

METHODS AND DEVICES FOR EXECUTING CROSS-CHAIN ANONYMOUS MULTI-SWAP CONTRACTS

TECHNICAL FIELD

[0001] The specification relates generally to computer technologies, and more particularly, to methods and devices for executing cross-chain anonymous multi-swap contracts.

BACKGROUND

[0002] 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.

[0003] 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.

[0004] Users of a blockchain system may utilize the blockchain system to carry out various types of transactions. For example, two parties, Party A and Party B, may both own assets recorded on a blockchain maintained in the blockchain system. Parties A and B may enter into an agreement to exchange, or swap, their assets on the blockchain. Such agreements are typically carried out using smart contracts, which 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.

[0005] While using smart contracts may be sufficient to facilitate swap of assets recorded on the same blockchain, situations may arise where Parties A and B may want to swap assets recorded on different blockchains. For example, Party A may want to transfer a first asset, e.g., bitcoins, recorded on a first blockchain to Party B in exchange for Party B transferring a second asset, e.g., ether, recorded on a second blockchain to Party A. Such operations may be referred to as cross-chain swaps.

[0006] Because cross-chain swaps require multiple transfers to be performed on multiple blockchains, it may be desirable to implement an atomic swap protocol to prevent the parties from unilaterally withholding, or withdrawing from, the execution of the transfer operations. For example, if Party A has already transferred its asset recorded on the first blockchain to Party B, implementing an atomic swap protocol may help prevent Party B from withholding the transfer of its asset recorded on the second blockchain to Party A.

[0007] A hashed time lock contract, or HTLC, is a type of smart contract that uses a hash lock and a time lock to require that the receiving party of a transferred

asset, e.g., a payment, to either acknowledge the receipt of the transfer prior to a deadline by generating cryptographic proof of the transfer, or forfeit the ability to claim the asset transferred, thereby returning the asset to the transferring party. HTLCs may be utilized to implement atomic swap protocols. However, current implementations of HTLC-based atomic swap protocols require substantial amount of setup time.

SUMMARY

[0008] In one aspect, a computer-implemented method for executing an anonymous multi-swap (AMS) contract between a first user and a second user includes: detecting a recordation of a first plurality of parameters for the AMS contract on a first blockchain, the first plurality of parameters including: an initial value of a first function lock on the first blockchain, at least one action to be performed on the first blockchain when the AMS contract is executed, and a maximum number of times the AMS contract can be executed, the maximum number of times the AMS contract can be executed being greater than one; recording a second plurality of parameters for the AMS contract on a second blockchain, the second plurality of parameters including: an initial value of a second function lock on the second blockchain, at least one action to be performed on the second blockchain when the AMS contract is executed, and the maximum number of times the AMS contract can be executed, the initial value of the second function lock being different from the initial value of the first function lock; detecting an invocation of a request to execute the AMS contract on the second blockchain; retrieving, from the request to execute the AMS contract on the second blockchain, a retrieved key used to unlock the second function lock on the second blockchain; calculating, based on the retrieved key, a calculated key for unlocking the first function lock on the first

blockchain; and invoking a request to execute the AMS contract on the first blockchain with the calculated key.

[0009] In another aspect, a device for executing an AMS contract between a first user and a second user 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: detect a recordation of a first plurality of parameters for the AMS contract on a first blockchain, the first plurality of parameters including: an initial value of a first function lock on the first blockchain, at least one action to be performed on the first blockchain when the AMS contract is executed, and a maximum number of times the AMS contract can be executed; record a second plurality of parameters for the AMS contract on a second blockchain, the second plurality of parameters including: an initial value of a second function lock on the second blockchain, at least one action to be performed on the second blockchain when the AMS contract is executed, and the maximum number of times the AMS contract can be executed, the initial value of the second function lock being different from the initial value of the first function lock; detect an invocation of a request to execute the AMS contract on the second blockchain; retrieve, from the request to execute the AMS contract on the second blockchain, a retrieved key used to unlock the second function lock on the second blockchain; calculate, based on the retrieved key, a calculated key for unlocking the first function lock on the first blockchain; and invoke a request to execute the AMS contract on the first blockchain with the calculated key.

[0010] In still another aspect, a non-transitory computer-readable medium has stored therein instructions that, when executed by a processor of a device, cause the device to perform a method for executing an AMS contract between a first user and

a second user. The method includes: detecting a recordation of a first plurality of parameters for the AMS contract on a first blockchain, the first plurality of parameters including: an initial value of a first function lock on the first blockchain, at least one action to be performed on the first blockchain when the AMS contract is executed, and a maximum number of times the AMS contract can be executed; recording a second plurality of parameters for the AMS contract on a second blockchain, the second plurality of parameters including: an initial value of a second function lock on the second blockchain, at least one action to be performed on the second blockchain when the AMS contract is executed, and the maximum number of times the AMS contract can be executed, the initial value of the second function lock being different from the initial value of the first function lock; detecting an invocation of a request to execute the AMS contract on the second blockchain; retrieving, from the request to execute the AMS contract on the second blockchain, a retrieved key used to unlock the second function lock on the second blockchain; calculating, based on the retrieved key, a calculated key for unlocking the first function lock on the first blockchain; and invoking a request to execute the AMS contract on the first blockchain with the calculated key.

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 method for setting an anonymous multi-swap contract, according an embodiment.

[0015] FIG. 4 is a flow chart of a method for executing an anonymous multi-swap contract, according an embodiment.

[0016] FIG. 5 is a flow chart of a method for executing an anonymous multi-swap contract, according an embodiment.

[0017] FIG. 6 is a block diagram of an apparatus for executing an anonymous multi-swap contract, according to an embodiment.

[0018] FIG. 7 is a flow chart of a method for executing an anonymous multi-swap contract, according an embodiment.

[0019] FIG. 8 is a block diagram of an apparatus for executing an anonymous multi-swap contract, according to an embodiment.

