METHODS AND DEVICES FOR DEADLOCK DETECTION IN REAL-TIME GROSS SETTLEMENT SYSTEMS

TECHNICAL FIELD

[0001] The specification relates generally to computer technologies, and more particularly, to methods and devices for deadlock detection in real-time gross settlement systems.

BACKGROUND

[0002] Real-time gross settlement (RTGS) systems are fund transfer systems where a transfer of funds (e.g., money or securities) takes place from one party (e.g., a bank) to another party (e.g., another bank) on a real-time and gross basis. When a settlement in the transfer of funds occurs, the transfers is generally final and irrevocable. The settlement is in real-time when, for example, a payment transaction is not subject to any waiting period. The settlement is on a gross basis when, for example, a payment transaction is settled on one-to-one basis without bundling or netting with any other transaction. RTGS systems may be operated by a country's central bank.

[0003] There are situations where one or more parties, e.g., banks, participating in an RTGS system may have insufficient liquidity to settle their payment transactions. For example, suppose that Bank A is set to receive $15 from Bank B and transfer $20 to Bank C, and further suppose that Bank A has $0 liquidity before settlement can take

place. Bank A in this situation is not able to settle the transactions unless Bank A receives additional supply of liquidity. Bank A in this situation is deemed to have caused a deadlock in the RTGS system.

[0004] Currently, a central bank operating the RTGS system may detect possible deadlocks. The central bank can detect deadlocks because it has the authority to obtain liquidity and transaction information from the parties, e.g., banks, participating in the RTGS system. However, with decentralized computing becoming a trend, technologies such as blockchains and the like are becoming more prevalent. Existing RTGS systems implemented based on decentralized computing technologies, such as blockchains and the like, lack the abilities to detect deadlocks.

[0005] 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.

[0006] 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. A blockchain is a data structure for storing data, such as transactions, that may prevent tampering and manipulation of the data by malicious parties.

[0007] Unlike RTGS systems operated by a central bank, where participating parties are willing to, or required to, disclose their liquidity and transaction information to the central bank, participating parties in a blockchain system may not be willing to disclose such information to other participating parties in the blockchain system. This lack of disclosure makes it difficult for RTGS systems implemented based on decentralized computing technologies, such as blockchains and the like, to detect deadlocks. Therefore, there is a need for privacy-preserving deadlock detection methods that are capable of detecting deadlocks without requiring any participating party to disclose their liquidity and transaction information.

SUMMARY

[0008] In one aspect, a computer-implemented method for detecting deadlocks in a real-time gross settlement system includes: instructing a plurality of users of the real-time gross settlement system to jointly determine a minimum post transfer liquidity, wherein the minimum post transfer liquidity is jointly determined based on a post transfer liquidity of each of the plurality of users without disclosing by each of the plurality of users of its post transfer liquidity to another user; receiving the jointly determined minimum post transfer liquidity; and determining whether a deadlock exists based on the minimum post transfer liquidity.

[0009] In another aspect, a device for detecting deadlocks in a real-time gross settlement system 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 instruct a plurality of users of the real-time gross settlement system to jointly determine a minimum post transfer liquidity, wherein the minimum post transfer liquidity is jointly determined based on a post transfer liquidity of each of the plurality of users without disclosing by each of the plurality of users of its post transfer liquidity to another user; receive the jointly determined minimum post transfer liquidity; and determine whether a deadlock exists based on the minimum post transfer liquidity.

[0010] 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 detecting deadlocks in a real-time gross settlement system. The method includes: instructing a plurality of users of the real-time gross settlement system to jointly determine a minimum post transfer liquidity, wherein the minimum post transfer liquidity is jointly determined based on a post transfer liquidity of each of the plurality of users without disclosing by each of the plurality of users of its post transfer liquidity to another user; receiving the jointly determined minimum post transfer liquidity; and determining whether a deadlock exists based on the minimum post transfer liquidity.

BRIEF DESCRIPTION OF THE DRAWINGS

[0011] 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.

[0012] FIG. 1 is a schematic diagram of a blockchain system, according to an embodiment.

[0013] FIG. 2 is a schematic diagram of a computing device for implementing a node in a blockchain system, according to an embodiment.

