US 20200059362A1

(19) United States

(12) Patent Application Publication (10) Pub. No.: US 2020/0059362 A1

BRODY et al. (43) Pub. Date: Feb. 20, 2020

(54) METHODS AND SYSTEMS FOR ENHANCING PRIVACY ON DISTRIBUTED LEDGER-BASED NETWORKS

(71) Applicants: **Ernst & Young U.S. LLP**, New York, NY (US); **Ernst & Young Services (UK) Limited**, London (GB); **Ernst & Young Advisory**, Courbevoie (FR)

(72) Inventors: **Paul Richard BRODY**, Woodside, CA (US); **Duncan James WESTLAND**, Addlestone (GB); **Chaitanya Reddy KONDA**, London (GB); **Quentin DROUOT**, Antony (FR); **Xavier De BOISSIEU**, Paris (FR)

(21) Appl. No.: **16/534,858**

(22) Filed: **Aug. 7, 2019**

Related U.S. Application Data

(60) Provisional application No. 62/719,636, filed on Aug. 18, 2018, provisional application No. 62/748,002, filed on Oct. 19, 2018.

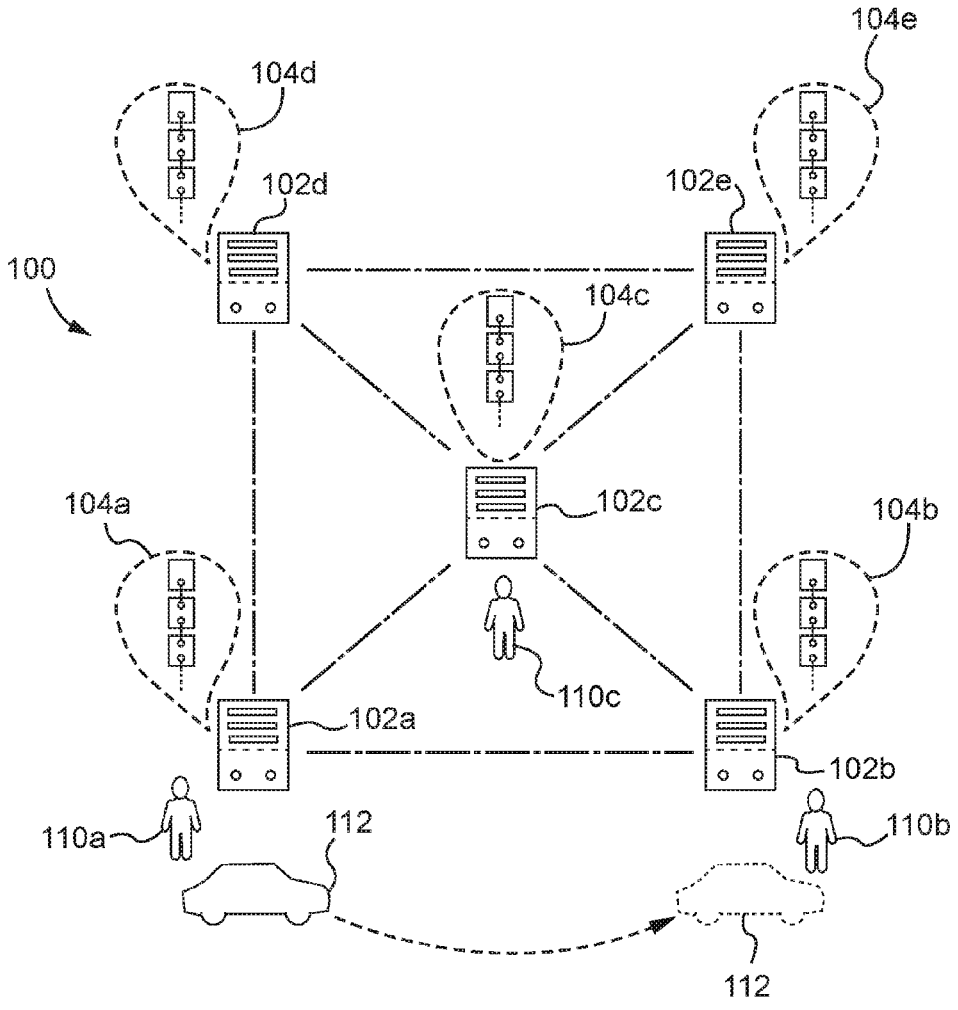
Publication Classification

(51) **Int. Cl.**
H04L 9/32 (2006.01)
H04L 9/30 (2006.01)

(52) **U.S. Cl.**
CPC ***H04L 9/3221*** (2013.01); ***H04L 9/30*** (2013.01); ***H04L 9/3242*** (2013.01); ***H04L 9/3213*** (2013.01)

(57) **ABSTRACT**

One or more embodiments described herein disclose methods and systems that are directed at providing enhanced privacy and security to distributed ledger-based networks (DLNs) via the implementation of zero-knowledge proofs (ZKPs) in the DLNs. ZKPs allow participants of DLNs to make statements on the DLNs about some private information and to prove the truth of the information without having to necessarily reveal the private information publicly. As such, the disclosed methods and systems directed at the ZKP-enabled DLNs provide privacy to participants of the DLNs while still allowing the DLNs to remain as consensus-based networks.