DETAILED DESCRIPTION

[0020] Embodiments of the specification provide methods and devices for setting and executing cross-chain anonymous multi-swap (AMS) contracts. The methods and devices support a one-time off-chain negotiation process where two parties, or users, can negotiate terms of an AMS contract to facilitate cross-chain swaps up to N times. In some embodiments, the methods and devices also support a one-time on-chain setup process to record the negotiated terms as parameters of the AMS contract. In some embodiments, the on-chain setup process can record the parameters on blockchains involved in the cross-chain swaps. In this manner, the blockchains can rely on the AMS contract to ensure that swaps can take place if all

users conform to the AMS contract and that all swaps are atomic and fair. If one user deviates from the AMS contract, that user will automatically forfeit its ability to claim the asset transferred, thereby returning the asset to the transferring user. Furthermore, in some embodiments, the methods and devices may implement a protocol that conceals the relationship between the parameters recorded on the blockchains to prevent a third-party from linking the parameters recorded on the blockchains together. In this manner, the privacy of the parties to the AMS contract can be preserved.

[0021] Embodiments disclosed in the specification have one or more technical effects. In some embodiments, the methods and devices support a one-time off-chain negotiation process for setting up an AMS contract so that it can be executed up to N times. This allows the users to eliminate the need to re-negotiate the contract up to N-1 times. In some embodiments, the methods and devices also support a one-time on-chain setup process to record the negotiated terms as parameters on blockchains. This allows the blockchains to record the parameters concisely and efficiently, and eliminates the need to repeat the on-chain setup process up to N-1 times. In some embodiments, the methods and devices utilize one-way function chains (e.g., hash chains) to implement function locks for the AMS contract. This allows each user to maintain only one key, which can work in conjunction with a one-way function chain to unlock the AMS contract up to N times. In some embodiments, the methods and devices implement a protocol to conceal the relationship between the function locks used on different blockchains. This allows the blockchains to use function locks for AMS contracts without revealing the relationships between the users who set up the AMS contracts, thereby preserving the privacy of the users. Furthermore, in some embodiments, the methods and

devices can automatically invalidate the AMS contract after N number of executions, or after the AMS contract expires. This allows the blockchains to prevent the users from executing unauthorized swaps. In some embodiments, the methods and devices further support ordering of up to N number of swaps in the AMS contract. This allows the users to specify not only the number of swaps they agreed to perform, but also the ordering of such swaps.

[0022] 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.

[0023] 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.

[0024] 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).

[0025] 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).

[0026] 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.

[0027] 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.

[0028] 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.

[0029] 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.

[0030] 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.

[0031] 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.

[0032] 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.

[0033] 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.

[0034] FIG. 3 illustrates a flow chart of a method 300 for setting a cross-chain atomic multi-swap (AMS) contract according to an embodiment. In the embodiment, the AMS contract set by the method 300 can be executed up to N times.

[0035] Referring to FIG. 3, first and second users, User 1 and User 2, may have accounts on first and second separate blockchains, Blockchain A and Blockchain B, respectively. Blockchains A and B may be implemented on one or more blockchain systems, e.g., the blockchain system 100 (FIG. 1) and the like, as described above. Blockchains A and B 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.

[0036] For illustrative purposes, suppose that User 1 wants to swap Asset X owned by User 1 and recorded on Blockchain A with Asset Y owned by User 2 and recorded on Blockchain B. Asset X may include one or more units of a certain type of asset, e.g., bitcoins, or a bundle of various types of assets, recorded on Blockchain A. Asset Y may include one or more units of a certain type of asset, e.g., ether, or a bundle of various types of assets, recorded on Blockchain B. Suppose further that User 1 is willing to swap with User 2 under the same terms for a number of times, such as up to N times, in the future. User 1 may reach out to User 2 using a communication channel and start a negotiation process with User 2. Users 1 and 2 may negotiate specific terms of the swap, including, e.g., an exchange rate between Asset X and Asset Y, a maximum number of swaps (or times the AMS contract can be executed), N , User 1 being allowed to invoke a swap operation, an expiration time T_1 when the agreement expires on Blockchain A, and an expiration time T_2 when the agreement expires on Blockchain B. An AMS contract may be set up after Users 1 and 2 reach an agreement.

[0037] At step 302, User 1 may generate one or more parameters for the AMS contract based on the agreement reached by Users 1 and 2. In some embodiments, the parameters may include the values of the maximum number N , the expiration time T_1 when the agreement expires on Blockchain A, and the expiration time T_2 when the agreement expires on Blockchain B.

[0038] In some embodiments, the parameters may also include a first one-way function value H_N and a second one-way function value Δ_N calculated by User

1. H_N may be used to serve as an initial function lock of the AMS contract on Blockchain A and $H_N + \Delta_N$ may be used to serve as an initial function lock of the AMS contract on Blockchain B. Alternatively, H_N may be used to serve as an initial function lock of the AMS contract on Blockchain B and $H_N + \Delta_N$ may be used to serve as an initial function lock of the AMS contract on Blockchain A.

[0039] A function lock may refer to a lock, L , that can be unlock only when a correct key, Key , is provided to a function $g()$, which can produce a value $g(Key)$ that matches the value of L . If the function $g()$ is a hash function, then the function lock may be referred to as a hash lock. One of the reasons for using different function locks on Blockchains A and B is to prevent a third-party from linking the two function locks together. If the third-party can link the two function locks together, e.g., by identifying two identical function locks, the third-party may be able to identify the transfers carried out on both ends of the AMS contract, which may reveal information that is private to Users 1 and 2. For example, suppose that Blockchain A records real estate ownership and Blockchain B records payment information. If Users 1 and 2 carry out a cross-chain swap that transfers a real property recorded on Blockchain A from User 1 to User 2 and transfers a payment recorded on Block B from User 2 to User 1, by linking the two transfers together, the third-party may be able to determine the price User 2 paid for the real property. In another example, if Blockchain A uses public identity for its users while Blockchain B enforces anonymity, linking two transfers together may allow the third-party to determine identities of users operating on Blockchain B, thereby compromising the anonymity enforced by Blockchain B. Accordingly, by requiring Blockchains A and B to use different function locks, e.g., H_N on Blockchain A and $H_N + \Delta_N$ on Blockchain B, the relationship between the transfers carried out on both ends of the AMS contract can

be concealed, thereby preserving the privacy of Users 1 and 2.

[0040] In some embodiments, both H_N and Δ_N may be calculated using a one-way function $g()$. 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 independent variable of a one-way function from a value of a dependent variable of the one-way function, making the function practically infeasible to invert and, thus, the function is called “one-way.” Cryptographic hash functions, such as SHA 256 and the like, are examples of one-way functions.