[0014] FIG. 3 is a flow chart of a privacy-preserving deadlock detection method, according an embodiment.

[0015] FIG. 4 is a flow chart of a method for detecting real-time gross settlement deadlocks, according an embodiment.

[0016] FIG. 5 is a block diagram of an apparatus for detecting real-time gross settlement deadlocks, according to an embodiment.

DETAILED DESCRIPTION

[0017] Embodiments of the specification provide methods and devices for providing privacy-preserving deadlock detection in real-time gross settlement (RTGS) systems. The methods and devices support local calculation of liquidity information by users of an RTGS system. The methods and devices also support a multiparty computation in which multiple users of the RTGS system can jointly determine the minimum liquidity amount without requiring any user to disclose its own liquidity information. In this manner, the determined minimum liquidity amount can be utilized to detect deadlocks in the RTGS system and still preserve privacy of each user.

[0018] Embodiments disclosed in the specification have one or more technical effects. In some embodiments, the methods and devices support local calculation of liquidity information by users of the RTGS system. This allows each user to perform calculations locally without having to disclose any private information that the user does not wish to share. In some embodiments, the methods and devices also support a multiparty computation that can be carried out without requiring any user to disclose its own liquidity information to any other user. This allows the users to jointly carry out the computation while preserving their privacy. Furthermore, in some embodiments, the methods and devices can detect deadlocks in RTGS systems implemented based on decentralized computing technologies, such as blockchains and the like. This allows the RTGS systems implemented based on decentralized computing technologies to take appropriate measures to address the detected deadlocks.

[0019] 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.

[0020] 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.

[0021] 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).

[0022] 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).

[0023] 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.

[0024] 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.

[0025] 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.

[0026] 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.

[0027] 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.

[0028] 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.

[0029] 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.

[0030] 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.

[0031] Referring back to FIG. 1, in some embodiments, the blockchain system 100 may be implemented to support various types of participating parties, or users, including, e.g., banks, financial institutions, credit unions, and the like. The various parties may utilize the blockchain system 100 to retrieve or record information, including, e.g., payment transfer information. Typically, records such as payment transfers and the like recorded on the blockchain system 100 are encrypted by relevant parties.

[0032] In some embodiments, the blockchain system 100 may be implemented to support real-time gross settlement (RTGS). For example, the blockchain system 100 may enforce a policy so that all transfers, once processed, are final and irrevocable. The blockchain system 100 may also enforce a policy so that none of the payment transactions submitted to the blockchain system 100 is subject to a waiting period. The blockchain system 100 may further enforce a policy so that payment transactions are settled on one-to-one basis without bundling or netting with any other transactions.

[0033] FIG. 3 illustrates a flow chart of a privacy-preserving deadlock detection method 300 for detecting deadlocks in an RTGS system implemented on a blockchain

system, e.g., the blockchain system 100 (FIG. 1), according to an embodiment. The privacy-preserving deadlock detection method 300 may detect deadlocks without requiring any participating parties to disclose their liquidity and transaction information.

[0034] At step 302, each party participating in the RTGS system, e.g., Parties A through N, may be instructed to independently calculate its liquidity information. In some embodiments, each party may independently calculate its liquidity information using its local computing resources. In some embodiments, each party $i \in \{A, B, \dots, N\}$ may calculate its current liquidity C_i, an amount T_i to be transferred to (receivable by) the party, an amount F_i to be transferred from (payable by) the party, and a post transfer liquidity P_i, where $P_i = C_i + T_i - F_i$.

[0035] At step 304, Parties A through N may be instructed to jointly carry out a multiparty computation to determine a minimum liquidity amount $P_{min} = \min_{i \in \{A,B,\dots,N\}}\{P_i\}$. In some embodiments, Parties A through N may jointly carry out the multiparty computation without disclosing their liquidity information, including their post transfer liquidity P_i. In this manner, Parties A through N can jointly determine P_{min} and preserve their privacy.

