METHODS AND DEVICES FOR MITIGATING INVOICE FINANCING FRAUD

TECHNICAL FIELD

[0001] The specification relates generally to computer technologies, and more particularly, to methods and devices for mitigating invoice financing fraud.

BACKGROUND

[0002] Invoice financing is a way for businesses to borrow money against accounts receivable, including, e.g., the amounts due from customers. Invoice financing helps businesses improve cash flow, pay employees and suppliers, and reinvest in operations and growth earlier than they could if they had to wait until their customers paid their balances in full.

[0003] One of the risks associated with invoice financing is fraud. For example, some businesses may commit fraud by unilaterally producing fake invoices and attempt to borrow money using such invoices. Some businesses may commit fraud by colluding with others. Some businesses may commit fraud by borrowing against the same invoice multiple times from multiple fund providers or financiers. Fraudulent activities as such are difficult to detect and mitigate.

SUMMARY

[0004] In one aspect, a computer-implemented method for mitigating invoice financing fraud includes: receiving a first transaction from a first user, the first transaction comprising a first commitment of an invoice and a first proof for proving the first user is an issuer of the invoice, the first commitment being generated based on a first token corresponding to the first user, the first token itself being excluded

from the first transaction; verifying the first proof; receiving a second transaction from a second user, the second transaction and the first transaction containing different payload so as to unlink the second transaction and the first transaction, the second transaction comprising the first token corresponding to the first user, a second commitment of the invoice, and a second proof for proving the second user has certified the invoice, the second commitment being generated based on a second token corresponding to the second user and a certifying document generated by the second user; determining that the first token is valid and verifying the second proof; receiving a third transaction from a third user, the third transaction comprising the second token corresponding to the second user and a third proof generated by the third user; and determining that the second token is valid and verifying the third proof, to determine no invoice financing fraud.

[0005] In another aspect, a computer-implemented method for mitigating invoice financing fraud includes: receiving a first transaction from a first user, the first transaction comprising a first commitment of an invoice and a first proof for proving the first user is an issuer of the invoice, the first commitment being generated based on a first token corresponding to the first user, the first token itself being excluded from the first transaction; verifying the first proof; receiving a second transaction from a second user, the second transaction and the first transaction containing different payload so as to unlink the second transaction and the first transaction, the second transaction comprising the first token corresponding to the first user, a second commitment of the invoice, and a second proof for proving the second user is a receiver of the invoice, the second commitment being generated based on a second token corresponding to the second user; determining that the first token is valid and verifying the second proof; receiving a third transaction from a third user, the third

transaction comprising the second token corresponding to the second user, a third commitment of the invoice, and a third proof for proving the third user has reviewed and certified the invoice, the third commitment being generated based on a third token corresponding to the third user and a certifying document generated by the third user; determining that the second

token is valid and verifying the third proof; receiving a fourth transaction from a fourth user, the fourth transaction comprising the third token corresponding to the third user and a fourth proof generated by the fourth user; and determining that the third token is valid and verifying the fourth proof, to determine no invoice financing fraud.

[0006] In another aspect, a device for mitigating invoice financing fraud includes: one or more processors; and one or more computer-readable memories coupled to the one or more processors and having instructions stored thereon that are executable by the one or more processors to: receive a first transaction from a first user, the first transaction comprising a first commitment of an invoice and a first proof for proving the first user is an issuer of the invoice, the first commitment being generated based on a first token corresponding to the first user; verify the first proof; receive a second transaction from a second user, the second transaction comprising the first token corresponding to the first user, a second commitment of the invoice, and a second proof for proving the second user has certified the invoice, the second commitment being generated based on a second token corresponding to the second user and a certifying document generated by the second user; determine that the first token is valid and verify the second proof; receive a third transaction from a third user, the third transaction comprising the second token corresponding to the second user and a third proof generated by the third user; and determine that the second token is valid and verify the third proof, to determine no invoice financing fraud.

[0007] In still another aspect, a non-transitory computer-readable medium have stored therein instructions that, when executed by a processor of a device, cause the device to perform a method for mitigating invoice financing fraud. The method includes: receiving a first transaction from a first user, the first transaction comprising a first commitment of an invoice and a first proof for proving the first user

is an issuer of the invoice, the first commitment being generated based on a first token corresponding to the first user; verifying the first proof; receiving a second transaction from a second user, the second transaction comprising the first token corresponding to the first user, a second commitment of the invoice, and a second proof for proving the second user has certified the invoice, the second commitment being generated based on a second token corresponding to the second user and a certifying document generated by the second user; determining that the first token is valid and verifying the second proof; receiving a third transaction from a third user, the third transaction comprising the second token corresponding to the second user and a third proof generated by the third user; and determining that the second token is valid and verifying the third proof, to determine no invoice financing fraud.

BRIEF DESCRIPTION OF THE DRAWINGS

[0008] The accompanying drawings, which are incorporated in and constitute a part of this specification, illustrate embodiments. In the following description, which refers to the drawings, the same numbers in different drawings represent the same or similar elements unless otherwise represented.

[0009] FIG. 1 is a schematic diagram of a blockchain system, according to an embodiment.

[0010] FIG. 2 is a schematic diagram of a computing device for implementing a node in a blockchain system, according to an embodiment.

[0011] FIG. 3 is a flow chart of a method for mitigating invoice financing fraud, according an embodiment.

[0012] FIG. 4 is a flow chart of a method for mitigating invoice financing fraud, according an embodiment.

[0013] FIG. 5 is a flow chart of a method for mitigating invoice financing fraud, according an embodiment.

[0014] FIG. 6 is a flow chart of a method for mitigating invoice financing fraud, according an embodiment.

[0015] FIG. 7 is a block diagram of an apparatus for mitigating invoice financing fraud, according to an embodiment.

DETAILED DESCRIPTION

[0016] Embodiments of the specification provide methods and devices for mitigating invoice financing fraud. The methods and devices may utilize blockchain systems to validate invoices submitted by users. Fake invoices can be detected and fraudulent attempts to borrow money using such invoices can be prevented. The methods and devices may also implement a protocol that can determine whether multiple users have colluded with each other. The methods and devices may further implement a protocol that can determine whether an invoice has already been used for invoice financing. If it is determined that the invoice has already been used for invoice financing, the methods and devices can prevent additional attempts to finance using the same invoice. The methods and devices may further implement the protocol to protect user privacy. In this manner, the methods and devices can validate the invoices using the blockchain system without revealing the details of the invoices to the public.

[0017] Embodiments disclosed in the specification have one or more technical effects. In some embodiments, the methods and devices require users to submit proof in order to validate invoices submitted by the users. This allows the methods and devices to detect fake invoices and prevent fraudulent attempts to

borrow money using fake invoices. In some embodiments, the methods and devices support validation of invoices using a blockchain system. This allows the methods and devices to store invoices in a data structure that can prevent tampering and manipulation by malicious parties. In some embodiments, the methods and devices also support a protocol that can determine whether multiple users have colluded with each other. This allows the methods and devices to detect and prevent collusion using data record on the blockchain system. In some embodiments, the methods and devices further support a protocol that can calculate commitment values for the invoices to be validated using the blockchain system. A commitment value calculated for an invoice may be associated with a token and a random value. The token may be used to nullify the commitment after it is revealed by a user. This allows the methods and devices to prevent double financing using the same invoice. The random value may be used to introduce additional randomness to further protect user privacy. This allows the methods and devices to utilize the blockchain system to validate invoices without revealing any private information to the public.

[0018] Blockchain systems, also known as distributed ledger systems (DLSs) or consensus systems, may enable participating parties to store data securely and immutably. Blockchain systems may include any DLSs, without referencing any particular use case, and may be used for public, private, and consortium blockchain networks. A public blockchain network is open for all entities to use the system and participate in the consensus process. A private blockchain network is provided for a particular entity, which centrally controls read and write permissions. A consortium blockchain network is provided for a select group of entities, which control the consensus process, and includes an access control layer.

[0019] A blockchain system is implemented using a peer-to-peer (P2P)

network, in which the nodes communicate directly with each other, e.g., without the need of a fixed, central server. Each node in the P2P network may initiate communication with another node in the P2P network. A blockchain system maintains one or more blockchains.

[0020] A blockchain is a data structure that stores data, e.g., transactions, in a way that may prevent tampering and manipulation of the data by malicious parties. The transactions stored in this manner may be immutable and subsequently verified. A blockchain includes one or more blocks. Each block is linked to a previous block immediately before it in the blockchain by including a cryptographic hash of the previous block. Each block also may include a timestamp, its own cryptographic hash, and one or more transactions. The transactions, which generally have already been verified by the nodes of the blockchain system, may be hashed and encoded into a data structure, such as a Merkle tree. In a Merkle tree, data at leaf nodes of the tree is hashed, and all hashes in each branch of the tree may be concatenated at a root of the branch. This process continues up the tree to the root of the entire tree, which stores a hash that is representative of all data in the tree. A hash purporting to be of a transaction stored in the tree can be quickly verified by determining whether it is consistent with the structure of the tree.

