

SENDER IDENTITY VERIFICATION USING ZERO-KNOWLEDGE PROOFS

BACKGROUND

[0001] In some cases, the sending party may wish to deliver information to the receiving party without exposing private underlying data used to generate the information. For example, the sending party may wish to report an average temperature measurement but may not wish to expose the individual sensor measurements from each temperature sensor. In this manner, the sending party may wish to use a mechanism that proves that the reported average temperature measurement sent to the receiving party used all of the sensor measurements and also applied the proper formula for calculating the average. One way of ensuring privacy while verifying accuracy is using a zero-knowledge proof (“ZKP”).

[0002] In other words, a party may compute a function on data that it wants to keep confidential and share the result of the function with another party. The ZKP allows a party to verify that the function was properly applied to the data, without having to share the data itself. This allows the other party to verify the integrity of the function and the accuracy of the result without knowing the underlying data. One example of a ZKP is the zero-knowledge succinct non-interactive argument of knowledge (“zk-SNARK”).

[0003] The zk-SNARK scheme may be useful because the succinct proofs may use relatively little memory space for a blockchain of proofs. Further, the relatively fast verification may allow multiple users that are observing the blockchain to quickly verify a large load of transactions.

[0004] The zk-SNARK cryptography may be described in the following three algorithms:

[0005] (1) $(pk, vk) = \text{SETUP}(CS)$

[0006] (2) $(\pi) = \text{PROVE}(pk, \text{public_data}, \text{private_data})$

[0007] (3) $(\text{verified}) = \text{VERIFY}(vk, \pi, \text{public_data})$

[0008] In these algorithms, “CS” represents a constraint system defining a non-probabilistic or nondeterministic polynomial statement. The constraint system may include the function sought to be proved. Public_data may be the result of the function

and `private_data` may be the underlying data that should remain confidential. In the example above, `public_data` may be the average temperature measurement, and the `private_data` may be the individual temperature sensor values.

[0009] The SETUP algorithm generates a proving key (“pk”) and a verification key (“vk”) based on the constraint system. The SETUP algorithm may require random numbers that must be kept secret to ensure that the proof is properly and securely generated.

[0010] The PROVE algorithm generates a proof from the proving data, the `public_data`, and the private data. The symbol π may represent the proof itself generated using the proving key as well as public data and private data. In some implementations, the proof π may be only 300 bytes long. zk-SNARK may have the benefit of proving a succinct proof, regardless of the complexity of the underlying function.

[0011] The VERIFY algorithm uses the vk and the proof, π , to verify the correctness of the public data. The VERIFY algorithm may return a binary one or zero value indicating whether or not the public data is correct.

[0012] Digital ledger technologies (DLTs) provide data storage mechanisms that operate at scale without central administration. This decentralization offers a variety of benefits over traditional storage mechanisms, and DLTs are being integrated into applications spanning varied sectors, industries, and technologies. For example, an asset transfer system may leverage a DLT to document asset transfers. Such assets may include cryptocurrencies, but also physical assets, such as goods, products, and art, security assets, bonds, insurance policies, titles to assets, actual currencies, and other suitable assets.

[0013] Typically, asset transfer systems fall into one of two categories: public and private. In a public asset transfer system, all transactions are viewable, knowable, and ascertainable to the other participants in the asset transfer system. An example of a public asset transfer system are the BITCOIN and ETHEREUM protocols. An advantage of a public asset transfer system is transparency, but that comes at the expense of privacy.

[0014] In a private asset transfer system, transactional details are encrypted, hidden, and unknowable to other participants in the asset transfer system. Examples of a private asset

transfer systems include the ZCASH, ZETHER, ZERO COIN, and AZTEC protocols, as well as confidential transactions on MONERO, etc.

[0015] There are instances where a receiver of assets must know where those assets came from. For example, know your customer (KYC) is the process of a business verifying the identity of its clients and assessing their suitability and any potential legal compliance risk of the client relationship. Bank regulations, anti-money laundering, and anti-bribery regulations exist that forbid taking money or assets from certain entities. Thus, there may be a need for banks, insurers, export creditors and other financial institutions to conduct due diligence on their customers.

[0016] Improved systems and methods are needed to provide KYC information while maintaining privacy.

BRIEF SUMMARY

[0017] Disclosed herein are system, apparatus, device, method and/or computer program product embodiments, and/or combinations and sub-combinations thereof, for providing the identity of a sender to a receiver of a transaction in a private asset transfer system using zero-knowledge proofs and cryptographic primitives without publically exposing the sender identity and other transactional details.

[0018] A computer-implemented method for privately verifying a sender may include receiving a transaction that identifies a transfer of an asset from the sender to a receiver, wherein the transaction is signed by the sender with a one-time key that does not link to a prior transaction of the sender. A public key identifying the sender may then be privately sent from the sender to the receiver. A zero-knowledge proof may be generated that when verified proves that the one-time key corresponds to the public key identifying the sender. The transaction and the zero-knowledge proof may be written to a public ledger without broadcasting to the public ledger the identity of the sender while allowing the receiver to verify the sender's identity by verifying the zero-knowledge proof.

[0019] In some embodiments, a system for privately verifying a sender may comprise a memory and at least one processor coupled to the memory. The processor may be configured to receive a transaction that identifies a transfer of an asset from the sender to a receiver, wherein the transaction is signed by the sender with a one-time key that does

not link to a prior transaction of the sender. The processor may be configured to privately send a public key identifying the sender to the receiver. The processor may be configured to generate a zero-knowledge proof that when verified proves that the one-time key corresponds to the public key identifying the sender. The processor may be configured to write the transaction and the zero-knowledge proof to a public ledger without broadcasting to the public ledger the identity of the sender in a manner that allows the receiver to verify the sender's identity by verifying the zero-knowledge proof.

[0020] In some embodiment, a non-transitory computer-readable device is disclosed, the non-transitory computer-readable device may have instructions stored thereon that, when executed by at least one computing device, may cause the at least one computing device to perform the computer-implemented method described above.

BRIEF DESCRIPTION OF THE DRAWINGS

[0021] The accompanying drawings are incorporated herein and form a part of the specification.

[0022] FIG. 1 depicts a block diagram of a transaction environment including an asset transfer system, according to some embodiments.