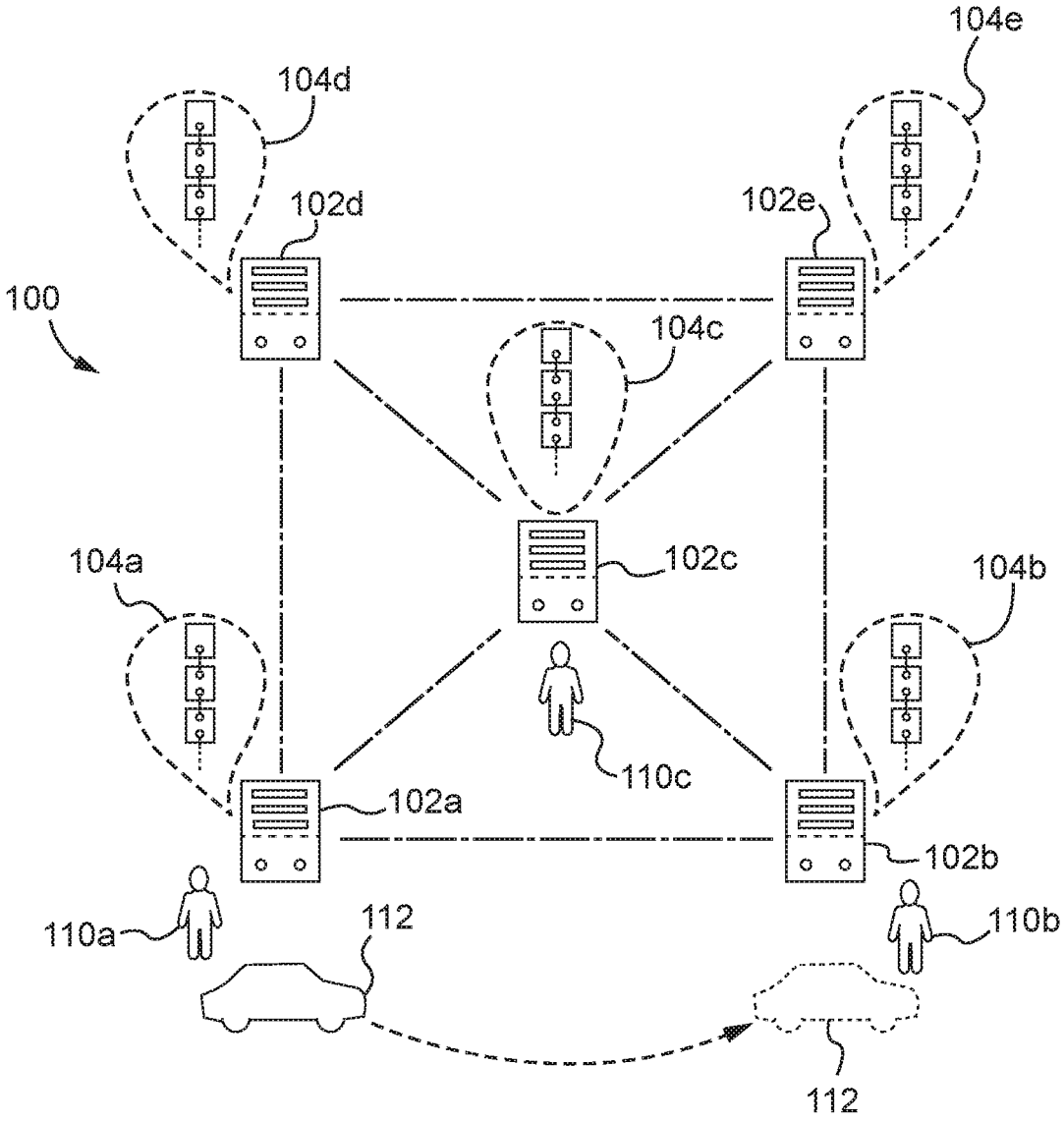

104e
104d
102d
102e
100
104c
104a
102c
104b
110c
102a
102b
110a
112
110b
112

FIG. 1

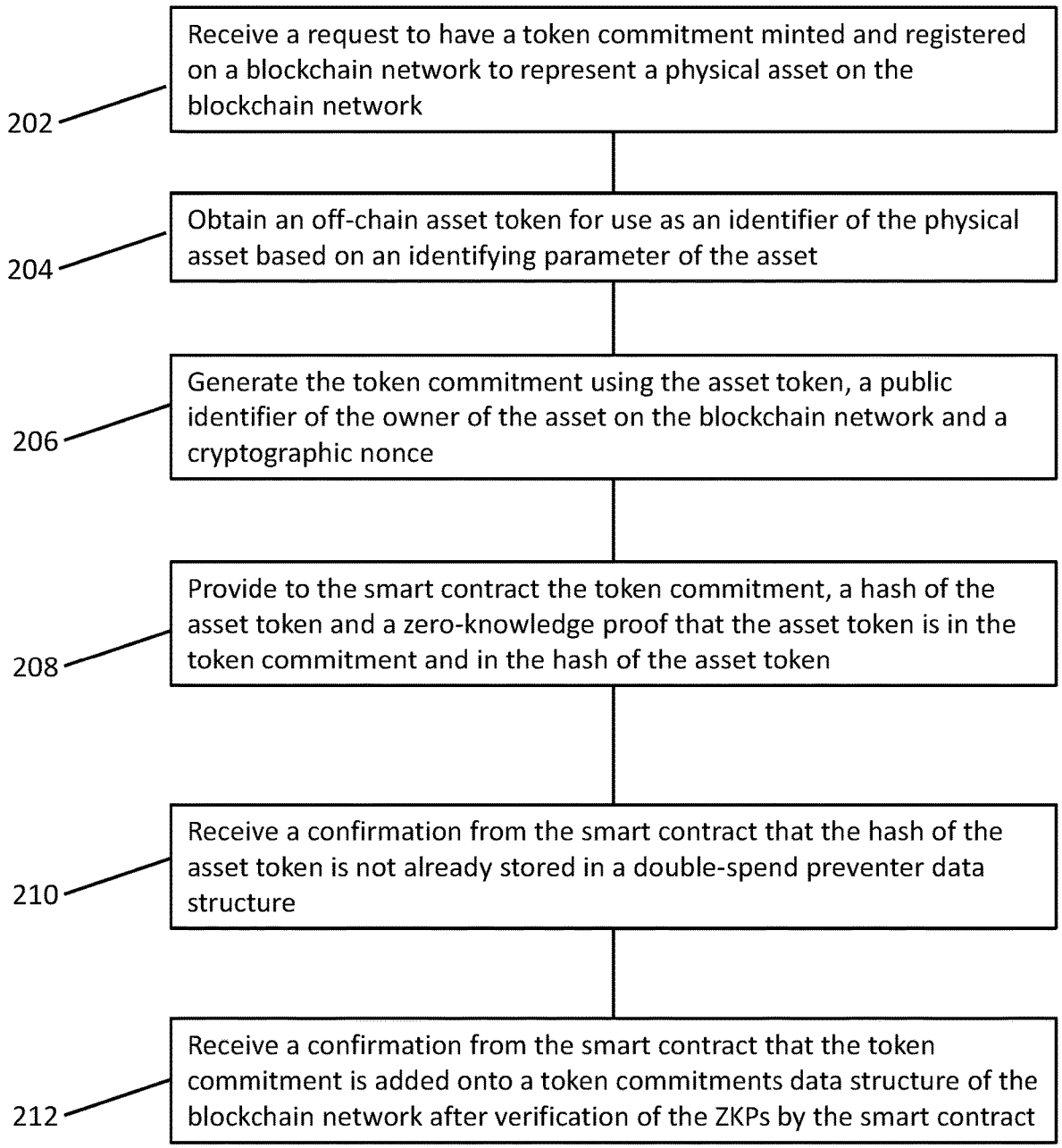

Receive a request to have a token commitment minted and registered on a blockchain network to represent a physical asset on the blockchain network
202
Obtain an off-chain asset token for use as an identifier of the physical asset based on an identifying parameter of the asset
204
Generate the token commitment using the asset token, a public identifier of the owner of the asset on the blockchain network and a cryptographic nonce
206
Provide to the smart contract the token commitment, a hash of the asset token and a zero-knowledge proof that the asset token is in the token commitment and in the hash of the asset token
208
Receive a confirmation from the smart contract that the hash of the asset token is not already stored in a double-spend preventer data structure
210
Receive a confirmation from the smart contract that the token commitment is added onto a token commitments data structure of the blockchain network after verification of the ZKPs by the smart contract
212

FIG. 2

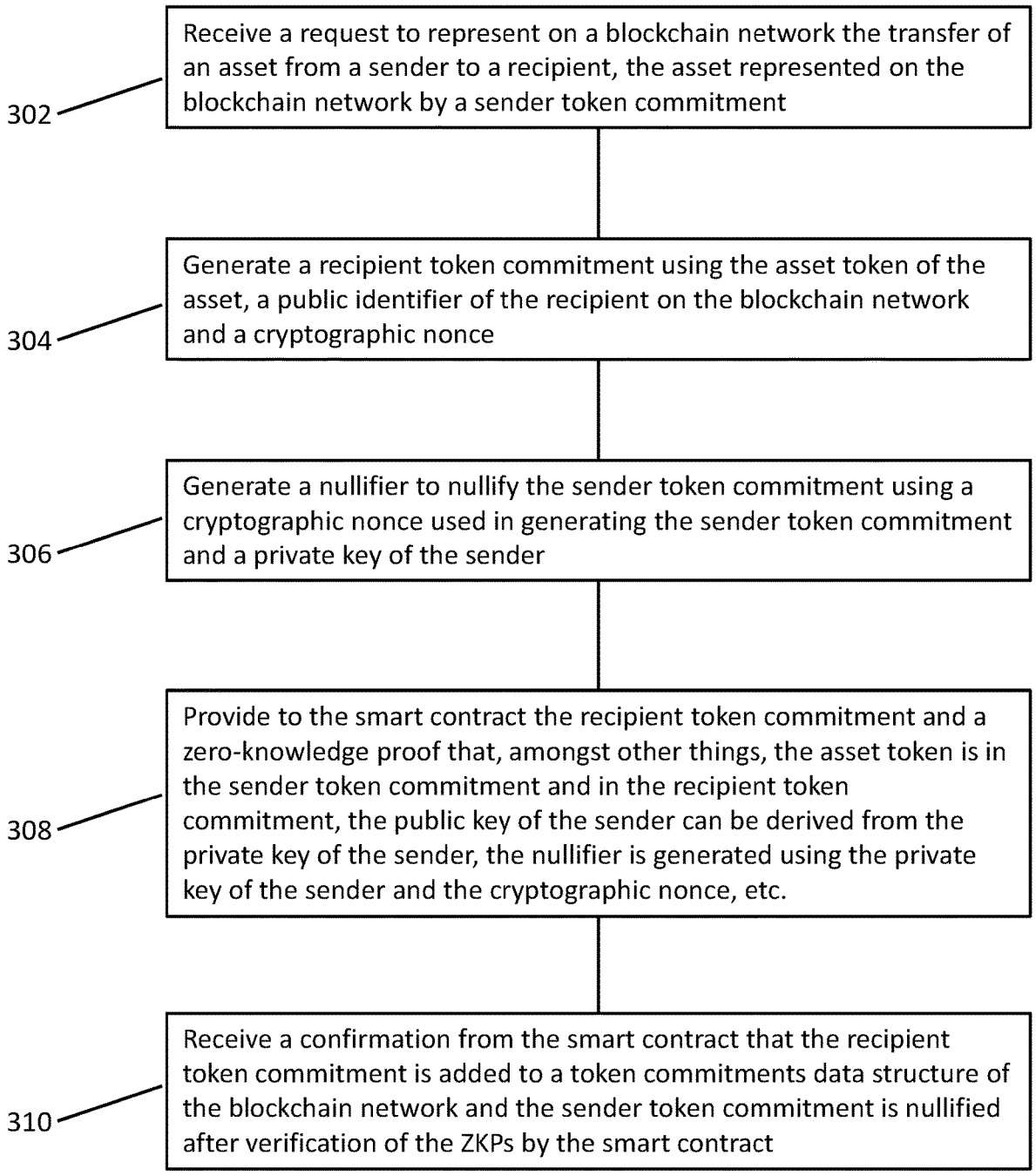

Receive a request to represent on a blockchain network the transfer of an asset from a sender to a recipient, the asset represented on the blockchain network by a sender token commitment
302
Generate a recipient token commitment using the asset token of the asset, a public identifier of the recipient on the blockchain network and a cryptographic nonce
304
Generate a nullifier to nullify the sender token commitment using a cryptographic nonce used in generating the sender token commitment and a private key of the sender
306
Provide to the smart contract the recipient token commitment and a zero-knowledge proof that, amongst other things, the asset token is in the sender token commitment and in the recipient token commitment, the public key of the sender can be derived from the private key of the sender, the nullifier is generated using the private key of the sender and the cryptographic nonce, etc.
308
Receive a confirmation from the smart contract that the recipient token commitment is added to a token commitments data structure of the blockchain network and the sender token commitment is nullified after verification of the ZKPs by the smart contract
310

FIG. 3

METHODS AND SYSTEMS FOR ENHANCING PRIVACY ON DISTRIBUTED LEDGER-BASED NETWORKS

CROSS-REFERENCE TO RELATED PATENT APPLICATION

[0001] This application claims priority to and the benefit of U.S. Provisional Application No. 62/719,636, filed Aug. 18, 2018, entitled "Methods and Systems of ZKP-Based Secure PE Transactions on Public Networks," and U.S. Provisional Application No. 62/748,002, filed Oct. 19, 2018, entitled "Methods and Systems of ZKP-Based Secure Private Enterprise Transactions on Public Networks," both of which are incorporated herein by reference in their entirety.

FIELD OF THE DISCLOSURE

[0002] Distributed ledger-based networks (DLNs) dispense with the need for a central authority to manage the operations of the networks due to their transparency and consensus-based verification mechanisms for validating actions occurring on the DLNs, which allow participants of the networks to trust the accuracy of the validations without the central authority. The transparency and consensus-based verification mechanisms, however, compromise the privacy of the actions and the involved parties, as relevant information has to be shared with at least a substantial portion of the participants of the DLNs for the actions to be validated. The disclosure illustrates how the privacy and security of such actions can be enhanced with the use of zero-knowledge proofs (ZKPs) that can be used to verify the validity of at least some aspects of the actions without private information related to the actions necessarily being revealed publicly. The disclosure discloses methods and systems that are directed at providing enhanced security and privacy to actions conducted on ZKP-enabled DLNs.

BACKGROUND

[0003] Organizations can use private networks as well as public networks such as the internet and distributed ledger-based networks (DLNs) to manage and track the production and shipping of large quantities of items or assets. The use of private networks, however, can be inefficient and costly, while public networks may not provide the desired level of privacy and/or security. For example, public DLNs can expose, by virtue of being public networks, details of private interactions occurring on the networks.

BRIEF DESCRIPTION OF THE FIGURES

[0004] FIG. **1** shows a distributed ledger-based network configured for use in managing and conducting a private transaction between two parties that are participants of the network, according to some embodiment.

[0005] FIG. **2** shows a flow chart illustrating the minting a token on a distributed ledger-based network to represent a real world or physical asset on the distributed ledger-based network, according to some embodiment.