[0021] A blockchain system includes a network of computing nodes that manage, update, and maintain one or more blockchains. The network may be a public blockchain network, a private blockchain network, or a consortium blockchain network. For example, numerous entities, such as hundreds, thousands, or even millions of entities, can operate in a public blockchain network, and each of the entities operates at least one node in the public blockchain network. Accordingly, the public blockchain network can be considered a public network with respect to the

participating entities. Sometimes, a majority of entities (nodes) must sign every block for the block to be valid and added to the blockchain of the blockchain network. Examples of public blockchain networks include particular peer-to-peer payment networks that leverage a distributed ledger, referred to as blockchain.

[0022] In general, a public blockchain network may support public transactions. A public transaction is shared with all of the nodes in the public blockchain network, and is stored in a global blockchain. A global blockchain is a blockchain replicated across all nodes, and all nodes are in perfect state consensus with respect to the global blockchain. To achieve consensus (e.g., agreement to the addition of a block to a blockchain), a consensus protocol is implemented in the public blockchain network. Examples of consensus protocols include proof-of-work (POW) (e.g., implemented in the some crypto-currency networks), proof-of-stake (POS), and proof-of-authority (POA).

[0023] In general, a private blockchain network may be provided for a particular entity, which centrally controls read and write permissions. The entity controls which nodes are able to participate in the blockchain network. Consequently, private blockchain networks are generally referred to as permissioned networks that place restrictions on who is allowed to participate in the network, and on their level of participation (e.g., only in certain transactions). Various types of access control mechanisms can be used (e.g., existing participants vote on adding new entities, a regulatory authority can control admission).

[0024] In general, a consortium blockchain network may be private among the participating entities. In a consortium blockchain network, the consensus process is controlled by an authorized set of nodes, one or more nodes being operated by a respective entity (e.g., a financial institution, insurance company). For

example, a consortium of ten (10) entities (e.g., financial institutions, insurance companies) can operate a consortium blockchain network, each of which operates at least one node in the consortium blockchain network. Accordingly, the consortium blockchain network can be considered a private network with respect to the participating entities. In some examples, each entity (node) must sign every block in order for the block to be valid, and added to the blockchain. In some examples, at least a sub-set of entities (nodes) (e.g., at least 7 entities) must sign every block in order for the block to be valid, and added to the blockchain.

[0025] FIG. 1 illustrates a schematic diagram of a blockchain system 100, according to an embodiment. Referring to FIG. 1, the blockchain system 100 may include a plurality of nodes, e.g., nodes 102-110, configured to operate on a blockchain 120. The nodes 102-110 may form a network 112, such as a peer-to-peer (P2P) network. Each of the nodes 102-110 may be a computing device, such as a computer or a computer system, configured to store a copy of the blockchain 120, or may be software running on the computing device, such as a process or an application. Each of the nodes 102-110 may have a unique identifier.

[0026] The blockchain 120 may include a growing list of records in the form of data blocks, such as blocks B1-B5 in FIG. 1. Each of the blocks B1-B5 may include a timestamp, a cryptographic hash of a previous block, and data of the present block, which may be transactions such as monetary transactions. For example, as illustrated in FIG. 1, block B5 may include a timestamp, a cryptographic hash of block B4, and transaction data of block B5. Also, for example, a hashing operation may be performed on the previous block to generate the cryptographic hash of the previous block. The hashing operation may convert inputs of various lengths into cryptographic outputs of a fixed length through a hash algorithm, such

as SHA-256.

[0027] The nodes 102-110 may be configured to perform an operation on the blockchain 120. For example, when a node, e.g., the node 102, wants to store new data onto the blockchain 120, that node may generate a new block to be added to the blockchain 120 and broadcast the new block to other nodes, e.g., the nodes 104-110, in the network 112. Based on legitimacy of the new block, e.g., validity of its signature and transactions, the other nodes may determine to accept the new block, such that the node 102 and the other nodes may add the new block to their respective copies of the blockchain 120. As this process repeats, more and more blocks of data may be added to the blockchain 120.

[0028] FIG. 2 illustrates a schematic diagram of a computing device 200 for implementing a node, e.g., the node 102 (FIG. 1), in a blockchain system, according to an embodiment. Referring to FIG. 2, the computing device 200 may include a communication interface 202, a processor 204, and a memory 206.

[0029] The communication interface 202 may facilitate communications between the computing device 200 and devices implementing other nodes, e.g., nodes 104-110 (FIG. 1), in the network. In some embodiments, the communication interface 202 is configured to support one or more communication standards, such as an Internet standard or protocol, an Integrated Services Digital Network (ISDN) standard, etc. In some embodiments, the communication interface 202 may include one or more of a Local Area Network (LAN) card, a cable modem, a satellite modem, a data bus, a cable, a wireless communication channel, a radio-based communication channel, a cellular communication channel, an Internet Protocol (IP) based communication device, or other communication devices for wired and/or wireless communications. In some embodiments, the communication interface 202

may be based on public cloud infrastructure, private cloud infrastructure, hybrid public/private cloud infrastructure.

[0030] The processor 204 may include one or more dedicated processing units, application-specific integrated circuits (ASICs), field-programmable gate arrays (FPGAs), or various other types of processors or processing units. The processor 204 is coupled with the memory 206 and is configured to execute instructions stored in the memory 206.

[0031] The memory 206 may store processor-executable instructions and data, such as a copy of the blockchain 120 (FIG. 1). The memory 206 may include any type of volatile or non-volatile memory devices, or a combination thereof, such as a static random-access memory (SRAM), an electrically erasable programmable read-only memory (EEPROM), an erasable programmable read-only memory (EPROM), a programmable read-only memory (PROM), a read-only memory (ROM), a magnetic memory, a flash memory, or a magnetic or optical disk. When the instructions in the memory 206 are executed by the processor 204, the computing device 200 may perform an operation on the blockchain 120.

[0032] FIG. 3 illustrates a flow chart of a method 300 for mitigating invoice financing fraud according to an embodiment. Referring to FIG. 3, multiple users may have accounts on a Blockchain, e.g., the blockchain 120 (FIG. 1). The Blockchain may be implemented to support various types of users, or parties, including, e.g., individuals, businesses, banks, financial institutions, hospitals, as well as other types of companies, organizations, and the like.

[0033] For illustrative purposes, three users are depicted in FIG. 3, and include a Seller, a Bank, and a trusted user referred to as an Authority. Suppose that the Seller is interested in obtaining invoice financing from the Bank. The Seller

may attempt to secure this financing by claiming to have an outstanding invoice issued to a Buyer (who may also be a user of the Blockchain, as will be describe below with reference to FIG. 4). The invoice may be issued by the Seller for various reasons, including, e.g., past or future services rendered or products sold to the Buyer. If the Seller indeed has an outstanding invoice issued to the Buyer, the Bank may be willing to provide invoice financing to the Seller if the Seller agrees to secure the financing using the outstanding invoice. However, the Bank may want to make sure that the invoice indeed exists and has not been used to secure financing from other institutions already. The Bank may therefore require the Seller to prove the validity of the invoice before the Bank agrees to provide invoice financing to the Seller.

[0034] In some embodiments, the Bank may be willing to accept the validity of the invoice if the Seller is able to request the Authority to confirm the validity of the invoice based on the method 300 described in detail below. In some embodiments, the Authority may be a trusted user such as a government agency or a customs official. The Authority may also be a trusted user representing government agencies or customs officials.

[0035] At step 302, the Seller may generate a digital invoice. The digital invoice may be generated in various formats, including, e.g., a standardized format that users of the Blockchain have agreed to. The digital invoice may contain data fields including, e.g., Seller's identity or identifier, Buyer's identity or identifier, description of product(s) or service(s) rendered, amount(s) due, payment terms, and the like.

[0036] The Seller may also generate a commitment value at step 302 for recordation on the Blockchain. In some embodiments, the Seller may calculate the

commitment value CM_INVOICE by taking into consideration information regarding the Seller, the Buyer, and at least a portion of the invoice itself. In some embodiments, CM_INVOICE may be calculated based on a one-way function. One of ordinary skill in the art will understand that a function $g\colon \mathcal{D} \to \mathcal{R}$ is one-way if, given a random element $x \in \mathcal{R}$, it is hard to compute a $y \in \mathcal{D}$ such that $g(y) = x$. In other words, it is difficult to compute a value of an input variable of a one-way function from a value of an output variable of the one-way function, making the function practically infeasible to invert and, thus, the function is called “one-way.” Hash functions, such as SHA 256 and the like, are examples of one-way functions.

[0037] For example, in some embodiments, CM_INVOICE may be calculated as a hash value of (PK_SELLER, PK_BUYER, INVOICE_DIGEST, TN, R), where PK_SELLER and PK_BUYER represent the public keys of the Seller and the Buyer, respectively, and INVOICE_DIGEST represents a hash value of the digital invoice, or a hash value of one or more portions of the digital invoice. TN may represent a token generated by the Seller that can be used to prevent the Seller from using the digital invoice to obtain financing more than once (i.e., prevent double financing). In some embodiments, TN may include a random number. However, it is to be understood that TN may also include alphanumeric values, and in some embodiments, TN may include a serial number. R may represent a random factor generated by the Seller to further protect information contained in PK_SELLER, PK_BUYER, and INVOICE_DIGEST. In some embodiments, R may be a random number or a random alphanumeric value.

