This property states that even in the presence of an adversary that can adaptively induce honest users to perform PBB operations, the ledger does not reveal any new information to the adversary except the publicly available information. Given two PBB scheme oracles $\mathsf{O}_0^{\mathsf{PBB}}$ and $\mathsf{O}_1^{\mathsf{PBB}}$, and two ledgers L_0 and L_1 constructed by a bounded adversary $\mathcal{A}$ using public consistent blockchain-bank queries to the two oracles, ledger indistinguishability implies that the adversary $\mathcal{A}$ cannot distinguish between L_0 and L_1.

Ledger indistinguishability is defined by an experiment $\mathsf{L\text{-}IND}$, which involves a polynomial-time adversary $\mathcal{A}$ attempting to break a given PBB scheme. Given a PBB scheme Π, an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$, the experiment $\mathsf{L\text{-}IND}(\Pi, \mathcal{A})$ proceeds as follows: First the challenger $\mathcal{C}$ samples a random bit b and initialises two PBB scheme oracles $\mathsf{O}_0^{\mathsf{PBB}}$ and $\mathsf{O}_1^{\mathsf{PBB}}$, maintaining ledgers L_0 and L_1. Throughout, the challenger $\mathcal{C}$ allows the adversary $\mathcal{A}$ to issue queries to $\mathsf{O}_0^{\mathsf{PBB}}$ and $\mathsf{O}_1^{\mathsf{PBB}}$, thus controlling the behaviour of honest parties on L_0 and L_1. Further, the challenger $\mathcal{C}$ provides the adversary $\mathcal{A}$ with the view of both ledgers, but in a randomized order: $\mathsf{L}_{\mathsf{Left}} := \mathsf{L}_b$ and $\mathsf{L}_{\mathsf{Right}} := \mathsf{L}_{1-b}$. The adversary's goal is to distinguish whether the view he sees corresponds to $(\mathsf{L}_{\mathsf{Left}}, \mathsf{L}_{\mathsf{Right}}) = (\mathsf{L}_0, \mathsf{L}_1)$, i.e. $b = 0$, or to $(\mathsf{L}_{\mathsf{Left}}, \mathsf{L}_{\mathsf{Right}}) = (\mathsf{L}_1, \mathsf{L}_0)$, i.e. $b = 1$. The formal description is here [18].

Transaction Indistinguishability. Transaction indistinguishability states that given two different queries of an adversary, only one of the two queries is

processed and the ledger is updated with the corresponding transaction. Transaction indistinguishability states that the adversary cannot distinguish which query maps to the recorded transaction. Based on the leakage from the queries sent by the adversary to the ledger, this property can also indicate other notions of privacy such as confidential transactions or anonymity of the transacting parties.

Transaction indistinguishability is defined by an experiment T-IND, which involves a polynomial-time adversary $\mathcal{A}$ attempting to break a given PBB scheme. Given a PBB scheme Π and an adversary $\mathcal{A}$, the (probabilistic) experiment $\mathsf{T\text{-}IND}(\Pi, \mathcal{A})$ proceeds as follows: The adversary $\mathcal{A}$ is allowed to send all the queries (defined in Fig. 1) except the reveal query for the secret key of the bank. In the challenge phase, first, the experiment randomly chooses $b \leftarrow \{0, 1\}$. The adversary is allowed to make multiple challenge queries. For each challenge query $\mathbf{Q} = \mathbf{Challenge}(\mathbf{Q_0}, \mathbf{Q_1})$ sent by the adversary $\mathcal{A}$, these two queries $\mathbf{Q_0}, \mathbf{Q_1}$ leak same information and the experiment only performs Q_b. At the end of the challenge phase, the adversary sends commit query $\mathbf{Q} = \mathbf{Commit}$ and receives the output trans_b. Finally, the adversary outputs a bit $b' \in \{0, 1\}$, and wins the game if $b' = b$. In this experiment T-IND, the queries sent by the adversary $\mathcal{A}$ during the challenge phase should also be public consistent.

3.2 Security

Following, we describe overdraft safety and balance properties and discuss the necessity to capture these in a privacy-preserving blockchain-based online bank.