[0041] In some embodiments, the one-way function $g()$ may also be homomorphic, which means that if $\mathcal{D}$ and $\mathcal{R}$ define two abelian groups, then for each pair $(a, b) \in \mathcal{D}^2$ it holds that $g(a \oplus b) = g(a) \oplus g(b)$, where $\oplus$ denotes the group operation, including, e.g., addition or multiplication. If $g()$ is additively homomorphic, for example, the characteristics of $g()$ will provide that $g(a + b) = g(a) + g(b)$. For simplicity of illustration, and without losing generality, the following descriptions and the accompanied drawings may use “+” to denote “ $\oplus$.”

[0042] In some embodiments, H_N may be calculated based on a one-way function chain, which is a successive application of $g()$ to an initial value H_0 . In some embodiments, the value H_0 may be randomly selected. H_1 may then be calculated as $H_1 = g(H_0)$, H_2 may be calculated as $H_2 = g(H_1)$, and so on. The successive application of $g()$ may continue until H_N is calculated as $H_N = g(H_{N-1})$. Δ_N may be calculated in the same or similar manner. That is, the value Δ_0 may be randomly selected. Δ_1 may then be calculated as $\Delta_1 = g(\Delta_0)$, Δ_2 may be calculated as $\Delta_2 = g(\Delta_1)$, and so on. The successive application of $g()$ may continue until Δ_N is calculated as $\Delta_N = g(\Delta_{N-1})$.

[0043] At step 304, User 1 may send the parameters, including, e.g., H_N , $H_N + \Delta_N$, Δ_0 , the maximum number N , the expiration time T_1 when the agreement expires on Blockchain A, and the expiration time T_2 when the agreement expires on Blockchain B, to User 2. However, User 1 may not share the value of H_0 with User 2. In some embodiments, User 2 may have the option to verify whether the values of N , T_1 , and T_2 conform to the terms of the agreement reached between Users 1 and 2. If the values of N , T_1 , and T_2 do not conform to the agreed terms, User 2 may send an error message to User 1 indicating the mismatch. Otherwise, if the values of N , T_1 , and T_2 conform to the agreed terms, User 2 may send a confirmation message to User 1.

[0044] In some embodiments, User 2 may also carry out step 306 to verify the validity of the values of H_N , $H_N + \Delta_N$, and Δ_0 . For example, User 2 may independently calculate $\Delta_1 = g(\Delta_0)$, $\Delta_2 = g(\Delta_1)$, ..., $\Delta_N = g(\Delta_{N-1})$. User 2 may then calculate the value of $(H_N) + (\Delta_N)$, where (H_N) represents the value of H_N as received from User 1 and (Δ_N) represents the value of Δ_N independently calculated by User 2. User 2 may compare the value of $(H_N) + (\Delta_N)$ to the value of $H_N + \Delta_N$ received from User 1. If $H_N + \Delta_N \neq (H_N) + (\Delta_N)$, User 2 may send an error message to User 1 indicating the mismatch. Otherwise, if $H_N + \Delta_N = (H_N) + (\Delta_N)$, User 2 may send a confirmation message to User 1. In some embodiments, User 2 may skip step 306 if User 1 is trusted by User 2. In some embodiments, User 2 may always carry out step 306 to verify the validity of H_N , $H_N + \Delta_N$, and Δ_0 .

[0045] In some embodiments, steps 302 through 306 may be performed off-chain. For example, Users 1 and 2 do not need to utilize Blockchains A and B to facilitate their negotiations regarding the terms of the AMS contract. In this manner, the terms of the AMS contract may be recorded on Blockchains A and B, after Users

1 and 2 reach an agreement. In some embodiments, steps 302 through 306 may only need to be performed once for the AMS contract.

[0046] After Users 1 and 2 have reached an agreement and User 1 has generated and sent the parameters to User 2, the AMS contract may be set up. At step 308, User 1 may setup the AMS contract by requesting recordation on Blockchain A of parameters including, e.g., H_N , which serves as the initial function lock of the AMS contract on Blockchain A, the maximum number N , the expiration time T_1 when the agreement expires on Blockchain A, and the action User 1 agrees to perform, i.e., transfer of Asset X to User 2. In some embodiments, User 1 may request the recordation of the parameters on Blockchain A by invoking a function, e.g., `setupAMS(Contract_ID_A, Object)`, made available on Blockchain A through a smart contract interface. At step 310, Blockchain A may carry out the instructions specified in `setupAMS(Contract_ID_A, Object)` and record the parameters specified by User 1 on Blockchain A. One of ordinary skill in the art will understand an object can be a variable, a data structure, a function, or a method in the field of computer technologies.

[0047] In some embodiments, the value of `Contract_ID_A` specified in `setupAMS(Contract_ID_A, Object)` may represent a unique identifier, e.g., a universally unique identifier, assigned or generated to identify the AMS contract on Blockchain A. In some embodiments, the value of the initial function lock of the AMS contract on Blockchain A, e.g., H_N , may be used as the unique identifier that identifies the AMS contract on Blockchain A. The value of `Object` may represent an object that contains the parameters, including H_N , N , T_1 , and the action User 1 agrees to perform, recorded on Blockchain A. In an embodiment, the object may be defined as follows:

```

Object
{
    N; // maximum number of swaps agreed by both parties
    L; // function lock of the contract, with initial value set to  $H_N$ 
    T; // contract expiration time, which is set to  $T_1$  for Blockchain A
    Action; // action to be performed on Blockchain A
}

```

In this manner, User 1 can specify, using the object recorded on Blockchain A, that User 1 agrees to the action of transferring Asset X to User 2 on Blockchain A (as specified in Object.Action) if User 2 can unlock the function lock (as specified in Object.L) before the AMS contract expires on Blockchain A (as specified in Object.T). User 1 can further specify that User 1 agrees to allow this action to be performed up to N times (as specified in Object.N).

[0048] User 2 may setup the AMS contract on Blockchain B in a similar manner. For example, at step 312, User 2 may setup the AMS contract by requesting recordation on Blockchain B of parameters including, e.g., $H_N + \Delta_N$, which serves as the initial function lock of the AMS contract on Blockchain B, the maximum number N , the expiration time T_2 when the agreement expires on Blockchain B, and the action User 2 agrees to perform, i.e., transfer of Asset Y to User 1. In some embodiments, User 2 may perform step 312 after step 310. Because User 2 has an account on Blockchain A, User 2 is able to identify the AMS contract parameters User 1 recorded on Blockchain A, including the Contract_ID_A User 1 used to identify the AMS contract. Alternatively or additionally, User 1 may provide the Contract_ID_A to User 2 through another communication channel.