[0038] The Seller may also generate a zero-knowledge proof, denoted as SELLER’S_PROOF, at step 302. Zero-knowledge proof refers to techniques that allow a prover to prove to a verifier that a statement is true without revealing any

information beyond the validity of the statement itself. At step 302, the Seller is the prover, who may prove to the Blockchain, or a smart contract executing on the Blockchain, which serves as the verifier, that the Seller is the true issuer of the invoice in question. The Seller may attempt to prove this by indicating that: (1) CM_INVOICE is well formed and (2) the Seller is indeed the Seller itself.

[0039] In an embodiment, to prove that CM_INVOICE is well formed, the Seller may prove to the Blockchain that the commitment value of (PK_SELLER, PK_BUYER, INVOICE_DIGEST, TN, R) is CM_INVOICE. In an embodiment, to prove that the Seller is the Seller itself, the Seller may prove to the Blockchain that the Seller is in possession of the Seller's private key, or that PK_SELLER = h(SK_SELLER), where SK_SELLER represents the private key that is only known to the Seller and h() is the hash function used to calculate the public key based on the private key.

[0040] In some embodiments, the Seller and the Blockchain may agree to implement zero-knowledge proof techniques such as Zero-Knowledge Succinct Non-Interactive Argument of Knowledge (zk-SNARK) or the like. The Seller may prove to the Blockchain that the Seller knows a secret input w such that for a public input x, a certain relationship between x and w holds true. In some embodiments, the relationship may be defined as an arithmetic circuit. In some embodiments, the arithmetic circuit may be defined based on an equation of polynomials that can be evaluated based on x and w. In some embodiments, a proving key and a verification key may be generated in a setup phase based on the arithmetic circuit and one or more security parameters established for the zero-knowledge proof. One of ordinary skill in the art will understand that the setup phase may be executed by a trusted party, or by multiple independent parties collaboratively using a multi-party

computation.

[0041] In some embodiments, the Seller may set x to CM_INVOICE and set w to a value generated based on PK_SELLER, PK_BUYER, INVOICE_DIGEST, TN, R, and SK_SELLER. For example, w may be generated by concatenating PK_SELLER, PK_BUYER, INVOICE_DIGEST, TN, R, and SK_SELLER together. In this manner, the Seller may generate SELLER'S_PROOF using the secret and public inputs as well as the proving key to prove to the Blockchain that the Seller is in possession of the secret input w. In some embodiments, the Blockchain may verify SELLER'S_PROOF using the public input and the verification key generated in the setup phase. In some embodiments, if the Blockchain accepts the Seller's proof that the Seller is in possession of the secret input w, the Blockchain may accept the Seller's statements as true.

[0042] At step 304, the Seller may submit a transaction to the Blockchain with payload containing {CM_INVOICE, SELLER'S_PROOF}. For illustrative purposes, this transaction may be referred to as an "ISSUE_INVOICE" transaction.

[0043] At step 306, the Blockchain may verify SELLER'S_PROOF to determine whether the Seller is in possession of the secret input w. In some embodiments, the Blockchain may utilize one or more smart contracts executing on the Blockchain to provide the determination. Smart contracts are computer protocols implemented in the form of computer code that are incorporated into the Blockchain, to facilitate, verify, or enforce the negotiation or performance of contracts. For example, users of the Blockchain may program agreed terms into a smart contract using a programming language, such as C++, Java, Solidity, Python, etc., and when the terms are met, the smart contract may be automatically executed on the Blockchain, e.g., to perform a transaction. Also for example, the smart contract may

include a plurality of subroutines or functions, each of which may be a sequence of program instructions that performs a specific task. The smart contract may be operational code that is fully or partially executed without human interaction.

[0044] In some embodiments, a smart contract may be incorporated into the Blockchain to determine whether SELLER'S_PROOF is acceptable. The smart contract may verify SELLER'S_PROOF using the public input and the verification key. If SELLER'S_PROOF cannot be verified, the smart contract may refuse to allow the Seller to proceed further. On the other hand, if SELLER'S_PROOF can be verified, then the smart contract may determine that SELLER'S_PROOF is acceptable and proceed to record CM_INVOICE in a seller-submitted invoice pool. In some embodiments, the seller-submitted invoice pool may be structured as a Merkle tree T_INVOICE and CM_INVOICE may be recorded on a leaf node of the Merkle tree T_INVOICE.

[0045] At step 308, the Seller may proceed to ask the Authority to validate the digital invoice generated by the Seller. In some embodiments, the Seller may send the digital invoice and the values of TN and R privately to the Authority through an off-chain communication channel. The Seller may also send CM_INVOICE to the Authority. Alternatively or additionally, the Authority may reconstruct CM_INVOICE based on the digital invoice and the values of TN and R received from the Seller.

[0046] At step 310, the Authority may examine the digital invoice to determine whether the digital invoice is valid. The Authority may, for example, compare the outstanding amount stated in the digital invoice against records maintained by or accessible to the Authority. For instance, if the Seller is an exporter and the digital invoice is issued for products exported to buyers overseas, the Authority may compare the digital invoice against records maintained by customs

officials to determine whether there are any discrepancies with respect to the description of products and the amount due. The Authority may also compare the digital invoice against other types of records, including, e.g., tax records, financial disclosures, and the like.

[0047] In some embodiments, the Authority may invalidate the digital invoice if the Authority determines that CM_INVOICE is not recorded in the seller-submitted invoice pool, or that the recorded CM_INVOICE does not match the commitment value of (PK_SELLER, PK_BUYER, INVOICE_DIGEST, TN, R). In this manner, if the Seller attempts to bypass steps 304-306, or attempts to change the digital invoice or the values of TN or R, such attempts can be detected by the Authority, who can then invalidate the digital invoice and prevent fraud. On the other hand, if the Authority determines that the digital invoice is valid, the Authority may proceed to generate a certified commitment value for recordation on the Blockchain. In some embodiments, the Authority may calculate the certified commitment value CM_CERT_INVOICE as:

COMMITMENT(PK_SELLER, PK_BUYER, INVOICE_DIGEST,

CERT_DIGEST, TN', R').

[0048] In some embodiments, CM_CERT_INVOICE may be calculated based on the same one-way function used by the Seller to calculate CM_INVOICE, with the values of PK_SELLER, PK_BUYER, and INVOICE_DIGEST being the same. CERT_DIGEST may represent a hash value of one or more certifying documents generated by the Authority. The certifying documents may be generated in various formats, including, e.g., a standardized format established by government agencies or customs officials for providing certifications. The Authority may also generate a new TN' and a new R'. In this manner, the Authority may calculate a

commitment value that is different from the commitment value calculated by the Seller, effectively preventing an unrelated user from linking the two commitment values together, which in turn prevents the unrelated user from linking the Seller and the Authority together.

[0049] The Authority may also generate a zero-knowledge proof, denoted as AUTHORITY'S_PROOF, at step 310. The Authority may generate AUTHORITY'S_PROOF to prove to the Blockchain that the Authority has reviewed and certified the invoice provided by the Seller. The Authority may attempt to prove this by indicating that: (1) CM_INVOICE is well formed, (2) CM_INVOICE is recorded in the seller-submitted invoice pool, and (3) CM_CERT_INVOICE is well formed.

[0050] In an embodiment, to prove that CM_INVOICE is well formed, the Authority may prove to the Blockchain that the Authority knows the values of PK_SELLER, PK_BUYER, INVOICE_DIGEST, TN, and R and that the commitment value of (PK_SELLER, PK_BUYER, INVOICE_DIGEST, TN, R) is CM_INVOICE. In an embodiment, to prove that CM_INVOICE is recorded in the seller-submitted invoice pool, the Authority may prove to the Blockchain that CM_INVOICE is a valid leaf node in the Merkle tree T_INVOICE. In an embodiment, to prove that CM_CERT_INVOICE is well formed, the Authority may prove to the Blockchain that the commitment value of (PK_SELLER, PK_BUYER, INVOICE_DIGEST, CERT_DIGEST, TN', R') is CM_CERT_INVOICE.

[0051] In some embodiments, the Authority and the Blockchain may agree to implement the zero-knowledge proof techniques described above. For example, the Authority may prove to the Blockchain that the Authority knows a secret input w such that for a public input x, a certain relationship between x and w holds true. In some embodiments, the relationship may be defined as an arithmetic circuit, which may be

defined based on an equation of polynomials that can be evaluated based on x and w. In some embodiments, a proving key and a verification key may be generated in a setup phase based on the arithmetic circuit and one or more security parameters established for the zero-knowledge proof. In some embodiments, the Authority may set x to a value containing TN and CM_CERT_INVOICE, and set w to a value generated based on PK_SELLER, PK_BUYER, INVOICE_DIGEST, CERT_DIGEST, R, TN', R', and CM_INVOICE. In this manner, the Authority may be able to generate AUTHORITY'S_PROOF to prove to the Blockchain that the Authority is in possession of the secret input w. In some embodiments, if the Blockchain accepts the Authority's proof that the Authority is in possession of the secret input w, the Blockchain may accept the Authority's statements as true.