Overdraft Safety. Informally, overdraft safety specifies that an honest user can withdraw all the balance that he owns in the blockchain. In UTXO based model, overdraft safety means that an honest user can always spend his unspent outputs inductively. In an account-based model, it means that an honest user can withdraw all the balance from his account (using smart contract). This security requirement prohibits an adversary to withdraw more than what it has since otherwise there must be an honest user who cannot withdraw all of his balance.

Experiment. Overdraft safety is defined by an experiment Overdraft, which involves a polynomial-time adversary $\mathcal{A}$ attempting to break a given PBB scheme. We now describe the Overdraft experiment for the PBB system. Given a PBB scheme Π and an adversary $\mathcal{A}$, the (probabilistic) experiment $\mathsf{Overdraft}(\Pi, \mathcal{A})$ proceeds as the adversary is capable of sending all the queries in the experiment that we define in Fig. 1. In addition, if the system model involves a transaction relay entity, the adversary can send $\mathbf{Q} = \mathbf{Reveal}$ for the secret key and state of this entity. In the challenge epoch, the adversary wins if in a certain epoch, there is an honest user who tries and fails to withdraw all his balance within one epoch.

Balance. This property requires that no bounded adversary $\mathcal{A}$ can own more money than what he minted or received via payments from others. In other words, adversary $\mathcal{A}$ cannot spend more than what he owns. This property states

that the total balance of honest users should not exceed the total balance of the system. In case of a UTXO-based system, an adversary $\mathcal{A}$ can spend more money (public unspent outputs) by directly putting a transaction on the ledger through pouring or by asking honest parties to create such transactions.

Experiment. The balance property is formalised by an experiment $\mathsf{Balance}$, which involves a polynomial-time adversary $\mathcal{A}$ attempting to break a given PBB scheme. Given a PBB scheme Π and adversary $\mathcal{A}$, in the experiment $\mathsf{Balance}(\Pi, \mathcal{A})$, the adversary $\mathcal{A}$ adaptively interacts with a challenger $\mathcal{C}$ and produces a set of unspent coins. Given Addr as a set of honest user's addresses in ledger L, adversary $\mathcal{A}$ wins the game if the total value the adversary can spend or has spent already is greater than the value it has minted or received that means $\mathsf{v}_{unspent} + \mathsf{v}_{\mathcal{A} \longrightarrow \mathsf{Addr}} > \mathsf{v}_{mint} + \mathsf{v}_{\mathsf{Addr} \longrightarrow \mathcal{A}}$.

Note: Security properties such as transaction non-malleability, transaction unlinkability and transaction untraceablility are not covered in this work.

4 Relation Between T-IND and L-IND

4.1 Transaction Indistinguishability Implies Ledger Indistinguishability

Theorem 1. *If there exists an adversary $\mathcal{A}_{\mathsf{T\text{-}IND}}$ that can win the $\mathsf{T\text{-}IND}$ experiment with advantage $\mathsf{Adv}^{\mathsf{PBB}}_{\mathcal{A}_{\mathsf{T\text{-}IND}}}$ within runtime t, then there must be an adversary $\mathcal{B}_{\mathsf{L\text{-}IND}}$ that can win the $\mathsf{L\text{-}IND}$ experiment with advantage $\mathsf{Adv}^{\mathsf{PBB}}_{\mathcal{B}_{\mathsf{L\text{-}IND}}}$ within runtime essentially t such that*

$$\mathsf{Adv}^{\mathsf{PBB}}_{\mathcal{A}_{\mathsf{T\text{-}IND}}} \leq 2\mathsf{Adv}^{\mathsf{PBB}}_{\mathcal{B}_{\mathsf{L\text{-}IND}}}.$$

Proof. The proof is a game among a challenger $\mathcal{C}_{\mathsf{L\text{-}IND}}$, an adversary $\mathcal{B}_{\mathsf{L\text{-}IND}}$ and an adversary $\mathcal{A}_{\mathsf{T\text{-}IND}}$. $\mathcal{B}_{\mathsf{L\text{-}IND}}$ simulates a challenger $\mathcal{C}_{\mathsf{T\text{-}IND}}$.