[0006] FIG. **3** shows a flow chart illustrating the creation or generation of new token commitment on the distributed ledger-based network to represent the transfer of a real-world or physical asset from a sender to a recipient, according to some embodiment,

SUMMARY

[0007] Some embodiments of the current disclosure disclose methods and systems that are directed at providing enhanced security and privacy to actions conducted on ZKP-enabled DLNs. For example, the actions may include representing the transfer of an asset from a sender to a recipient, where the asset is represented on the ZKP-enabled DLN by a first token commitment. In such embodiments, the methods may include the steps of: receiving a request that is configured to cause a transfer of the asset from the sender to the recipient; providing by a provider, in response to the request and to a self-executing code segment on the distributed ledger, a zero-knowledge proof (ZKP) that the provider of the ZKP has knowledge of an identity of an asset token, where (1) an application of a first hashing function on the asset token generating the first token commitment, and/or (2) an application of a second hashing function on the asset token generating a second token commitment representing the asset on the distributed ledger. Further, the methods may include the step of receiving, upon verification of the ZKP by the self-executing code segment, a confirmation confirming an addition of the second token commitment onto a token commitments data structure on the distributed ledger.

DETAILED DESCRIPTION

[0008] In some embodiments, parties participating in a transaction may elect to use a public distributed ledger-based network (DLN) to document the details of the transaction and manage its operations. DLNs can provide decentralized platforms that are transparent to at least all the participants of the networks, if not to the public at large, and as such, can be viewed as consensus-based platforms that facilitate trust between transaction participants without the need for a central authority to administer the network. For example, parties participating in a transaction for a sale of a digital music file can use a self-executing code or program (e.g., a smart contract) on the DLN (e.g., a blockchain) to manage the sale of the music file. The self-executing code or smart contract can regulate the exchange of the music file and the correct payment for the file between the parties without involvement from a third party. In some embodiments, the DLNs can also be used to manage transactions involving physical (e.g., non-digital) assets. In some implementations, this can be accomplished by using tokens to represent the assets, and a sale of an asset can be represented by the transfer of the token representing the asset from one party (e.g., the seller) to a second party (e.g., the buyer).

[0009] In some embodiments, a DLN can be and/or support a blockchain. Throughout the instant disclosure, in some embodiments, the terms "distributed ledger-based network" and "blockchain network" may be used interchangeably. Similarly, in some embodiments, the terms "self-executing code" or "self-executing code segment" and "smart contract" may be used interchangeably. Further, in some embodiments, the term "transaction" may be used to refer to off-chain transactions (e.g., transactions involving the sale of physical or digital assets between parties) and/or on-chain representation of these off-chain transactions (e.g., the transaction of tokens that represent the assets on the blockchain network). Whether the term refers to the former or the latter case should be clear from context. The terms "off-chain" or "off-the DLN" are to be understood to mean "not on the blockchain network" or "not on the DLN." For

example, a statement such as "the application of a hashing function is performed off-the DLN" is to be understood as meaning "the application of the hashing function is not performed on the DLN (and is performed elsewhere)".

[0010] As noted above, in some embodiments, the trust the distributed ledger-based networks provide with no need for a central authority derives from the transparency of the networks to at least all the participants of the network (and in the case of public networks, to the public at large). This transparency, however, can reduce or even eliminate any privacy or confidentiality that participants need or seek when interacting with the network or its participants. For example, in the case of public networks, any interested person can access and inspect the distributed ledgers on the networks to obtain detailed information on all transactions that are represented on the ledgers since the inception of the networks (as the ledgers are, in at least most cases, largely immutable). In some implementations, the lack of privacy or confidentiality can render the use of a public ledger-based network untenable. For instance, a pharmacy using a public blockchain network to manage the fulfillment of orders for shipment of prescription drugs without a mechanism to conceal at least some aspects of the transaction would publicly expose personal and health-related data of its customers (thereby violating their privacy and possibly health privacy laws).

[0011] In some cases, private DLNs can be used to provide participants a measure of privacy that may not be available on public networks. The privacy afforded by private (non-ZKP-enabled) DLNs, however, is far from adequate for most purposes (how ZKPs can be used to provide privacy to private and/or public blockchain networks will be discussed in details below). For example, with reference to the above example, the personal and health-related data of customers would still be available for inspection by other members of the private non-ZKP-enabled DLN (even if the data is hidden from the public). Further, private non-ZKP-enabled DLNs would be burdensome to maintain as, amongst other reasons, applications developed for public blockchain networks would not seamlessly interoperate on private non-ZKP-enabled blockchain networks.

[0012] The inefficiency and cost associated with private non-ZKP-enabled DLNs may be illustrated with reference to the internet, which suffers from several privacy and security-related ills due to the openness of the network to anyone capable of accessing the network. Setting up a "private" intranet network can be one way to combat the noted privacy and security-related ills. Such private networks, however, are likely to severely lag in their developments, and even then to be costly to maintain, compared to the open internet, as the closed nature of the private networks would limit interoperability of applications developed for the open or public internet. Analogously, a private DLN would lag in its development compared to a public DLN and still be costly to maintain. One or more embodiments described herein disclose methods and systems that are directed at providing enhanced privacy and security to DLNs via the implementation of ZKPs in the DLNs. It is to be noted that, although descriptions of these embodiments refer to public DLNs, the methods and systems equally apply to private DLNs.

[0013] In some embodiments, as noted above, the current disclosure discloses methods and systems that provide privacy to participants of a transaction on a ZKP-enabled DLN while retaining the level of trust afforded by decentralized networks (i.e., with no central authority) such as DLNs. For example, one or more of the methods and systems disclosed herein allow for the identities of parties to a transaction (e.g., a sale or transfer of an asset between the parties) as well as details of the transaction (e.g., details of the assets being transferred) to remain secret when a public blockchain network is used to manage the transaction. Referring to the example provided above, one or more of the disclosed methods and systems allow the pharmacy to use a public blockchain network to facilitate the shipment of the drugs without revealing on the blockchain network (or publicly) any identifying information related to the assets (i.e., the drugs), the sender (i.e., the pharmacy) and/or the recipient of the assets (i.e., the clients), while depending on the trust afforded by the blockchain network at least partly as a result of the transparency inherent to public blockchain networks. In such examples, the sender and the recipient may be represented by their respective public keys on the blockchain network.

[0014] FIG. **1** shows a ZKP-enabled DLN configured for use in managing and representing a private transaction between two parties that are participants of the network, in particular a public network, according to some embodiment. In some embodiments, the ZKP-enabled DLN or blockchain network **100** includes a plurality of computing nodes **102***a*-**102***e* configured to communicate amongst each other via a peer-to-peer (P2P) connection. In some implementations, the computing nodes **102***a*-**102***e* can be computing devices including but not limited to computers, servers, processors, data/information processing machines or systems, and/or the like, and may include data storage systems such as databases, memories (volatile and/or non-volatile), etc. In some implementations, the P2P connections may be provided by wired and/or wireless communications systems or networks (not shown) such as but not limited to the internet, intranet, local area networks (LANs), wide area networks (WANs), etc., utilizing wireless communication protocols or standards such as WiFi®, LTE®, WiMAX®, and/or the like.

[0015] In some embodiments, the ZKP-enabled DLN **100** may include self-executing codes or smart contracts that are configured to execute upon fulfillment of conditions that are agreed upon between transacting parties. For example, some or all of the computing nodes **102***a*-**102***e* may include copies of a self-executing code that self-execute upon fulfillment of the conditions. In some implementations, the computing nodes **102***a*-**102***e* may communicate amongst each other with the results of the executions of their respective self-executing codes, for example, to arrive at a consensus on the results. In some implementations, one or a few of the computing nodes **102***a*-**102***e* may have self-executing codes that self-execute, and the results would be transmitted to the rest of the computing nodes **102***a*-**102***e* for confirmation.

[0016] In some embodiments, a self-executing code or a smart contract can facilitate the completion of transactions on the ZKP-enabled DLN **100** by providing the transacting parties confidence that the other party would deliver the promised product or payment. For example, with reference to the above example related to the sale of a digital music file, a smart contract can be used to verify that the seller of the file is in fact an owner of the file, the buyer of the music file has adequate resource to pay for the music, etc. Further, the smart contract can facilitate the exchange of the music file by allowing the transfer of a payment to occur only after the transfer of the music file is completed (and validated).

[0017] In some embodiments, the ZKP-enabled DLN **100** may be linked to one or more oracles (not shown) or data feeds that provide external data to the ZKP-enabled DLN **100**. In some implementations, as discussed above, self-executing codes or smart contracts can automatically execute upon realization of some conditions of a transaction, and the oracles may provide the data that can be used to evaluate whether the conditions are met. For example, a transaction may be contingent on the price of a stock, a weather condition, etc., and an oracle may provide the requisite information to the smart contract facilitating the transaction. The smart contract, upon receiving the information, may self-execute after determining that the condition for the transaction has been fulfilled. In some embodiments, the oracles may facilitate for the smart contracts to send data out to external systems. For example, a smart contract may be configured to send out information to a smartphone when an account on the ZKP-enabled DLN **100** receives a payment, and an oracle may serve as a transit hub for the data including the information during its transmission to the smartphone.

[0018] In some embodiments, at least a substantial number of the computing nodes **102***a*-**102***e* include copies of a distributed ledger **104***a*-**104***e* onto which transactions that occur on the network are recorded. The recording of the transactions on the distributed ledger **104***a*-**104***e* may occur when some substantial proportion of the computing nodes **102***a*-**102***e*, or a subset thereof, agree on the validity of the transactions. The distributed ledger **104***a*-**104***e* can be immutable or nearly immutable in the sense that to alter the distributed ledger **104***a*-**104***e*, at least this substantial portion of the computing nodes **102***a*-**102***e* would have to agree, which can be increasingly difficult when the number of computing nodes **102***a*-**102***e* is large (and the distributed ledger **104***a*-**104***e* gets longer).