[0052] At step 312, the Authority may send the certifying documents and the values of TN' and R' to the Seller so that the Seller can later use the certifying documents to obtain financing from the Bank. At step 314, the Authority may submit a transaction to the Blockchain with payload containing {TN, CM_CERT_INVOICE, AUTHORITY'S_PROOF}. For illustrative purposes, this transaction may be referred to as an "AUTHORITY_CERTIFIED_INVOICE" transaction. In some embodiments, the Authority may have a public trusted identity registered on the Blockchain. In such embodiments, the Authority may sign the transaction and the payload using the Authority's private signing key.

[0053] At step 316, the Blockchain may utilize a smart contract to verify the identity of the Authority (e.g., based on the signature utilized by the Authority to sign the transaction) and check whether TN contained in the payload has been used or spent already. In some embodiments, the smart contract may maintain a used token list on the Blockchain to record used tokens. If TN contained in the payload is listed

in the used token list, the smart contract may refuse to proceed further because the Seller is attempting to use the same digital invoice to obtain financing more than once. On the other hand, if TN contained in the payload is not listed in the used token list, the smart contract may proceed to verify whether AUTHORITY'S_PROOF is acceptable. The smart contract may verify AUTHORITY'S_PROOF using the public input and the verification key. If AUTHORITY'S_PROOF can be verified, then the smart contract may determine that AUTHORITY'S_PROOF is acceptable and proceed to record CM_CERT_INVOICE in an authority-certified invoice pool. In some embodiments, the authority-certified invoice pool may be structured as a Merkle tree T_CERT_INVOICE and CM_CERT_INVOICE may be recorded on a leaf node of the Merkle tree T_CERT_INVOICE. The smart contract may also add TN to the used token list to indicate that TN has been used, thereby invalidating TN for future use.

[0054] At step 318, the Seller may proceed to request for invoice financing from the Bank. In some embodiments, the Seller may send the digital invoice, the certifying documents, and the values of TN' and R' privately to the Bank through an off-chain communication channel. The Seller may also calculate the value of CM_CERT_INVOICE and send the value of CM_CERT_INVOICE to the Bank. Alternatively or additionally, the Bank may reconstruct CM_CERT_INVOICE based on the digital invoice, the certifying documents, and the values of TN' and R' received from the Seller.

[0055] At step 320, the Bank may determine whether the digital invoice and the certifying documents are valid. The Bank may invalidate the digital invoice if, e.g., the digital invoice has not been certified by the Authority. The Bank may make this determination by determining whether CM_CERT_INVOICE is recorded in the

authority-certified invoice pool (e.g., in the Merkle tree T_CERT_INVOICE). If CM_CERT_INVOICE is not recorded in the authority-certified invoice pool, the Bank may determine that the digital invoice is invalid and refuse to allow the Seller to proceed further. On the other hand, if CM_CERT_INVOICE is recorded in the authority-certified invoice pool, the Bank may determine that the digital invoice is valid. The Bank may then proceed to decide whether to provide invoice financing to the Seller. It is noted that because CM_CERT_INVOICE is generated at least partially based on the certifying documents, the Bank may be able to locate CM_CERT_INVOICE in the authority-certified invoice pool only if the Bank receives true certifying documents from the Seller. In this manner, any attempts by the Seller to produce fake certifying documents can be detected and such fraudulent activities can be prevented.

[0056] In some embodiments, the Bank may also determine whether TN' provided by the Seller has been used already. In some embodiments, the Bank may call a smart contract executing on the Blockchain to check whether TN' is listed in the used token list. If TN' is listed in the used token list, the Bank may refuse to proceed further because the Seller is attempting to use the same digital invoice to obtain financing more than once. If TN' is not listed in the used token list, the Bank may proceed to decide whether to provide invoice financing to the Seller.

[0057] It is to be understood that the Bank may take various factors into consideration in determining whether to provide invoice financing to the Seller. Such factors may include, e.g., the size of the invoice, the Seller's credit score and credit history, the Bank's prior dealings with the Seller, and the like. Additional factors may include, e.g., the Buyer's credit score and the Bank's prior dealings with the Buyer.

[0058] If the Bank decides to deny the Seller's request for invoice financing,

the Bank may communicate that decision to the Seller privately through an off-chain communication channel, in which case the Seller may choose to repeat step 318 and request for invoice financing from a different fund provider. On the other hand, if the Bank decides to provide invoice financing to the Seller, the Bank may proceed to generate a zero-knowledge proof, denoted as BANK'S_PROOF, at step 320, to prove to the Blockchain that the Bank has verified the validity of the Seller's request to provide invoice financing. The Bank may attempt to prove this by indicating that: (1) CM_CERT_INVOICE is well formed and (2) CM_CERT_INVOICE is recorded in the authority-certified invoice pool.

[0059] In an embodiment, to prove that CM_CERT_INVOICE is well formed, the Bank may prove to the Blockchain that the Bank knows the values of PK_SELLER, PK_BUYER, INVOICE_DIGEST, CERT_DIGEST, TN', and R' and that the commitment value of (PK_SELLER, PK_BUYER, INVOICE_DIGEST, CERT_DIGEST, TN', R') is CM_CERT_INVOICE. In an embodiment, to prove that CM_CERT_INVOICE is recorded in the authority-certified invoice pool, the Bank may prove to the Blockchain that CM_CERT_INVOICE is a valid leaf node in the Merkle tree T_CERT_INVOICE.

[0060] In some embodiments, the Bank and the Blockchain may agree to implement the zero-knowledge proof techniques described above. For example, the Bank may prove to the Blockchain that the Bank knows a secret input w such that for a public input x, a certain relationship between x and w holds true. In some embodiments, the relationship may be defined as an arithmetic circuit, which may be defined based on an equation of polynomials that can be evaluated based on x and w. In some embodiments, a proving key and a verification key may be generated in a setup phase based on the arithmetic circuit and one or more security parameters

established for the zero-knowledge proof. In some embodiments, the Bank may set x to TN' and set w to a value generated based on PK_SELLER, PK_BUYER, INVOICE_DIGEST, CERT_DIGEST, R', and CM_CERT_INVOICE. In this manner, the Bank may be able to generate BANK'S_PROOF to prove to the Blockchain that the Bank is in possession of the secret input w. In some embodiments, if the Blockchain accepts the Bank's proof that the Bank is in possession of the secret input w, the Blockchain may accept the Bank's statements as true.

[0061] At step 322, the Bank may submit a transaction to the Blockchain with payload containing {TN', BANK'S_PROOF}. For illustrative purposes, this transaction may be referred to as an "ISSUE_FINANCING" transaction.

[0062] At step 324, the Blockchain may utilize a smart contract to check whether TN' contained in the payload has been used already. In some embodiments, the smart contract may check whether TN' is listed in the used token list. If TN' is listed in the used token list, the smart contract may refuse to proceed further because the Seller is attempting to use the same digital invoice to obtain financing more than once. If TN' is not listed in the used token list, the smart contract may proceed to verify whether BANK'S_PROOF is acceptable. The smart contract may verify BANK'S_PROOF using the public input and the verification key. If BANK'S_PROOF can be verified, then the smart contract may determine that BANK'S_PROOF is acceptable and proceed to invalidate TN' (e.g., by adding TN' to the used token list) and report to the Bank at step 326 that no fraud is detected based on the transactions received so that the Bank can proceed to issue invoice financing to the Seller.

[0063] It is to be understood that while the examples above utilized one used token list to record all used tokens, such an implementation is merely provided as an

example and is not meant to be limiting. In some embodiments, one used token list may be utilized to record used tokens generated by the Seller and a separate used token list may be utilized to record used tokens generated by the Authority. It is to be understood that other types of data structures may also be utilized to record used tokens. Furthermore, it is to be understood that the declarations of the functions, variables, and transactions described above are merely presented as examples and are not meant to be limiting.

[0064] It is noted that by requiring users to confirm the validity of the invoice, the method 300 can avoid double financing an invalidated invoice. The method 300 can also mitigate fraudulent activities where the Seller may produce fake invoices and attempt to borrow money using such invoices. The method 300 may further protect privacy by hiding user identities, amounts due, payment terms, and other invoice-related information using commitment values. The method 300 may also ensure that the transactions recorded on the Blockchain are unlinkable by unrelated users, thereby preventing unrelated users from knowing the users related to these transactions. Furthermore, in some embodiments, if a user wants to further conceal the number of “ISSUE_INVOICE” transactions or “ISSUE_FINANCING” transactions the user submits to the Blockchain, the user may choose to use one-time anonymous blockchain identities to submit such transactions.

[0065] It is also noted that the method 300 may be extended to allow the Buyer to confirm the validity of the digital invoice generated by the Seller. The Buyer may, for example, examine the digital invoice and invalidate the digital invoice if the Seller misstated the outstanding amount or falsified the invoice.

[0066] FIG. 4 illustrates a flow chart of a method 400 for mitigating invoice financing fraud according to an embodiment. For illustrative purposes, four users

are depicted in FIG. 4, and include a Seller, a Buyer, a Bank, and a trusted user referred to as an Authority. Suppose that the Seller is interested in obtaining invoice financing from the Bank, the Bank may require the Seller to prove the validity of the invoice before the Bank agrees to provide invoice financing to the Seller. In some embodiments, the Bank may be willing to accept the validity of the invoice if the Seller is able to request the Buyer and the Authority to confirm the validity of the invoice based on the method 400 described in detail below.