[0036] Multiparty computation is a subfield of cryptography with the goal of creating methods for parties to jointly compute a function over their inputs while keeping those inputs private. In a multiparty computation, a given number of parties, e.g., Parties A through N, may each have private data, e.g., the post transfer liquidity amount P_i, $i \in \{A, B, \dots, N\}$, that they want to keep as secret. The parties may also want to compute a value of a function on the private data, e.g., $f(P_A, P_B, \dots, P_N)$, while keeping their private data secret. For example, at step 304, Parties A through N may be

instructed to compute the value of $f(P_A, P_B, \ldots, P_N) = \min\ (P_A, P_B, \ldots, P_N)$. The goal of multiparty computation is to provide a protocol where Parties A through N can compute the $\min\ (P_A, P_B, \ldots, P_N)$ without revealing $P_A, P_B, \ldots, P_N$.

[0037] In some embodiments, Parties A through N may jointly carry out a multiparty computation protocol similar to that disclosed in Gagan Aggarwal et al., "Secure Computation of the Median (and Other Elements of Specified Ranks)," Journal of Cryptology (2010) 23: 373–401, which is herein incorporated by reference in its entirety. For instance, Parties A through N may jointly carry out a multiparty computation protocol defined as follows:

1: Provide each party with initial values of α and β set by the RTGS system, the initial value of α being sufficiently small to represent a possible minimum post transfer liquidity and the initial value of β being sufficiently large to represent a possible maximum post transfer liquidity.

2: *Repeat* until "done":

3: Set $m = \lceil(\alpha + \beta)/2\rceil$.

4 Instruct each party $i \in \{A, B, \ldots, N\}$ to compare P_i to m; if P_i is strictly less than m, set $l_i = 1$ and $g_i = 0$; otherwise, if P_i is strictly greater than m, set $l_i = 0$ and $g_i = 1$.

5: Calculate $\sum_{i\in\{A,B,\ldots,N\}} l_i$ and $\sum_{i\in\{A,B,\ldots,N\}} g_i$.

6: If $\sum_{i\in\{A,B,\ldots,N\}} l_i \leq 0$ and $\sum_{i\in\{A,B,\ldots,N\}} g_i \leq N - 1$, then report m as $\min\ (P_A, P_B, \ldots, P_N)$ and proceed to "done."

7: If $\sum_{i\in\{A,B,\ldots,N\}} l_i \geq 1$, then set $\beta = m - 1$ (meaning that $\min\ (P_A, P_B, \ldots, P_N)$ is smaller than m, so m should be decreased).

8: If $\sum_{i\in\{A,B,\ldots,N\}} g_i \geq N$, then set $\alpha = m + 1$ (meaning that $\min\ (P_A, P_B, \ldots, P_N)$ is greater than m, so m should be increased).

9: Done.

[0038] It is noted that the multiparty computation protocol described above may allow Parties A through N to compute the minimum liquidity amount, P_{min}, without revealing $P_A, P_B, \ldots, P_N$. It is to be understood, however, that the multiparty computation protocol described above is provided merely as an example and is not meant to be limiting. It is contemplated that Parties A through N may implement other types of multiparty computation techniques to carry out step 304.

[0039] It is also contemplated that in some embodiments, Parties A through N may utilize a multiparty summation process to calculate $\sum_{i\in\{A,B,\ldots,N\}} l_i$ and $\sum_{i\in\{A,B,\ldots,N\}} g_i$. In this manner, Parties A through N may calculate $\sum_{i\in\{A,B,\ldots,N\}} l_i$ and $\sum_{i\in\{A,B,\ldots,N\}} g_i$ without requiring the parties to disclose their l and g values to each other, further preserving the privacy of Parties A through N. For instance, in some embodiments, Parties A through N may jointly carry out the multiparty summation process disclosed in Yiping Shen et al., "On the Confidential Auditing of Distributed Computing Systems," Proceedings of the 24th International Conference on Distributed Computing Systems (2004) 600-607, or that disclosed in L. Jean Camp et al., "Providing Auditing While Protecting Privacy," The Information Society, 10(1), pp. 59-72, 1998, both of which are herein incorporated by reference in their entirety.