[0023] FIG. 2 depicts a block diagram of public state stored in a public ledger and reconstructed on member nodes, according to some embodiments.

[0024] FIG. 3 depicts a block diagram of private state stored on a node in an asset transfer system, according to some embodiments.

[0025] FIG. 4 depicts a flowchart illustrating a method for generating and validating a transaction in an asset transfer system, according to some embodiments.

[0026] FIG. 5 depicts a flowchart illustrating a method for transferring an asset from a sender to a receiver in an asset transfer system, according to some embodiments.

[0027] FIG. 6 depicts an example computer system useful for implementing various embodiments.

[0028] In the drawings, like reference numbers generally indicate identical or similar elements. Additionally, generally, the left-most digit(s) of a reference number identifies the drawing in which the reference number first appears.

DETAILED DESCRIPTION

[0029] Provided herein are system, apparatus, device, method and/or computer program product embodiments, and/or combinations and sub-combinations thereof, for providing the identity of a sender to a receiver of a transaction in a private asset transfer system using zero-knowledge proofs (“ZKPs”) and cryptographic primitives without publically exposing the sender identity and other details about the transaction.

[0030] An asset transfer system may facilitate asset issuance and transfer. A sender may use an asset transfer system to transfer an asset to a receiver. Such assets may be physical assets, e.g., goods, products, art, etc., security assets, e.g., bonds, insurance policies, titles to assets, currency, or cryptocurrency, and any other suitable assets. An asset transfer system may leverage a DLT, e.g., a blockchain, directed acyclic graph, database, or other suitable public ledger, to immutably store transactions over time. The use of a DLT by an asset transfer system may obviate the need for centralized control and management of the public ledger. Such decentralization may be especially useful where parties transact at arm’s length in the absence of a trusted intermediary.

[0031] In a public asset transfer system, transactions between participants are visible to other participants in and observers of the asset transfer system. Thus, the general public may ascertain the sender, receiver, amount or quantity, asset type, etc. involved in a transaction by examining the public ledger used in a public asset transfer system. However, this transparency may not be agreeable to organizations for a variety of reasons, e.g., a business may be hesitant to place all of the business’s financial transactions on a public ledger where they may be accessed by competitors.

[0032] Thus, private asset transfer systems exist that provide total confidentiality of transactions. In a private asset transfer system, the privacy of transactions may be preserved by rendering transactional details classified, private, and unknowable by default using cryptographic techniques. In such a system, the underlying ledger does not reveal any information about a transaction, such as the sender, receiver, asset type, or transfer amounts. Moreover, one transaction does not link to any other transactions from which additional information might be ascertained by participants and the general public.

[0033] However, transactional integrity remains essential in a private asset transfer system. Parties must trust that transaction parameters are accurate before executing a transaction that transfers a potentially valuable asset. Participants do not want to complete a transaction with another participant if the recipient can renege on promises, refuse to exchange assets, or engage in other bad conduct. Accordingly, provenance and verifiability remain a paramount concern within a private asset transfer system. Integrity guarantees may cover the issuance of assets, asset balances, avoidance of double spends, non-malleability of past transactions, ownership authentication, and a host of other concepts related to transactional integrity.

[0034] To achieve the twin aims of anonymity and transactional integrity, an asset transfer system may employ ZKPs coupled with cryptographic primitives. A ZKP provides a mechanism by which participants may provide trustable information while maintaining confidentiality using encrypted proof results. Generally speaking, a ZKP allows a participant node to compute a function on confidential transactional data and share only the result of the function with other members of the asset transfer system. The other members may then verify the integrity of the function and the accuracy of the result without ever being privy to or accessing underlying, confidential data. ZKPs may be decomposed into several smaller proofs and created/proved using parallel processing techniques to enhance performance and improve scalability of the system.

[0035] In legacy private asset transfer systems, all transactional details are confidential, and thus, even the receiver of assets does not know and cannot know the sender's identity. Because no link exists identifying a sender from one transaction to another, the receiver may not even be aware that the same sender created two transactions. But this inability to know the sender of assets generates real-world consequences and problems for the receivers of assets across various industries and use cases.

[0036] For example, many regulations, e.g., Anti-Money Laundering (AML), require banks and other organizations to know, track, and prove the identity of a sender that transferred assets to the organization. In AML scenarios, know-your-customer (KYC) guidelines require organizations to assess the suitability of clients and analyze the risk of illegal intentions and behaviors. Thus, organizations may need to know the identity of a sender to comply with regulations. However, even in such a scenario, transactional

details should remain hidden to other industry participants and observers, e.g., other organizations lacking privity to the asset transfer.

[0037] Outside of the regulatory context, an organization may need to know the identity of a sender for a variety of business and/or practical reasons, e.g., to accurately track and manage inventory, maintain accurate budgets, etc. For example, a manufacturer may place orders for widgets (i.e., the asset) from multiple suppliers. When the manufacturer receives a particular shipment of widgets, the manufacturer may need to know the identity of the sender to render payment to the appropriate supplier, update tracking systems, and perform other suitable managerial tasks. A multitude of other real-world scenarios exist in which the receiver of an asset may need to verify and prove the identity of a sender while preserving transactional anonymity among other participants and the general public.

[0038] Accordingly, embodiments use zero-knowledge proofs and cryptographic primitives to provide the identity of a sender to a receiver of a transaction in a private asset transfer system without publically exposing the sender identity and other transactional details. In this way, embodiments improve the functioning of a computer system by preserving sender anonymity among all members of a private asset transfer system except for the receiver. The encapsulating of a verifiable sender within a transaction in a private asset transfer system provides a further technical improvement by eliminating the need for a central arbiter or authority that stores information about the senders of transactions in a private asset transfer system, thereby saving the memory needed to maintain such ledgers. This provides a further technical improvement of saving processing power and CPU resources needed to provide such information to network participants in response to requests and similarly eliminates network traffic. The embodiments provide another technical improvement of enhanced security of transactions by removing data storage and maintenance responsibilities from third parties, limiting exposure to hackers and other bad actors. Moreover, the zk-SNARK proofs that are employed are non-interactive, thus providing a further technical improvement by eliminating the need for interactions among participants, thereby optimizing the transacting processes.

