

METHODS AND DEVICES FOR MANAGING STANDBY LETTER OF CREDIT

TECHNICAL FIELD

[0001] The specification relates generally to computer technologies, and more particularly, to methods and devices for managing a standby letter of credit.

BACKGROUND

[0002] A standby letter of credit (SBLC) is generally a document that guarantees payment by a bank on behalf of the bank's client. SBLCs are commonly used to facilitate transactions, including, e.g., international transactions. In a typical use case, for example, a first client having a deposit account with a first bank may agree to serve as a guarantor for a loan which a second client wants to take out from a second bank. For the second bank to accept the first client as the guarantor, the second bank may require the first bank to issue an SBLC. The SBLC is used to guarantee the first bank's payment to the second bank if the second client defaults on repayment of the loan. If the second client defaults, the first bank may pay the second bank and recover its cost by charging against the first client's deposit account maintained by the first bank. In this example, the first bank may be referred to as the issuing bank, the second bank may be referred to as the beneficiary bank, the first client may be referred to as the principal, and the second client may be referred to as the borrower.

[0003] Because of the importance of SBLCs in facilitating certain types of transactions, parties involved in such transactions may have incentives to record the SBLCs in blockchain systems. Blockchain systems, also known as distributed ledger systems (DLSS) or consensus systems, may enable participating entities 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.

[0004] While recording SBLCs in blockchain systems may help prevent tampering and manipulation by malicious parties, doing so may also expose the information contained in the SBLCs to unrelated parties. Because SBLCs may contain sensitive information, including, e.g., the identities of the issuing bank, the beneficiary bank, the principal, the borrower, the maximum amount the borrower is allowed to borrow, the type(s) of loan the borrower is allowed to arrange and the like, recording SBLCs in blockchain systems may compromise privacy of the parties related to the SBLCs.

SUMMARY

[0005] In one aspect, a computer-implemented method for managing a standby letter of credit (SBLC) includes: receiving a first transaction from a first user, the first transaction comprising a first SBLC commitment and a first proof for proving the first user is an issuer of the SBLC, the first SBLC commitment being generated based on a first SBLC token; in response to a determination that the first proof is acceptable, recording the first SBLC commitment in an SBLC commitment pool; receiving a second transaction from a second user, the second transaction comprising the first SBLC token, a second SBLC commitment, a first loan

commitment, and a second proof for proving the second user has issued a loan secured by the SBLC, the second SBLC commitment being generated based on a second SBLC token and the first loan commitment being generated based on a first loan token; and in response to a determination that the first SBLC token is valid and the second proof is acceptable, recording the second SBLC commitment in the SBLC commitment pool, recording the first loan commitment in a loan commitment pool, and invalidating the first SBLC token.

[0006] In another aspect, a device for managing an SBLC 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 SBLC commitment and a first proof for proving the first user is an issuer of the SBLC, the first SBLC commitment being generated based on a first SBLC token; in response to a determination that the first proof is acceptable, record the first SBLC commitment in an SBLC commitment pool; receive a second transaction from a second user, the second transaction comprising the first SBLC token, a second SBLC commitment, a first loan commitment, and a second proof for proving the second user has issued a loan secured by the SBLC, the second SBLC commitment being generated based on a second SBLC token and the first loan commitment being generated based on a first loan token; and in response to a determination that the first SBLC token is valid and the second proof is acceptable, record the second SBLC commitment in the SBLC commitment pool, record the first loan commitment in a loan commitment pool, and invalidate the first SBLC token.

[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 managing an SBLC. The method includes: receiving a first transaction from a first user, the first transaction comprising a first SBLC commitment and a first proof for proving the first user is an issuer of the SBLC, the first SBLC commitment being generated based on a first SBLC token; in response to a determination that the first proof is acceptable, recording the first SBLC commitment in an SBLC commitment pool; receiving a second transaction from a second user, the second transaction comprising the first SBLC token, a second SBLC commitment, a first loan commitment, and a second proof for proving the second user has issued a loan secured by the SBLC, the second SBLC commitment being generated based on a second SBLC token and the first loan commitment being generated based on a first loan token; and in response to a determination that the first SBLC token is valid and the second proof is acceptable, recording the second SBLC commitment in the SBLC commitment pool, recording the first loan commitment in a loan commitment pool, and invalidating the first SBLC token.

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 managing a standby letter of credit, according an embodiment.

[0012] FIG. 4 is a flow chart of a method for managing a standby letter of credit, according an embodiment.

[0013] FIG. 5 is a block diagram of an apparatus for managing a standby letter of credit, according to an embodiment.

DETAILED DESCRIPTION

[0014] Embodiments of the specification provide methods and devices for managing standby letters of credit (SBLCs). The methods and devices may utilize blockchain systems to validate SBLCs submitted by users. Fake SBLCs can be detected and fraudulent attempts to borrow money using such SBLCs can be prevented. The methods and devices may also implement a protocol that can manage an SBLC. A borrower may be allowed to borrow up to the maximum amount allowed under the SBLC, but if it is determined that the borrower has already borrowed up to the maximum amount allowed, the methods and devices can prevent additional attempts to borrow money using the same SBLC. The methods and devices may further implement a protocol to protect privacy. In this manner, the methods and devices can validate and record the SBLC using the blockchain system without revealing the details of the SBLC to the public.

[0015] 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 SBLCs submitted by the users. This allows the methods and devices to verify correctness, detect fake SBLCs, and prevent fraudulent attempts to borrow money using fake SBLCs. In some embodiments, the methods

and devices support validation of SBLCs using a blockchain system. This allows the methods and devices to store SBLCs 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 uses two types of commitments, e.g., SBLC commitments and loan commitments, to manage an SBLC. This allows a borrower to borrow up to the maximum amount allowed under the SBLC, but if it is determined that the borrower has already borrowed up to the maximum amount allowed, the methods and devices can prevent additional attempts to borrow money using the same SBLC. In some embodiments, the methods and devices further support a protocol that only record SBLC commitments and loan commitments on the blockchain system. This allows the methods and devices to utilize the blockchain system to validate and record the SBLC without revealing any private information to the public, thereby preserving privacy of users who are parties to the SBLC.