Initialisation. As described in the $\mathsf{L\text{-}IND}$ experiment, $\mathcal{C}_{\mathsf{L\text{-}IND}}$ at the beginning samples a random bit b and initialises two **PBB** scheme oracles $\mathsf{O}^{\mathsf{PBB}}_0$ and $\mathsf{O}^{\mathsf{PBB}}_1$, it provides adversary $\mathcal{B}_{\mathsf{L\text{-}IND}}$ with two ledgers $(\mathsf{L}_{\mathsf{Left}}, \mathsf{L}_{\mathsf{Right}}) = (\mathsf{L}_b, \mathsf{L}_{1-b})$. $\mathcal{B}_{\mathsf{L\text{-}IND}}$ is expected to distinguish the order of the ledgers.

Query Phase. When $\mathcal{C}_{\mathsf{T\text{-}IND}}(\mathcal{B}_{\mathsf{L\text{-}IND}})$ receives queries from $\mathcal{A}_{\mathsf{T\text{-}IND}}$, $\mathcal{B}_{\mathsf{L\text{-}IND}}$ forwards the queries (Q, Q) to $\mathsf{O}^{\mathsf{PBB}}_0$ and $\mathsf{O}^{\mathsf{PBB}}_1$ respectively. When the ledger $\mathsf{L}_{\mathsf{Left}}$ gets updated by one round, denoted as $\mathsf{L}'_{\mathsf{Left}}$, $\mathcal{C}_{\mathsf{T\text{-}IND}}$ extracts the new transaction that is added in the last round, and forwards it back to $\mathcal{A}_{\mathsf{T\text{-}IND}}$.

Challenge Phase. When $\mathcal{C}_{\mathsf{T\text{-}IND}}(\mathcal{B}_{\mathsf{L\text{-}IND}})$ receives two public consistent challenge queries Q_0, Q_1 from $\mathcal{A}_{\mathsf{T\text{-}IND}}$, $\mathcal{C}_{\mathsf{T\text{-}IND}}$ forwards queries (Q_0, Q_1) to oracles $\mathsf{O}^{\mathsf{PBB}}_0$ and $\mathsf{O}^{\mathsf{PBB}}_1$ respectively. When the ledger $\mathsf{L}_{\mathsf{Left}}$ gets updated to $\mathsf{L}'_{\mathsf{Left}}$, $\mathcal{C}_{\mathsf{T\text{-}IND}}$ extracts the new transaction and returns it back to $\mathcal{A}_{\mathsf{T\text{-}IND}}$.

Response Phase. If $\mathcal{A}_{\mathsf{T\text{-}IND}}$ responds with 0, $\mathcal{B}_{\mathsf{L\text{-}IND}}$ sets the view to be $(\mathsf{L}_{\mathsf{Left}}, \mathsf{L}_{\mathsf{Right}}) = (\mathsf{L}_0, \mathsf{L}_1)$, otherwise sets the view as $(\mathsf{L}_{\mathsf{Left}}, \mathsf{L}_{\mathsf{Right}}) = (\mathsf{L}_1, \mathsf{L}_0)$. Return the answer to $\mathcal{C}_{\mathsf{L\text{-}IND}}$.

Analysis. The left ledger contains the transactions corresponding to L_b, so the transactions returned to $\mathcal{A}_{\mathsf{T\text{-}IND}}$ always correspond to Q_b. It follows that the challenger $\mathcal{C}_{\mathsf{L\text{-}IND}}$ and adversary $\mathcal{B}_{\mathsf{L\text{-}IND}}$ always simulate the challenger $\mathcal{C}_{\mathsf{T\text{-}IND}}$ with the random bit b. It follows that $\mathcal{B}_{\mathsf{L\text{-}IND}}$ guesses the order of the ledgers correctly exactly as often as $\mathcal{A}_{\mathsf{T\text{-}IND}}$ guesses its challenge bit correctly. The claim follows. □

4.2 Ledger Indistinguishability Implies Transaction Indistinguishability

In this section, we prove that ledger indistinguishability implies transaction indistinguishability. We first construct an adversary $\mathcal{A}_{\mathsf{T\text{-}IND}}$ that only sends one challenge query from an adversary $\mathcal{A}_{\mathsf{MultiT\text{-}IND}}$ that sends multiple (l_c) challenge queries. Then we prove that if there exists an adversary $\mathcal{B}_{\mathsf{L\text{-}IND}}$ that can break ledger indistinguishability, there must be an adversary $\mathcal{A}_{\mathsf{MultiT\text{-}IND}}$ can break transaction indistinguishability.