- [0039]** Various embodiments of these features will now be discussed with respect to the corresponding figures.
- [0040]** FIG. 1 depicts a block diagram of transaction environment 100 including an asset transfer system, according to some embodiments. Any operation herein may be performed by any type of structure in the diagram, such as a module or dedicated device, in hardware, software, or any combination thereof. Any block in the block diagram of FIG. 1 may be regarded as a module, apparatus, dedicated device, general-purpose processor, engine, state machine, application, functional element, or related technology capable of and configured to perform its corresponding operation(s) described herein. Transaction environment 100 may include asset transfer system 110, database 120, message queue 130, proof workers 140, ledger interface 150, public ledger 160, and nodes 170 exemplified by node 170A and node 170B.
- [0041]** Asset transfer system 110 may facilitate the issuance, ownership, and transfer of assets. Such assets may be physical assets, e.g., goods, products, art, etc., security assets, e.g., bonds, insurance policies, titles to assets, currency, or cryptocurrency, and any other suitable assets. Asset transfer system 110 may track asset type identifiers, asset amounts, asset value, and other suitable qualities, aspects, and characteristics of assets.
- [0042]** Asset transfer system 110 may include one or more processors, memory, servers, routers, modems, and/or antennae configured to interface with other components in FIG. 1. Asset transfer system 110 may share hardware and/or software components with database 120, message queue 130, and/or ledger interface 150. In another embodiment, asset transfer system 110 may communicate with database 120, message queue 130, and/or ledger interface 150 via a local area network (LAN), a wireless network, and/or via the Internet. Asset transfer system 110 may expose an HTTP protocol server to receive communications from nodes 170, message queue 130, and/or ledger interface 150 and provide appropriate communications in return. In another embodiment, asset transfer system 110 may provide lightweight functions and API calls to nodes 170 that facilitate communication between nodes 170 and public ledger 160 and/or message queue 130.
- [0043]** Asset transfer system 110 may be a private asset transfer system that maintains and preserves confidentiality of transactional details among participants and the general public (as contrasted with a public asset transfer system where transactional details, e.g.,

amount, sender, receiver, etc., are ascertainable to anyone with access to the public ledger). Thus, asset transfer system 110 may generally maintain the privacy of transactions by rendering transactional details classified, private, and unknowable by default. In such a private asset transfer system, transactional integrity and trust remains fundamental.

[0044] Asset transfer system 110 may achieve these fundamental requirements of transactional integrity and verifiability while maintaining privacy by using an appropriate commitment scheme. Under such a scheme, instead of publishing the asset owner and amount of ownership in clear text, asset transfer system 110 may write an appropriate transaction to digital ledger 160 in encrypted and encapsulated form. Towards this end, asset transfer system 110 may employ an appropriate ZKP or series of ZKPs. Through a ZKP, other participants/nodes in asset transfer system 110 may independently verify the authenticity of the transaction without accessing the actual information and private data used to generate the ZKP by the creator of the transaction.

[0045] The disclosure below will refer to a note, i.e., a representation of the ownership of an asset within asset transfer system 110. A note may be associated with a particular address. Such an address may be derived from a diversifier and a private key associated with the node (a process described in further detail below). Thus, only the node owning the address and having knowledge of their private key may later transfer/spend an acquired note. A note may include other relevant fields, for example: an indication of the asset identifier, an amount, an asset type, a description field, etc. A note may be further encrypted so as not to reveal fields to participants and the general public when the transaction is written to the public ledger. In an embodiment, to create a transfer transaction, a sender may generate a spend note, an output note, a conversion note, and other suitable notes, as described in further detail below.

[0046] Asset transfer system 110 may allow the receiver of a transaction to verify the identity of the sender. To allow the receiver, and only the receiver, to verify the sender's identity, asset transfer system 110 may privately send the sender's public key to the receiver. To privately send the sender's public key, asset transfer system 110 may generate a shared key between the sender and the receiver using an address associated with the receiver. Asset transfer system 110 may then encrypt a public key associated

with the sender's using the shared key. The encrypted public key may then be conveyed or sent to the receiver. In an embodiment, the encrypted public key may be included in a generated transaction or note. The receiver may then use the received public key of the sender to ascertain the sender's identity and verify the identity of the sender by comparing the sender's identity to the signer of the transaction using a ZKP as set out below. Because the public key is encrypted, other participants/nodes in asset transfer system 110 will not be able to ascertain transactional details or the sender's identity without the shared key. These sender identity verification processes are described in further detail below. In this way, a symmetric key encryption process may be used to privately transmit the sender's public key.

[0047] In another embodiment, an asymmetric key encryption process may be used to privately transmit the sender's public key. In that embodiment, the sender's public key is encrypted using a public key of the receiver. Then, the encrypted message can only be decrypted using a private key of the receiver corresponding to and used to generate the receiver's public key. In this way, an asymmetric key encryption process may be used to privately transmit the sender's public key.

[0048] Database 120 may store a variety of information and data used by asset transfer system 110. Database 120 may be a relational database, a NoSQL database or other horizontally scaling database, a file repository, a digital ledger technology or blockchain, or any other suitable storage mechanism. Database 120 may employ a centralized storage area network (SAN), network-attached storage (NAS), redundant array of independent disks, and/or any other configuration of storage devices to supply sufficient storage capacity to store database tables and supporting structures used by asset transfer system 110.

[0049] Message queue 130 may be leveraged by asset transfer system 110 to interface with proof generators 140 that generate ZKPs and perform related proof tasks. In this fashion, message queue 130 may distribute proof tasks to proof workers 140, receive responses to the proof tasks from proof workers 140, and return appropriate responses back to asset transfer system 110 and/or nodes 170. In one embodiment, message queue 130 may be an open-source message broker software or middleware implementing an appropriate message-queueing protocol.