[0049] In some embodiments, User 2 may request the recordation of the parameters on Blockchain B by invoking a function, e.g., setupAMS(Contract_ID_B, Object), made available on Blockchain B through a smart contract interface. At step 314, Blockchain B may carry out the instructions specified in

setupAMS(Contract_ID_B, Object) and record the parameters specified by User 2 on Blockchain B.

[0050] In some embodiments, the value of Contract_ID_B specified in setupAMS(Contract_ID_B, Object) may be different from the value of Contract_ID_A to prevent a third-party from linking the contract identifiers recorded on Blockchains A and B together. For example, the value of Contract_ID_B specified in setupAMS(Contract_ID_B, Object) may represent a different unique identifier, e.g., a universally unique identifier, assigned or generated to identify the AMS contract on Blockchain B. In some embodiments, the value of the initial function lock of the AMS contract on Blockchain B, e.g., $H_N + \Delta_N$, may be used as the unique identifier that identifies the AMS contract on Blockchain B. The value of Object may represent an object that contains the parameters, including $H_N + \Delta_N$, N , T_2 , and the action User 2 agrees to perform on Blockchain B. In an embodiment, the object may be defined as follows:

```
Object
{
    N; // maximum number of swaps agreed by both parties
    L; // function lock of the contract, with initial value set to  $H_N + \Delta_N$ 
    T; // contract expiration time, which is set to  $T_2$  for Blockchain B
    Action; // action to be performed on Blockchain B
}
```

In this manner, User 2 can specify, using the object recorded on Blockchain B, that User 2 agrees to the action of transferring Asset Y to User 1 on Blockchain B (as specified in Object.Action) if User 1 can unlock the function lock (as specified in Object.L) before the AMS contract expires on Blockchain B (as specified in Object.T). User 2 can further specify that User 2 agrees to allow this action to be performed up to N times (as specified in Object.N).

[0051] In some embodiments, Users 1 and 2 may only need to carry out steps

308 through 314 once to setup the AMS contract. After the AMS contract is set up, Users 1 and 2 may execute the AMS contract to facilitate swap of Assets X and Y up to N times.

[0052] FIG. 4 illustrates a flow chart of a method 400 for executing an AMS contract, e.g., the AMS contract setup by Users 1 and 2, according to an embodiment. At step 402, User 1 may initiate a first swap by invoking a function on Blockchain B, e.g., `callAMS(Contract_ID_B, Key)`, to request transfer of Asset Y from User 2 to User 1 on Blockchain B. User 1 may use `Contract_ID_B` to identify the AMS contract setup by Users 1 and 2, and use `Key` to specify a key that can be used to unlock the function lock imposed by the AMS contract on Blockchain B. In some embodiments, the function lock imposed by the AMS contract on Blockchain B is set to $H_N + \Delta_N$ initially, which can be unlocked using $H_{N-1} + \Delta_{N-1}$.

[0053] At step 404, Blockchain B may carry out the instructions specified in `callAMS(Contract_ID_B, Key)`. These instructions can be illustrated using pseudo code defined below:

```
callAMS(Contract_ID_B, Key)
{
    Retrieve recorded Object from Blockchain B using Contract_ID_B;

    IF g(Key) == Object.L AND time < Object.T AND Object.N > 0
    {
        Perform Object.Action on Blockchain B;
        Set Object.L to Key;
        Reduce Object.N by 1;
    }
}
```

[0054] It is noted that because the function lock imposed by the AMS contract on Blockchain B is set to $H_N + \Delta_N$ initially, invoking `callAMS(Contract_ID_B, Key)` and setting `Key` to $H_{N-1} + \Delta_{N-1}$ will allow User 1 to unlock the initial function lock. Because User 1 calculated H_N and Δ_N based on a function chain (step 302, FIG. 3),

User 1 has the values of the entire chain, including $H_{N-1} = g(H_{N-2})$, $H_{N-2} = g(H_{N-3})$, ... H_0 and $\Delta_{N-1} = g(\Delta_{N-2})$, $\Delta_{N-2} = g(\Delta_{N-3})$, ... Δ_0 . Therefore, as long as User 1 invokes `callAMS(Contract_ID_B, Key)` before `Object.T` (which is T_2 because the function is invoked on Blockchain B), User 1 will be able to satisfy both the function lock and time lock imposed by the AMS contract. Blockchain B may then proceed to perform the recorded action (i.e., `Object.Action`, which is to transfer Asset Y from User 2 to User 1 on Blockchain B), update the function lock by setting `Object.L` to Key (i.e., setting `Object.L` to $H_{N-1} + \Delta_{N-1}$, which can be unlocked using the preceding value in the function chain, $H_{N-2} + \Delta_{N-2}$), and reduce the value of `Object.N` by one.

[0055] At step 406, User 2 may respond to the swap operation initiated by User 1 by invoking a function on Blockchain A, e.g., `callAMS(Contract_ID_A, Key)`, to request transfer of Asset X from User 1 to User 2 on Blockchain A. User 2 may use `Contract_ID_A` to identify the AMS contract setup by Users 1 and 2, and use Key to specify the key that can be used to unlock the function lock imposed by the AMS contract. In some embodiments, the function lock imposed by the AMS contract on Blockchain A is set to H_N initially, which can be unlocked using H_{N-1} .

[0056] User 2 may be able to calculate the value of H_{N-1} after step 404 because User 2 has an account on Blockchain B and can retrieve records from Blockchain B, which contains a record of User 1's invocation of `callAMS(Contract_ID_B, Key)` performed at step 404. Furthermore, because User 1 invoked `callAMS(Contract_ID_B, Key)` at step 404 by setting Key to $H_{N-1} + \Delta_{N-1}$, User 2 can retrieve the value of $H_{N-1} + \Delta_{N-1}$, which can be used to calculate the value of H_{N-1} by subtracting Δ_{N-1} from $H_{N-1} + \Delta_{N-1}$. The value of Δ_{N-1} may be readily available to User 2 based on calculations User 2 had already performed at

step 306 (FIG. 3). Alternatively, User 2 may calculate the value of Δ_{N-1} in real-time using the function chain as defined at step 306 (FIG. 3). In either case, User 2 can calculate $H_{N-1} = (H_{N-1} + \Delta_{N-1}) - \Delta_{N-1}$ and utilize the value of H_{N-1} to unlock the AMS contract on Blockchain A.