[0067] At step 402, the Seller may generate a digital invoice, a commitment value CM_INVOICE, and a zero-knowledge proof SELLER'S_PROOF as described above. CM_INVOICE may be calculated as a hash value of (PK_SELLER, PK_BUYER, INVOICE_DIGEST, TN, R), where PK_SELLER and PK_BUYER represent the public keys of the Seller and the Buyer, respectively, and INVOICE_DIGEST represents a hash value of the digital invoice, or a hash value of one or more portions of the digital invoice. TN may represent a token generated by the Seller that can be used to prevent the Seller from using the digital invoice to obtain financing more than once (i.e., prevent double financing). R may represent a random factor generated by the Seller to further protect information contained in PK_SELLER, PK_BUYER, and INVOICE_DIGEST.

[0068] At step 404, the Seller may submit an "ISSUE_INVOICE" transaction to the Blockchain with payload containing {CM_INVOICE, SELLER'S_PROOF}. At step 406, the Blockchain may verify SELLER'S_PROOF to determine whether SELLER'S_PROOF is acceptable. In some embodiments, the Blockchain may utilize a smart contract to verify SELLER'S_PROOF using the public input and the verification key, as described above. If SELLER'S_PROOF cannot be verified, the smart contract may refuse to allow the Seller to proceed further. On the other hand,

if SELLER'S_PROOF can be verified, then the smart contract may determine that SELLER'S_PROOF is acceptable and proceed to record CM_INVOICE in a seller-submitted invoice pool. In some embodiments, the seller-submitted invoice pool may be structured as a Merkle tree T_INVOICE and CM_INVOICE may be recorded on a leaf node of the Merkle tree T_INVOICE.

[0069] At step 408, the Seller may proceed to ask the Buyer to validate the digital invoice generated by the Seller. In some embodiments, the Seller may send the digital invoice and the values of TN and R privately to the Buyer through an off-chain communication channel. The Seller may also send CM_INVOICE to the Buyer. Alternatively or additionally, the Buyer may reconstruct CM_INVOICE based on the digital invoice and the values of TN and R received from the Seller.

[0070] At step 410, the Buyer may examine the digital invoice to determine whether the digital invoice is valid. The Buyer may invalidate the digital invoice if, e.g., the Seller misstated the outstanding amount or falsified the invoice. The Buyer may also invalidate the digital invoice if the Buyer determines that CM_INVOICE is not recorded in the seller-submitted invoice pool, or that the recorded CM_INVOICE does not match the commitment value of (PK_SELLER, PK_BUYER, INVOICE_DIGEST, TN, R). In this manner, if the Seller attempts to bypass steps 404-406, or attempts to change the digital invoice or the values of TN or R, such attempts can be detected by the Buyer, who can then invalidate the digital invoice and prevent fraud.

[0071] If the Buyer determines that the digital invoice is valid, the Buyer may proceed to generate a buyer-verified commitment value for recordation on the Blockchain. In some embodiments, the Buyer may calculate the buyer-validated commitment value CM_BV_INVOICE as:

COMMITMENT(PK_SELLER, PK_BUYER, INVOICE_DIGEST, TN', R').

[0072] In some embodiments, CM_BV_INVOICE may be calculated based on the same function used by the Seller to calculate CM_INVOICE. It is noted that while the values of PK_SELLER, PK_BUYER, and INVOICE_DIGEST may be the same, the Buyer may be responsible for generating a new TN' and a new R'. In this manner, the Buyer may calculate a commitment value that is different from the commitment value calculated by the Seller, effectively preventing an unrelated user from linking the two commitment values together, which in turn prevents the unrelated user from linking the Seller and the Buyer together.

[0073] The Buyer may also generate a zero-knowledge proof, denoted as BUYER'S_PROOF, at step 410. The Buyer may generate BUYER'S_PROOF to prove to the Blockchain that the Buyer is the true receiver of the invoice. The Buyer may attempt to prove this by indicating that: (1) CM_INVOICE is well formed, (2) CM_INVOICE is recorded in the seller-submitted invoice pool, (3) CM_BV_INVOICE is well formed, and (4) the Buyer is indeed the Buyer itself.

[0074] In an embodiment, to prove that CM_INVOICE is well formed, the Buyer may prove to the Blockchain that the Buyer knows the values of PK_SELLER, PK_BUYER, INVOICE_DIGEST, TN, and R and that the commitment value of (PK_SELLER, PK_BUYER, INVOICE_DIGEST, TN, R) is CM_INVOICE. In an embodiment, to prove that CM_INVOICE is recorded in the seller-submitted invoice pool, the Buyer may prove to the Blockchain that CM_INVOICE is a valid leaf node in the Merkle tree T_INVOICE. In an embodiment, to prove that CM_BV_INVOICE is well formed, the Buyer may prove to the Blockchain that the commitment value of (PK_SELLER, PK_BUYER, INVOICE_DIGEST, TN', R') is CM_BV_INVOICE. In an embodiment, to prove that the Buyer is indeed the Buyer itself, the Buyer may prove

to the Blockchain that the Buyer is in possession of the Buyer's private key, or that PK_BUYER = h(SK_BUYER), where SK_BUYER represents the private key that is only known to the Buyer.

[0075] In some embodiments, the Buyer and the Blockchain may agree to implement the zero-knowledge proof techniques described above. For example, the Buyer may prove to the Blockchain that the Buyer knows a secret input w such that for a public input x, a certain relationship between x and w holds true. In some embodiments, the relationship may be defined as an arithmetic circuit, which may be defined based on an equation of polynomials that can be evaluated based on x and w. In some embodiments, a proving key and a verification key may be generated in a setup phase based on the arithmetic circuit and one or more security parameters established for the zero-knowledge proof. In some embodiments, the Buyer may set x to a value containing TN and CM_BV_INVOICE, and set w to a value generated based on PK_SELLER, PK_BUYER, INVOICE_DIGEST, R, TN', R', and SK_SELLER. In this manner, the Buyer may be able to generate BUYER'S_PROOF to prove to the Blockchain that the Buyer is in possession of the secret input w. In some embodiments, if the Blockchain accepts the Buyer's proof that the Buyer is in possession of the secret input w, the Blockchain may accept the Buyer's statements as true.

[0076] At step 412, the Buyer may send the values of TN' and R' to the Seller so that the Seller can later use these values to request for certification from the Authority. At step 414, the Buyer may submit a transaction to the Blockchain with payload containing {TN, CM_BV_INVOICE, BUYER'S_PROOF}. For illustrative purposes, this transaction may be referred to as a "BUYER_VALIDATED_INVOICE" transaction.

[0077] At step 416, the Blockchain may utilize a smart contract to check whether TN contained in the payload has been used or spent already. In some embodiments, the smart contract may maintain a used token list on the Blockchain to record used tokens. If TN contained in the payload is listed in the used token list, the smart contract may refuse to proceed further because the Seller is attempting to use the same digital invoice to obtain financing more than once. On the other hand, if TN contained in the payload is not listed in the used token list, the smart contract may proceed to verify whether BUYER'S_PROOF is acceptable. The smart contract may verify BUYER'S_PROOF using the public input and the verification key. If BUYER'S_PROOF can be verified, then the smart contract may determine that BUYER'S_PROOF is acceptable and proceed to record CM_BV_INVOICE in a buyer-validated invoice pool. In some embodiments, the buyer-validated invoice pool may be structured as a Merkle tree T_BV_INVOICE and CM_BV_INVOICE may be recorded on a leaf node of the Merkle tree T_BV_INVOICE. The smart contract may also add TN to the used token list to indicate that TN has been used, thereby invalidating TN for future use.

[0078] At step 418, the Seller may proceed to ask the Authority to validate the digital invoice generated by the Seller. In some embodiments, the Seller may send the digital invoice and the values of TN' and R' privately to the Authority through an off-chain communication channel. The Seller may also send CM_BV_INVOICE to the Authority. Alternatively or additionally, the Authority may reconstruct CM_BV_INVOICE based on the digital invoice and the values of TN' and R' received from the Seller.

[0079] At step 420, the Authority may examine the digital invoice to determine whether the digital invoice is valid. In some embodiments, the Authority

may invalidate the digital invoice if the Authority determines that CM_BV_INVOICE is not recorded in the buyer-validated invoice pool, or that the recorded CM_BV_INVOICE does not match the commitment value of (PK_SELLER, PK_BUYER, INVOICE_DIGEST, TN', R'). In this manner, if the Seller attempts to bypass any of the preceding steps, or attempts to change the digital invoice or the values of TN' or R', such attempts can be detected by the Authority, who can then invalidate the digital invoice and prevent fraud. On the other hand, if the Authority determines that the digital invoice is valid, the Authority may proceed to generate a certified commitment value for recordation on the Blockchain. In some embodiments, the Authority may calculate the certified commitment value CM_CERT_INVOICE as:

COMMITMENT(PK_SELLER, PK_BUYER, INVOICE_DIGEST,

CERT_DIGEST, TN'', R'').