[0016] 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.

[0017] A blockchain is a data structure that stores data, e.g., transactions, SBLCs, and the like, 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.

[0018] 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.

[0019] 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).

[0020] 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).

[0021] 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.

[0022] 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.

[0023] 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.

[0024] 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.

[0025] 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.

[0026] 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.

[0027] 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.

[0028] 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.

[0029] FIG. 3 illustrates a flow chart of a method 300 for managing a standby letter of credit (SBLC) 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.

[0030] For illustrative purposes, two users are depicted in FIG. 3, which includes an Issuing Bank and a Beneficiary Bank. A Principal having an account with the Issuing Bank may agree to serve as a guarantor for a loan which a Borrower wants to take out from the Beneficiary Bank. For the Beneficiary Bank to accept the Principal as the guarantor, the Beneficiary Bank may require the Issuing Bank to issue an SBLC. The SBLC can guarantee the Issuing Bank's payment to the Beneficiary Bank if the Borrower defaults on repayment of the loan. In some embodiments, the Beneficiary Bank may be willing to accept the SBLC issued by the Issuing Bank if the Issuing Bank and the Beneficiary Bank both agree to record and manage the SBLC based on the method 300.

[0031] At step 302, the Issuing Bank may issue an SBLC. The SBLC may be issued in various formats, including, e.g., digital or paper-based formats. The SBLC may contain information including, e.g., the identities of the Issuing Bank, the Beneficiary Bank, the Principal, the Borrower, the maximum amount the Borrower is allowed to borrow, the type(s) of loan the Borrower is allowed to arrange (e.g., long-

term, short-term, fixed-rate, or variable-interest), the expiration date of the SBLC, and the like. The SBLC may also include an identifier, which may be a number or an alphanumeric value that uniquely identifies the SBLC.

[0032] Once the SBLC is issued, the Issuing Bank may generate an SBLC commitment CM_{SBLC} for the SBLC at step 302 for recordation on the Blockchain. In some embodiments, the Issuing Bank may calculate CM_{SBLC} by taking into consideration information regarding the Issuing Bank, the Beneficiary Bank, and certain information contained in the SBLC. In some embodiments, CM_{SBLC} may be calculated based on a one-way function. One of ordinary skill in the art will understand that a function $g: \mathcal{D} \rightarrow \mathcal{R}$ 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.

[0033] For example, in some embodiments, CM_{SBLC} may be calculated as a hash value of $(PK_{IB}, PK_{BB}, ID_{SBLC}, V_{REMAIN}, V_{SUM}, D_{SBLC}, TN_{SBLC}, R_{SBLC})$, where PK_{IB} and PK_{BB} represent the publickeys of the Issuing Bank (IB) and the Beneficiary Bank (BB), respectively, and ID_{SBLC} represents the identifier that can be used to identify the SBLC. V_{REMAIN} may represent the amount that the Borrower is allowed to borrow from the Beneficiary Bank and V_{SUM} may represent the amount already borrowed by the Borrower. In some embodiments, the initial value of V_{REMAIN} may be set to the maximum amount the Issuing Bank is willing to guarantee and the initial value of V_{SUM} may be set to 0. As will be explained below, V_{REMAIN} may be

decreased and V_{SUM} may be increased each time the Borrower borrows money from the Beneficiary Bank. Also, as will be explained below, V_{SUM} may be decreased each time the Borrower repays the Beneficiary Bank. In some embodiments, V_{REMAIN} may remain unchanged when the Borrower repays the Beneficiary Bank because the Borrower has already borrowed against the total amount allowed under the SBLC.

[0034] Other variables used to calculate CM_{SBLC} may include D_{SBLC} , TN_{SBLC} , and R_{SBLC} . D_{SBLC} may represent a hash value of certain information contained in the SBLC, including, e.g., the identity of the Principal, the identity of the Borrower, the type(s) of loan the Borrower is allowed to arrange, the expiration date of the SBLC, and the like. TN_{SBLC} may represent a token number generated by the Issuing Bank that can be used to invalidate CM_{SBLC} later. In some embodiments, TN_{SBLC} may include a random number. However, it is to be understood that TN_{SBLC} may also include alphanumeric values, and in some embodiments, TN_{SBLC} may include a serial number. R_{SBLC} may represent a randomness factor generated by the Issuing Bank to further protect information contained in CM_{SBLC} . In some embodiments, R_{SBLC} may be a random number or a random alphanumeric value.

[0035] In some embodiments, the Issuing Bank may also generate a zero-knowledge proof, denoted as π_{SBLC} , 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 Issuing Bank is the prover, who may prove to the Blockchain, or a smart contract executing on the Blockchain, which serves as the verifier, that the Issuing Bank is the true issuer of the SBLC in question. The Issuing Bank may attempt to prove this

by indicating that: (1) CM_{SBLC} is well-formed and (2) the Issuing Bank is indeed the Issuing Bank itself.

[0036] In an embodiment, to prove that CM_{SBLC} is well-formed, the Issuing Bank may prove to the Blockchain that the commitment value of PK_{IB} , PK_{BB} , ID_{SBLC} , V_{REMAIN} , V_{SUM} , D_{SBLC} , TN_{SBLC} , and R_{SBLC} is CM_{SBLC} . In an embodiment, to prove that the Issuing Bank is the Issuing Bank itself, the Issuing Bank may prove to the Blockchain that the Issuing Bank is in possession of the Issuing Bank's private key, or that $PK_{IB} = h(SK_{IB})$, where SK_{IB} represents the private key that is only known to the Issuing Bank and $h()$ is the hash function used to calculate the public key based on the private key.