[0040] In some embodiments, Parties A through N may carry out the multiparty summation process by masking their l and g values. For illustrative purposes, suppose that there are three parties, Parties A, B, and C, and their l and g values are l_A, l_B, and

l_C, and g_A, g, and g_C, respectively. Each party i may be instructed to select two random integers, a and b, to be used in a second-degree polynomial $ax^2 + bx + c$, where the constant in the polynomial, c, is set to represent the party's l value or the party's g value, depending on whether the parties are instructed to calculate $\sum_{i\in\{A,B,\ldots,N\}} l_i$ or $\sum_{i\in\{A,B,\ldots,N\}} g_i$. Suppose that the parties are instructed to calculate $\sum_{i\in\{A,B,\ldots,N\}} l_i$, Party A may select, for example, $a = 25$ and $b = 83$, which means Party A will have its polynomial defined as $25x^2 + 83x + l_A$. Party A may mask its l value by calculating masked values using the polynomial at $x = 1$, $x = 2$, and $x = 3$. Parties B and C may carry out similar operations with their own selection of random integers a and b to mask their own l values.

[0041] Each party i may be further instructed to keep one of its masked values and share the rest of its masked values with other parties to facilitate the calculation of $\sum_{i\in\{A,B,\ldots,N\}} l_i$. Continuing with the three-party example above, Party A may keep a masked value calculated using the polynomial $25x^2 + 83x + l_A$ at $x = 1$, share a masked value calculated using the polynomial $25x^2 + 83x + l_A$ at $x = 2$ with Party B, and share another masked value calculated using the polynomial $25x^2 + 83x + l_A$ at $x = 3$ with Party C. Parties B and C may carry out similar operations with respect to their masked values.

[0042] Each party i may then proceed to calculate its share of $\sum_{i\in\{A,B,\ldots,N\}} l_i$ based on the masked value it kept to itself and the masked values it received from other parties. Continuing with the three-party example above, Party A may calculate a sum based on the masked value it kept (e.g., calculated by Party A using Party A's own polynomial at $x = 1$), the masked value it received from Party B (e.g., calculated by

Party B using Party B's polynomial at $x = 1$), and the masked value it received from Party C (e.g., calculated by Party C using Party C's polynomial at $x = 1$). Party B may calculate a sum based on the masked value it kept (e.g., calculated by Party B using Party B's own polynomial at $x = 2$), the masked value it received from Party A (e.g., calculated by Party A using Party A's polynomial at $x = 2$), and the masked value it received from Party C (e.g., calculated by Party C using Party C's polynomial at $x = 2$). Party C may calculate a sum based on the masked value it kept (e.g., calculated by Party C using Party C's own polynomial at $x = 3$), the masked value it received from Party A (e.g., calculated by Party A using Party A's polynomial at $x = 3$), and the masked value it received from Party B (e.g., calculated by Party B using Party B's polynomial at $x = 3$).

[0043] Parties A through N may then consolidate the results to calculate $\sum_{i\in\{A,B,\ldots,N\}} l_i$ and $\sum_{i\in\{A,B,\ldots,N\}} g_i$ If the parties used the constants in their polynomials to represent their l values, the sum representing the combined value of all constants used in the polynomials will equal to $\sum_{i\in\{A,B,\ldots,N\}} l_i$. Likewise, if the parties used the constants in their polynomials to represent their g values, the sum representing the combined value of all constants used in the polynomials will equal to $\sum_{i\in\{A,B,\ldots,N\}} g_i$. In this manner, Parties A through N can calculate $\sum_{i\in\{A,B,\ldots,N\}} l_i$ and $\sum_{i\in\{A,B,\ldots,N\}} g_i$ without requiring the parties to disclose their l and g values to each other, further preserving the privacy of Parties A through N.

[0044] It is to be understood that while the examples described above discussed using second-degree polynomials to facilitate masking of values provided by three participating parties, such examples are depicted merely for illustrative purposes and

are not meant to be limiting. It is to be understood that additional parties may participate in the multiparty summation process discussed above. It is also to be understood that the parties may implement other types of multiparty computation processes to facilitate the calculation of $\sum_{i\in\{A,B,\ldots,N\}} l_i$ and $\sum_{i\in\{A,B,\ldots,N\}} g_i$. Alternatively, in some embodiments, Parties A through N may calculate $\sum_{i\in\{A,B,\ldots,N\}} l_i$ and $\sum_{i\in\{A,B,\ldots,N\}} g_i$ without using multiparty computation. It is further to be understood that regardless of how Parties A through N choose to calculate $\sum_{i\in\{A,B,\ldots,N\}} l_i$ and $\sum_{i\in\{A,B,\ldots,N\}} g_i$, the calculation is merely one sub-step of step 304. The ultimate goal of step 304 is to facilitate the calculation of the minimum liquidity amount, P_{min}, without requiring the parties to reveal their post transfer liquidity information, $P_A, P_B, \ldots, P_N$, as described above.