[0057] At step 408, Blockchain A may carry out the instructions specified in `callAMS(Contract_ID_A, Key)`. These instructions can be illustrated using pseudo code defined below:

```
callAMS(Contract_ID_A, Key)
{
    Retrieve recorded Object from Blockchain A using Contract_ID_A;

    If g(Key) == Object.L and time < Object.T and Object.N > 0
    {
        Perform Object.Action on Blockchain A;
        Set Object.L to Key;
        Reduce Object.N by 1;
    }
}
```

[0058] It is noted that because the function lock imposed by the AMS contract on Blockchain A is set to H_N initially, invoking `callAMS(Contract_ID_A, Key)` and setting Key to H_{N-1} will allow User 2 to unlock the initial function lock. Therefore, as long as User 2 invokes `callAMS(Contract_ID_A, Key)` before Object.T (which is T_1 because the function is invoked on Blockchain A), User 2 will be able to satisfy both the function lock and time lock imposed by the AMS contract. Blockchain A may then proceed to perform the recorded action (i.e., Object.Action, which is to transfer Asset X from User 1 to User 2 on Blockchain A), update the function lock by setting Object.L to Key (i.e., setting Object.H to H_{N-1} , which can be unlocked using H_{N-2}), and reduce the value of Object.N by one. At this point, User 1 is in possession of Asset Y on Blockchain B and User 2 is in possession of Asset X on Blockchain A. The first swap operation is considered complete.

[0059] The AMS contract may be executed again to carry out additional swap operations. For example, User 1 may initiate a second swap by invoking the callAMS(Contract_ID_B, Key) function at step 410, which may cause Blockchain B to carry out the instructions specified in callAMS(Contract_ID_B, Key) at step 412. It is noted that when User 1 invokes callAMS(Contract_ID_B, Key) for the second time, User 1 may set Key to $H_{N-2} + \Delta_{N-2}$, which is known to User 1 (calculated by User 1 at step 302, FIG. 3) and can be used by User 1 to unlock the updated function lock, Object.L on Blockchain B, because Object.L on Blockchain B has been set to $H_{N-1} + \Delta_{N-1}$ after the first swap. User 2 may retrieve the record of User 1's second invocation of callAMS(Contract_ID_B, Key) performed at step 412 and obtain the value of Key, which is set to $H_{N-2} + \Delta_{N-2}$ by User 1. In this manner, User 2 can calculate $H_{N-2} = (H_{N-2} + \Delta_{N-2}) - \Delta_{N-2}$ as the key to unlock the AMS contract on Blockchain A for the second time, allowing User 2 to respond by invoking the callAMS(Contract_ID_A, Key) function at step 414, which may cause Blockchain A to carry out the instructions specified in callAMS(Contract_ID_A, Key) at step 416.

[0060] In some embodiments, Users 1 and 2 may invoke the callAMS() function up to N times to perform up to N swaps. User 1 may set Key to $H_{N-n} + \Delta_{N-n}$ every time User 1 invokes callAMS(Contract_ID_B, Key), where n is the number of times callAMS(Contract_ID_B, Key) is being invoked. For example, the first time User 1 invokes callAMS(Contract_ID_B, Key), User 1 may set Key to $H_{N-1} + \Delta_{N-1}$, the second time User 1 invokes callAMS(Contract_ID_B, Key), User 1 may set Key to $H_{N-2} + \Delta_{N-2}$, the third time User 1 invokes callAMS(Contract_ID_B, Key), User 1 may set Key to $H_{N-3} + \Delta_{N-3}$, and so on. In this manner, User 1 can utilize values in the function chain $H_{N-1} = g(H_{N-2})$, $H_{N-2} = g(H_{N-3})$, ... H_0 and $\Delta_{N-1} = g(\Delta_{N-2})$, $\Delta_{N-2} = g(\Delta_{N-3})$, ... Δ_0 to unlock the AMS contract on Blockchain B

as User 1 continues to invoke callAMS(Contract_ID_B, Key) on Blockchain B. User 2 can calculate the values of $H_{N-1} = (H_{N-1} + \Delta_{N-1}) - \Delta_{N-1}$, $H_{N-2} = (H_{N-2} + \Delta_{N-2}) - \Delta_{N-2}$, ... $H_0 = (H_0 + \Delta_0) - \Delta_0$ and use the calculated values to unlock the AMS contract on Blockchain A as User 2 continues to invoke callAMS(Contract_ID_A, Key) on Blockchain A.

[0061] It is noted that because Blockchain A uses function locks expressed as $H_N, H_{N-1}, H_{N-2}, \dots H_0$ and Blockchain B uses function locks expressed as $H_N + \Delta_N, H_{N-1} + \Delta_{N-1}, H_{N-2} + \Delta_{N-2}, \dots H_0 + \Delta_0$, a third-party will not have the ability to link them together because the values of $\Delta_N, \Delta_{N-1}, \Delta_{N-2}, \dots \Delta_0$ are unknown to the third-party. Accordingly, the transactions carried out on Blockchains A and B in response to callAMS() can be concealed, thereby preserving privacy of Users 1 and 2.