[0019] As noted above, the ZKP-enabled DLN **100** can be used to facilitate transactions that involve digital assets (e.g., sale of digital music files). In some embodiments, the ZKP-enabled DLN **100** can also be used to facilitate transactions of assets that occur off-chain or off-line (e.g., transactions of physical assets). In some implementations, off-chain assets can be represented by tokens (e.g., token commitments) on the ZKP-enabled DLN **100**, and the sale or transfer of the off-chain assets can be represented on the ZKP-enabled DLN **100** by the transfer of the tokens between the blockchain accounts of the transacting parties. In some implementations, the types of tokens used to represent the off-chain assets can depend on the nature of the assets themselves. For example, fungible products (e.g., some amount of gasoline or a currency) can be represented with fungible tokens while non-fungible products (e.g., distinguishable products such as a product with a serial number) can be represented by non-fungible tokens. FIG. **1** shows an example embodiment of a transaction that involves the sale of an off-chain asset (e.g., vehicle **112**) from a first transaction participant **110***a* to a second transaction participant **110***b*. In such example, the vehicle may be represented on the ZKP-enabled DLN **100** with a non-fungible token that can be transferred from the first transaction participant **110***a* to the second transaction participant **110***b* to represent the sale or transfer of the vehicle **112** during the transaction between the two parties. In some embodiments, tokens may be stored off-chain, i.e., off of the ZKP-enabled DLN **100**. For example, tokens may be stored in storage systems or databases that are linked with the ZKP-enabled DLN **100**. For instance, if the ZKP-enabled DLN **100** is a ZKP-enabled Ethereum blockchain network, the tokens may be stored in the Swarm database. In some embodiments, the tokens may be stored on the ZKP-enabled DLN **100** (e.g., in the storage systems associated with the computing nodes **102***a*-**102***e*).

[0020] In some embodiments, as noted above, transactions that occur on the ZKP-enabled DLN **100** (including off-chain transactions that are represented on the ZKP-enabled DLN **100** with the use of tokens, for example) are recorded onto at least a substantial number of the distributed ledgers **104***a*-**104***e* that exist on the ZKP-enabled DLN **100**. For example, a transaction between a first transaction participant **110***a* and a second transaction participant **110***b* on the ZKP-enabled DLN **100** representing the transfer of an off-chain asset **112** from the former to the latter would be recorded on all or nearly all of the distributed ledgers **104***a*-**104***e* once the transaction details are accepted as valid by the participants of the ZKP-enabled DLN **100**. In the case of a blockchain network that is not ZKP-enabled, however, the first transaction participant **110***a* and the second transaction participant **110***b* are afforded little or no privacy as all or nearly all the details of the transaction are made public or visible to all that have access to the blockchain network (the public, in case of public blockchains), such details including confidential information on the transacting participants, the asset being transacted, the tokens used to represent the asset and the representation of its transfer on the blockchain network, and/or the like. In some embodiments, the present disclosure discloses methods and systems for providing privacy to transactions that occur or are represented on public blockchains with the use of zero knowledge proofs (ZKPs).

[0021] In some embodiments, ZKPs can be used by a first entity, the "prover" or "provider" of the proofs, to convince a second entity, the "verifier" of the proofs, that a statement about some secret information is truthful without having to reveal the secret information to the verifier. ZKPs can be interactive, i.e., require interaction from the prover for the verifier to verify the truthfulness of the statement. In some embodiments, the ZKPs can be non-interactive, requiring no further interaction from the prover for the verifier to verify the statement. Examples of non-interactive ZKPs include zero-knowledge succinct non-interactive argument of knowledge (zk-SNARK), zero-knowledge scalable transparent argument of knowledge (zk-STARK), etc. Discussions of ZKPs can be found in U.S. patent application Ser. No. 16/383,845, filed on Apr. 15, 2019 and entitled "Methods and Systems for Identifying Anonymized Participants of Distributed Ledger-Based Networks Using Zero-Knowledge Proofs," which is incorporated by reference herein in its entirety.

[0022] FIG. **2** shows a flow chart illustrating the steps of minting a token on a distributed ledger to represent a real world or physical asset on the distributed ledger, according to some embodiment. To represent an off-chain transaction for a non-fungible asset such as the transfer of a vehicle asset **112** from a first transaction participant **110***a* (referred hereinafter as the sender **110***a*) to a second transaction participant **110***b* (referred hereinafter as the recipient **110***b*) on the ZKP-enabled DLN **100**, in some embodiments, the sender **110***a* may generate, at **204** and using the computing node **102***a*, an asset identifier that can serve as a unique identifier for the asset while concealing the asset's identity. In some

implementations, the asset identifier may be generated in response to a request, at **202**, to have a physical asset represented on a ZKP-enabled DLN **100**. For example, the sender **110***a* can generate, using the computing node **102***a*, an alpha-numeric value that is uniquely associated with some identifying parameters (e.g., serial numbers, model numbers, etc.) of the asset, and the alpha-numeric value can be used as the asset identifier that hides the real identity of the asset. As another example, a unique asset identifier can be generated by cryptographically hashing the identifying parameters of the non-fungible asset to generate an asset token that can serve as the unique asset identifier. The cryptographic hashing includes the application of a cryptographic hashing algorithm such as but not limited to the SHA-256 algorithm, on the identifying parameters. For instance, an asset token can be generated for the vehicle asset **112** by applying a hashing function (e.g., SHA-256) on one or more of the identifying parameters of the vehicle such as the vehicle identification number, model type, model year, etc. Accordingly, the asset token can serve as a unique asset identifier without exposing or revealing to other participants of the ZKP-enabled DLN **100** any of the identifying parameters of the asset (i.e., vehicle). In some embodiments, the hashing can occur off the ZKP-enabled DLN **100**. For example, if the ZKP-enabled DLN **100** is a ZKP-enabled Ethereum blockchain network, in some implementations, the asset token can be generated and stored off the Ethereum blockchain network at the Swarm storage network/database.

[0023] At **206**, an off-chain non-fungible asset (e.g., physical asset such as the vehicle **112**) can be registered or represented on the ZKP-enabled DLN **100** for the first time by generating or minting a non-fungible token commitment that encodes at least some aspects of the non-fungible asset and/or the ownership of the asset on the ZKP-enabled DLN **100**. In some embodiments, minting of a token commitment may refer to the registration or representation of an asset on the ZKP-enabled DLN **100** by a token commitment for the first time. As will be discussed below, new token commitments may be generated later to represent an asset that is being represented on the ZKP-enabled DLN **100** by an existing token commitment. In such cases, however, the asset is being transferred to a new owner, and the generation of the new token commitment may be accompanied by the nullification of the existing token commitment (which indicates that the asset does not belong to the initial owner anymore). In any case, whether an asset (e.g., non-fungible asset) is being registered or represented on the ZKP-enabled DLN **100** for the first time by the minting of a token commitment or the transfer of the asset from one owner to another is being registered on the ZKP-enabled DLN **100** by generation of a new token commitment, the minted token commitment and/or the new token commitment may encode at least some aspects of the asset and/or the ownership of the asset. To encode the aspects of the non-fungible asset, in some implementations, a cryptographic hashing function or algorithm can be applied to the unique asset identifier of the asset such as the asset token that itself was obtained via an application of a hashing function on the identifying parameters of the non-fungible asset, as discussed in the example above. Further, to encode some aspects of the ownership of the asset, in some implementations, the cryptographic hashing function can also be applied to a public identifier on the ZKP-enabled DLN **100** that is associated with the owner (e.g., sender **110***a* when the sender **110***a* is minting the token commitment for the first time). An example of such public identifier includes the public key of the sender on the ZKP-enabled DLN **100** (i.e., the public key that is associated with the sender **110***a* on the ZKP-enabled DLN **100**).

[0024] In some embodiments, the cryptographic hashing function can also be applied to a random nonce such as, but not limited to, a random or serial number that is securely generated and used, at least in most cases, only once. In some implementations, the random nonce can be used as a handle of the non-fungible token commitment independent of the non-fungible asset (e.g., encoded by the asset token) and/or its ownership (e.g., encoded by the public key). For example, as discussed below, the transfer of a physical asset to the recipient **110***b* can be represented by the generation and registration on the ZKP-enabled DLN **100** of a new token commitment that associates the asset with the new owner, the recipient **110***b*, and the nullification of the existing token commitment that associated the asset with the sender **110***a*. In such implementations, the token commitment handle, the random nonce, can be used to nullify the existing token commitment, as discussed below.

[0025] In some embodiments, the minting of a non-fungible token commitment to represent an asset ZKP-enabled DLN **100** for the first time may include the computation of a token commitment (Z-token) as follows: $Z=H(S \circledast P_k \circledast A)$, where A is the asset token identifying the asset (e.g., obtained by hashing, off-chain, at least some identifying parameters of the asset), P_k is the public key on the blockchain that is associated with the sender **110***a* (i.e., the current owner of the asset), S is a random nonce, H is a cryptographic hashing function or algorithm (e.g., SHA-256), and $\circledast$ represents a combining operator (e.g., the XOR operator $\oplus$, the concatenation operator |, etc.). In some embodiments, the computation of the token commitment Z may include application of the hashing function on additional elements besides or instead of S, P_k and A. In some embodiments, the token commitment comprises or consists of a random nonce (e.g., a securely and randomly generated serial number), a public identifier on the ZKP-enabled DLN **100** of the sender **110***a* (e.g., public key of the sender **110***a*) and an asset identifier (e.g., asset token A). In some embodiments, the application of the hashing function in computing for the Z-token (i.e., token commitment) allows for the generation or construction of the non-fungible token (e.g., Z-token or token commitment) without revealing the identities of the random nonce S and/or the asset token A on the ZKP-enabled DLN **100** (i.e., S and A may be kept secret by the sender **110***a*, except when A is transmitted (privately) to the recipient **110***b* as discussed below during an asset transfer transaction).

[0026] After the token commitment is computed, at **208**, the sender **110***a* may provide or publish, anonymously and using the computing node **102***a*, the Z-token and/or a hash of the asset token A, H(A), to a self-executing code or smart contract on the ZKP-enabled DLN **100** to have the token commitment (i.e., Z-token) minted or registered for the first time on the ZKP-enabled DLN **100**. Prior to the Z-token being included in the distributed ledgers **104***a*-**104***e* of the ZKP-enabled DLN **100** as a representation of the off-chain asset (e.g., vehicle asset **112**) on the ZKP-enabled DLN **100**, however, the sender **110***a* may have to demonstrate to the ZKP-enabled DLN **100** that (1) the Z-token in fact includes the asset token A, and/or (2) the asset is not already represented on the ZKP-enabled DLN **100**, i.e., a Z-token is

not already minted for the asset on the ZKP-enabled DLN **100** (e.g., to avoid "double minting," which can lead to undesirable "double spend" or "double transfer" on the ZKP-enabled DLN **100** of multiple token commitments all representing the same asset), according to some embodiments. In some implementations, the sender **110***a* may generate and provide anonymously to the smart contract, using the computing node **102***a*, a ZKP that the Z-token includes the asset token A. Further, the ZKP may also include a proof that a hash of the asset token A, H(A), includes the asset token A. In some implementations, the hash H(A) can be used by the smart contract to verify that the asset is not already represented on the ZKP-enabled DLN **100**. That is, as discussed below, H(A) can be used to prevent undesirable "double spend" by prohibiting a future attempt to mint or register a new token commitment for the asset identified off-chain by the asset token A while another valid token commitment for the same asset (i.e., the asset identified by the asset token A) exists on the ZKP-enabled DLN **100**. In other words, in some embodiments, H(A) can be used to prevent the minting of a new token commitment if there is an existing token commitment representing, on the ZKP-enabled DLN **100**, the asset identified off-chain by the asset token A.