Lemma 1. *Let* $\mathsf{T\text{-}IND}_{l_c}$ *denote the transaction indistinguishability game in terms of* l_c *challenge queries, and* $\mathsf{T\text{-}IND}$ *denote the original transaction indistinguishability game in terms of one challenge query.*

If there exists an adversary $\mathcal{A}_{\mathsf{T\text{-}IND}_{l_c}}$ *that can win* $\mathsf{T\text{-}IND}_{l_c}$ *game with advantage* $\mathsf{Adv}^{\mathsf{PBB}}_{\mathcal{A}_{\mathsf{T\text{-}IND}_{l_c}}}$ *within run time* t*, there must be an adversary* $\mathcal{A}_{\mathsf{T\text{-}IND}}$ *that can win* $\mathsf{T\text{-}IND}$ *game with advantage* $\mathsf{Adv}^{\mathsf{PBB}}_{\mathcal{A}_{\mathsf{T\text{-}IND}}}$ *within run time essentially* t *such that*

$$\mathsf{Adv}^{\mathsf{PBB}}_{\mathcal{A}_{\mathsf{T\text{-}IND}_{l_c}}} \leq l_c \mathsf{Adv}^{\mathsf{PBB}}_{\mathcal{A}_{\mathsf{T\text{-}IND}}}.$$

This follows from a standard hybrid argument.

Theorem 2. *If there exists an adversary* $\mathcal{B}_{\mathsf{L\text{-}IND}}$ *that can win the* $\mathsf{L\text{-}IND}$ *game with advantage* $\mathsf{Adv}^{\mathsf{PBB}}_{\mathcal{B}_{\mathsf{L\text{-}IND}}}$ *within runtime* t*, then there exists an adversary* $\mathcal{A}_{\mathsf{T\text{-}IND}_{l_c}}$ *that can win the* $\mathsf{T\text{-}IND}_{l_c}$ *game in terms of* l_c *challenge queries, with advantage* $\mathsf{Adv}^{\mathsf{PBB}}_{\mathcal{A}_{\mathsf{T\text{-}IND}_{l_c}}}$ *and within runtime essentially* t *such that*

$$\mathsf{Adv}^{\mathsf{PBB}}_{\mathcal{B}_{\mathsf{L\text{-}IND}}} \leq 2\mathsf{Adv}^{\mathsf{PBB}}_{\mathcal{A}_{\mathsf{T\text{-}IND}_{l_c}}}.$$

Proof. We consider three hybrid games $\mathcal{H}_0$, $\mathcal{H}_1$ and $\mathcal{H}_2$ involving the adversary $\mathcal{B}_{\mathsf{L\text{-}IND}}$, and two oracles $\mathsf{O}_0^{\mathsf{PBB}}$ and $\mathsf{O}_1^{\mathsf{PBB}}$. The game constructs two ledgers L_0 and L_1 that the adversary is allowed to see. When the adversary makes a query (Q_0, Q_1), the game provides queries to the oracles and records their transactions on the appropriate ledger. The hybrid $\mathcal{H}_0$ gives the query Q_0 to $\mathsf{O}_0^{\mathsf{PBB}}$ and Q_1 to $\mathsf{O}_1^{\mathsf{PBB}}$. The hybrid $\mathcal{H}_1$ gives the query Q_0 to both oracles. The hybrid $\mathcal{H}_2$ gives the query Q_1 to $\mathsf{O}_0^{\mathsf{PBB}}$ and Q_0 to $\mathsf{O}_1^{\mathsf{PBB}}$. Denote by ϵ_i, $i \in \{0, 1, 2\}$, the probability that the adversary outputs 0 in hybrid $\mathcal{H}_i$.