- [0050]** Proof workers 140 may be stateless workers that generate ZKPs, prove ZKPs, and perform other proofing operations related to ZKPs. Because generating and proving ZKPs may be computationally demanding, proof workers 140 may be pooled and take advantage of various parallel-processing techniques to improve the overall scalability of asset transfer system 110. Proof workers 140 may receive a proof task from message queue 130, complete the task, and return a result to asset transfer system 110 and/or nodes 170 via message queue 130. In an embodiment, proof generators 140 may be specialized processors and/or CPU cores devoted entirely to generating and proving ZKPs.
- [0051]** Ledger interface 150 may provide an abstraction layer by which asset transfer system 110 and/or nodes 170 may communicate with public ledger 160. Ledger interface 150 may broadcast to (write) and verify transactions (read) from public ledger 160. By abstracting ledger interface 150 in this fashion, multiple underlying technologies/solutions may be deployed as public ledger 160. In other words, ledger interface 150 allows asset transfer system 110 to remain storage-mechanism/ledger agnostic, and accordingly, asset transfer system 110 may function on blockchain, databases, and other suitable DLTs or combinations thereof. In an embodiment, ledger interface 150 may expose an HTTP server or servers and authenticate transactions and requests using a suitable methodology or set of provided functions/API calls.
- [0052]** Public ledger 160 may store immutable transaction data related to issuance and ownership of assets within asset transfer system 110. Public ledger 160 may be a blockchain, directed acyclic graph, database, other suitable DLT, or some combination thereof. Public ledger 160 may immutably record transactions that occur within asset transfer system 110 in a manner that is not subject to tampering or manipulation. Public ledger 160 may employ any appropriate data structure that may be used to store transactions. In an embodiment, public ledger 160 may employ an append-only Merkle tree or other suitable hash structure to store transactional details. Public ledger 160 may house a public state, described in further detail below in FIG. 2 as public state 200, or, in an alternative embodiment, public ledger 160 may store information from which such a public state may be recreated.

[0053] Nodes 170, such as node 170A and node 170B, may be participants in an asset transfer system, such as asset transfer system 110. In an embodiment, nodes 170 may adhere to an appropriate asset transfer protocol that allows nodes 170 to interface with asset transfer 110, message queue 130, and/or ledger interface 150. Nodes 170 may operate as both senders and receivers of assets. Nodes 170 may thus interface with asset transfer system 110 to write transactions to public ledger 160 and to read transactions from public ledger 160 to determine when an asset is received. Moreover, nodes 170 may operate as verifiers of transactions conducted by other nodes 170 in the system. Thus, as transactions are written to public ledger 160, nodes 170 may verify the legitimacy of the transactions by verifying the ZKPs contained therein, thus establishing a consensus view of asset ownership among participants of the private asset transfer system. Nodes 170 may maintain a public state that may be recreated from information in public ledger 160, and a private state that includes confidential information about assets owned by the particular node and cryptographic information known only to the particular node. Such a private state is described as private state 300 in further detail below with reference to FIG. 3.

[0054] FIG. 2 depicts a block diagram of public state 200 stored in a public ledger such as public ledger 160 and reconstructed on participant nodes, according to some embodiments. Any operation herein may be performed by any type of structure in the diagram, such as a module or dedicated device, in hardware, software, or any combination thereof. Any block in the block diagram of FIG. 2 may be regarded as a module, apparatus, dedicated device, general-purpose processor, engine, state machine, application, functional element, or related technology capable of and configured to perform its corresponding operation(s) described herein.

[0055] Generally speaking, public state 200 may store issuance rules, completed transactions, parameters for ZKPs, and other information relevant to the administration of asset transfer system 110. In an embodiment, public state 200 may be stored on public ledger 160. In another embodiment, public state 200 may be stored on each node among nodes 170 and generated from information and data stored on public ledger 160. In such either embodiment, public state 200 may be recreated from public ledger 160. Thus, when a new node among nodes 170 joins the asset transfer system, the new node may

create a local version of public state 200 from the contents of public ledger 160. Public state 200 may include nullifier set 202, commitment tree 204, and rules 206.

[0056] Nullifier set 202 may store the nullifiers for all notes ever spent in asset transfer system 110. In an embodiment, when a sender creates a transfer transaction, the sender may generate a nullification or spend of a note signifying to the system that the previously owned note is being spent. Asset transfer system 110 and/or node 170 may create a unique nullifier identify based on the spender's private key and/or other cryptographic information known to the sender. Such a unique nullifier protects against double spending, i.e., a sender attempting to send the same asset twice to two different receivers. The unique nullifier may be included in nullifier set 202 so that all spent notes may be verified by the system during later transactions. Nullifier set 202 may employ any appropriate data structure that may be used to store the nullifiers. In an embodiment, nullifier set 202 employs an append-only Merkle tree or other suitable hash structure.

[0057] Commitment tree 204 may store all finalized notes created in asset transfer system 110. In an embodiment, when a sender creates a transfer transaction, the sender may generate an output note or a note commitment, containing information about the receiver, the amount to transfer, etc. Commitment tree 204 may include with transactions a reference to a recent state of commitment tree 204. Commitment tree 204 may employ any appropriate data structure that may be used to store the nullifiers. In an embodiment, commitment tree 204 employs an append-only Merkle tree or other suitable hash structure.

[0058] Rules 206 may store the active issuance rules and administration rules. For example, rules 206 may specify permissions to issue a type of asset, e.g., the types and/or quantities of assets or asset tokens that the issuance system may issue. Rules 206 may specify protocols by which verifiers may confirm the issuances of assets in conformance with the rules. Rules 206 may provide for the governance of asset transfer system 110 and specify how transactions are to be created and signed. Rules 206 may be derived from public ledger 160. For example, rules 206 may be derived from a specific protocol associated with public ledger 160 or stored in public ledger 160 (e.g., in a header block of a blockchain).

[0059] FIG. 3 depicts a block diagram of private state 300 stored on a node in an asset transfer system. Any operation herein may be performed by any type of structure in the diagram, such as a module or dedicated device, in hardware, software, or any combination thereof. Any block in the block diagram of FIG. 3 may be regarded as a module, apparatus, dedicated device, general-purpose processor, engine, state machine, application, functional element, or related technology capable of and configured to perform its corresponding operation(s) described herein.

[0060] Private state 300 may include confidential information about assets owned by the particular node as well as cryptographic information known only to the particular node. One skilled in the relevant arts(s) will appreciate that private state 300 may differ across nodes 170, i.e., different nodes may store different private states than other nodes. For example, node 170A may house a different private state than node 170B. Private state 300 may include wallets 302, owned notes 304, and proof witnesses 306.