[0045] At step 306, Parties A through N may be instructed to submit the minimum liquidity amount, P_{min}, to the blockchain system 100 for recordation. In some embodiments, the blockchain system 100 may utilize a consensus protocol to validate the value of P_{min}. The blockchain system 100 may reject the value of P_{min} if no consensus is reached. In some embodiments, the blockchain system 100 may utilize other verification techniques to verify the validity of the value submitted. In some embodiments, the blockchain system 100 may accept and incorporate the submitted value of P_{min} into the blockchain system 100 without additional verification.

[0046] At step 308, the blockchain system 100 may determine whether a deadlock situation exists based on the value of P_{min}. In some embodiments, the blockchain system 100 may determine that a deadlock situation exists if $P_{min} < 0$. The blockchain system 100 may notify one or more Parties A through N that additional

supply of liquidity is needed to resolve the deadlock situation. On the other hand, the blockchain system 100 may determine that the RTGS system is not deadlocked if $P_{min} \geq 0$, and report that no additional liquidity is needed.

[0047] In some embodiments, the blockchain system 100 may implement one or more smart contracts and utilize the smart contracts to determine whether a deadlock situation exists. Smart contracts are computer protocols implemented in the form of computer code that are incorporated into the blockchain 120 (FIG. 1), to facilitate, verify, or enforce the negotiation or performance of contracts. For example, a user of the blockchain system 100 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 by the blockchain system 100, e.g., to perform a task. 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. In some embodiments, the blockchain system may implement a smart contract that is programmed to determine whether or not a deadlock situation exists based on whether $P_{min} < 0$ or $P_{min} \geq 0$.

[0048] In some embodiments, the blockchain system 100 may keep track of a total number of transactions pending in the RTGS system and invoke the privacy-preserving deadlock detection method 300 when the total number of pending transactions reaches a predetermined threshold. FIG. 4 illustrates a flow chart of a method 400 for invoking the privacy-preserving deadlock detection method 300, according to an embodiment. The method 400 may be performed by one or more

nodes in a blockchain system that maintain the blockchain, e.g., the nodes 102-110 in the blockchain system 100 (FIG. 1).

[0049] At step 402, a node, e.g., the node 102, may determine the number of transactions pending execution on the blockchain system 100. For example, a number of transactions may be included in a new block that is to be accepted by the blockchain system 100. These transactions are still pending execution on the blockchain system 100. The node 102 may determine the number of such transactions in step 402. At step 404, the node 102 may determine whether the number of transactions pending execution on the blockchain system 100 has exceeded a predetermined threshold. At step 406, in response to the determination that the number of transactions pending execution on the blockchain system 100 has exceeded the predetermined threshold, the node 102 may invoke the privacy-preserving deadlock detection method 300 (Fig. 3).

[0050] At step 408, the node 102 may instruct participating parties, e.g., Parties A through N, to independently calculate their post transfer liquidity as described in step 302 above. At step 410, the node 102 may instruct the parties to carry out a multiparty computation to determine a minimum post transfer liquidity amount $P_{min} = \min_{i \in \{A,B,\ldots,N\}}\{P_i\}$ as described in step 304 above. At step 412, the node 102 may instruct the parties to submit the minimum post transfer liquidity amount, P_{min}, to the blockchain system 100 for recordation. In some embodiments, the node 102 may send instructions to the parties via wired or wireless communication channels. The parties may submit the minimum post transfer liquidity amount to the blockchain system 100 via wired or wireless communication channels. Upon receiving the minimum post transfer liquidity amount, the node 102 may, at step 414, determine whether or not a deadlock

situation exists based on whether $P_{min} < 0$ or $P_{min} \geq 0$.