We see that $\mathcal{H}_0$ behaves the same as the experiment $\mathsf{L\text{-}IND}$ with $b = 0$, while $\mathcal{H}_2$ behaves the same as the experiment $\mathsf{L\text{-}IND}$ with $b = 1$. Therefore,

$$\mathsf{Adv}^{\mathsf{PBB}}_{\mathcal{B}_{\mathsf{L\text{-}IND}}} = |\epsilon_0 - \epsilon_2| \leq |\epsilon_0 - \epsilon_1| + |\epsilon_1 - \epsilon_2|.$$

Now we consider two adversaries $\mathcal{A}_0$ and $\mathcal{A}_1$ against $\mathsf{T\text{-}IND}$. They run a copy of $\mathcal{B}_{\mathsf{L\text{-}IND}}$ and an oracle $\mathsf{O}^{\mathsf{PBB}}$. They construct two ledgers L_0 and L_1 that the adversary $\mathcal{B}_{\mathsf{L\text{-}IND}}$ is allowed to see. When $\mathcal{B}_{\mathsf{L\text{-}IND}}$ makes a query (Q_0, Q_1), the adversaries make a query to their own challenge oracle, submit the query Q_0 to $\mathsf{O}^{\mathsf{PBB}}$, and update the ledgers. The adversary $\mathcal{A}_0$ makes the challenge query (Q_0, Q_1) and appends the result to L_1, appending the output of $\mathsf{O}^{\mathsf{PBB}}$ to L_0. The adversary $\mathcal{A}_1$ makes the challenge query (Q_1, Q_0) and appends the result to L_0, appending the output of $\mathsf{O}^{\mathsf{PBB}}$ to L_1. When $\mathcal{B}_{\mathsf{L\text{-}IND}}$ outputs b', the adversaries $\mathcal{A}_0$ and $\mathcal{A}_1$ output b'. Note, the queries made by each adversary are public consistent.

Denote by $\delta_{i,\beta}$, $i \in \{0,1\}$, $\beta \in \{0,1\}$, the probability that the adversary $\mathcal{A}_{\mathsf{L\text{-}IND}}$ outputs 0 when $\mathcal{A}_i$ interacts with a $\mathsf{T\text{-}IND}$ experiment with $b = \beta$. Then the advantage of $\mathcal{A}_i$ is $|\delta_{i,0} - \delta_{i,1}|$.

We see that the adversary $\mathcal{A}_0$ interacting with a $\mathsf{T\text{-}IND}$ experiment with $b = 1$ behaves exactly as the hybrid $\mathcal{H}_0$, so $\delta_{0,1} = \epsilon_0$. The adversary $\mathcal{A}_0$ interacting with a $\mathsf{T\text{-}IND}$ experiment with $b = 0$ behaves exactly as the adversary $\mathcal{A}_1$ interacting with a $\mathsf{T\text{-}IND}$ experiment with $b = 1$, which again behaves exactly as $\mathcal{H}_1$, so $\delta_{0,0} = \delta_{1,1} = \epsilon_1$. And the adversary $\mathcal{A}_1$ interacting with a $\mathsf{T\text{-}IND}$ experiment with $b = 0$ behaves exactly as the hybrid $\mathcal{H}_2$, so $\delta_{1,0} = \epsilon_2$.

Finally, we consider the adversary $\mathcal{A}_{\mathsf{T\text{-}IND}_{l_c}}$. It samples $d \leftarrow_{\$} \{0,1\}$ and runs $\mathcal{A}_d$ and outputs its guess b'. Its advantage is the average advantage of $\mathcal{A}_0$ and $\mathcal{A}_1$. The claim follows. □

5 Security Analysis of Monero

Monero is a cryptocurrency based on CryptoNote protocol [17]. Monero uses ring signature to obfuscate a sender's address among other users' addresses. The value of a user's transaction is hidden by using the zero-knowledge protocol Bulletproof. The receiver's address is a one-time address that cannot be linked. By these techniques, Monero provides unlinkability among transactions to a certain extent, enabling the untraceable payment scheme. The idea of Monero is: An output set O (of size s) of public keys $(pk_1, pk_2, \ldots, pk_m)$ from the existing public keys of monero users $(U_1, U_2, \ldots, U_m)$ is chosen. A user U_k from the set O creates a ring signature σ on its transaction which can be verified by the set of public keys. The ring signature σ makes the signer U identity indistinguishable from the users of set O and hence provides the property of plausible deniability. We present Monero system (algorithms) according to our PBB system as follows.