[0061] Wallets 302 may house private keys, public keys, addresses, and other cryptographic information related to nodes 170. Wallets 302 may house a master private key that may be used to establish ownership of assets. This private key may be known only to a particular node and allows the node to determine a history of transfers to and from the node. Wallets 302 may also include additional public and private keys used to sign transactions and derive additional addresses that may be used to perform transactions in asset transfer system 110. For example, wallets 302 may include a public key that represents the node, is known to other participants, and allows other participants to send assets to the node. Wallets 302 may also store numerous addresses derived from the private key and an appropriate diversifier that can be used to receive transfers. Wallets 302 may further include a spend authorizing key, proof authorizing keys, viewing keys, and additional keys, addresses, and cryptographic primitives. In an embodiment, a single node may store a set of wallets associated. In another embodiment, a single node may be limited to only one wallet.

[0062] Owned notes 304 may store a representation of a node's assets, i.e., the assets that a node presently owns. When an addition is made to public ledger 160, each node may examine the addition and attempt to decrypt the notes for each wallet contained in the node. If the node successfully decrypts the outgoing note, the node is the recipient (new

owner) of the note and saves the note for later use by that node by adding the note to owned notes 304. Owned notes 304 may include suitable descriptive fields including: the payment address used to receive the note, the value of the note, a randomness element, a note nullifier, and information about the note in the note commitment tree such as the anchor, the path, and the position.

[0063] Proof witnesses 306 may house data needed to generate various ZKPs employed by asset transfer system 110. Proof witnesses 306 may contain data that is kept confidential and not available to the general public. The data included in proof witnesses 306 may vary based on the nature of the transaction, i.e., spend-note proofs (for nullifications) versus output-note proofs (for transferring) may require different data in proof witnesses 306. Proof witnesses 306 may further allow the receiver of a transaction to verify the identity of the sender by creating a shared key based on the sender's information and an address associated with the receiver.

[0064] FIG. 4 depicts a flowchart illustrating a method 400 for generating and validating a transaction in an asset transfer system. Method 400 presents a general overview of a transaction occurring within asset transfer system 110, to be contrasted with FIG. 5 below, which presents an asset transfer transaction that provides for sender-identity verification. Method 400 may be performed by processing logic that can comprise hardware (e.g., circuitry, dedicated logic, programmable logic, microcode, etc.), software (e.g., instructions executing on a processing device), or a combination thereof. It is to be appreciated that not all steps may be needed to perform the disclosure provided herein. Further, some of the steps may be performed simultaneously, or in a different order than shown in FIG. 4, as will be understood by a person of ordinary skill in the art(s).

[0065] In 402, node 170A computes data for a desired transaction. The computation performed may be relevant to an asset transfer, as described in further detail below with reference to FIG. 5. However, the transaction may also be an issuance, rule-change, or other suitable transaction within asset transfer system 110. This computation considers public data from public state 200 and private data from private state 300. Node 170A further interacts with asset transfer system 110 and/or message queue 130 to generate an appropriate ZKP or ZKPs based on appropriate public and private data related to the transaction.

[0066] In 404, node 170A may broadcast the transaction to public ledger 160. The nature of the broadcast may vary based on the type of technology employed on public ledger 160. For example, where public ledger 160 is a relational database, broadcasting the transaction may take the form of an insert statement or other transact-SQL. Where public ledger 160 is a blockchain technology, an appropriate transaction may be verified by consensus and added as a block to public ledger 160.

[0067] In 406, node 170B validates the transaction that was broadcast in 404. Here, node 170B may be another participating node in the private asset transfer system, and node 170B is not necessarily the receiver of the transaction. All nodes among nodes 170 may independently validate the data that node 170A broadcast to public ledger 160 by verifying the ZKPs contained in the transaction. Nodes 170 may also verify that node 170A did not break any consensus rules and other administrative protocols in compiling and broadcasting the transaction. In a situation where the transaction does not comply with rules 206 and/or does not provide verifiable ZKPs, the transaction may be ignored. Ignored transactions may remain on public ledger 160, but may not be considered when performing calculations based on public ledger 160.

[0068] In 408, node 170B updates public state 200 locally based on the transaction. For example, if the broadcast transaction was a rule-change, node 170B may update local administration rules stored in rules 206. All nodes among nodes 170 may independently update local data about the public state based on a broadcast transaction. The foregoing description of method 400 provides background information for transaction broadcasting and verification generally, and method 500 will be described next which characterizes an asset transfer transaction in particular.

[0069] FIG. 5 depicts a flowchart illustrating a method 500 for transferring an asset from a sender to a receiver in an asset transfer system. Method 500 may be performed by processing logic that can comprise hardware (e.g., circuitry, dedicated logic, programmable logic, microcode, etc.), software (e.g., instructions executing on a processing device), or a combination thereof. It is to be appreciated that not all steps may be needed to perform the disclosure provided herein. Further, some of the steps may be performed simultaneously, or in a different order than shown in FIG. 5, as will be understood by a person of ordinary skill in the art(s).

[0070] Throughout the below description, node 170A will be referred to as the sender of an asset in a transaction and node 170B will be referred to as the receiver of the asset in of transaction, while node 170C will be serving as another participant in the private asset transfer system lacking privity to the transaction. However, a node among nodes 170 that sends an asset may subsequently serve as a receiver in a future transaction or a mere validator. Accordingly, the delegation of these roles within the context of FIG. 5 and the associated disclosure is merely exemplary.

[0071] In 502, node 170B may communicate to node 170A an address associated with node 170B to send an asset. Node 170B may derive the address using a diversifier and a private key known only to node 170B. Node 170B may thus house numerous such addresses in wallets 302 as derived from a private key. Node 170B may communicate the address to node 170A using any suitable communication protocol. In one embodiment, this communication may occur via functions or API calls offered by asset transfer system 110 and accessed by node 170B. However, in another embodiment, the communication of the address may occur externally, i.e., through any suitable conventional method of communication such as email, text, instant message, orally, etc. By receiving the address, node 170A (the sender) knows where to send the asset.

[0072] In 504, node 170A may generate a one-time key, also referred to as a one-time signature. This one-time key may be used in subsequent steps to sign a transfer transaction. Node 170A may derive the one-time key from a private key stored in wallets 302. The one-time key may be unique to a transaction, i.e., each transaction created by node 170A may be signed by a different one-time key, which prevents the general public from linking transactions stored on public ledger 160 back to a single sender. In an embodiment, node 170A may also generate a unique nullifier based on the private key. The unique nullifier may be associated with a spend note to prevent against double-spending of assets, while the one-time signature is associated with an output note that details the recipient of the asset, the amount, the asset identifier, and other suitable information.