[0062] It is also noted that using function locks expressed as $H_N, H_{N-1}, H_{N-2}, \dots H_0$ on Blockchain A and using function locks expressed as $H_N + \Delta_N, H_{N-1} + \Delta_{N-1}, H_{N-2} + \Delta_{N-2}, \dots H_0 + \Delta_0$ on Blockchain B are merely presented as examples and are not meant to be limiting. It is contemplated that function locks expressed as $H_N, H_{N-1}, H_{N-2}, \dots H_0$ may be used on Blockchain B (instead of Blockchain A) and function locks expressed as $H_N + \Delta_N, H_{N-1} + \Delta_{N-1}, H_{N-2} + \Delta_{N-2}, \dots H_0 + \Delta_0$ may be used on Blockchain A (instead of Blockchain B) without compromising their abilities to preserve privacy of Users 1 and 2. For example, in some embodiments, User 1 may set $H_N + \Delta_N$ as the initial function lock of the AMS contract on Blockchain A at step 308 (FIG. 3) and User 2 may set H_N as the initial function lock of the AMS contract on Blockchain B at step 312 (FIG. 3). Subsequently, at step 402 (FIG. 2) User 1 may use H_{N-1} as Key when invoking callAMS(Contract_ID_B, Key) to unlock the function lock H_N on Blockchain B, and at step 406, User 2 may retrieve the value of H_{N-1} from User 1's invocation of

callAMS(Contract_ID_A, Key) and calculate the value of $H_{N-1} + \Delta_{N-1}$, which can be used by User 2 to unlock the function lock $H_N + \Delta_N$ on Blockchain A. Users 1 and 2 may invoke the callAMS() function up to N times to perform up to N swaps, as described above.

[0063] In some embodiments, Blockchains A and B may keep track of the number of times Users 1 and 2 are allowed to execute the swap operation specified in the AMS contract by reducing the value of N (which is recorded on the blockchains as Object.N) every time a swap operation is performed. In this manner, Blockchains A and B can prevent unauthorized execution of the swap operation once the value of N reaches 0. In some embodiments, Blockchains A and B may delete an object once its N value reaches 0.

[0064] In some embodiments, Blockchain B may only allow User 1 to perform the swap operation before the AMS contract expires on Blockchain B at time T_2 . Likewise, Blockchain A may only allow User 2 to perform the swap operation before the AMS contract expires on Blockchain A at time T_1 . In this manner, Blockchains A and B can prevent unauthorized execution of swap operations after the AMS contract is set to expire. In some embodiments where User 2 is the user who responds to the swap operation initiated by User 1, User 2 may request setting the AMS contract to expire on Blockchain A at time T_1 after T_2 , i.e., $T_2 < T_1$. In some embodiments, User 2 may verify whether T_2 is indeed set to be less than T_1 when User 2 receives the parameters from User 1 (FIG. 3 at step 304). If the T_2 is greater than or equal to T_1 , User 2 may send an error message to User 1.

[0065] In some embodiments, Users 1 and 2 may further specify the ordering of up to N number of swaps in the AMS contract. In other words, Users 1 and 2 may specify not only the number of swaps they agreed to perform, but also the ordering

of such swaps. For example, Users 1 and 2 may agree to swap Asset X1 with Asset Y1 first, followed by the swap of Asset X2 with Asset Y2, and so on. In some embodiments, the ordering may be specified as an ordered list in an object defined as follows:

```
Object
{
    N; // maximum number of swaps agreed by both parties
    L; // function lock of the contract
    T; // contract expiration time
    ActionArray; // an ordered list of actions to be performed
}
```

[0066] It is to be understood that the ordered list of actions may be executed in manners similar to that described above. However, instead of repeating executions of Object.Action up to N times, a blockchain that supports the execution of an ordered list of actions may utilize a smart contract to execute instructions similar to the pseudo code defined as follows:

```
callAMS(Contract_ID, Key)
{
    Retrieve recorded Object from Blockchain using Contract_ID;

    IF g(Key) == Object.L AND time < Object.T AND Object.N > 0
    {
        Perform Object.ActionArray[N-Object.N];
        Set Object.L to Key;
        Reduce Object.N by 1;
    }
}
```

[0067] As illustrated in the pseudo code above, the index to Object.ActionArray, [N-Object.N], may point to the first element of the array when callAMS() is invoked for the first time, allowing the blockchain to perform the action specified in Object.ActionArray[0]. Subsequently, the index [N-Object.N] may increase as the value of Object.N decreases, allowing the blockchain to perform subsequent actions specified in Object.ActionArray in an ordered manner.

[0068] It is to be understood that the objects defined above are presented as examples and are not meant to be limiting. It is to be understood that the parameters contained in these objects may be stored on the blockchains in various manners. For example, in some embodiments, the various parameters described above may be stored in formats such as key-value pairs or the like. In some embodiments, prefixes may also be utilized to help distinguish key-value pairs associated with different AMS contracts. It is also to be understood that the declarations of the functions and the pseudo code described above are presented as examples and are not meant to be limiting.

[0069] FIG. 5 illustrates a flow chart of a method 500 for executing an AMS contract, according to an embodiment. The method 500 may be performed by one or more nodes in a blockchain system, e.g., the nodes 102-110 in the blockchain system 100 (FIG. 1). The nodes 102-110 in the blockchain system 100 may perform operations on a blockchain, e.g., the blockchain 120 (FIG. 1). The blockchain 120 may be implemented as either Blockchain A or B in the examples described above.

[0070] At step 502, a node, e.g., the node 102, may record parameters for an AMS on the blockchain 120. Continuing with the examples described above, the node 102 may receive the parameters from a first user, e.g., User 1 (FIG. 3). The parameters may include a function lock, an expiration time, at least one action to be performed on the blockchain 120 when the AMS contract is executed, and a maximum number of times the AMS contract can be executed, which is also the maximum number of agreed swaps, prior to the expiration time. In this manner, User 1 can specify, using the parameters recorded on the blockchain 120, that User 1 agrees to let the at least one action to be performed on the blockchain 120 if a second user can unlock the function lock before the expiration time. User 1 can also

specify that User 1 agrees to allow the AMS contract to be executed up to N times.

[0071] At step 504, the node 102 may receive, from the second user, e.g., User 2 (FIG. 4), a first request to execute the AMS contract and a first key for unlocking the function lock of the AMS contract. In some embodiments, User 2 may submit the first request through a function call. For example, User 2 may call the function callAMS(Contract_ID_A, Key) described above (step 406 of FIG. 4).

[0072] At step 506, the node 102 may determine whether to process the first request based on one or more of the first key, the expiration time, and the maximum number of times the AMS contract can be executed. In some embodiments, the node 102 may apply a one-way function to the first key to obtain a first function value and determine whether the first function value matches the function lock. If the first function value does not match the function lock, the node 102 may determine not to process the first request. In some embodiments, the node 102 may determine whether the AMS contract has expired based on the expiration time. If the AMS contract has expired, the node 102 may determine not to process the first request. In some embodiments, the node 102 may determine whether the AMS contract has been executed the maximum number of times. If the AMS contract has been executed the maximum number of times, the node 102 may determine not to process the first request. In some embodiments, the node 102 may determine to process the first request only when it is determined that the first function value matches the function lock, the AMS contract has not expired, and the AMS contract has not been executed the maximum number of times.