- **Setup** The algorithm generates the public parameters pp.
- **KeyGen** The algorithm takes pp and generates the key pair for users.
- **Deposit/Mint** The algorithm is run by a miner to generate original coins (base coins) of value v. It takes pp, the ledger state $\mathsf{St_{bc}}$ (similar to $\mathsf{St_b}$ in PBB), and the key pairs of the miner u, outputs a new unspent transaction and update miner's temporary state.

$$(\mathsf{trans}, \mathsf{TempSt_u}) \leftarrow \mathsf{Deposit}(\mathsf{pk_u}, \mathsf{sk_u}, \mathsf{St_{bc}}, \mathsf{St_u}, \mathsf{pp}, \mathsf{v})$$

– **Pay** The protocol is run by a user (payer) p to send transactions to other users. It takes an anonymity set (a ring): a set of other irrelevant users' public keys, the size of the set n, the public key of the receiver r, the key pair of the payer, the state of the payer, an unspent transaction with amount v' and the ledger state $\mathsf{St_{bc}}$ as inputs, it outputs a transaction with its one time output address, and payer's temporary state.

$$(\mathsf{trans}, \mathsf{TempSt}'_{\mathsf{p}}) \leftarrow \mathsf{Pay}(\mathsf{pk_p}, \mathsf{sk_p}, \mathsf{pk_r}, \{\mathsf{pk_i}\}_{\mathsf{i}=0}^{\mathsf{n}-1}, \mathsf{St_p}, \mathsf{St_{bc}}, \mathsf{pp}, \mathsf{v}')$$

– **Contract** The algorithm takes the public parameters, a trans, the public state and all users' states as inputs and then updates all of them.

$$(\mathsf{St}'_{\mathsf{bc}}, \{\mathsf{St}'_{\mathsf{u}}\}) \leftarrow \mathsf{Contract}(\mathsf{St_{bc}}, \{\mathsf{St_u}\}, \{\mathsf{TempSt_u}\}, \mathsf{trans}, \mathsf{pp})$$

Security Proof Sketch for Monero. Following we first describe the T-IND experiment customised to the algorithms of Monero, and further we define its leakage function to capture the privacy of Monero to give the security proofs.

Experiment. T-IND Given a (candidate) PBB scheme Π, an adversary $\mathcal{A}$, and security parameter λ, the (probabilistic) experiment $\mathsf{T\text{-}IND}(\Pi, \mathcal{A}, \lambda)$ proceeds as follows. In the query phase, the adversary is capable of sending the queries for the Monero system to the experiment. In the challenge phase, the experiment randomly chooses $b \leftarrow \{0, 1\}$, adversary then sends a challenge query as $\mathbf{Q} = \mathbf{Challenge}(\mathbf{Q_0}, \mathbf{Q_1})$ where $\mathbf{Q_0}, \mathbf{Q_1}$ leak the some information and the experiment only performs Q_b. The adversary gets the output trans_b. At the end, the adversary outputs $b' \in \{0, 1\}$. The adversary wins the game if $b' = b$. During the challenge, the queries sent by the adversary need to be *Public Consistent*.

Leakage Function. The leakage function of Monero takes the total number N of available public keys that can be added into a ring signature , all these public keys $\{pk_i\}_{i=1}^{N}$, the size of the ring signature n, and a $\mathbf{Q} = (\mathsf{Pay}, \mathsf{pk_p}, \mathsf{pk_r}, \mathsf{v})$ query as inputs, and outputs a set of public keys that is included in the ring signature.

$$\{\mathsf{pk}\}_{j=1}^{n} \leftarrow \mathsf{Leakage}(\mathbf{Q}, \mathbf{N}, \mathbf{n}, \{\mathsf{pk_i}\}_{\mathbf{i}=\mathbf{1}}^{\mathbf{N}})$$

We adopt a game-hopping approach.
Experiment $\mathbf{Exp}_0$. The same as the T-IND experiment.
Experiment $\mathbf{Exp}_1$. The difference between $\mathbf{Exp}_1$ and $\mathbf{Exp}_0$ is that $\mathbf{Exp}_1$ does not use the sender's keys to generate the ring signature, instead, it uses a freshly generated public key in the ring to generate the ring signature on the output transaction. Due to a ring signature, an adversary is not able to distinguish which public key is the real signer, $\mathbf{Exp}_1$ is indistinguishable with $\mathbf{Exp}_0$.
Experiment $\mathbf{Exp}_2$. The experiment $\mathbf{Exp}_2$ modifies $\mathbf{Exp}_1$ by replacing the zero-knowledge proofs for balances by the simulated zero-knowledge proof using the Bulletproof SHVZK simulator. Because of the zero-knowledge property of zero-knowledge proof, the simulated proof reveals no information about the statement, i.e., the balances, thus $\mathbf{Exp}_2$ is indistinguishable with $\mathbf{Exp}_1$. Furthermore, the soundness property of zero-knowledge proof preserves the overdraft.