[0073] In 506, node 170A may encrypt a public key identifying the sender in a fashion that may only be decrypted by the receiver (here, node 170B). Node 170A may employ various suitable cryptographic primitives to achieve such encryption. For example, node

170A may generate a shared key using an address associated with the receiver and then use the shared key to encrypt the public key identifying node 170A stored in private state 300. The shared key may subsequently be used by node 170B to decode the sender's identity, which can be further verified by verifying ZKPs, as described in further detail below in step 518.

[0074] In 508, node 170A may generate, create, or cause to be created a ZKP or ZKPs needed that will be used by nodes 170 to validate the transfer while preserving confidentiality. The ZKP(s) may allow the transfer to be performed confidentially, preserving the anonymity of both the sender and the receiver while hiding details about the asset being transferred. In an embodiment, node 170A may generate three different ZKPs to be verified by the other nodes in the asset transfer system to authenticate the transfer of the asset:

- (1) the spend proof (verifying facts about the notes owned by the sender);
- (2) the output proof (proving facts about the notes owned by the receiver); and
- (3) the converter proof (ensuring that the spent note and the output note are consistent).

[0075] In addition, node 170A may generate a ZKP that when verified proves that the one-time key generated in 504 corresponds to the public key identifying node 170A, thus allowing the receiver to verify that the received transaction and sender identifier agree. In an example the ZKP may be generated using zk-SNARKS. As mentioned above, zk-SNARKS has three algorithms: SETUP, PROVE, and VERIFY.

[0076] For the SETUP algorithm, a constraint system may be passed to the SETUP() algorithm. The constraint system passed to the SETUP() algorithm may specify how the that a one-time key generated, as was generated in 504, corresponds to the public key identifying node 170A. For example, both the public key identifying the sender and the one-time key may be derived from the same private key of the sender, perhaps using one-way cryptographic function. From the constraint system, the SETUP algorithm generates a proving key (pk) and a verification key (vk).

[0077] The PROVE algorithm generates a proof from the proving key, public data, and the private data. Here, the public data and private data algorithm along with the proving key. The private data may include the sender's private key used to generate both the one-time key generated and the sender's public key. The public data may include the one-time

key created by the sender, the public key of the sender and other suitable information for the transaction. The PROVE() algorithm may generate a ZKP (π) from the public data and private data. Although the foregoing describes the ZKPs as being separate and distinct, in an alternate embodiment, a single ZKP may be generated that accomplishes the requisite proofs.

[0078] In 510, node 170A may build appropriate notes to complete a transfer transaction using the ZKPs and cryptographic primitives created in prior steps. In an embodiment, node 170A may create a note to destroy an owned note among owned notes 304 (the spent asset) and also create a new note (the output note) to effectively complete a transaction in asset transfer system 110. Node 170A may also create a unique nullifier that may be associated with the spend note that allows asset transfer system 110 to check that the sender is not engaging in a double-spend, i.e., sending the same asset twice to separate receivers. The unique nullifier may be distinctly and deterministically derived from a note using the private key known only to node 170A. In such an embodiment, attempting to spend the same asset twice would result in the same unique nullifier appearing twice in nullifier set 202, which would be an invalid happenstance under rules 206, and result in an ignored transaction.

[0079] In 512, node 170A may broadcast the transfer transaction to public ledger 160 in accordance with a transfer protocol as specified by ledger interface 150 and/or asset transfer system 110. The transfer transaction may include any notes generated in 510. In an embodiment, these notes may include a spent note, an output note, and an asset converter. Additionally, the notes may include or be associated with the one-time signature key generated by the node 170A in 504. Asset transfer system 110 may add the spent note as a note nullifier to nullifier set 202 or otherwise store the spent note on public ledger 106. Asset transfer system 110 may add the output note as a note commitment to commitment tree 204 or otherwise broadcast the output note to public ledger 106. Asset transfer system 110 may also include a conversion note and additional information related to the sender-identity verification in the broadcast transaction.

[0080] In 514, nodes 170, including node 170B and node 170C may independently validate the broadcast transaction by verifying the ZKPs contained in the transaction. For example, the broadcast transaction may be verified by nodes 170 using the VERIFY()

algorithm by passing the information in public data (e.g., the one-time transaction key and the public key of the sender), the verification key returned by the SETUP() algorithm, and the ZKP π . Though FIG. 5 includes only node 170A, node 170B, and node 170C, many additional nodes may participate in asset transfer system 110, and all nodes in the system may independently validate the broadcast transaction. For example, nodes 170 may employ message queue 130 to performing proofing tasks related to the ZKPs generated in 504, e.g., the spend proof, the output proof, and the converter proof. Nodes 170 may also validate that consensus rules and other administrative protocols were not broken. In a situation where the transaction does not comply with rules 206 and/or does not provide verifiable ZKPs, the transaction may be ignored. Such a transaction may continue to reside on public ledger 160, but may not be considered in future calculations (e.g., when nodes 170 calculate private state 300 and determine owned notes 304). Notably, when node 170B (the receiver) validates the transaction during this step, node 170B may be able to decrypt the transaction to ascertain additional information that remains hidden to node 170C. By being able to decrypt the transaction, node 170B becomes aware that it is the receiver of the transaction. Node 170B may then perform additional steps, described below, that node 170C, as a non-receiving participant, cannot and does not perform.

[0081] In 516, node 170C, a node that is neither the sender nor the receiver for the above described transaction, apply the validated transaction from 514 to public state 200 associated with node 170C. Each node among nodes 170 may validate the data, confirm adherence to the consensus rules, and update the local public state 200 in this fashion. This ensures transactional integrity across asset transfer system 110.

[0082] In 518, node 170B (the receiver) may determine, verify, and validate the identity of the sender. Node 170B may be able to decrypt additional details about the transaction (e.g., the amount, the asset identifier, etc.) in addition to validating it in step 514 because the transaction may be encrypted using an address or other suitable cryptographic primitives associated with node 170B. Node 170B may also determine the identity of the sender using the shared key generated by node 170A and used to encrypt the public key. Additionally, node 170B may validate that the sender's identity is correct with respect to the transaction by verifying a ZKP generated by the sender that proves that the one-time

key used to sign the transaction actually identifies the sender. In this sense, the identity of the participant that signed the transaction and the identity of the sender sent to the receiver need to agree for the transaction to be valid.