[0027] In some embodiments, if a token commitment (Z-token) representing the asset (as identified by the asset token A) already exists on the ZKP-enabled DLN **100**, then a new token commitment Z' representing the same asset can be generated on the ZKP-enabled DLN **100** only upon the nullification or invalidation of the existing token commitment Z, as discussed below (by nullified or invalidated, in some embodiments, it is meant, without limitations, that the existing token commitment is no longer valid to represent the asset on the DLN **100** (e.g., the smart contract would reject the token commitment if it were provided to it as a representation of the asset)).

[0028] In some embodiments, the sender **110***a*, using the computing node **102***a*, provides the ZKP to the smart contract without having revealed A to the ZKP-enabled DLN **100** (i.e., without exposing A to the participants of the blockchain), thereby protecting the identity of the physical asset being transacted between the sender **110***a* and the recipient **110***b*. The hashing of the asset token A also allows the sender **110***a* to hide the identity of the asset token A (and hence the asset itself) from the ZKP-enabled DLN **100** or the smart contract (and hence from the other participants on the ZKP-enabled DLN **100**).

[0029] Upon receiving the token commitment Z, the hash of the asset token H(A), and/or the ZKPs, in some embodiments, the self-executing code or smart contract may verify the ZKPs. For example, the smart contract may obtain or retrieve a public input and/or a verification key (e.g., from the sender **110***a*) and compute the ZKPs to verify statements included in the ZKPs, such as the statements that H(A) includes A (i.e., the statement that H(A) is obtained by applying a hashing function or algorithm on the asset token A) and the statement that the token commitment Z also includes A (i.e., the statement that Z is obtained by applying a hashing function or algorithm on the asset token A). Further, the smart contract may verify that there has never been an H(A) provided to the smart contract previously (e.g., if the asset has never been represented on the ZKP-enabled DLN **100**). For example, at **210**, the ZKP-enabled DLN **100** may include a double-spend preventer data structure that includes all the hashes of asset tokens that have been provided to the smart contract previously. In such embodiments, the smart contract may check the double-spend preventer data structure for the presence of a hash of the asset token A, and if there is one, this may be understood as the asset identified off-chain by the asset token A has already been minted or registered on the ZKP-enabled DLN **100**, and as such, the smart contract may reject the token commitment Z provided by the sender **110***a*, and prevent its inclusion into a commitments data structure on the ZKP-enabled DLN **100**. In some embodiments, the commitments data structure includes token commitments (e.g., a token commitment such as the Z-token $Z=H(S \circledast P_k \circledast A)$ that serve as a representation, on the ZKP-enabled DLN **100**, of an asset identified by the asset token A), the token commitment Z having been included into the commitments data structure after the smart contract verifies that the double-spend preventer data structure does not contain the hashes of the asset tokens (e.g., after the smart contract verifies that H(A) is not included in the double-spend preventer data structure). As such, the double-spend preventer data structure can be used to prevent the undesirable problem of "double spend," where a user of the ZKP-enabled DLN **100** may mint or generate two (or in general, multiple) token commitments $Z_n=H(S_n \circledast P_k \circledast A)$ for a single asset identified by A, and attempt to transfer the two (or multiple) Z_n to different entities (which is what a "double spend" is, since there is only a single underlying asset for the multiple transfers). Once there is a H(A) in the double-spend preventer data structure, in some implementations, the smart contract would not allow adding, into the commitments data structure, a new token commitment representing the asset identified by the asset token A. That is, the smart contract would not allow the minting of a new token commitment $Z=H(S \circledast P_k \circledast A)$ for an asset identified by the asset token A if H(A) is present in the double-spend preventer data structure. In some embodiments, the double-spend preventer data structure and/or the commitments data structure may be stored on the ZKP-enabled DLN **100** (i.e., these data structures may be stored on storage systems that are linked to or part of the computing nodes **102***a*-**102***e* that make up the ZKP-enabled DLN **100**). As noted above, the combining operator $\circledast$ may include the XOR operator $\oplus$, the concatenation operator |, and/or the like.

[0030] In some embodiments, the smart contract may discover that the hash of the asset token A, H(A), is not in the double-spend preventer data structure. In such embodiments, the smart contract may add H(A) into double-spend preventer data structure and allow the addition of the token commitment into the commitments data structure on the ZKP-enabled DLN **100**. The addition of the token commitment into the commitments data structure, at **212**, may signify the representation or registration of the asset on the ZKP-enabled DLN **100**. Further, since the token commitment includes an identifier (e.g., a public identifier) of the sender **110***a* on the ZKP-enabled DLN **100** (e.g., public key of the sender **110***a*), in some implementations, the token commitment also serves as a notice or a record of the ownership of the asset (e.g., ownership belonging to the entity that is behind the public key on the ZKP-enabled DLN **100**, i.e., the sender **110***a*). It is to be noted, however, that, in some embodiments, other participants on the ZKP-enabled DLN **100** **100** may not be privy to A (i.e., the identity of the asset being transacted) and/or the owner/sender **110***a*

of the asset. That is, the privacy of the sender **110***a* (as it related to ownership and the identity of the asset, for example) can be protected as a result of the use of ZKP in the methods and systems disclosed herein.

[0031] FIG. **3** shows a flow chart illustrating the generation of new token commitment on the distributed ledger to represent the transfer of a real-world or physical asset from a sender to a recipient, according to some embodiment. At **302**, the new token commitment may be generated in response to a request to represent the asset transfer on the ZKP-enabled DLN **100**. Once the asset **112** (e.g., non-fungible asset) is represented on the ZKP-enabled DLN **100** as discussed above, at **304**, the transfer of the asset **112** from the sender **110***a* to the recipient **110***b* can be represented on the ZKP-enabled DLN **100** by the generation of a new non-fungible token commitment that is linked or associated with the recipient **110***b*, and the nullification or invalidation of the existing token commitment Z that included the identifier (e.g., public key) of sender **110***a*. For example, the new non-fungible token commitment (the "recipient token commitment") can be generated by an application of a hashing function or algorithm on an identifier (e.g., public identifier) of the recipient **110***b*, an example of such identifier including the public key of the recipient on the ZKP-enabled DLN **100**. Further, since the recipient token commitment should represent the same off-chain asset **112** as the existing token commitment (the "sender token commitment", which was obtained by an application of a hashing function on the asset token A), in some implementations, the application of the hashing function to obtain the recipient token commitment may also include applying the hashing function on the same asset token A. In addition, in some implementations, the hashing function may also be applied on a random nonce for reasons discussed above with reference to the computation of the sender token commitment. In some implementations, the random nonce used to generate the sender token commitment Z may be different from the random nonce that would be used to generate the recipient token commitment.

[0032] An example implementation of the generation of a non-fungible recipient token commitment as discussed above can include the computation of a recipient Z'-token as follows: $Z'=H(S' \circledast P_k' \circledast A)$, where A is the asset token used in generating the sender token commitment Z, P_k' is the public key on the ZKP-enabled DLN **100** that is associated with the recipient **110***b*, S' is a random nonce, H is a cryptographic hashing function or algorithm (e.g., SHA-256), and $\circledast$ represents a combining operator (e.g., the XOR operator $\oplus$, the concatenation operator |, etc.). In some embodiments, S' may be different from S. In some embodiments, the computation of the recipient token commitment Z' may include application of the hashing function on additional elements besides or instead of S', P_k' and A. In some embodiments, the recipient token commitment Z' may comprise or consist S', P_k' and A. In some embodiments, the recipient token commitment Z' may be generated by the sender **110***a* and provided, via the computing node **102***a*, to the smart contract of the ZKP-enabled DLN **100** anonymously. Further, the sender **110***a* may secretly provide the recipient **110***b* the random nonce S' and/or the asset token A (i.e., without divulging or revealing S' and/or A ZKP-enabled DLN **100** (i.e., to the public or the other participants of the ZKP-enabled DLN **100**)).

[0033] Before the smart contract can allow the addition of the recipient token commitment Z' onto the commitments data structure, thereby passing the representation of the ownership of the asset **112** on the ZKP-enabled DLN **100** from the sender **110***a* to the recipient **110***b*, in some embodiments, the sender **110***a* may have to demonstrate to the smart contract that the Z token (i.e., the sender token commitment) belongs to the sender **110***a* (signifying that the asset **112** belongs to the sender **110***a*). Further, the sender **110***a* would have to demonstrate to the smart contract that a new token commitment, the recipient token commitment Z', representing the same asset **112** but assigning ownership to the recipient **110***b*, has been generated and that the sender token commitment, which is already on the commitments data structure, has been nullified or invalidated. These various demonstrations are described below.

[0034] In some embodiments, to demonstrate to the smart contract that the sender token commitment Z belongs to the sender **110***a*, the sender **110***a* can provide the smart contract anonymously a ZKP that the sender **110***a* knows that the token commitment Z is obtained by an application of a hashing function on a combination of a random nonce, an asset identifier of the off-chain asset that is being transacted and/or an identifier associated with the sender **110***a* on the ZKP-enabled DLN **100** such as but not limited to a public identifier. For example, the ZKP can include a proof that the sender **110***a* has knowledge that Z is obtained by applying a hashing function or algorithm on a combination of a random nonce S, an asset token A that can be used as an identifier of the asset **112**, and/or the public key of the sender **110***a* on the ZKP-enabled DLN **100**. As a specific example, the ZKP can include a proof that Z is obtained by the computation $H(S \circledast P_k \circledast A)$, where the combining operator $\circledast$ may include the XOR operator $\oplus$, the concatenation operator |, and/or the like.