Experiment $\mathbf{Exp}_3$. The $\mathbf{Exp}_3$ modifies $\mathbf{Exp}_2$ by replacing the one time address of the output with a randomly generated address, which is irrelevant to the receiver's public key. Therefore $\mathbf{Exp}_3$ is indistinguishable with $\mathbf{Exp}_2$.

Balance + Overdraft Safety : Given the experiment $\mathsf{Balance}$ involving polynomial-time adversary $\mathcal{A}$ and challenger $\mathcal{C}$, the Monero scheme Π is $\mathsf{Balance}$-secure, if

$$\Pr[\mathsf{Balance}(\Pi, \mathcal{A}) = 1] \leq \mathrm{negl}(\lambda)$$

We present a proof sketch for the above claim. Monero blockchain involves certain types of transaction assertions and for which a number of transaction-related knowledge proofs are implemented. In Monero, each user U has two pairs of private/public keys. The first key pair is the view key pair (k^v, K^v) that allows a user to view the transactions directed to him/her. The other pair is spend key pair (k^s, K^s) that allows the user to spend his/her coins. For each transaction tx, a sender first chooses a random number r and generates a One Time Address (OTA) K^o using r and the receiver's public keys (view and spend keys). The sender sends the K^o along with transaction public key PK_{tx} to the network. The transaction public key $PK_{tx} = rG$, where G is the base point in the elliptic curve. Further, the OTA is recorded in the blockchain.

We say the Monero ledger is balanced if the following conditions hold. In other words at anytime, it is possible to check any user's transaction history by verifying its incoming and outgoing transactions along with its unspent outputs.

- For each valid Pay transaction tx directed to a user U (receiver), U can always verify tx by creating a new OTA and by matching it with the OTA created by the sender. To prove the ownership of incoming transaction tx, the receiver U proves the knowledge of k^v in K^v and with the combination of transaction public key rG, proves the knowledge of $k^v * rG = rK^v$. A verifier having rK^v can check that the OTA is owned by the receiver's address.
- For each valid Pay transaction tx made by a user U (sender), U can verify tx by creating a valid key image for the owned output conditioning the key image has not appeared in the blockchain before. To prove this to a verifier, the sender U provides rK^v to the verifier and proves that it corresponds with the transaction public key rG and the receiver's address.
- For each valid address owned by a user U, the user can always verify to others that the provided address contains a minimum balance. To do that, the user creates key images for the unspent outputs conditioning the key images that have not appeared in the blockchain.

6 Conclusion

Privacy-preserving cryptocurrency systems should ensure the security properties of their systems. Despite having many such systems, there is no unified model to ensure or formalise the security of these systems. This paper attempts to create a general model referred to as a privacy-preserving blockchain-based bank that

can be used to prove the security of privacy-preserving cryptocurrency systems. We presented the privacy properties of these systems and also the properties related to the integrity of these systems. Further, to show the usefulness of our model, we analysed the security of Monero system using our model. More details such as security analysis of Zcash, discussion of different security properties and their relationships with each other can be found in the full version of the paper.