[0051] In some embodiments, each party, e.g., Parties A through N, may keep track of the number of transactions pending actions by that particular party. A particular party may indicate its intention to invoke the privacy-preserving deadlock detection method 300 when the number of transactions pending actions by that particular party reaches a threshold established by that party. In some embodiments, the privacy-preserving deadlock detection method 300 may be invoked if Parties A through N reach a consensus that the privacy-preserving deadlock detection method 300 should be invoked. Once a consensus is reached, a node, e.g., the node 102, may carry out steps 408-414 as described above.

[0052] In some embodiments, the node 102 may carry out additional verification steps to verify the computation correctness of the privacy-preserving deadlock detection method 300. For example, in some embodiments, the node 102 may instruct participating parties, e.g., Parties A through N, to utilize a commitment scheme to calculate commitment values of their independently calculated liquidity information and submit the commitment values to the blockchain system 100 for recordation. Such a commitment scheme may include a cryptographic primitive that allows the parties to commit to certain values while keeping the values hidden from others. In this manner, the parties can submit only the commitment values (as opposed to their calculated liquidity information) to the blockchain system 100, which can utilize these commitment values to confirm correctness of the liquidity information during the execution of the privacy-preserving deadlock detection method 300. If someone has changed their liquidity information during the execution of the privacy-preserving deadlock detection

method 300, a node in the blockchain system 100, e.g., the node 102, can detect the change and invalidate the computation result of the method 300. In such a case, the node 102 may invoke the method 300 again, or report an error.

[0053] The node 102 may also instruct participating parties, e.g., Parties A through N, to implement additional verification techniques. For example, in some embodiments, the node 102 may instruct participating parties to implement a liquidity-based verification technique. Such a liquidity-based verification technique may instruct the parties to calculate a total liquidity and compare the calculated total liquidity to a total liquidity known to exist in the RTGS system. If the calculated total liquidity does not equal to the total liquidity known to exist in the RTGS system, a node in the blockchain system 100, e.g., the node 102, may determine that an error has occurred and invalidate the computation result of the method 300. Additionally, or alternatively, in some embodiments, the node 102 may instruct participating parties to implement a transaction-based verification technique. Such a transaction-based verification technique may instruct the parties to calculate an aggregated total amount receivable and an aggregated total amount payable in the RTGS system. If the aggregated total amount receivable does not equal to the aggregated total amount payable, a node in the blockchain system 100, e.g., the node 102, may determine that an error has occurred and invalidate the computation result of the method 300.

[0054] Referring back to FIG. 3. In some embodiments, the privacy-preserving deadlock detection method 300 may be utilized in decentralized computing platforms that do not form blockchains. For example, the steps depicted in the privacy-preserving deadlock detection method 300 may be carried out by the nodes participating in a P2P

network without forming a blockchain. In such embodiments, instead of submitting the determined minimum post transfer liquidity amount, P_{min}, to a blockchain for recordation, the nodes participating in the P2P network may record the value of P_{min} on one or more peer nodes in step 306. These peer nodes may also be utilized to carry out step 308 to determine whether or not a deadlock situation exists based on whether $P_{min} < 0$ or $P_{min} \geq 0$. It is contemplated that the privacy-preserving deadlock detection method 300 may be implemented in other types of decentralized computing platforms as well.

[0055] In some embodiments, the privacy-preserving deadlock detection method 300 may be utilized in a RTGS system operated by a central bank. For example, the central bank may instruct other banks participating in the RTGS system to carry out steps 302 and 304 as described above. The banks may then be instructed to submit the determined minimum post transfer liquidity amount P_{min} to the central bank in step 306. The central bank may carry out step 308 to determine whether or not a deadlock situation exists based on whether $P_{min} < 0$ or $P_{min} \geq 0$. In this manner, the central bank can distribute some of the computation steps to other banks participating in the RTGS system while still preserving privacy of the participating banks.

[0056] FIG. 5 is a block diagram of a deadlock detection 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 determination module 502, an instruction module 504, a receiving module 506, and a detection module 508.

[0057] The determination module 502 may determine whether to invoke privacy-

preserving deadlock detection in an RTGS system. In some embodiments, the determination module 502 may make the determination based on the number of transactions pending execution in the RTGS system. Alternatively, or additionally, parties participating in the RTGS system may keep track of the number of transactions pending locally and determine whether to invoke privacy-preserving deadlock detection in the RTGS system based on whether the parties reach a consensus to invoke privacy-preserving deadlock detection.