[0073] If the node 102 determines to process the first request, the node 102 may proceed to step 508. At step 508, the node 102 may execute the at least one action defined in the AMS contract on the blockchain 120. The node 102 may also

update the function lock of the AMS contract with the first key so that the first key will serve as the function lock for a second request to execute the AMS contract. In this manner, the node 102 may repeat steps 504-508 again when the node 102 receives a second request from User 2. The node 102 may repeat steps 504-508 up to the maximum number of times that the AMS contract can be executed.

[0074] In some embodiments, the at least one action to be performed on the blockchain 120 may include at least one action to transfer an asset recorded on the blockchain 120 from the first user, e.g., User 1, to the second user, e.g., User 2.

[0075] In some embodiments, the at least one action to be performed on the blockchain may include an ordered list of actions to be performed on the blockchain 120. The node 102 may execute a first action listed in the ordered list of actions on the blockchain 120 when the node 102 determines to process the first request to execute the AMS contract. The node 102 may execute a second action listed in the ordered list of actions on the blockchain 120 when the node 102 determines to process the second request to execute the AMS contract. In this manner, the node 102 may allow the users to specify not only the maximum number of times the AMS contract is allowed to be executed, but also the ordering of the actions to be executed.

[0076] FIG. 6 is a block diagram of an AMS contract execution apparatus 600, according to an embodiment. The apparatus 600 may be an implementation of a software process, and may correspond to the method 500 (FIG. 5). Referring to FIG. 6, the apparatus 600 may include a receiving module 602, a recording module 604, a determination module 606, and an execution module 608.

[0077] The receiving module 602 may receive information from users of a blockchain. In some embodiments, the receiving module 602 may receive, from a

first user, parameters for an AMS contract. The parameters may include a function lock, an expiration time, at least one action to be performed on the blockchain when the AMS contract is executed, and a maximum number of times the AMS contract can be executed prior to the expiration time. The receiving module 602 may provide the received parameters to the recording module 604.

[0078] The recording module 604 may record the received parameters on the blockchain. The recording module 604 may record the received parameters in various formats. For example, the recording module 604 may create an object that contains the received parameters as properties of the object and record the object on the blockchain. The recording module 604 may also record the received parameters in other formats, including key-values pairs and the like.

[0079] In some embodiments, the receiving module 602 may also receive requests from users of the blockchain. In some embodiments, the receiving module 602 may receive, from the second user, a first request to execute the AMS contract and a first key for unlocking the function lock of the AMS contract. The receiving module 602 may provide the received request to the determination module 606.

[0080] The determination module 606 may determine whether to process the first request based on the first key, the expiration time, and the maximum number of times the AMS contract can be executed. In some embodiments, the determination module 606 may apply a one-way function to the first key to obtain a first function value and determine whether the first function value matches the function lock. If the first function value does not match the function lock, the determination module 606 may determine not to process the first request. In some embodiments, the determination module 606 may determine whether the AMS contract has expired based on the expiration time. If the AMS contract has expired, the determination

module 606 may determine not to process the first request. In some embodiments, the determination module 606 may determine whether the AMS contract has been executed the maximum number of times. If the AMS contract has been executed the maximum number of times, the determination module 606 may determine not to process the first request. In some embodiments, the determination module 606 may determine to process the first request only when it is determined that the first function value matches the function lock, the AMS contract has not expired, and the AMS contract has not been executed the maximum number of times. The determination module 606 may then provide the first request to the execution module 608.

[0081] The execution module 608 may execute the at least one action defined in the AMS contract on the blockchain. The execution module 608 may also update the function lock of the AMS contract with the first key so that the first key will serve as the function lock for a second request to execute the AMS contract. In this manner, if the second user submits a second request to the receiving module 602 to execute the AMS contract again, the receiving module 602 may provide the second request to the determination module 606, which may determine whether to provide the second request to the execution module 608 for execution.

[0082] FIG. 7 illustrates a flow chart of a method 700 for executing an AMS contract, according to an embodiment. The method 700 may be performed by users using one or more computing devices, e.g., the computing device 200 (FIG. 2). These computing devices may include nodes in one or more blockchain systems, but may also include devices outside of the networks forming the blockchain systems.

[0083] At step 702, a first user, e.g., User 1 (FIG. 3), may request a recordation of a first plurality of parameters for the AMS contract on a first

blockchain, e.g., Blockchain A (FIG. 3). The first plurality of parameters may include an initial value of a first function lock on the first blockchain, at least one action to be performed on the first blockchain when the AMS is executed, and a maximum number of times the AMS contract can be executed. The recordation of the first plurality of parameters may be detected by a second user, e.g., User 2 (FIG. 3).

[0084] At step 704, the second user may request a recordation of a second plurality of parameters for the AMS contract on a second blockchain, e.g., Blockchain B (FIG. 3). The second plurality of parameters may include an initial value of a second function lock on the second blockchain, at least one action to be performed on the second blockchain when the AMS contract is executed, and the maximum number of times the AMS contract can be executed.

[0085] In some embodiments, the initial value of the second function lock is different from the initial value of the first function lock. In some embodiments, the initial value of the first function lock may be set to H_N , which may be calculated by the first user based on a first successive application of a one-way function $g()$ to a first selected value, H_0 . The first user may, for example, apply the one-way function $g()$ N times to calculate $H_1 = g(H_0)$, $H_2 = g(H_1)$, ... $H_N = g(H_{N-1})$, where N is the maximum number of times the AMS contract can be executed. The initial value of the second function lock may be set to $H_N + \Delta_N$, which may be calculated by the first user based on H_N and a second successive application of the one-way function $g()$ to a second selected value, Δ_0 . The first user may, for example, apply the one-way function $g()$ N times to calculate $\Delta_1 = g(\Delta_0)$, $\Delta_2 = g(\Delta_1)$, ... $\Delta_N = g(\Delta_{N-1})$, and calculate the value of $H_N + \Delta_N$ accordingly. In some embodiments, the initial value of the first function lock may be set to $H_N + \Delta_N$ instead of H_N , and the initial value of the second function lock may be set to H_N instead of $H_N + \Delta_N$, as described above.