[0080] In some embodiments, CM_CERT_INVOICE may be calculated based on the same function used by the Seller to calculate CM_INVOICE, with the values of PK_SELLER, PK_BUYER, and INVOICE_DIGEST being the same. CERT_DIGEST may represent a hash value of one or more certifying documents generated by the Authority. The certifying documents may be generated in various formats, including, e.g., a standardized format established by government agencies or customs officials for providing certifications. The Authority may also generate a new TN'' and a new R''. In this manner, the Authority may calculate a commitment value that is different from the commitment values calculated by the Seller and the Buyer, effectively preventing an unrelated user from linking the three commitment values together, which in turn prevents the unrelated user from linking the Seller, the Buyer, and the Authority together.

[0081] The Authority may also generate a zero-knowledge proof, denoted as AUTHORITY'S_PROOF, at step 420. The Authority may generate AUTHORITY'S_PROOF to prove to the Blockchain that the Authority has reviewed and certified the invoice provided by the Buyer. The Authority may attempt to prove this by indicating that: (1) CM_BV_INVOICE is well formed, (2) CM_BV_INVOICE is recorded in the buyer-submitted invoice pool, and (3) CM_CERT_INVOICE is well formed.

[0082] In an embodiment, to prove that CM_BV_INVOICE is well formed, the Authority may prove to the Blockchain that the Authority knows the values of PK_SELLER, PK_BUYER, INVOICE_DIGEST, TN', and R' and that the commitment value of (PK_SELLER, PK_BUYER, INVOICE_DIGEST, TN', R') is CM_BV_INVOICE. In an embodiment, to prove that CM_BV_INVOICE is recorded in the buyer-submitted invoice pool, the Authority may prove to the Blockchain that CM_BV_INVOICE is a valid leaf node in the Merkle tree T_BV_INVOICE. In an embodiment, to prove that CM_CERT_INVOICE is well formed, the Authority may prove to the Blockchain that the commitment value of (PK_SELLER, PK_BUYER, INVOICE_DIGEST, CERT_DIGEST, TN'', R'') is CM_CERT_INVOICE.

[0083] In some embodiments, the Authority and the Blockchain may agree to implement the zero-knowledge proof techniques described above. For example, the Authority may prove to the Blockchain that the Authority knows a secret input w such that for a public input x, a certain relationship between x and w holds true. In some embodiments, the relationship may be defined as an arithmetic circuit, which may be defined based on an equation of polynomials that can be evaluated based on x and w. In some embodiments, a proving key and a verification key may be generated in a setup phase based on the arithmetic circuit and one or more security parameters

established for the zero-knowledge proof. In some embodiments, the Authority may set x to a value containing TN and CM_CERT_INVOICE, and set w to a value generated based on PK_SELLER, PK_BUYER, INVOICE_DIGEST, CERT_DIGEST, R', TN'', R'', and CM_BV_INVOICE. In this manner, the Authority may be able to generate AUTHORITY'S_PROOF to prove to the Blockchain that the Authority is in possession of the secret input w. In some embodiments, if the Blockchain accepts the Authority's proof that the Authority is in possession of the secret input w, the Blockchain may accept the Authority's statements as true.

[0084] At step 422, the Authority may send the certifying documents and the values of TN'' and R'' to the Seller so that the Seller can later use the certifying documents to obtain financing from the Bank. At step 424, the Authority may submit an "AUTHORITY_CERTIFIED_INVOICE" transaction to the Blockchain with payload containing {TN', CM_CERT_INVOICE, AUTHORITY'S_PROOF}.

[0085] At step 426, the Blockchain may utilize a smart contract to verify the identity of the Authority (e.g., based on the signature provided by the Authority) and check whether TN' contained in the payload has been used or spent already. In some embodiments, the smart contract may maintain a used token list on the Blockchain to record used tokens. If TN' contained in the payload is listed in the used token list, the smart contract may refuse to proceed further because the Seller is attempting to use the same digital invoice to obtain financing more than once. On the other hand, if TN' contained in the payload is not listed in the used token list, the smart contract may proceed to verify whether AUTHORITY'S_PROOF is acceptable. The smart contract may verify AUTHORITY'S_PROOF using the public input and the verification key. If AUTHORITY'S_PROOF can be verified, then the smart contract may determine that AUTHORITY'S_PROOF is acceptable and proceed to record

CM_CERT_INVOICE in an authority-certified invoice pool. In some embodiments, the authority-certified invoice pool may be structured as a Merkle tree T_CERT_INVOICE and CM_CERT_INVOICE may be recorded on a leaf node of the Merkle tree T_CERT_INVOICE. The smart contract may also add TN' to the used token list to indicate that TN has been used, thereby invalidating TN' for future use.

[0086] At step 428, the Seller may proceed to request for invoice financing from the Bank. In some embodiments, the Seller may send the digital invoice, the certifying documents, and the values of TN'' and R'' privately to the Bank through an off-chain communication channel. The Seller may also calculate the value of CM_CERT_INVOICE and send the value of CM_CERT_INVOICE to the Bank. Alternatively or additionally, the Bank may reconstruct CM_CERT_INVOICE based on the digital invoice, the certifying documents, and the values of TN'' and R'' received from the Seller.

[0087] At step 430, the Bank may determine whether the digital invoice and the certifying documents are valid. The Bank may invalidate the digital invoice if, e.g., the digital invoice has not been certified by the Authority. The Bank may make this determination by determining whether CM_CERT_INVOICE is recorded in the authority-certified invoice pool (e.g., in the Merkle tree T_CERT_INVOICE). If CM_CERT_INVOICE is not recorded in the authority-certified invoice pool, the Bank may determine that the digital invoice is invalid and refuse to allow the Seller to proceed further. On the other hand, if CM_CERT_INVOICE is recorded in the authority-certified invoice pool, the Bank may determine that the digital invoice is valid. The Bank may then proceed to decide whether to provide invoice financing to the Seller.

[0088] In some embodiments, the Bank may also determine whether TN''

provided by the Seller has been used already. In some embodiments, the Bank may call a smart contract executing on the Blockchain to check whether TN'' is listed in the used token list. If TN'' is listed in the used token list, the Bank may refuse to proceed further because the Seller is attempting to use the same digital invoice to obtain financing more than once. If TN'' is not listed in the used token list, the Bank may proceed to decide whether to provide invoice financing to the Seller.

[0089] If the Bank decides to deny the Seller's request for invoice financing, the Bank may communicate that decision to the Seller privately through an off-chain communication channel, in which case the Seller may choose to repeat step 428 and request for invoice financing from a different fund provider. On the other hand, if the Bank decides to provide invoice financing to the Seller, the Bank may proceed to generate a zero-knowledge proof, denoted as BANK'S_PROOF, at step 430, to prove to the Blockchain that the Bank has verified the validity of the Seller's request to provide invoice financing. The Bank may attempt to prove this by indicating that: (1) CM_CERT_INVOICE is well formed and (2) CM_CERT_INVOICE is recorded in the authority-certified invoice pool.

[0090] In an embodiment, to prove that CM_CERT_INVOICE is well formed, the Bank may prove to the Blockchain that the Bank knows the values of PK_SELLER, PK_BUYER, INVOICE_DIGEST, CERT_DIGEST, TN'', and R'' and that the commitment value of (PK_SELLER, PK_BUYER, INVOICE_DIGEST, CERT_DIGEST, TN'', R'') is CM_CERT_INVOICE. In an embodiment, to prove that CM_CERT_INVOICE is recorded in the authority-certified invoice pool, the Bank may prove to the Blockchain that CM_CERT_INVOICE is a valid leaf node in the Merkle tree T_CERT_INVOICE.

[0091] In some embodiments, the Bank and the Blockchain may agree to

implement the zero-knowledge proof techniques described above. For example, the Bank may prove to the Blockchain that the Bank knows a secret input w such that for a public input x, a certain relationship between x and w holds true. In some embodiments, the relationship may be defined as an arithmetic circuit, which may be defined based on an equation of polynomials that can be evaluated based on x and w. In some embodiments, a proving key and a verification key may be generated in a setup phase based on the arithmetic circuit and one or more security parameters established for the zero-knowledge proof. In some embodiments, the Bank may set x to TN'' and set w to a value generated based on PK_SELLER, PK_BUYER, INVOICE_DIGEST, CERT_DIGEST, R'', and CM_CERT_INVOICE. In this manner, the Bank may be able to generate BANK'S_PROOF to prove to the Blockchain that the Bank is in possession of the secret input w.

[0092] At step 432, the Bank may submit an "ISSUE_FINANCING" transaction to the Blockchain with payload containing {TN', BANK'S_PROOF}. At step 434, the Blockchain may utilize a smart contract to check whether TN'' contained in the payload has been used already. In some embodiments, the smart contract may check whether TN'' is listed in the used token list. If TN'' is listed in the used token list, the smart contract may refuse to proceed further because the Seller is attempting to use the same digital invoice to obtain financing more than once. If TN'' is not listed in the used token list, the smart contract may proceed to verify whether BANK'S_PROOF is acceptable. The smart contract may verify BANK'S_PROOF using the public input and the verification key. If BANK'S_PROOF can be verified, then the smart contract may determine that BANK'S_PROOF is acceptable and proceed to invalidate TN'' (e.g., by adding TN'' to the used token list) and report to the Bank at step 436 that no fraud is detected based on the

transactions received so that the Bank can proceed to issue invoice financing to the Seller.