[0058] The instruction module 504 may provide instructions to the parties participating in the RTGS system. For example, the instruction module 504 may instruct the participating parties to independently calculate their liquidity information. In some embodiments, each party $i \in \{A, B, \ldots, N\}$ may calculate its current liquidity C_i, an amount T_i to be transferred to (receivable by) the party, an amount F_i to be transferred from (payable by) the party, and a post transfer liquidity P_i, where $P_i = C_i + T_i - F_i$. The instruction module 504 may also instruct the participating parties to carry out a multiparty computation to determine a minimum post transfer liquidity amount $P_{min} = \min_{i \in \{A,B,\ldots,N\}}\{P_i\}$. The instruction module 504 may further instruct the participating parties to submit the minimum post transfer liquidity amount, P_{min}, to the receiving module 506.

[0059] The receiving module 506 may receive information provided by the participating parties and provide the information received to the detection module 508. The detection module 508 may determine whether a deadlock situation exists based on the value of P_{min}. In some embodiments, the detection module 508 may communicate with a blockchain system that utilizes one or more smart contracts to determine whether

a deadlock situation exists. In some embodiments, the detection module 508 may include other types of computing modules capable of determining whether a deadlock situation exists based on the value of P_{min}. In some embodiments, the detection module 508 may determine whether or not a deadlock situation exists based on whether $P_{min} < 0$ or $P_{min} \geq 0$.

[0060] 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.

[0061] 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.

[0062] 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.

[0063] 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.

[0064] 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).

[0065] 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.

[0066] 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.

[0067] 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.

[0068] 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 detecting deadlocks in a real-time gross settlement system, the method comprising:

instructing a plurality of users of the real-time gross settlement system to jointly determine a minimum post transfer liquidity, wherein the minimum post transfer liquidity is jointly determined based on a post transfer liquidity of each of the plurality of users without disclosing by each of the plurality of users of its post transfer liquidity to another user;

receiving the jointly determined minimum post transfer liquidity; and

determining whether a deadlock exists based on the minimum post transfer liquidity.

2. The method of claim 1, further comprising:

determining a number of transactions pending execution in the real-time gross settlement system;

determining whether the number of transactions pending execution has exceeded a predetermined threshold; and

in response to a determination that the number of transactions pending execution has exceeded the predetermined threshold, instructing each of the plurality of users to independently calculate the post transfer liquidity of that user for jointly determining the minimum post transfer liquidity.

3. The method of any preceding claim, wherein the real-time gross settlement system is implemented on a peer-to-peer network.

4. The method of any preceding claim, wherein the real-time gross settlement system is implemented with a blockchain system.

5. The method of claim 4, further comprising:

incorporating the minimum post transfer liquidity into a blockchain of the blockchain system.

6. The method of claim 5, further comprising:

executing a smart contract recorded on the blockchain to determine whether the deadlock exists.

7. The method of any preceding claim, wherein the deadlock is determined to exist when the minimum post transfer liquidity is less than zero.

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

notifying at least one of the plurality of users of the deadlock when the minimum post transfer liquidity is less than zero.

9. The method of any preceding claim, wherein each user i of the plurality of users $A, B, \ldots, N$, where $i \in \{A, B, \ldots, N\}$, is instructed to independently calculate a current

liquidity C_i, an amount T_i to be transferred to user i, an amount F_i to be transferred from user i, and the post transfer liquidity P_i, where $P_i = C_i + T_i - F_i$.

10. The method of claim 9, wherein the plurality of users $A, B, \ldots, N$ jointly implement a multiparty computation protocol to compute the minimum post transfer liquidity $\min_{i \in \{A,B,\ldots,N\}}\{P_i\}$ without disclosing by each user i of its post transfer liquidity P_i to another user, where $i \in \{A, B, \ldots, N\}$.

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

verifying a computation correctness of the jointly determined minimum post transfer liquidity.

12 A device for detecting deadlocks in a real-time gross settlement system, 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 11.

13. An apparatus for detecting deadlocks in a real-time gross settlement system, the apparatus comprising a plurality of modules for performing the method of any of claims 1 to 11.

14. 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 11.