[0035] In some embodiments, providing a proof that the sender **110***a* knows that the token commitment Z is obtained by an application of a hashing function on a combination of a random nonce, an asset identifier of the off-chain asset and/or an identifier associated with the sender **110***a* on the blockchain may not be sufficient as a proof that the sender token commitment Z belongs to the sender **110***a*, since there could be other participants of the ZKP-enabled DLN **100** that can have the stated information. For example, in some embodiments, the asset **112** may not have been minted or represented on the ZKP-enabled DLN **100** initially by the sender **110***a*, but rather by a prior owner or sender (not shown) that then transferred the asset to the sender **110***a*. In such cases, the prior sender may have been the one that generated the asset token A (off-chain, for example) and represented the transfer of the asset **112** from the prior sender to the sender **110***a* on the ZKP-enabled DLN **100** by having the smart contract on the ZKP-enabled DLN **100** add the token commitment $Z=H(S \circledast P_k \circledast A)$ to the token commitments data structure, where P_k is the public key of the sender **110***a* that is receiving the asset **112** from the prior owner. In such embodiments, the prior owner would have knowledge or possession of S, P_k and/or A, and as such can generate similar or same ZKP as one generated by the sender **110***a* and provided to the smart contract to represent the transfer of the asset to the recipient **110***b*. As such, to demonstrate that the sender **110***a* is the rightful (e.g., current) owner of the asset, in some implementations, the sender **110***a* may also provide to the smart contract, via the

computing node **102***a*, a ZKP that the sender **110***a* can generate the public identifier associated with the sender **110***a* on the ZKP-enabled DLN **100** from a corresponding secret identifier associated with the sender **110***a* on the ZKP-enabled DLN **100**. For example, the public identifier associated with the sender **110***a* can be the public key of the sender **110***a*, and the sender **110***a* can provide the smart contract a ZKP that the sender **110***a* can derive or obtain the public key P_k from the private key V_k of the sender **110***a*. As the private key V_k is known only to the sender **110***a*, at least nominally, in such implementations, the prior sender or any other party or participant of the ZKP-enabled DLN **100** may not be able to generate such ZKP. As such, in some embodiments, the sender's claim that the sender token commitment Z belongs to the sender **110***a* may be proved by verifying the ZKP, generated and provided to the smart contract by the sender **110***a*, that the sender **110***a* has knowledge that Z can be obtained by computing $H(S \circledast P_k \circledast A)$ and that P_k can be obtained from V_k.

[0036] In some embodiments, the sender **110***a* may also have to demonstrate to the smart contract that the sender token commitment Z is no longer valid before the smart contract can allow the addition of the recipient token commitment Z' onto the commitments data structure. The smart contract may enforce this condition to avoid a "double-spend" by the sender **110***a*, where the sender **110***a* can generate a plurality of token commitments for the same off-chain asset **112** using the same asset token A but different random nonces S and different public keys of other participants **110***c*-**110***e* in the ZKP-enabled DLN **100**. In some embodiments, "double spend" by the sender **110***a* can be prevented by having the sender **110***a* generate and provide to the smart contract, via the computing node **102***a*, a nullifier that nullifies the sender token commitment that is already on the token commitments data structure. By requiring the sender **110***a* to nullify an existing valid token commitment Z (i.e., a token commitment that is stored in the token commitments data structure) prior to the addition of a new token commitment Z' into the token commitments data structure, in some implementations, the smart contract prevents the "double spend" problem, since the sender **110***a* can nullify Z only once (hence only one Z' can be added into the commitments data structure, i.e., no "double spend").

[0037] In some embodiments, the nullifier can be constructed or generated out of the random nonce S that was used to generate the token commitment Z. The random nonce S, however, may be known to other participants of the ZKP-enabled DLN **100** (i.e., besides the sender **110***a*), such as a previous owner of the asset. To demonstrate to the smart contract that the nullifier is in fact constructed or generated by the sender **110***a* (and not by a previous owner of the asset **112**, for example), in some embodiments, the sender **110***a* may include in the nullifier a secret element or identifier that is known only to the sender **110***a*. For example, in some embodiments, the nullifier can be computed via an application of a hashing function H on the random nonce S and the private key of the sender **110***a*, V_k, as follows: $N=H(S \circledast V_k)$, where the combining operator $\circledast$ may include the XOR operator $\oplus$, the concatenation operator |, and/or the like.

[0038] At **306**, the sender **110***a* may provide the nullifier N to the smart contract, via the computing node **102***a*, along with a ZKP that the sender **110***a* knows N is obtained via an application of the hashing function H on the random nonce S and the private key V_k. In some implementations, the hashing allows the sender **110***a* to hide the identity of the random nonce S and/or the private key V_k, and the ZKP allows the sender **110***a* to convince the smart contract, if the proof is verified, that N includes S and V_k, without the sender **110***a* having to reveal S and V_k themselves to the smart contract or the participants of the blockchain **100**. In some embodiments, the ZKP-enabled DLN **100** may include a nullifier data structure that includes all the nullifiers that have been provided to the smart contract. For example, the nullifier data structure may be stored on the ZKP-enabled DLN **100** (i.e., the data structure may be stored on storage systems that are linked to or part of the computing nodes **102***a*-**102***e* that make up the ZKP-enabled DLN **100**). In such embodiments, the smart contract may check to see if the nullifier N provided by the sender **110***a* is already present in the nullifier data structure, and if so, may reject the addition of the recipient token commitment Z' onto the commitments data structure as the presence of N in the nullifier data structure indicates that the sender token commitment Z has already been nullified (which indicates that the sender **110***a* does not own the asset **112**, and as such cannot be in the position to transfer the asset **112** to the recipient **110***b*).

[0039] Before the smart contract can allow the addition of the recipient token commitment Z' onto the commitments data structure, in some embodiments, the sender **110***a* may also have to demonstrate to the smart contract that the recipient token commitment Z' includes an identifier associated with the recipient (e.g., a public identifier such as the public key P_k' of the recipient), an identifier that can be used to identify the asset (e.g., the asset token A) and/or the random nonce S'. To accomplish this goal without revealing identifying information about the public key P_k', the asset token A and/or the random nonce S' on the ZKP-enabled DLN **100**, at **308**, the sender **110***a* may generate and provide to the smart contract, via the computing node **102***a*, a ZKP that Z' includes, or is generated using, the identifier associated with the recipient (e.g., the public key P_k' of the recipient), the asset identifier (e.g., the asset token A) and/or the random nonce S'. In particular, the ZKP includes the proof that the asset token in the recipient token commitment Z' is the same asset token as the asset token in the sender token commitment Z, which demonstrates to the smart contract, when verified, that the sender token commitment Z and the recipient token commitment Z' represent the same asset on the ZKP-enabled DLN **100** that is being transacted between the sender **110***a* and the recipient **110***b*.

[0040] Upon receiving the above-identified ZKPs, in some embodiments, the smart contract may verify the ZKPs and/or check that the nullifier N is not already included in the nullifier data structure of the ZKP-enabled DLN **100**. As N is configured to nullify the sender token commitment Z upon addition onto the nullifier data structure, in some implementations, its prior presence on nullifier data structure would indicate that Z has already been nullified, which would cause the smart contract to reject the addition of the recipient token commitment Z' onto the commitments data structure. In some embodiments, upon verifying that the nullifier N is not included in the nullifier data structure, the smart contract may add the nullifier into the nullifier data structure (after all the ZKPs are verified, for example).

[0041] As discussed above, the ZKPs provided by the sender **110***a* include (a) the ZKP that the nullifier N is

obtained via an application of a hashing function or algorithm on a random nonce and/or a private key on the ZKP-enabled DLN **100** of the sender **110***a*, (b) the ZKP that the sender token commitment Z is obtained via an application of a hashing function on the same random nonce used in generating the nullifier, a public identifier on the ZKP-enabled DLN **100** of the sender **110***a* (e.g., the public key of the sender **110***a*) and/or the asset token, (c) the ZKP that the sender **110***a* can generate or derive the public identifier associated with the sender **110***a* from a secret identifier associated with the sender **110***a* (e.g., the sender **110***a* can derive its public key from its private key), and/or the (d) ZKP that the recipient token commitment Z' is obtained via an application of a hashing function on a random nonce (e.g., different from the random nonce used to generate Z), a public identifier on the ZKP-enabled DLN **100** of the recipient **110***b* (e.g., the public key of the recipient **110***b*) and/or the same asset token used in generating or computing the sender token commitment Z. After verifying one or more of the above-identified ZKPs, at **310**, the smart contract may allow the recipient token commitment Z' to be added onto the commitments data structure of the ZKP-enabled DLN **100**, which signifies the representation, on the ZKP-enabled DLN **100**, of the transfer of the asset **112** from the sender **110***a* to the recipient **110***b*.

[0042] In some embodiments, if the recipient **110***b* wishes to transfer the asset **112** to another participant **110***c* after receiving it from the sender **110***a*, the recipient **110***b* can carry out all or nearly all the same steps undertaken by the sender **110***a* to represent the transfer transaction on the ZKP-enabled DLN **100** between the sender **110***a* and the recipient **110***b*. The recipient **110***b* may not, however, be able to generate an asset token to identify the asset **112**, as the presence of the hash of the asset token H(A) in the preventer data structure would cause the smart contract to reject the token commitments the recipient **110***b* would have to provide to the smart contract to represent the transfer of the asset to the new recipient **110***c*. As such, once an asset is minted on the blockchain **100** for the first time (and identified with an asset token A), it cannot be minted again, but can only be transferred from one participant of the ZKP-enabled DLN **100** to another via the generation of a token commitment that includes the same asset token A that was created during the initial minting or representation of the asset **112** on the ZKP-enabled DLN **100**. As such, the double-spend preventer data structure can be used to prevent "double-spend," where multiple token commitments Z are created for the same asset **112**, and are transferred to a plurality of participants.

[0043] As discussed above, with the use of ZKPs, a sender **110***a* can represent on the ZKP-enabled DLN **100** a physical, off-chain asset **112**, without having to disclose or reveal to other participants of the ZKP-enabled DLN **100** (or to the public) any identifying information about the asset **112** (e.g., without revealing asset token A obtained by hashing identifying parameters of the asset **112** such as serial numbers, model numbers, asset name, etc.). Further, the identity of the sender **110***a* can also remain hidden from the blockchain **100**. For example, in minting a token commitment Z on the ZKP-enabled DLN **100** for the asset **112**, the sender **110***a* provides to the smart contract anonymously a hash of A, H(A), and $Z=H(S \circledast P_k \circledast A)$, which conceal or cloak the identity of the sender **110***a*, the random nonce S, the public key P_k, the serial token A (which can be obtained by hashing some identifying information of the asset **112**), etc. The smart contract can verify the provided ZKPs, and allow the token commitment Z to be added onto the token commitments data structure, without having access to any of these information (including the identity of the sender **110***a*, as H(A) and Z are provided by the sender **110***a* anonymously). As such, the use of ZKPs in the ZKP-enabled DLN **100** allows for the representation of an asset **112** on the ZKP-enabled DLN **100** while preserving the confidentiality or privacy of participants or users of the ZKP-enabled DLN **100** (such as sender **110***a*) and their assets (such as the vehicle **112**).