[0093] FIG. 5 illustrates a flow chart of a method 500 for mitigating invoice financing fraud, according to an embodiment. The method 500 may be performed by one or more nodes in a blockchain system, e.g., the nodes 102-110 in the blockchain system 100 (FIG. 1). The nodes 102-110 in the blockchain system 100 may perform operations on a blockchain, e.g., the blockchain 120 (FIG. 1). The blockchain 120 may be implemented as the Blockchain in the examples described above.

[0094] At step 502, a node, e.g., the node 102, may receive a first transaction submitted by a first user. The first user may be, e.g., the Seller (FIG. 3), who is the issuer of an invoice and is attempting to secure financing using the invoice. The first transaction may include the "ISSUE_INVOICE" transaction (FIG. 3) described above, which may include a first commitment of the invoice, e.g., CM_INVOICE, and a first proof, e.g., SELLER'S_PROOF.

[0095] At step 504, the node 102 may determine whether the first proof is acceptable and report a detection of fraud in response to a determination that the first proof is unacceptable. In some embodiments, the first proof may be a zero-knowledge proof generated by the first user to prove that the first user is the true issuer of the invoice. In some embodiments, the first user may attempt to prove to the node 102 that the first commitment of the invoice is well formed and that the first user is indeed the first user itself. In some embodiments, if the node 102 determines that the first proof is unacceptable, the node 102 may report a detection of suspected fraud. On the other hand, if the node 102 determines that the first proof is acceptable, the node 102 may record the first commitment in a first invoice pool, which is referred to as the seller-submitted invoice pool in FIG. 3.

[0096] At step 506, the node 102 may receive a second transaction submitted by a second user. The second user may be, e.g., the Authority (FIG. 3), who is trusted by the blockchain system 100 to certify the invoice issued by the first user. The second transaction may include the "AUTHORITY_CERTIFIED_INVOICE" transaction (FIG. 3) described above, which may include a first token, e.g., TN, generated by the first user, a second commitment of the invoice, e.g., CM_CERT_INVOICE, and a second proof, e.g., AUTHORITY'S_PROOF.

[0097] At step 508, the node 102 may determine whether the first token is valid and whether the second proof is acceptable. The node 102 may report a detection of fraud in response to a determination that the first token is invalid or the second proof is unacceptable. In some embodiments, the node 102 may determine whether the first token is valid by determining whether the first token is listed in a used token list. In some embodiments, the second proof may be a zero-knowledge proof generated by the second user to prove that the second user is the receiver of the invoice. In some embodiments, if the node 102 determines that the first token is invalid or the second proof is unacceptable, the node 102 may report a detection of suspected fraud. On the other hand, if the node 102 determines that the first token is valid and that the second proof is acceptable, the node 102 may record the second commitment in a second invoice pool, which is referred to as the authority-certified invoice pool in FIG. 3. The node 102 may also invalidate the first token. In some embodiments, the node 102 may invalidate the first token by adding the first token to a used token list.

[0098] At step 510, the node 102 may receive a third transaction submitted by a third user. The third user may be, e.g., the Bank (FIG. 3), who is a fund

provider having received a request to provide invoice financing to the first user. The third transaction may include the "ISSUE_FINANCING" transaction (FIG. 3) described above, which may include a second token, e.g., TN', generated by the second user, and a third proof, e.g., BANK'S_PROOF, generated by the third user.

[0099] At step 512, the node 102 may determine whether the second token is valid and whether the third proof is acceptable. The node 102 may report a detection of fraud in response to a determination that the second token is invalid or the third proof is unacceptable. In some embodiments, the node 102 may determine whether the second token is valid by determining whether the second token is listed in a used token list. In some embodiments, the third proof may be a zero-knowledge proof generated by the third user to prove that the third user has verified validity of the request to provide invoice financing to the first user. In some embodiments, if the node 102 determines that the second token is invalid or the third proof is unacceptable, the node 102 may report a detection of suspected fraud. On the other hand, if the node 102 determines that the second token is valid and that the third proof is acceptable, the node 102 may invalidate the second token and report to the third user that no fraud is detected based on the transactions received. In some embodiments, the node 102 may invalidate the second token by adding the second token to a used token list. The third user may proceed to issue invoice financing to the first user.

[00100] FIG. 6 illustrates a flow chart of a method 600 for mitigating invoice financing fraud, according to an embodiment. The method 600 may be performed by one or more nodes in a blockchain system, e.g., the nodes 102-110 in the blockchain system 100 (FIG. 1). The nodes 102-110 in the blockchain system 100 may perform operations on a blockchain, e.g., the blockchain 120 (FIG. 1). The blockchain 120

may be implemented as the Blockchain in the examples described above.

[00101] At step 602, a node, e.g., the node 102, may receive a first transaction submitted by a first user. The first user may be, e.g., the Seller (FIG. 4), who is the issuer of an invoice and is attempting to secure financing using the invoice. The first transaction may include the "ISSUE_INVOICE" transaction (FIG. 4) described above, which may include a first commitment of the invoice, e.g., CM_INVOICE, and a first proof, e.g., SELLER'S_PROOF.

[00102] At step 604, the node 102 may determine whether the first proof is acceptable and report a detection of fraud in response to a determination that the first proof is unacceptable. On the other hand, if the node 102 determines that the first proof is acceptable, the node 102 may record the first commitment in a first invoice pool, which is referred to as the seller-submitted invoice pool in FIG. 4.

[00103] At step 606, the node 102 may receive a second transaction submitted by a second user. The second user may be, e.g., the Buyer (FIG. 4), who is the receiver of the invoice. The second transaction may include the "BUYER_VALIDATED_INVOICE" transaction (FIG. 4) described above, which may include a first token, e.g., TN, generated by the first user, a second commitment of the invoice, e.g., CM_BV_INVOICE, and a second proof, e.g., BUYER'S_PROOF.

[00104] At step 608, the node 102 may determine whether the first token is valid and whether the second proof is acceptable. The node 102 may report a detection of fraud in response to a determination that the first token is invalid or the second proof is unacceptable. On the other hand, if the node 102 determines that the first token is valid and that the second proof is acceptable, the node 102 may record the second commitment in a second invoice pool, which is referred to as the buyer-validated invoice pool in FIG. 4. The node 102 may also invalidate the first

token. In some embodiments, the node 102 may invalidate the first token by adding the first token to a used token list.

[00105] At step 610, the node 102 may receive a third transaction submitted by a third user. The third user may be, e.g., the Authority (FIG. 4), who is trusted by the blockchain system 100 to certify the invoice issued by the first user. The third transaction may include the "AUTHORITY_CERTIFIED_INVOICE" transaction (FIG. 4) described above, which may include a second token, e.g., TN', generated by the second user, a third commitment of the invoice, e.g., CM_CERT_INVOICE, and a third proof, e.g., AUTHORITY'S_PROOF.

[00106] At step 612, the node 102 may determine whether the second token is valid and whether the third proof is acceptable. The node 102 may report a detection of fraud in response to a determination that the second token is invalid or the third proof is unacceptable. On the other hand, if the node 102 determines that the second token is valid and that the third proof is acceptable, the node 102 may record the third commitment in a third invoice pool, which is referred to as the authority-certified invoice pool in FIG. 4. The node 102 may also invalidate the second token. In some embodiments, the node 102 may invalidate the second token by adding the second token to a used token list.

[00107] At step 614, the node 102 may receive a fourth transaction submitted by a fourth user. The fourth user may be, e.g., the Bank (FIG. 4), who is a fund provider having received a request to provide invoice financing to the first user. The fourth transaction may include the "ISSUE_FINANCING" transaction (FIG. 4) described above, which may include a third token, e.g., TN'', generated by the third user, and a fourth proof, e.g., BANK'S_PROOF, generated by the fourth user.

[00108] At step 616, the node 102 may determine whether the third token is

valid and whether the fourth proof is acceptable. The node 102 may report a detection of fraud in response to a determination that the third token is invalid or the fourth proof is unacceptable. On the other hand, if the node 102 determines that the third token is valid and that the fourth proof is acceptable, the node 102 may invalidate the third token and report to the fourth user that no fraud is detected based on the transactions received. In some embodiments, the node 102 may invalidate the third token by adding the third token to a used token list. The fourth user may proceed to issue invoice financing to the first user.

[00109] FIG. 7 is a block diagram of an invoice financing fraud mitigation apparatus 700, according to an embodiment. The apparatus 700 may be an implementation of a software process, and may correspond to the method 500 (FIG. 5) and method 600 (FIG. 6). Referring to FIG. 7, the apparatus 700 may include a receiving module 702, a determination module 704, a recording module 706, and a reporting module 708.

[00110] The receiving module 702 may receive transactions submitted by users. The transactions may include the “ISSUE_INVOICE” transaction, the “AUTHORITY_CERTIFIED_INVOICE” transaction, the “BUYER_VALIDATED_INVOICE” transaction, and “ISSUE_FINANCING” transaction described above. The receiving module 702 may provide the received transactions to the determining module 704.