[0037] In some embodiments, the Issuing Bank 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 Issuing Bank may prove to the Blockchain that the Issuing 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. 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.

[0038] In some embodiments, the Issuing Bank may set x to CM_{SBLC} and set w

to a value generated based on PK_{IB} , PK_{BB} , ID_{SBLC} , V_{REMAIN} , D_{SBLC} , TN_{SBLC} , R_{SBLC} , and SK_{IB} . For example, w may be generated by concatenating PK_{IB} , PK_{BB} , ID_{SBLC} , V_{REMAIN} , D_{SBLC} , TN_{SBLC} , R_{SBLC} , and SK_{IB} together. In this manner, the Issuing Bank may generate π_{SBLC} using the secret and public inputs as well as the proving key to prove to the Blockchain that the Issuing Bank is in possession of the secret input w . In some embodiments, the Blockchain may be able to verify π_{SBLC} using the public input and the verification key generated in the setup phase. In some embodiments, if the Blockchain accepts the Issuing Bank's proof that the Issuing Bank is in possession of the secret input w , the Blockchain may accept the Issuing Bank's statements as true.

[0039] At step 304, the Issuing Bank may submit a transaction to the Blockchain with payload containing $\{CM_{SBLC}, \pi_{SBLC}\}$. For illustrative purposes, this transaction may be referred to as a "SBLC" transaction.

[0040] At step 306, the Blockchain may check π_{SBLC} to determine whether the Issuing Bank 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.

[0041] In some embodiments, a smart contract may be incorporated into the Blockchain to determine whether π_{SBLC} is acceptable. The smart contract may verify π_{SBLC} using the public input and the verification key. If π_{SBLC} cannot be verified, the smart contract may refuse to allow the Issuing Bank to proceed further. On the other hand, if π_{SBLC} can be verified, then the smart contract may determine that π_{SBLC} is acceptable and proceed to record CM_{SBLC} in an SBLC commitment pool. In some embodiments, the SBLC commitment pool may be structured as a Merkle tree T_{SBLC} and CM_{SBLC} may be recorded on a leaf node of the Merkle tree T_{SBLC} .

[0042] At step 308, the Issuing Bank may proceed to provide the Beneficiary Bank with the SBLC. In some embodiments, the Issuing Bank may send the SBLC and the values of TN_{SBLC} and R_{SBLC} privately to the Beneficiary Bank through an off-chain communication channel. The Issuing Bank may also send CM_{SBLC} to the Beneficiary Bank. Alternatively or additionally, the Beneficiary Bank may reconstruct CM_{SBLC} based on the SBLC and the values of TN_{SBLC} and R_{SBLC} received from the Issuing Bank.

[0043] At step 310, the Beneficiary Bank may examine the SBLC to determine whether it is willing to accept the SBLC before issuing a loan to the borrower. It is to be understood that the Beneficiary Bank may take various factors into consideration in determining whether to accept the SBLC. Such factors may include, e.g., the value of V_{REMAIN} , the value of V_{SUM} , the Principal's credit score and credit history, the Borrower's credit score and credit history, the Beneficiary Bank's prior dealings with the Issuing Bank, the Principal, the Borrower, and the like.

[0044] The Beneficiary Bank may refuse to accept the SBLC if the Beneficiary

Bank determines that CM_{SBLC} is not recorded in the SBLC commitment pool, or that the recorded CM_{SBLC} does not match the commitment value of PK_{IB} , PK_{BB} , ID_{SBLC} , V_{REMAIN} , V_{SUM} , D_{SBLC} , TN_{SBLC} , and R_{SBLC} . In this manner, if the Issuing Bank attempts to bypass steps 304-306, or attempts to change the SBLC or the values of TN_{SBLC} or R_{SBLC} , such attempts can be detected by the Beneficiary Bank, who can then refuse to accept the SBLC and prevent fraud.

[0045] If the Beneficiary Bank decides to accept the SBLC, then the Beneficiary Bank may proceed to issue one or more loans to the Borrower. If the Beneficiary Bank issues a loan to the Borrower, the Beneficiary Bank may generate a loan commitment CM_{SBLC} for recordation on the Blockchain. The Beneficiary Bank may also generate a new SBLC commitment CM_{SBLC}^{new} for recordation on the Blockchain. The new SBLC commitment CM_{SBLC}^{new} may reflect changes in values of V_{REMAIN} (which may be decreased each time the Borrower borrows money from the Beneficiary Bank) and V_{SUM} (which may be increased each time the Borrower borrows money from the Beneficiary Bank). The Beneficiary Bank may then request the Blockchain to record the new SBLC commitment CM_{SBLC}^{new} and invalidate the previously (or most recently) recorded SBLC commitment CM_{SBLC} . In this manner, CM_{SBLC}^{new} can effectively replace CM_{SBLC} .

[0046] In some embodiments, the Beneficiary Bank may request the Blockchain to invalidate CM_{SBLC} by presenting to the Blockchain the token number TN_{SBLC} contained in CM_{SBLC} . The Blockchain may add TN_{SBLC} to a used token list, indicating that TN_{SBLC} has been used or spent, which may in turn indicate that CM_{SBLC} is no longer valid.