[0083] In 520, node 170B may update its private state 300 based on the information contained in the transaction. For example, node 170B may update owned notes 304 to indicate the additionally owned asset. As described above, node 170B knows that the transaction was directed to node 170B based on the fact that node 170B was able to decrypt the transaction.

[0084] In 522, node 170B may apply the validated transaction from 514 to public state 200 associated with node 170B. Thus, node 170B may validate the data, confirm adherence to the consensus rules, and update the local public state 200. This ensures transactional integrity across asset transfer system 110. As a result of the foregoing, a transaction may be completed in a private asset transfer system that transfers an asset from a sender to a receiver. The receiver may determine and validate the identity of the sender. Neither the sender identity nor the transactional details are publically exposed.

[0085] Various embodiments may be implemented, for example, using one or more well-known computer systems, such as computer system 600 shown in FIG. 6. One or more computer systems 600 may be used, for example, to implement any of the embodiments discussed herein, as well as combinations and sub-combinations thereof.

[0086] Computer system 600 may include one or more processors (also called central processing units, or CPUs), such as a processor 604. Processor 604 may be connected to a communication infrastructure or bus 606.

[0087] Computer system 600 may also include user input/output device(s) 608, such as monitors, keyboards, pointing devices, etc., which may communicate with communication infrastructure 606 through user input/output interface(s) 602.

[0088] One or more of processors 604 may be a graphics processing unit (GPU). In an embodiment, a GPU may be a processor that is a specialized electronic circuit designed to process mathematically intensive applications. The GPU may have a parallel structure that is efficient for parallel processing of large blocks of data, such as mathematically intensive data common to computer graphics applications, images, videos, etc.

- [0089]** Computer system 600 may also include a main or primary memory 608, such as random access memory (RAM). Main memory 608 may include one or more levels of cache. Main memory 608 may have stored therein control logic (i.e., computer software) and/or data.
- [0090]** Computer system 600 may also include one or more secondary storage devices or memory 610. Secondary memory 610 may include, for example, a hard disk drive 612 and/or a removable storage device or drive 614. Removable storage drive 614 may be a floppy disk drive, a magnetic tape drive, a compact disk drive, an optical storage device, tape backup device, and/or any other storage device/drive.
- [0091]** Removable storage drive 614 may interact with a removable storage unit 618. Removable storage unit 618 may include a computer usable or readable storage device having stored thereon computer software (control logic) and/or data. Removable storage unit 618 may be a floppy disk, magnetic tape, compact disk, DVD, optical storage disk, and/ any other computer data storage device. Removable storage drive 614 may read from and/or write to removable storage unit 618.
- [0092]** Secondary memory 610 may include other means, devices, components, instrumentalities or other approaches for allowing computer programs and/or other instructions and/or data to be accessed by computer system 600. Such means, devices, components, instrumentalities or other approaches may include, for example, a removable storage unit 622 and an interface 620. Examples of the removable storage unit 622 and the interface 620 may include a program cartridge and cartridge interface (such as that found in video game devices), a removable memory chip (such as an EPROM or PROM) and associated socket, a memory stick and USB port, a memory card and associated memory card slot, and/or any other removable storage unit and associated interface.
- [0093]** Computer system 600 may further include a communication or network interface 624. Communication interface 624 may enable computer system 600 to communicate and interact with any combination of external devices, external networks, external entities, etc. (individually and collectively referenced by reference number 628). For example, communication interface 624 may allow computer system 600 to communicate with external or remote devices 628 over communications path 626, which may be wired

and/or wireless (or a combination thereof), and which may include any combination of LANs, WANs, the Internet, etc. Control logic and/or data may be transmitted to and from computer system 600 via communication path 626.

[0094] Computer system 600 may also be any of a personal digital assistant (PDA), desktop workstation, laptop or notebook computer, netbook, tablet, smart phone, smart watch or other wearable, appliance, part of the Internet-of-Things, and/or embedded system, to name a few non-limiting examples, or any combination thereof.

[0095] Computer system 600 may be a client or server, accessing or hosting any applications and/or data through any delivery paradigm, including but not limited to remote or distributed cloud computing solutions; local or on-premises software (“on-premise” cloud-based solutions); “as a service” models (e.g., content as a service (CaaS), digital content as a service (DCaaS), software as a service (SaaS), managed software as a service (MSaaS), platform as a service (PaaS), desktop as a service (DaaS), framework as a service (FaaS), backend as a service (BaaS), mobile backend as a service (MBaaS), infrastructure as a service (IaaS), etc.); and/or a hybrid model including any combination of the foregoing examples or other services or delivery paradigms.

[0096] Any applicable data structures, file formats, and schemas in computer system 600 may be derived from standards including but not limited to JavaScript Object Notation (JSON), Extensible Markup Language (XML), Yet Another Markup Language (YAML), Extensible Hypertext Markup Language (XHTML), Wireless Markup Language (WML), MessagePack, XML User Interface Language (XUL), or any other functionally similar representations alone or in combination. Alternatively, proprietary data structures, formats or schemas may be used, either exclusively or in combination with known or open standards.

[0097] In some embodiments, a tangible, non-transitory apparatus or article of manufacture comprising a tangible, non-transitory computer useable or readable medium having control logic (software) stored thereon may also be referred to herein as a computer program product or program storage device. This includes, but is not limited to, computer system 600, main memory 608, secondary memory 610, and removable storage units 618 and 622, as well as tangible articles of manufacture embodying any combination of the foregoing. Such control logic, when executed by one or more data

processing devices (such as computer system 600), may cause such data processing devices to operate as described herein.

[0098] Based on the teachings contained in this disclosure, it will be apparent to persons skilled in the relevant art(s) how to make and use embodiments of this disclosure using data processing devices, computer systems and/or computer architectures other than that shown in FIG. 6. In particular, embodiments can operate with software, hardware, and/or operating system implementations other than those described herein.

[0099] It is to be appreciated that the Detailed Description section, and not any other section, is intended to be used to interpret the claims. Other sections can set forth one or more but not all exemplary embodiments as contemplated by the inventor(s), and thus, are not intended to limit this disclosure or the appended claims in any way.