References

1. Almashaqbeh, G., Solomon, R.: SoK: privacy-preserving computing in the blockchain era. Cryptology ePrint Archive, Report 2021/727 (2021). https://ia.cr/2021/727
2. Bowe, S., Chiesa, A., Green, M., Miers, I., Mishra, P., Wu, H.: ZEXE: enabling decentralized private computation. In: 2020 IEEE Symposium on Security and Privacy (SP), pp. 947–964 (2020). https://doi.org/10.1109/SP40000.2020.00050
3. Bünz, B., Agrawal, S., Zamani, M., Boneh, D.: Zether: towards privacy in a smart contract world. In: Bonneau, J., Heninger, N. (eds.) FC 2020. LNCS, vol. 12059, pp. 423–443. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-51280-4_23
4. Cremers, C., Loss, J., Wagner, B.: A holistic security analysis of monero transactions. Cryptology ePrint Archive, Paper 2023/321 (2023). https://eprint.iacr.org/2023/321
5. Duffield, E., Diaz, D.: Dash: a privacy centric cryptocurrency (2015)
6. Fauzi, P., Meiklejohn, S., Mercer, R., Orlandi, C.: Quisquis: a new design for anonymous cryptocurrencies. In: Galbraith, S.D., Moriai, S. (eds.) ASIACRYPT 2019. LNCS, vol. 11921, pp. 649–678. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-34578-5_23
7. Gjøsteen, K., Raikwar, M., Wu, S.: PriBank: confidential blockchain scaling using short commit-and-proof NIZK argument. In: Galbraith, S.D. (ed.) CT-RSA 2022. LNCS, vol. 13161, pp. 589–619. Springer, Cham (2022). https://doi.org/10.1007/978-3-030-95312-6_24
8. Jivanyan, A.: Lelantus: Towards confidentiality and anonymity of blockchain transactions from standard assumptions. IACR Cryptol. ePrint Arch. **2019**, 373 (2019)
9. Jivanyan, A., Feickert, A.: Lelantus spark: secure and flexible private transactions. Cryptology ePrint Archive (2021)
10. Kerber, T., Kiayias, A., Kohlweiss, M.: Kachina-foundations of private smart contracts. In: 2021 IEEE 34th Computer Security Foundations Symposium (CSF), pp. 1–16. IEEE (2021)
11. Kosba, A., Miller, A., Shi, E., Wen, Z., Papamanthou, C.: Hawk: the blockchain model of cryptography and privacy-preserving smart contracts. In: 2016 IEEE Symposium on Security and Privacy (SP), pp. 839–858 (2016). https://doi.org/10.1109/SP.2016.55
12. Kosba, A., Miller, A., Shi, E., Wen, Z., Papamanthou, C.: Hawk: The blockchain model of cryptography and privacy-preserving smart contracts. In: 2016 IEEE Symposium on Security and Privacy (SP), pp. 839–858. IEEE (2016)
13. Lai, R.W., Ronge, V., Ruffing, T., Schröder, D., Thyagarajan, S.A.K., Wang, J.: Omniring: scaling private payments without trusted setup. In: Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, pp. 31–48 (2019)

14. Liang, M., Karantaidou, I., Baldimtsi, F., Gordon, S.D., Varia, M.: (ϵ, δ)-indistinguishable mixing for cryptocurrencies. Proc. Priv. Enhanc. Technol. **2022**(1), 49–74 (2021)
15. Mitani, T., Otsuka, A.: Confidential and auditable payments. In: Bernhard, M., et al. (eds.) FC 2020. LNCS, vol. 12063, pp. 466–480. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-54455-3_33
16. Nakamoto, S.: Bitcoin: a peer-to-peer electronic cash system (2009). http://bitcoin.org/bitcoin.pdf
17. Saberhagen, N.V.: CryptoNote v 2.0 (2013). https://bytecoin.org/old/whitepaper.pdf
18. Sasson, E.B., et al.: Zerocash: decentralized anonymous payments from bitcoin. In: 2014 IEEE Symposium on Security and Privacy, pp. 459–474. IEEE (2014)
19. Steffen, S., Bichsel, B., Gersbach, M., Melchior, N., Tsankov, P., Vechev, M.: zkay: Specifying and enforcing data privacy in smart contracts. In: Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, pp. 1759–1776 (2019)
20. The Monero Project: Monero (2014). https://web.getmonero.org
21. Wijaya, D.A., Liu, J., Steinfeld, R., Liu, D., Yuen, T.H.: Anonymity reduction attacks to monero. In: Guo, F., Huang, X., Yung, M. (eds.) Inscrypt 2018. LNCS, vol. 11449, pp. 86–100. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-14234-6_5
22. Wood, G.: Ethereum: a secure decentralised generalised transaction ledger. Yellow Paper (2014)
23. Zhang, Y., Long, Y., Liu, Z., Liu, Z., Gu, D.: Z-channel: scalable and efficient scheme in zerocash. Comput. Secur. **86**, 112–131 (2019)