[0047] The Beneficiary Bank may generate CM_{SBLC}^{new} based on the same one-

way function used by the Issuing Bank to generate CM_{SBLC} . For example, the Beneficiary Bank may calculate CM_{SBLC}^{new} as a hash value of $(PK_{IB}, PK_{BB}, ID_{SBLC}, V_{REMAIN}^{new}, V_{SUM}^{new}, D_{SBLC}, TN_{SBLC}^{new}, R_{SBLC}^{new})$, where PK_{IB} , PK_{BB} , ID_{SBLC} , and D_{SBLC} are the same as described above. The Beneficiary Bank may calculate the values of V_{REMAIN}^{new} and V_{SUM}^{new} based on the loan issued to the Borrower. If the loan amount is V_{LOAN} , for example, then the values of V_{REMAIN}^{new} and V_{SUM}^{new} may be calculated as $V_{REMAIN}^{new} = V_{REMAIN} - V_{LOAN}$ and $V_{SUM}^{new} = V_{SUM} + V_{LOAN}$, respectively. The Beneficiary Bank may also generate new values of TN_{SBLC}^{new} and R_{SBLC}^{new} . In this manner, the Beneficiary Bank may calculate a commitment value that is different from the commitment value calculated by the Issuing Bank, effectively preventing an unrelated user from linking the two commitment values together, which in turn prevents the unrelated user from linking the Issuing Bank and the Beneficiary Bank together.

[0048] The Beneficiary Bank may also generate the loan commitment CM_{LOAN} based on a one-way function. In some embodiments, CM_{LOAN} may be calculated as a hash value of $(PK_{BB}, V_{LOAN}, D_{LOAN}, ID_{SBLC}, TN_{LOAN}, R_{LOAN})$, where PK_{BB} may represent the public key of the Beneficiary Bank (BB), V_{LOAN} may represent the outstanding amount of the loan, and D_{LOAN} may represent a hash value of certain information contained in the loan, including, e.g., the identity of the Borrower, the issue date, the maturity date, reference documents, other terms and conditions of the loan and the like. ID_{SBLC} may represent the identifier that can be used to identify the SBLC, TN_{LOAN} may represent a token number generated by the Beneficiary Bank that can be used to invalidate CM_{LOAN} later, and R_{LOAN} may represent a random factor generated by the Beneficiary Bank to further protect information contained in

CM_{LOAN} .

[0049] The Beneficiary Bank may further generate a zero-knowledge proof, denoted as π_{LOAN} , at step 310 to prove to the Blockchain that the Beneficiary Bank has issued a loan secured by the SBLC. In some embodiments, the Beneficiary Bank may generate π_{LOAN} when the Beneficiary Bank has determined that: (1) CM_{SBLC} , CM_{LOAN} , and CM_{SBLC}^{new} are well-formed (i.e., the commitment values are calculated correctly), (2) CM_{SBLC} is recorded in the SBLC commitment pool (e.g., the Merkle tree T_{SBLC}), (3) $V_{REMAIN}^{new} = V_{REMAIN} - V_{LOAN}$ and $V_{REMAIN}^{new} \geq 0$, (4) $V_{SUM}^{new} = V_{SUM} + V_{LOAN}$, and (5) the Beneficiary Bank is in possession of its private key SK_{BB} and that $PK_{BB} = h(SK_{BB})$.

[0050] In some embodiments, the Beneficiary Bank and the Blockchain may agree to implement the zero-knowledge proof techniques described above. For example, the Beneficiary Bank may prove to the Blockchain that the Beneficiary 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 Beneficiary Bank may set x to a value containing TN_{SBLC} , CM_{LOAN} , and CM_{SBLC}^{new} , and set w to a value generated based on CM_{SBLC} , PK_{IB} , PK_{BB} , ID_{SBLC} , D_{SBLC} , D_{LOAN} , V_{REMAIN} , V_{REMAIN}^{new} , V_{SUM} , V_{SUM}^{new} , V_{LOAN} , TN_{SBLC} , TN_{SBLC}^{new} , TN_{LOAN} , R_{SBLC} , R_{SBLC}^{new} , R_{LOAN} , and SK_{BB} . In this manner, the Beneficiary Bank may be able to generate a zero-knowledge proof π_{LOAN} to prove to the Blockchain that the Beneficiary Bank is in possession of the secret input w . In some embodiments, if the Blockchain accepts the Beneficiary Bank's proof that the Beneficiary Bank is in possession of the secret input w , the Blockchain may accept the Beneficiary Bank's statements as true.

[0051] At step 312, the Beneficiary Bank may submit a transaction to the

Blockchain with payload containing $\{TN_{SBLC}, CM_{SBLC}^{new}, CM_{LOAN}, \pi_{LOAN}\}$. For illustrative purposes, this transaction may be referred to as a “LOAN” transaction.

[0052] At step 314, the Blockchain may utilize a smart contract to check whether TN_{SBLC} 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_{SBLC} contained in the payload is listed in the used token list, the smart contract may refuse to proceed further because the Beneficiary Bank is attempting to use a token that has already been used. On the other hand, if TN_{SBLC} contained in the payload is not listed in the used token list, the smart contract may proceed to verify whether π_{LOAN} is acceptable. The smart contract may verify π_{LOAN} using the public input and the verification key.

[0053] If π_{LOAN} can be verified, then the smart contract may determine that π_{LOAN} is acceptable and proceed to record CM_{SBLC}^{new} in the SBLC commitment pool (described above) and record CM_{LOAN} in a loan commitment pool. In some embodiments, the loan commitment pool may be structured as a Merkle tree T_{LOAN} and CM_{LOAN} may be recorded on a leaf node of the Merkle tree T_{LOAN} . The smart contract may also add TN_{SBLC} to the used token list to indicate that TN_{SBLC} has been used, thereby invalidating TN_{SBLC} (and hence, CM_{SBLC}) for future use. In this manner, CM_{SBLC}^{new} may become the most recently recorded SBLC commitment for the SBLC and may therefore be referred to as CM_{SBLC} in subsequent steps. Also, in this manner, information regarding the SBLC and the loan issued against the SBLC can be verified and recorded on the Blockchain using the method 300 without revealing the details of the SBLC to the public.