[0044] Further, the use of ZKPs also allows participants to represent, on the ZKP-enabled DLN **100**, their off-chain asset **112** transactions without revealing or exposing the transaction participants as well as details of the transaction itself on the ZKP-enabled DLN **100** (i.e., to the other participants of the ZKP-enabled DLN **100**, which may include the general public). As noted above, the sender **110***a* provides to the smart contract anonymously (and hence hidden from at least the other participants **110***c*-**110***e*) the sender token commitment $Z=H(S \circledast P_k \circledast A)$, the recipient token commitment $Z'=H(S' \circledast P_k' \circledast A)$ and the nullifier $N=H(S \circledast S_k)$, without revealing S, P_k, S', P_k', A and/or S_k, thereby protecting the identities of the sender **110***a*, the recipient **110***b* and/or the asset identified by A. As noted above, the combining operator $\circledast$ may include the XOR operator $\oplus$, the concatenation operator |, and/or the like. In some implementations, the identity of the sender **110***a* can be protected as the sender **110***a* communicates with the smart contract anonymously. The smart contract can, without having access to any of these information, verify the provided ZKPs and allow (i) the addition of the recipient token commitment Z' onto the token commitments data structure and (ii) the nullification of the sender token commitment Z. As such, the use of ZKPs in the on the ZKP-enabled DLN **100** allows for the representation of a transaction including the transfer of the asset **112** on the blockchain while preserving the confidentiality or privacy of the participants of the transaction (such as sender **110***a* and recipient **110***b*) as well as the assets (such as the vehicle **112**) and the transaction itself (e.g., random nonces, asset tokens, etc. generated during the transaction).

[0045] Some embodiments of the current disclosure include a method comprising: sending a request to cause an asset identifiable by an identifying parameter be registered on a distributed ledger-based network (DLN); and receiving, in response to the request, a confirmation confirming the registration of the asset on the DLN, the confirmation being issued after verification of a zero-knowledge proof (ZKP), provided by a prover, that the prover has knowledge of an identity of an asset token that when a hashing function is applied to the asset token, a token commitment representing the asset on the DLN is generated.

[0046] In some embodiments, the hashing function is a first hashing function; and the asset token is obtained via an application of a second hashing function on the identifying parameter of the asset.

[0047] In some embodiments, the hashing function is a first hashing function; the ZKP includes the ZKP that the prover of the ZKP has knowledge of the identity of the asset token that when a second hashing function is applied to the asset token, a first hash value is generated; the verification of the ZKP includes verifying that the first hash value is not

stored in a double-spend preventer data structure on the DLN prior to the issuance of the confirmation.

[0048] In some embodiments, the hashing function is a first hashing function; the ZKP includes the ZKP that the prover of the ZKP has knowledge of the identity of the asset token that when a second hashing function is applied to the asset token, a first hash value is generated; the confirmation is issued after the first hash value is added onto a double-spend preventer data structure on the DLN after a verification that the first hash value is not stored in the double-spend preventer data structure.

[0049] In some embodiments, the application of the hashing function includes the application of the hashing function on an identifier associated with an owner of the token commitment and/or a random nonce, the identifier including a public key of the owner on the DLN.

[0050] In some embodiments, the ZKP and/or the verification of the ZKP do not reveal the identifying parameter of the asset and/or a random nonce used to generate the token commitment on the DLN.

[0051] In some embodiments, the confirmation is issued without revealing any identifying information of the asset, an owner of the asset, and/or a random nonce used to generate the token commitment on the DLN.

[0052] Some embodiments of the current disclosure include a method comprising: receiving a request that is configured to cause a transfer of an asset from a sender to a recipient, the asset represented on a distributed ledger-based network (DLN) by a first token commitment; and causing, in response to the request and on the DLN, a registration of the transfer of the asset from the sender to the recipient, the registration occurring after verification of a zero-knowledge proof (ZKP), provided by a prover, that the prover has knowledge of an identity of an asset token, (1) the first token commitment obtained via an application of a first hashing function on the asset token, and/or (2) a second token commitment representing the asset on the DLN obtained via an application of a second hashing function on the asset token.

[0053] In some embodiments, the ZKP includes the ZKP that the prover has knowledge of an identity of: (a) a first identifier associated with the sender, the first token commitment obtained via the application of the first hashing function on the first identifier, and/or (b) a second identifier associated with the recipient, the second token commitment obtained via the application of the second hashing function on the second identifier.

[0054] In some embodiments, the ZKP includes the ZKP that the prover has knowledge of an identity of: (a) a first identifier associated with the sender, the first token commitment obtained via the application of the first hashing function on the first identifier, the first identifier including a public key of the sender on the DLN, and/or (b) a second identifier associated with the recipient, the second token commitment obtained via the application of the second hashing function on the second identifier, the second identifier including a public key of the recipient on the DLN.

[0055] In some embodiments, the ZKP includes the ZKP that the prover is capable of deriving a first identifier associated with the sender from a secret identifier associated with the sender, the first identifier and the secret identifier including a public key and a private key, respectively, of the sender on the DLN.

[0056] In some embodiments, the registration of the transfer occurs after a verification that a nullifier is not stored in a nullifier data structure on the DLN, a presence of the nullifier in the nullifier data structure indicating invalidity of the first token commitment.

[0057] In some embodiments, the registration of the transfer occurs after a nullifier is added into a nullifier data structure on the DLN after a verification that the nullifier is not stored in the nullifier data structure, a presence of the nullifier in the nullifier data structure indicating invalidity of the first token commitment.

[0058] In some embodiments, the ZKP includes the ZKP that the prover has knowledge of an identity of a nullifier obtained via an application of a third hashing function on a random nonce and/or a secret identifier associated with the sender, a presence of the nullifier in a nullifier data structure on the DLN indicating invalidity of the first token commitment.

[0059] In some embodiments, the ZKP includes the ZKP that the prover has knowledge of an identity of a nullifier obtained via an application of a third hashing function on a random nonce and/or a secret identifier associated with the sender, a presence of the nullifier in a nullifier data structure on the DLN indicating invalidity of the first token commitment, the application of the first hashing function including the application of the first hashing function on the random nonce.

[0060] In some embodiments, the ZKP includes the ZKP that the prover has knowledge of an identity of a nullifier obtained via an application of a third hashing function on a random nonce and/or a secret identifier associated with the sender, a presence of the nullifier in a nullifier data structure on the DLN indicating invalidity of the first token commitment, the secret identifier including the private key of the sender.

[0061] In some embodiments, the verification of the ZKP occurs without any identifying information of the asset including the identifying parameter of the asset being revealed on the DLN.

[0062] In some embodiments, the verification of the ZKP occurs without any identifying information of the sender and/or the recipient being revealed, the identifying information of the sender and/or the recipient including a public key of the sender, a private key of the sender, a public key of the recipient and/or a private key of the recipient, on the DLN.

[0063] In some embodiments, the verification of the ZKP occurs without any identifying information of the first token commitment and/or a random nonce being revealed on the DLN.

[0064] While various embodiments have been described and illustrated herein, one will readily envision a variety of other means and/or structures for performing the function and/or obtaining the results and/or one or more of the advantages described herein, and each of such variations and/or modifications is deemed to be within the scope of the embodiments described herein. More generally, one will readily appreciate that all parameters, dimensions, materials, and configurations described herein are meant to be exemplary and that the actual parameters, dimensions, materials, and/or configurations will depend upon the specific application or applications for which the teachings is/are used. One will recognize, or be able to ascertain using no more than routine experimentation, many equivalents to the specific embodiments described herein. It is, therefore, to be

understood that the foregoing embodiments are presented by way of example only and that, within the scope of the disclosure, including the appended claims and equivalents thereto, disclosed embodiments may be practiced otherwise than as specifically described and claimed. Embodiments of the present disclosure are directed to each individual feature, system, tool, element, component, and/or method described herein. In addition, any combination of two or more such features, systems, articles, elements, components, and/or methods, if such features, systems, articles, elements, components, and/or methods are not mutually inconsistent, is included within the scope of the present disclosure.

[0065] The above-described embodiments can be implemented in any of numerous ways. For example, embodiments may be implemented using hardware, software or a combination thereof. When implemented in software, the software code can be stored (e.g., on non-transitory memory) and executed on any suitable processor or collection of processors, whether provided in a single computer or distributed among multiple computers.

[0066] Further, it should be appreciated that a computer may be embodied in any of a number of forms, such as a rack-mounted computer, a desktop computer, a laptop computer, netbook computer, or a tablet computer. Additionally, a computer may be embedded in a device not generally regarded as a computer but with suitable processing capabilities, including a smart phone, smart device, or any other suitable portable or fixed electronic device.

[0067] Also, a computer can have one or more input and output devices. These devices can be used, among other things, to present a user interface. Examples of output devices that can be used to provide a user interface include printers or display screens for visual presentation of output and speakers or other sound generating devices for audible presentation of output. Examples of input devices that can be used for a user interface include keyboards, and pointing devices, such as mice, touch pads, and digitizing tablets. As another example, a computer can receive input information through speech recognition or in other audible format.

[0068] Such computers can be interconnected by one or more networks in any suitable form, including a local area network or a wide area network, such as an enterprise network, and intelligent network (IN) or the Internet. Such networks can be based on any suitable technology and can operate according to any suitable protocol and can include wireless networks, wired networks or fiber optic networks.

[0069] The various methods or processes outlined herein can be coded as software that is executable on one or more processors that employ any one of a variety of operating systems or platforms. Additionally, such software can be written using any of a number of suitable programming languages and/or programming or scripting tools, and also can be compiled as executable machine language code or intermediate code that is executed on a framework or virtual machine.

[0070] In this respect, various disclosed concepts can be embodied as a computer readable storage medium (or multiple computer readable storage media) (e.g., a computer memory, one or more floppy discs, compact discs, optical discs, magnetic tapes, flash memories, circuit configurations in Field Programmable Gate Arrays or other semiconductor devices, or other non-transitory medium or tangible computer storage medium) encoded with one or more programs that, when executed on one or more computers or other processors, perform methods that implement the various embodiments of the disclosure discussed above. The computer readable medium or media can be transportable, such that the program or programs stored thereon can be loaded onto one or more different computers or other processors to implement various aspects of the present disclosure as discussed above.