[00111] The determining module 704 may process the received transactions as described above. If a suspected fraud is detected, the determining module 704 may request the reporting module 708 to report the detection of fraud. Otherwise, the determining module 704 may provide commitment values contained in the received transactions to the recording module 706, which may record the

commitment values in corresponding invoice pool as described above. In some embodiments, if the received transaction is an “ISSUE_FINANCING” transaction and the determining module 704 detects no fraud, the determining module 704 may request the reporting module 708 to report that no fraud is detected so that the invoice financing request submitted by the first user may be processed.

[00112] Each of the above described modules may be implemented as software, or hardware, or a combination of software and hardware. For example, each of the above described modules may be implemented using a processor executing instructions stored in a memory. Also, for example, each the above described modules may be implemented with one or more application specific integrated circuits (ASICs), digital signal processors (DSPs), digital signal processing devices (DSPDs), programmable logic devices (PLDs), field programmable gate arrays (FPGAs), controllers, micro-controllers, microprocessors, or other electronic components, for performing the described methods. Further for example, each of the above described modules may be implemented by using a computer chip or an entity, or implemented by using a product having a certain function. In one embodiment, the apparatus 700 may be a computer, and the computer may be a personal computer, a laptop computer, a cellular phone, a camera phone, a smartphone, a personal digital assistant, a media player, a navigation device, an email receiving and sending device, a game console, a tablet computer, a wearable device, or any combination of these devices.

[00113] For an implementation process of functions and roles of each module in the apparatus 700, references can be made to corresponding steps in the above-described methods. Details are omitted here for simplicity.

[00114] In some embodiments, a computer program product may include a

non-transitory computer-readable storage medium having computer-readable program instructions thereon for causing a processor to carry out the above-described methods.

[00115] The computer-readable storage medium may be a tangible device that can store instructions for use by an instruction execution device. The computer-readable storage medium may be, for example, but is not limited to, an electronic storage device, a magnetic storage device, an optical storage device, an electromagnetic storage device, a semiconductor storage device, or any suitable combination of the foregoing. A non-exhaustive list of more specific examples of the computer-readable storage medium includes the following: a portable computer diskette, a hard disk, a random access memory (RAM), a read-only memory (ROM), an erasable programmable read-only memory (EPROM), a static random access memory (SRAM), a portable compact disc read-only memory (CD-ROM), a digital versatile disk (DVD), a memory stick, a floppy disk, a mechanically encoded device such as punch-cards or raised structures in a groove having instructions recorded thereon, and any suitable combination of the foregoing.

[00116] The computer-readable program instructions for carrying out the above-described methods may be assembler instructions, instruction-set-architecture (ISA) instructions, machine instructions, machine dependent instructions, microcode, firmware instructions, state-setting data, or source code or object code written in any combination of one or more programming languages, including an object oriented programming language, and conventional procedural programming languages. The computer-readable program instructions may execute entirely on a computing device as a stand-alone software package, or partly on a first computing device and partly on a second computing device remote from the first

computing device. In the latter scenario, the second, remote computing device may be connected to the first computing device through any type of network, including a local area network (LAN) or a wide area network (WAN).

[00117] The computer-readable program instructions may be provided to a processor of a general-purpose or special-purpose computer, or other programmable data processing apparatus to produce a machine, such that the instructions, which execute via the processor of the computer or other programmable data processing apparatus, create means for implementing the above-described methods.

[00118] The flow charts and diagrams in the figures illustrate the architecture, functionality, and operation of possible implementations of devices, methods, and computer program products according to various embodiments of the specification. In this regard, a block in the flow charts or diagrams may represent a software program, segment, or portion of code, which comprises one or more executable instructions for implementing specific functions. It should also be noted that, in some alternative implementations, the functions noted in the blocks may occur out of the order noted in the figures. For example, two blocks shown in succession may, in fact, be executed substantially concurrently, or the blocks may sometimes be executed in the reverse order, depending upon the functionality involved. It will also be noted that each block of the diagrams and/or flow charts, and combinations of blocks in the diagrams and flow charts, may be implemented by special purpose hardware-based systems that perform the specified functions or acts, or combinations of special purpose hardware and computer instructions.

[00119] It is appreciated that certain features of the specification, which are, for clarity, described in the context of separate embodiments, may also be provided in combination in a single embodiment. Conversely, various features of the

specification, which are, for brevity, described in the context of a single embodiment, may also be provided separately or in any suitable subcombination or as suitable in any other described embodiment of the specification. Certain features described in the context of various embodiments are not essential features of those embodiments, unless noted as such.

[00120] Although the specification has been described in conjunction with specific embodiments, many alternatives, modifications and variations will be apparent to those skilled in the art. Accordingly, the following claims embrace all such alternatives, modifications and variations that fall within the terms of the claims.

What Is Claimed Is:

1. A computer-implemented method for mitigating invoice financing fraud, the method comprising:

receiving a first transaction from a first user, the first transaction comprising a first commitment of an invoice and a first proof for proving the first user is an issuer of the invoice, the first commitment being generated based on a first token corresponding to the first user, the first token itself being excluded from the first transaction;

verifying the first proof;

receiving a second transaction from a second user, the second transaction and the first transaction containing different payload so as to unlink the second transaction and the first transaction, the second transaction comprising the first token corresponding to the first user, a second commitment of the invoice, and a second proof for proving the second user has certified the invoice, the second commitment being generated based on a second token corresponding to the second user and a certifying document generated by the second user;

determining that the first token is valid and verifying the second proof;

receiving a third transaction from a third user, the third transaction comprising the second token corresponding to the second user and a third proof generated by the third user; and

determining that the second token is valid and verifying the third proof, to determine no invoice financing fraud.

2. The method of claim 1, wherein the method is performed by one or more nodes in a blockchain system.

3. The method of claim 2, wherein the second user has a trusted identity registered on the blockchain system, the method further comprising:

verifying the trusted identity of the second user based on a signature utilized by the second user to sign the second transaction.

4. The method of any preceding claim, further comprising one of:

reporting a detection of fraud in response to a determination that the first proof is unacceptable;

reporting the detection of fraud in response to a determination that the first token is invalid or the second proof is unacceptable; or

reporting the detection of fraud in response to a determination that the second token is invalid or the third proof is unacceptable.

5. The method of any preceding claim, further comprising:

recording the first commitment of the invoice in a first invoice pool in response to a determination that the first proof is acceptable; and

recording the second commitment of the invoice in a second invoice pool and invaliding the first token in response to a determination that the first token is valid and that the second proof is acceptable.

6. The method of any preceding claim, further comprising:

invaliding the second token and reporting that no fraud is detected in response to a determination that the second token is valid and that the third proof is acceptable.

7. The method of any preceding claim, wherein the first proof, the second proof, and the third proof are zero-knowledge proofs.

8. The method of any preceding claim, wherein the first user is an issuer of the invoice and the first proof is a proof generated by the first user to prove that the first user is the issuer of the invoice.

9. The method of any preceding claim, wherein the second user is a trusted user and the second proof is a proof generated by the second user to prove that the second user has certified the invoice.

10. The method of any preceding claim, wherein the third user is a fund provider having received a request to provide invoice financing to the first user and the third proof is a proof generated by the third user to prove that the third user has verified validity of the request to provide invoice financing to the first user.

11. The method of any preceding claim, wherein the first commitment is generated by the first user with a first random value and the second commitment is generated by the second user with a second random value to unlink the first commitment and the second commitment.

12. A computer-implemented method for mitigating invoice financing fraud, the method comprising:

receiving a first transaction from a first user, the first transaction comprising a first commitment of an invoice and a first proof for proving the first user is an issuer of the invoice, the first commitment being generated based on a first token corresponding to the first user, the first token itself being excluded from the first transaction;

verifying the first proof;

receiving a second transaction from a second user, the second transaction and the first transaction containing different payload so as to unlink the second transaction and the first transaction, the second transaction comprising the first token corresponding to the first user, a second commitment of the invoice, and a second proof for proving the second user is a receiver of the invoice, the second commitment being generated based on a second token corresponding to the second user;

determining that the first token is valid and verifying the second proof;

receiving a third transaction from a third user, the third transaction comprising the second token corresponding to the second user, a third commitment of the invoice, and a third proof for proving the third user has reviewed and certified the invoice, the third commitment being generated based on a third token corresponding to the third user and a certifying document generated by the third user;

determining that the second token is valid and verifying the third proof;

receiving a fourth transaction from a fourth user, the fourth transaction comprising the third token corresponding to the third user and a fourth proof generated by the fourth user; and

determining that the third token is valid and verifying the fourth proof, to determine no invoice financing fraud.

13. The method of claim 12, wherein the method is performed by one or more nodes in a blockchain system, and the third user has a trusted identity registered on the blockchain system, the method further comprising:

verifying the trusted identity of the third user based on a signature utilized by the third user to sign the third transaction.

14. A device for mitigating invoice financing fraud, comprising:

one or more processors; and

one or more computer-readable memories coupled to the one or more processors and having instructions stored thereon that are executable by the one or more processors to perform the method of any of claims 1 to 13.

15. An apparatus for mitigating invoice financing fraud, the apparatus comprising a plurality of modules for performing the method of any of claims 1 to 13.

16. A non-transitory computer-readable medium having stored therein instructions that, when executed by a processor of a device, cause the device to perform the method of any of claims 1 to 13.