[0054] In some embodiments, the Beneficiary Bank may repeat steps 310-312

if the Beneficiary Bank issues another loan to the Borrower, and the Blockchain may repeat step 314 in response.

[0055] In some embodiments, the method 300 may include additional steps defined to facilitate recordation of repayment information on the Blockchain. For example, the Beneficiary Bank may utilize the method 300 to record repayment information when a repayment is received from the Borrower. Alternatively, the Beneficiary Bank may utilize the method 300 to record repayment information according to a schedule (e.g., monthly, quarterly, or annually).

[0056] To record repayment information, at step 316, the Beneficiary Bank may generate a new loan commitment CM_{LOAN}^{new} for recordation on the Blockchain and request the Blockchain to invalidate the most recently recorded loan commitment CM_{LOAN} . The Beneficiary Bank may also generate a new SBLC commitment CM_{SBLC}^{new} for recordation on the Blockchain and request the Blockchain to invalidate the most recently recorded SBLC commitment CM_{SBLC} (e.g., the SBLC commitment previously generated at step 310 and recorded at step 314).

[0057] The Beneficiary Bank may generate CM_{LOAN}^{new} based on the same one-way function used by the Beneficiary Bank to generate CM_{LOAN} . For example, the Beneficiary Bank may calculate CM_{LOAN}^{new} as a hash value of $(PK_{BB}, V_{LOAN}^{new}, D_{LOAN}, ID_{SBLC}, TN_{LOAN}^{new}, R_{LOAN}^{new})$, where PK_{BB} , D_{loan} , and ID_{SBLC} are the same as described above. The Beneficiary Bank may calculate the values of V_{LOAN}^{new} based on the repayment(s) made by the Borrower. In some embodiments, the value of V_{LOAN}^{new} may be reduced by the repayment amount $V_{REPAYMENT}$ so that $V_{LOAN}^{new} = V_{LOAN} - V_{REPAYMENT}$. The Beneficiary Bank may also generate new values of TN_{LOAN}^{new} and R_{LOAN}^{new} as described above.

[0058] The Beneficiary Bank may generate CM_{LOAN}^{new} in manners similar to that described above. For example, the Beneficiary Bank may calculate CM_{LOAN}^{new} as a hash value of $(PK_{IB}, PK_{BB}, ID_{SBLC}, V_{REMAIN}^{new}, V_{SUM}^{new}, D_{SBLC}, TN_{SBLC}^{new}, R_{SBLC}^{new})$, where PK_{IB} , PK_{BB} , ID_{SBLC} , and D_{SBLC} are the same as described above. The Beneficiary Bank may calculate the values of V_{REMAIN}^{new} and V_{SUM}^{new} based on the repayment(s) made by the Borrower. In some embodiments, the value of V_{REMAIN}^{new} may remain unchanged when the Borrower repays an existing loan. The value of V_{SUM}^{new} , on the other hand, may be reduced by the repayment amount $V_{REPAYMENT}$ so that $V_{SUM}^{new} = V_{SUM} - V_{REPAYMENT}$. The Beneficiary Bank may also generate new values of TN_{SBLC}^{new} and R_{SBLC}^{new} as described above.

[0059] The Beneficiary Bank may request the Blockchain to invalidate CM_{LOAN} by presenting to the Blockchain the token number TN_{LOAN} contained in CM_{LOAN} . Similarly, the Beneficiary Bank may request the Blockchain to invalidate CM_{SBLC} by presenting to the Blockchain the token number TN_{SBLC} contained in CM_{SBLC} .

[0060] The Beneficiary Bank may further generate a zero-knowledge proof, denoted as $\pi_{REPAYMENT}$, at step 316 to prove to the Blockchain that the Beneficiary Bank has received a repayment of a loan secured by the SBLC. In some embodiments, the Beneficiary Bank may generate $\pi_{REPAYMENT}$ when it has determined that: (1) CM_{SBLC} , CM_{SBLC}^{new} , CM_{LOAN} , and CM_{LOAN}^{new} are well-formed, (2) CM_{SBLC} is recorded in the SBLC commitment pool (e.g., the Merkle tree T_{SBLC}), (3) CM_{LOAN} is recorded in the loan commitment pool (e.g., the Merkle tree T_{LOAN}), (4) $V_{SUM}^{new} = V_{SUM} - V_{REPAYMENT}$, (5) $V_{LOAN}^{new} = V_{LOAN} - V_{REPAYMENT}$, and (6) the Beneficiary Bank is in possession of its private key SK_{BB} and that $PK_{BB} = h(SK_{BB})$.

[0061] In some embodiments, the Beneficiary Bank and the Blockchain may

agree to implement the zero-knowledge proof techniques described above. For example, the Beneficiary Bank may prove to the Blockchain that the Beneficiary 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 Beneficiary Bank may set x to a value containing TN_{SBLC} , CM_{LOAN}^{new} , and CM_{SBLC}^{new} , and set w to a value generated based on CM_{SBLC} , CM_{LOAN} , PK_{IB} , PK_{BB} , ID_{SBLC} , D_{SBLC} , D_{LOAN} , V_{REMAIN} , V_{SUM} , V_{SUM}^{new} , V_{LOAN} , V_{LOAN}^{new} , $V_{REPAYMENT}$, TN_{SBLC} , TN_{SBLC}^{new} , TN_{LOAN} , TN_{LOAN}^{new} , R_{SBLC} , R_{SBLC}^{new} , R_{LOAN} , R_{LOAN}^{new} , and SK_{BB} . In this manner, the Beneficiary Bank may be able to generate a zero-knowledge proof $\pi_{REPAYMENT}$ to prove to the Blockchain that the Beneficiary Bank is in possession of the secret input w . In some embodiments, if the Blockchain accepts the Beneficiary Bank's proof that the Beneficiary Bank is in possession of the secret input w , the Blockchain may accept the Beneficiary Bank's statements as true.