[0071] The terms "program" or "software" are used herein in a generic sense to refer to any type of computer code or set of computer-executable instructions that can be employed to program a computer or other processor to implement various aspects of embodiments as discussed above. Additionally, it should be appreciated that according to one aspect, one or more computer programs that when executed perform methods of the present disclosure need not reside on a single computer or processor, but can be distributed in a modular fashion amongst a number of different computers or processors to implement various aspects of the disclosure.

[0072] Computer-executable instructions can be in many forms, such as program modules, executed by one or more computers or other devices. Generally, program modules include routines, programs, objects, components, data structures, etc. that perform particular tasks or implement particular abstract data types. Typically the functionality of the program modules can be combined or distributed as desired in various embodiments.

[0073] Also, data structures can be stored in computer-readable media in any suitable form. For simplicity of illustration, data structures may be shown to have fields that are related through location in the data structure. Such relationships can likewise be achieved by assigning storage for the fields with locations in a computer-readable medium that convey relationship between the fields. However, any suitable mechanism can be used to establish a relationship between information in fields of a data structure, including through the use of pointers, tags or other mechanisms that establish relationship between data elements.

[0074] Also, various concepts can be embodied as one or more methods, of which an example has been provided. The acts performed as part of the method may be ordered in any suitable way. Accordingly, embodiments can be constructed in which acts are performed in an order different than illustrated, which can include performing some acts simultaneously, even though shown as sequential acts in illustrative embodiments. All publications, patent applications, patents, and other references mentioned herein are incorporated by reference in their entirety.

[0075] All definitions, as defined and used herein, should be understood to control over dictionary definitions, definitions in documents incorporated by reference, and/or ordinary meanings of the defined terms.

[0076] The indefinite articles "a" and "an," as used herein in the specification and in the claims, unless clearly indicated to the contrary, should be understood to mean "at least one."

[0077] The phrase "and/or," as used herein in the specification and in the claims, should be understood to mean "either or both" of the elements so conjoined, i.e., elements that are conjunctively present in some cases and disjunctively present in other cases. Multiple elements listed with "and/or" should be construed in the same fashion, i.e., "one or more" of the elements so conjoined. Other elements may optionally be present other than the elements specifically

identified by the "and/or" clause, whether related or unrelated to those elements specifically identified. Thus, as a non-limiting example, a reference to "A and/or B", when used in conjunction with open-ended language such as "comprising" can refer, in one embodiment, to A only (optionally including elements other than B); in another embodiment, to B only (optionally including elements other than A); in yet another embodiment, to both A and B (optionally including other elements); etc.

[0078] As used herein, "or" should be understood to have the same meaning as "and/or" as defined above. For example, when separating items in a list, "or" or "and/or" shall be interpreted as being inclusive, i.e., the inclusion of at least one, but also including more than one, of a number or list of elements, and, optionally, additional unlisted items. Only terms clearly indicated to the contrary, such as "only one of" or "exactly one of," or, when used in claims, "consisting of," will refer to the inclusion of exactly one element of a number or list of elements. In general, the term "or" as used herein shall only be interpreted as indicating exclusive alternatives (i.e. "one or the other but not both") when preceded by terms of exclusivity, such as "either," "one of," "only one of," or "exactly one of" "Consisting essentially of," when used in claims, shall have its ordinary meaning as used in the field of patent law.

[0079] As used herein, the phrase "at least one," in reference to a list of one or more elements, should be understood to mean at least one element selected from any one or more of the elements in the list of elements, but not necessarily including at least one of each and every element specifically listed within the list of elements and not excluding any combinations of elements in the list of elements. This definition also allows that elements may optionally be present other than the elements specifically identified within the list of elements to which the phrase "at least one" refers, whether related or unrelated to those elements specifically identified. Thus, as a non-limiting example, "at least one of A and B" (or, equivalently, "at least one of A or B," or, equivalently "at least one of A and/or B") can refer, in one embodiment, to at least one, optionally including more than one, A, with no B present (and optionally including elements other than B); in another embodiment, to at least one, optionally including more than one, B, with no A present (and optionally including elements other than A); in yet another embodiment, to at least one, optionally including more than one, A, and at least one, optionally including more than one, B (and optionally including other elements); etc.

[0080] All transitional phrases such as "comprising," "including," "carrying," "having," "containing," "involving," "holding," "composed of," and the like are to be understood to be open-ended, i.e., to mean including but not limited to. Only the transitional phrases "consisting of" and "consisting essentially of" shall be closed or semi-closed transitional phrases, respectively, as set forth in the United States Patent Office Manual of Patent Examining Procedures, Section 2111.03.

1. A method, comprising:

receiving a request that is configured to cause a registration of a token commitment on a distributed ledger-based network (DLN), the token commitment representing an asset on the DLN and the request including an identifying parameter of the asset;

providing by a provider, in response to the request and to a self-executing code segment on the DLN, a zero-knowledge proof (ZKP) that the provider has knowledge of an identity of an asset token where when a hashing function is applied to the asset token, the token commitment is generated; and

receiving, after verification of the ZKP by the self-executing code segment, a confirmation confirming the registration of the token commitment on the DLN including an addition of the token commitment onto a token commitments data structure on the DLN.

2. The method of claim **1**, wherein:

the hashing function is a first hashing function;

the asset token is obtained via an application of a second hashing function on the identifying parameter of the asset.

3. The method of claim **1**, wherein:

the hashing function is a first hashing function;

the ZKP includes the ZKP that the provider of the ZKP has knowledge of the identity of the asset token that when a second hashing function is applied to the asset token, a first hash value is generated;

the token commitment is registered after the self-executing code segment verifies that the first hash value is not stored in a double-spend preventer data structure on the DLN prior to the registration of the token commitment.

4. The method of claim **1**, wherein:

the hashing function is a first hashing function;

the ZKP includes the ZKP that the provider of the ZKP has knowledge of the identity of the asset token that when a second hashing function is applied to the asset token, a first hash value is generated;

the token commitment is registered after the self-executing code segment adds the first hash value into a double-spend preventer data structure on the DLN after verifying that the first hash value is not stored in the double-spend preventer data structure.

5. The method of claim **1**, wherein the token commitment represents a non-fungible token.

6. The method of claim **1**, wherein the application of the hashing function includes the application of the hashing function on an identifier associated with an owner of the token commitment and/or a random nonce.

7. The method of claim **1**, wherein the application of the hashing function includes the application of the hashing function on an identifier associated with an owner of the token commitment and/or a random nonce, the identifier including a public key of the owner on the DLN.

8. The method of claim **1**, wherein the ZKP and/or the verification of the ZKP do not reveal the identifying parameter of the asset and/or a random nonce used to generate the token commitment on the DLN.

9. The method of claim **1**, wherein the registration of the token commitment occurs without revealing any identifying information of the asset, any identifying information of the asset, and/or a random nonce used to generate the token commitment on the DLN.

10. A method, comprising:

receiving a request that is configured to cause a transfer of an asset from a sender to a recipient, the asset represented on a distributed ledger-based network (DLN) by a first token commitment;

providing by a provider, in response to the request and to a self-executing code segment on the DLN, a zero-knowledge proof (ZKP) that the provider of the ZKP has knowledge of an identity of an asset token,

(1) an application of a first hashing function on the asset token generating the first token commitment, and/or

(2) an application of a second hashing function on the asset token generating a second token commitment representing the asset on the DLN; and

receiving, upon verification of the ZKP by the self-executing code segment, a confirmation confirming an addition of the second token commitment onto a token commitments data structure on the DLN.

11. The method of claim **10**, wherein the ZKP includes the ZKP that the provider has knowledge of an identity of:

(a) a first identifier associated with the sender, the application of the first hashing function including the application of the first hashing function on the first identifier, and/or

(b) a second identifier associated with the recipient, the application of the second hashing function including the application of the second hashing function on the second identifier.

12. The method of claim **10**, wherein the ZKP includes the ZKP that the provider has knowledge of an identity of:

(a) a first identifier associated with the sender, the application of the first hashing function including the application of the first hashing function on the first identifier, the first identifier including a public key of the sender on the DLN, and/or

(b) a second identifier associated with the recipient, the application of the second hashing function including the application of the second hashing function on the second identifier, the second identifier including a public key of the recipient on the DLN.

13. The method of claim **10**, wherein the ZKP includes the ZKP that the provider has knowledge of an identity of a nullifier obtained via an application of a third hashing function on a random nonce and/or a secret identifier associated with the sender,

a presence of the nullifier in a nullifier data structure on the DLN indicating invalidity of the first token commitment.

14. The method of claim **10**, wherein the ZKP includes the ZKP that the provider has knowledge of an identity of a nullifier obtained via an application of a third hashing function on a random nonce and/or a secret identifier associated with the sender,

a presence of the nullifier in a nullifier data structure on the DLN indicating invalidity of the first token commitment,

the application of the first hashing function including the application of the first hashing function on the random nonce.

15. The method of claim **10**, wherein the ZKP includes the ZKP that the provider has knowledge of an identity of a nullifier obtained via an application of a third hashing function on the random nonce and/or a secret identifier associated with the sender,

a presence of the nullifier in a nullifier data structure on the DLN indicating invalidity of the first token commitment,

the secret identifier including the private key of the sender.

16. The method of claim **10**, wherein the ZKP includes the ZKP that the provider is capable of deriving a public identifier associated with the sender from a secret identifier associated with the sender.

17. The method of claim **10**, wherein the ZKP includes the ZKP that the provider is capable of deriving a public identifier associated with the sender from a secret identifier associated with the sender, the public identifier and the secret identifier including a public key and a private key, respectively, of the sender on the DLN.

18. The method of claim **10**, wherein the second token commitment is added onto the commitments data structure after the self-executing code segment verifies a nullifier is not stored in a nullifier data structure on the DLN, a presence of the nullifier in the nullifier data structure indicating invalidity of the first token commitment.

19. The method of claim **10**, wherein the second token commitment is added onto the commitments data structure after the self-executing code segment adds a nullifier into a nullifier data structure on the DLN after verifying that the nullifier is not stored in the nullifier data structure, a presence of the nullifier in the nullifier data structure indicating invalidity of the first token commitment.

20. The method of claim **10**, wherein the verification of the ZKP by the self-executing code segment occurs without revealing any identifying information of the asset on the DLN.

21-30. (canceled)

* * * * *