[0100] While this disclosure describes exemplary embodiments for exemplary fields and applications, it should be understood that the disclosure is not limited thereto. Other embodiments and modifications thereto are possible, and are within the scope and spirit of this disclosure. For example, and without limiting the generality of this paragraph, embodiments are not limited to the software, hardware, firmware, and/or entities illustrated in the figures and/or described herein. Further, embodiments (whether or not explicitly described herein) have significant utility to fields and applications beyond the examples described herein.

[0101] Embodiments have been described herein with the aid of functional building blocks illustrating the implementation of specified functions and relationships thereof. The boundaries of these functional building blocks have been arbitrarily defined herein for the convenience of the description. Alternate boundaries can be defined as long as the specified functions and relationships (or equivalents thereof) are appropriately performed. Also, alternative embodiments can perform functional blocks, steps, operations, methods, etc. using orderings different than those described herein.

[0102] References herein to “one embodiment,” “an embodiment,” “an example embodiment,” or similar phrases, indicate that the embodiment described can include a particular feature, structure, or characteristic, but every embodiment can not necessarily include the particular feature, structure, or characteristic. Moreover, such phrases are not necessarily referring to the same embodiment. Further, when a particular feature,

structure, or characteristic is described in connection with an embodiment, it would be within the knowledge of persons skilled in the relevant art(s) to incorporate such feature, structure, or characteristic into other embodiments whether or not explicitly mentioned or described herein. Additionally, some embodiments can be described using the expression “coupled” and “connected” along with their derivatives. These terms are not necessarily intended as synonyms for each other. For example, some embodiments can be described using the terms “connected” and/or “coupled” to indicate that two or more elements are in direct physical or electrical contact with each other. The term “coupled,” however, can also mean that two or more elements are not in direct contact with each other, but yet still co-operate or interact with each other.

[0103] The breadth and scope of this disclosure should not be limited by any of the above-described exemplary embodiments, but should be defined only in accordance with the following claims and their equivalents.

WHAT IS CLAIMED IS:

1. A computer-implemented method for privately verifying a sender, comprising:
receiving a transaction that identifies a transfer of an asset from the sender to a receiver,
wherein the transaction is signed by the sender with a one-time key that does not link
to a prior transaction of the sender;
privately sending, from the sender to the receiver, a public key identifying the sender;
generating at least one zero-knowledge proof that, when verified, proves that the one-
time key corresponds to the public key identifying the sender; and
writing the transaction and the at least one zero-knowledge proof to a public ledger,
wherein the transaction does not broadcast to the public ledger the identity of the
sender, and
wherein the receiver can verify the sender's identity by verifying the zero-knowledge
proof with the public key.
2. The computer-implemented method of claim 1, the privately sending further
comprising:
generating a shared key using an address associated with the receiver;
creating an encrypted public key using the shared key and the public key; and
including the encrypted public key in the transaction.
3. The computer-implemented method of claim 1, wherein the identity of the
receiver, the asset, and asset amount are encrypted on the public ledger and cannot be ascertained
from the public ledger by a third node.
4. The computer-implemented method of claim 1, wherein verifying the zero-
knowledge proof further validates the transfer on the public ledger by proving that the sender
owned the asset.
5. The computer-implemented method of claim 1, wherein the transaction further
comprises a unique nullifier that protects against a double-spend of the asset.

6. The computer-implemented system of claim 1, wherein the asset is a cryptocurrency.

7. The computer-implemented method of claim 1, wherein the public ledger is a blockchain.

8. The computer-implemented method of claim 1, the generating further comprising: generate the zero-knowledge proof using parallel processing.

9. A system for privately verifying a sender, comprising:
a memory; and

at least one processor coupled to the memory and configured to:

receive a transaction that identifies a transfer of an asset from the sender to a receiver, wherein the transaction is signed by the sender with a one-time key that does not link to a prior transaction of the sender;

privately send, from the sender to the receiver, a public key identifying the sender;
generate a zero-knowledge proof that, when verified, proves that the one-time key corresponds to the public key identifying the sender; and

write the transaction and the zero-knowledge proof to a public ledger, wherein the transaction does not broadcast to the public ledger the identity of the sender, and wherein the receiver can verify the sender's identity by verifying the zero-knowledge proof with the public key.

10. The system of claim 9, wherein to privately send the at least one processor is further configured to:

generate a shared key using an address associated with the receiver;
create an encrypted public key using the shared key and the public key; and
include the encrypted public key in the transaction.

11. The system of claim 9, wherein the identity of the receiver, the asset, and asset amount are encrypted on the public ledger and cannot be ascertained from the public ledger by a third node.
12. The system of claim 9, wherein verifying the zero-knowledge proof further validates the transfer on the public ledger by proving that the sender owned the asset.
13. The system of claim 9, wherein to generate the zero-knowledge proof the at least one processor is further configured to:
generate the zero-knowledge proof using parallel processing.
14. The system of claim 9, wherein the transaction further comprises a unique nullifier that protects against a double-spend of the asset.
15. The system of claim 9, wherein the asset is a cryptocurrency.
16. The system of claim 9, wherein the public ledger is a blockchain.
17. A non-transitory computer-readable device having instructions stored thereon that, when executed by at least one computing device, cause the at least one computing device to perform operations comprising:
receiving a transaction that identifies a transfer of an asset from the sender to a receiver, wherein the transaction is signed by the sender with a one-time key that does not link to a prior transaction of the sender;
privately sending, from the sender to the receiver, a public key identifying the sender;
generating a zero-knowledge proof that, when verified, proves that the one-time key corresponds to the public key identifying the sender; and
writing the transaction and the zero-knowledge proof to a public ledger, wherein the transaction does not broadcast to the public ledger the identity of the sender, and wherein the receiver can verify the sender's identity by verifying the zero-knowledge proof with the public key.

18. The non-transitory computer-readable device of claim 17, wherein verifying the zero-knowledge proof further validates the transfer on the public ledger by proving that the sender owned the asset.

19. The non-transitory computer-readable device of claim 17, wherein the identity of the receiver, the asset, and asset amount are encrypted on the public ledger and cannot be ascertained from the public ledger by a third node.

20. The non-transitory computer-readable device of claim 17, the operations further comprising:
generating the zero-knowledge proof using parallel processing.