[0062] At step 318, the Beneficiary Bank may submit a transaction to the Blockchain with payload containing $\{TN_{SBLC}, TN_{LOAN}, CM_{SBLC}^{new}, CM_{LOAN}^{new}, \pi_{REPAYMENT}\}$. For illustrative purposes, this transaction may be referred to as a "REPAYMENT" transaction.

[0063] At step 320, the Blockchain may utilize a smart contract to check whether TN_{SBLC} or TN_{LOAN} contained in the payload have been used or spent already. If TN_{SBLC} or TN_{LOAN} contained in the payload have been used or spent already (e.g., listed in the used token list), the smart contract may refuse to proceed further. On the other hand, if neither TN_{SBLC} nor TN_{LOAN} is listed in the used token list, the smart contract may proceed to verify whether $\pi_{REPAYMENT}$ is acceptable. The smart contract may verify $\pi_{REPAYMENT}$ using the public input and the verification key.

If $\pi_{REPAYMENT}$ can be verified, then the smart contract may determine that $\pi_{REPAYMENT}$ is acceptable and proceed to record CM_{SBLC}^{new} in the SBLC commitment pool (described above) and record CM_{LOAN}^{new} in the loan commitment pool (described above). The smart contract may also add TN_{SBLC} and TN_{LOAN} to the used token list to indicate that TN_{SBLC} and TN_{LOAN} have been used, thereby invalidating TN_{SBLC} and TN_{LOAN} for future use. In this manner, CM_{SBLC}^{new} and CM_{LOAN}^{new} may become the most recently recorded SBLC commitment and loan commitment, which may be referred to as CM_{SBLC} and CM_{LOAN} , respectively, in subsequent steps. Also, in this manner, information regarding the SBLC and repayments made by the Borrower can be verified and recorded on the Blockchain using the method 300 without revealing the details of the SBLC to the public.

[0064] In some embodiments, the Beneficiary Bank may repeat steps 316-318 to record repayment information when a repayment is received from the Borrower. Alternatively, the Beneficiary Bank may repeat steps 316-318 to record repayment information according to a schedule (e.g., monthly, quarterly, or annually). The Blockchain may repeat step 320 in response.

[0065] In some embodiments, the method 300 may also support recording of amendments made to an SBLC. In some embodiments, only the Issuing Bank may have permission to amend certain information contained in the SBLC. Such information may include, e.g., the maximum amount the Borrower is allowed to borrow, the type(s) of loan the Borrower is allowed to arrange, and the expiration date of the SBLC.

[0066] In some embodiments, when the Issuing Bank amends an SBLC, the Issuing Bank may generate a new SBLC commitment CM_{SBLC}^{new} for the amended SBLC

using the same one-way function used by the Issuing Bank to generate CM_{SBLC} at step 302. The new SBLC commitment CM_{SBLC}^{new} may be generated to reflect the amended values of, e.g., V_{REMAIN} and other values contained in D_{SBLC} . The Beneficiary Bank may then request the Blockchain to record the new SBLC commitment CM_{SBLC}^{new} and invalidate the most recently recorded SBLC commitment CM_{SBLC} . In this manner, CM_{SBLC}^{new} can effectively replace CM_{SBLC} .

[0067] The Issuing Bank may further generate a zero-knowledge proof π_{AMEND} . In some embodiments, the Issuing Bank may generate π_{AMEND} when the Issuing Bank has determined that: (1) CM_{SBLC} and CM_{SBLC}^{new} are well-formed (i.e., the commitment values are calculated correctly), (2) CM_{SBLC} is recorded in the SBLC commitment pool (e.g., the Merkle tree T_{SBLC}), (3) the Issuing Bank is in possession of its private key SK_{IB} and that $PK_{IB} = h(SK_{IB})$, and (4) PK_{IB} , PK_{BB} , ID_{SBLC} , and V_{SUM} are the same in CM_{SBLC} and CM_{SBLC}^{new} .

[0068] The Issuing Bank may then submit a transaction to the Blockchain with payload containing $\{TN_{SBLC}, CM_{SBLC}^{new}, \pi_{AMEND}\}$. This transaction may be referred to as an “AMENDMENT” transaction. The Blockchain may utilize a smart contract to check whether TN_{SBLC} (i.e., the token associated with the most recently recorded SBLC commitment) contained in the payload has been used or spent already. If TN_{SBLC} contained in the payload is listed in the used token list, the smart contract may refuse to proceed further. On the other hand, if TN_{SBLC} contained in the payload is not listed in the used token list, the smart contract may proceed to verify whether π_{AMEND} is acceptable.

[0069] If π_{AMEND} can be verified, then the smart contract may determine that π_{AMEND} is acceptable and proceed to record CM_{SBLC}^{new} in the SBLC commitment pool

(described above). The smart contract may also add TN_{SBLC} to the used token list to indicate that TN_{SBLC} has been used, thereby invalidating TN_{SBLC} (and hence, CM_{SBLC}) for future use. In this manner, CM_{SBLC}^{new} may become the most recently recorded SBLC commitment for the amended SBLC. Also, in this manner, information regarding the amended SBLC can be verified and recorded on the Blockchain using the method 300 without revealing the details of the amended SBLC to the public.

[0070] In some embodiments, the method 300 may further support recording of cancellation of an SBLC. In some embodiments, only the Issuing Bank may have the permission to cancel the SBLC, provided that there is no outstanding loan against the SBLC (e.g., $V_{SUM} = 0$). In some embodiments, recording of cancellation of an SBLC may be performed if the SBLC has a corresponding SBLC commitment CM_{SBLC} recorded on the Blockchain. In such a case, the Issuing Bank may simply request the Blockchain to invalidate the token TN_{SBLC} contained in CM_{SBLC} . However, in some embodiments, the Issuing Bank may still be required to generate a zero-knowledge proof π_{CANCEL} in order to carry out this cancellation process.

[0071] In some embodiments, the Issuing Bank may generate π_{CANCEL} when the Issuing Bank has determined that: (1) CM_{SBLC} is well-formed (i.e., the commitment value is calculated correctly), (2) CM_{SBLC} is recorded in the SBLC commitment pool (e.g., the Merkle tree T_{SBLC}), (3) the Issuing Bank is in possession of its private key SK_{IB} and that $PK_{IB} = h(SK_{IB})$, and (4) the total loan amount V_{SUM} contained in CM_{SBLC} is equal to 0.

[0072] The Issuing Bank may then submit a transaction to the Blockchain with payload containing $\{TN_{SBLC}, \pi_{CANCEL}\}$. This transaction may be referred to as a “CANCELLATION” transaction. The Blockchain may utilize a smart contract to

check whether TN_{SBLC} contained in the payload has been used or spent already. If TN_{SBLC} contained in the payload is listed in the used token list, the smart contract may refuse to proceed further. On the other hand, if TN_{SBLC} contained in the payload is not listed in the used token list, the smart contract may proceed to verify whether π_{CANCEL} is acceptable. If π_{CANCEL} can be verified, then the smart contract may determine that π_{CANCEL} is acceptable and proceed to add TN_{SBLC} to the used token list to indicate that TN_{SBLC} has been used, thereby invalidating TN_{SBLC} (and hence, CM_{SBLC}) for future use.

[0073] 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 for SBLC commitments and a separate used token list may be utilized to record used tokens generated for loan commitments. 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.

[0074] FIG. 4 illustrates a flow chart of a method 400 for managing an SBLC, according to an embodiment. The method 400 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.

[0075] At step 402, 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 Issuing Bank (FIG. 3), who is the issuer of the SBLC. The first transaction may include the “SBLC” transaction (FIG. 3) described above, which may include a first SBLC commitment CM_{SBLC} and a first proof π_{SBLC} for proving the first user is the issuer of the SBLC. As described above, the first SBLC commitment CM_{SBLC} may be generated based on a first SBLC token TN_{SBLC} .

[0076] At step 404, the node 102 may determine whether the first proof π_{SBLC} is acceptable. In some embodiments, the first proof π_{SBLC} may be a zero-knowledge proof. In some embodiments, the first user may use the first proof π_{SBLC} to prove to the node 102 that the first user is in possession of the secret input w described above. In some embodiments, if the node 102 determines that the first proof is unacceptable, the node 102 may report an error. On the other hand, if the node 102 determines that the first proof is acceptable, the node 102 may record the first SBLC commitment CM_{SBLC} in an SBLC commitment pool T_{SBLC} .

[0077] At step 406, the node 102 may receive a second transaction submitted by a second user. The second user may be, e.g., the Beneficiary Bank (FIG. 3), who issued a loan secured by the SBLC. The second transaction may include the first SBLC token TN_{SBLC} , a second SBLC commitment CM_{SBLC}^{new} , a first loan commitment CM_{LOAN} , and a second proof π_{LOAN} . As described above, the second SBLC commitment CM_{SBLC}^{new} may be generated based on a second SBLC token TN_{SBLC}^{new} and the first loan commitment CM_{LOAN} may be generated based on a first loan token TN_{LOAN} .

[0078] At step 408, the node 102 may determine whether the first SBLC token TN_{SBLC} is valid and whether the second proof π_{LOAN} is acceptable. In some

embodiments, the second proof π_{LOAN} may be a zero-knowledge proof. In some embodiments, the second user may use the second proof π_{LOAN} to prove to the node 102 that the second user is in possession of the secret input w described above. In some embodiments, if the node 102 determines that the second proof is unacceptable, the node 102 may report an error. On the other hand, if the node 102 determines that the second proof is acceptable, the node 102 may record the second SBLC commitment CM_{SBLC}^{new} in the SBLC commitment pool T_{SBLC} , recording the first loan commitment CM_{LOAN} in a loan commitment pool T_{LOAN} , and invalidate the first SBLC token TN_{SBLC} . In this manner, CM_{SBLC}^{new} may become the most recently recorded SBLC commitment for the SBLC and may therefore be referred to as CM_{SBLC} in subsequent steps.

[0079] At step 410, the node 102 may receive a third transaction submitted by the second user. The third transaction may include the second SBLC token, the first loan token, a third SBLC commitment, a second loan commitment, and a third proof $\pi_{REPAYMENT}$ for proving the second user has received a repayment of the loan secured by the SBLC. In some embodiments, the third SBLC commitment may be generated based on a third SBLC token and the second loan commitment may be generated based on a second loan token.

[0080] At step 412, the node 102 may determine whether the second SBLC token and the first loan token are valid and whether the third proof $\pi_{REPAYMENT}$ is acceptable. In some embodiments, the third proof $\pi_{REPAYMENT}$ may be a zero-knowledge proof. In some embodiments, the second user may use the third proof $\pi_{REPAYMENT}$ to prove to the node 102 that the second user is in possession of the secret input w described above. In some embodiments, if the node 102 determines

that the third proof $\pi_{REPAYMENT}$ is unacceptable, the node 102 may report an error. On the other hand, if the node 102 determines that the third proof $\pi_{REPAYMENT}$ is acceptable, the node 102 may record the third SBLC commitment in the SBLC commitment pool, record the second loan commitment in the loan commitment pool, and invalidate the second SBLC token and the first loan token.

[0081] In some embodiments, the method 400 may further support reception of amendment and cancellation transactions described above. For example, at step 414, the node 102 may receive an amendment transaction submitted by the first user. The amendment transaction may include an SBLC token associated with the most recently recorded SBLC commitment, an amended SBLC commitment, and an amendment proof for proving the first user is the issuer of the SBLC. The node 102 may determine whether the SBLC token associated with the most recently recorded SBLC commitment is valid and the amendment proof is acceptable. In response to a determination that the SBLC token associated with the most recently recorded SBLC commitment is valid and the amendment proof is acceptable, the node 102 may record the amended SBLC commitment in the SBLC commitment pool and invalidate the SBLC token associated with the most recently recorded SBLC commitment.

[0082] In another example, at step 416, the node 102 may receive a cancellation transaction from the first user. The cancellation transaction may include an SBLC token associated with the most recently recorded SBLC commitment and a cancellation proof for proving the first user is the issuer of the SBLC. The node 102 may determine whether the SBLC token associated with the most recently recorded SBLC commitment is valid and the cancellation proof is acceptable. In response to a determination that the SBLC token associated with the most recently recorded SBLC commitment is valid and the cancellation proof is acceptable, the node 102 may

invalidate the SBLC token associated with the most recently recorded SBLC commitment.

[0083] FIG. 5 is a block diagram of an SBLC management apparatus 500, according to an embodiment. The apparatus 500 may be an implementation of a software process, and may correspond to the method 400 (FIG. 4). Referring to FIG. 5, the apparatus 500 may include a receiving module 502, a determination module 504, a recording module 506, and a reporting module 508.

[0084] The receiving module 502 may receive transactions submitted by users and provide the received transactions to the determining module 504, such as perform steps 402, 406, 410, 414, and 416 (FIG. 4). The determining module 504 may determine whether the tokens contained in the transactions are still valid and whether the proofs contained in the transactions are acceptable, such as perform steps 404, 408, 412, 414, and 416 (FIG. 4). If one or more tokens contained in a transaction is invalid or the proof contained in the transaction is unacceptable, the determining module 504 may request the reporting module 508 to report an error; otherwise, the determining module 504 may request the recording module 506 to record the transaction and invalidate the token(s) contained in the transaction, as described above.

[0085] 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 500 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.

[0086] For an implementation process of functions and roles of each module in the apparatus 500, references can be made to corresponding steps in the above-described methods. Details are omitted here for simplicity.

[0087] 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.

[0088] 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.

[0089] 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).

[0090] 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

[0091] 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.

[0092] 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.

[0093] 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 managing a standby letter of credit (SBLC), the method comprising:

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

in response to a determination that the first proof is acceptable, recording the first SBLC commitment in an SBLC commitment pool;

receiving a second transaction from a second user, the second transaction comprising the first SBLC token, a second SBLC commitment, a first loan commitment, and a second proof for proving the second user has issued a loan secured by the SBLC, the second SBLC commitment being generated based on a second SBLC token and the first loan commitment being generated based on a first loan token; and

in response to a determination that the first SBLC token is valid and the second proof is acceptable, recording the second SBLC commitment in the SBLC commitment pool, recording the first loan commitment in a loan commitment pool, and invalidating the first SBLC token.

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

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

reporting an error in response to a determination that the first proof is unacceptable; or

reporting the error in response to a determination that the first SBLC token is invalid or the second proof is unacceptable.

4. The method of any preceding claim, wherein each of the first SBLC commitment, the second SBLC commitment, and the first loan commitment is generated with a random value to unlink the first SBLC commitment, the second SBLC commitment, and the first loan commitment.

5. The method of any preceding claim, wherein whether the first SBLC token is valid is determined based on whether the first SBLC token is listed in a used token list.

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

receiving a third transaction from the second user, the third transaction comprising the second SBLC token, the first loan token, a third SBLC commitment, a second loan commitment, and a third proof for proving the second user has received a repayment of the loan secured by the SBLC, the third SBLC commitment being generated based on a third SBLC token and the second loan commitment being generated based on a second loan token; and

in response to a determination that the second SBLC token and the first loan token are valid and the third proof is acceptable, recording the third SBLC commitment in the SBLC commitment pool, recording the second loan commitment

in the loan commitment pool, and invalidating the second SBLC token and the first loan token.

7. The method of claim 6, further comprising:

reporting an error in response to a determination that the second SBLC token is invalid, the first loan token is invalid, or the third proof is unacceptable.

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

receiving an amendment transaction from the first user, the amendment transaction comprising an SBLC token associated with a most recently recorded SBLC commitment, an amended SBLC commitment, and an amendment proof for proving the first user is the issuer of the SBLC; and

in response to a determination that the SBLC token associated with the most recently recorded SBLC commitment is valid and the amendment proof is acceptable, recording the amended SBLC commitment in the SBLC commitment pool and invalidating the SBLC token associated with the most recently recorded SBLC commitment.

9. The method of claim 8, further comprising:

reporting an error in response to a determination that the SBLC token associated with the most recently recorded SBLC commitment is invalid or the amendment proof is unacceptable.

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

receiving a cancellation transaction from the first user, the cancellation transaction comprising an SBLC token associated with a most recently recorded SBLC commitment and a cancellation proof for proving the first user is the issuer of the SBLC; and

in response to a determination that the SBLC token associated with the most recently recorded SBLC commitment is valid and the cancellation proof is acceptable, invalidating the SBLC token associated with the most recently recorded SBLC commitment.

11. The method of claim 10, further comprising:

reporting an error in response to a determination that the SBLC token associated with the most recently recorded SBLC commitment is invalid or the cancellation proof is unacceptable.

12. The method of any preceding claim, wherein the proofs are zero-knowledge proofs.

13. A device for managing a standby letter of credit (SBLC), 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 12.

14. An apparatus for managing a standby letter of credit (SBLC), the apparatus comprising a plurality of modules for performing the method of any of claims 1 to 12.

15. 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 12.