[0086] At step 706, the first user may request to execute the AMS contract on the second blockchain (FIG. 4, step 402). The first user may invoking a function on the second blockchain, e.g., $\text{callAMS}(\text{Contract_ID_B}, \text{Key})$, as described above. The first user may use Contract_ID_B to identify the AMS contract and use Key to specify a key that can be used to unlock the second function lock on the second blockchain. If the initial value of the second function lock is set to $H_N + \Delta_N$, for example, the first user may use $H_{N-1} + \Delta_{N-1}$ as key to unlock the second function lock. If the initial value of the second function lock is set to H_N , the first user may use H_{N-1} as key to unlock the second function lock. The invocation of the request to execute the AMS contract on the second blockchain may be detected by the second user.

[0087] At step 708, the second user may retrieve, from the request to execute the AMS contract on the second blockchain, the key used by the first user to unlock the second function lock on the second blockchain. At step 710, the second user may calculate, based on the retrieved key, a key that the second user may use to unlock the first function lock on the first blockchain (FIG. 4, step 406). For example, if the initial value of the first function lock is set to H_N and the initial value of the second function lock is set to $H_N + \Delta_N$, then the first user uses $H_{N-1} + \Delta_{N-1}$ as the key to invoke $\text{callAMS}(\text{Contract_ID_B}, \text{Key})$ on the second blockchain, which means that the second user can retrieve the value of $H_{N-1} + \Delta_{N-1}$ and subtract Δ_{N-1} from $H_{N-1} + \Delta_{N-1}$ to calculate the value of H_{N-1} . In this manner, the second user can calculate the key needed to unlock the first function lock on the first blockchain (which is set to H_N in this example). Similarly, if the initial value of the first function lock is set to $H_N + \Delta_N$ and the initial value of the second function lock is set to H_N , then the first user uses H_{N-1} as the key to invoke $\text{callAMS}(\text{Contract_ID_B}, \text{Key})$ on

the second blockchain, which means that the second user can retrieve the value of H_{N-1} and calculate the value of $H_{N-1} + \Delta_{N-1}$ accordingly. In this manner, the second user can calculate the key needed to unlock the first function lock on the first blockchain (which is set to $H_N + \Delta_N$ in this example).

[0088] At step 712, the second user may invoke a request to execute the AMS contract on the first blockchain, e.g., `callAMS(Contract_ID_A, Key)`, as described above. The second user may provide the key calculated above along with the request to execute the AMS contract on the first blockchain, and based on the descriptions above, the key calculated by the second user should allow the second user to successfully unlock the first function lock on the first blockchain.

[0089] In some embodiments, steps 702 through 712 may be repeated up to N times (FIG. 4, Swaps 2 through N). For instance, the first user may invoke a second request to execute the AMS contract on the second blockchain. The second user may detect this invocation and retrieve, from the second request, a second key used by the first user to unlock the second function lock on the second blockchain. The second user may then calculate the key needed for the second user to unlock the first function lock on the first blockchain and invoke a second request to execute the AMS contract on the first blockchain.

[0090] In some embodiments, the at least one action to be performed on the first blockchain may include at least one action to transfer a first asset recorded on the first blockchain from the first user to the second user. The at least one action to be performed on the second blockchain may include at least one action to transfer a second asset recorded on the second blockchain from the second user to the first user. In some embodiments, the at least one action to be performed on the first blockchain may further include a first ordered list of actions to be performed on the

first blockchain and the at least one action to be performed on the second blockchain may further include a second ordered list of actions to be performed on the second blockchain.

[0091] FIG. 8 is a block diagram of an AMS contract execution apparatus 800, according to an embodiment. The apparatus 800 may be an implementation of a software process, and may correspond to the method 700 (FIG. 7). Referring to FIG. 8, the apparatus 800 may include a detection module 802, a recording module 804, a retrieving module 806, a calculation module 808, and an execution module 810.

[0092] The detection module 802 may detect a recordation of a first plurality of parameters for the AMS contract on a first blockchain. The first plurality of parameters may include an initial value of a first function lock on the first blockchain, at least one action to be performed on the first blockchain when the AMS is executed, and a maximum number of times the AMS contract can be executed.

[0093] The recording module 804 may record a second plurality of parameters for the AMS contract on a second blockchain. The second plurality of parameters may include an initial value of a second function lock on the second blockchain, at least one action to be performed on the second blockchain when the AMS contract is executed, and the maximum number of times the AMS contract can be executed.

[0094] In some embodiments, the detection module 802 may also detect an invocation of a request to execute the AMS contract on the second blockchain. The retrieving module 806 may then retrieve, from the request to execute the AMS contract on the second blockchain, a retrieved key used to unlock the second function lock on the second blockchain. The calculation module 808 may calculate, based on the retrieved key, a calculated key for unlocking the first function lock on the first blockchain. The execution module 810 may invoke a request to execute the

AMS contract on the first blockchain with the calculated key.

[0095] 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 apparatuses 600 and 800 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.

[0096] For an implementation process of functions and roles of each module in the apparatuses 600 and 800, references can be made to corresponding steps in the above-described methods. Details are omitted here for simplicity.

[0097] 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.

[0098] 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.

[0099] 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).

[00100] 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.

[00101] 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.

[00102] 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.

[00103] 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 executing an anonymous multi-swap (AMS) contract between a first user and a second user, the method comprising:

detecting a recordation of a first plurality of parameters for the AMS contract on a first blockchain, the first plurality of parameters including: an initial value of a first function lock on the first blockchain, at least one action to be performed on the first blockchain when the AMS contract is executed, and a maximum number of times the AMS contract can be executed, the maximum number of times the AMS contract can be executed being greater than one;

recording a second plurality of parameters for the AMS contract on a second blockchain, the second plurality of parameters including: an initial value of a second function lock on the second blockchain, at least one action to be performed on the second blockchain when the AMS contract is executed, and the maximum number of times the AMS contract can be executed, the initial value of the second function lock being different from the initial value of the first function lock;

detecting an invocation of a request to execute the AMS contract on the second blockchain;

retrieving, from the request to execute the AMS contract on the second blockchain, a retrieved key used to unlock the second function lock on the second blockchain;

calculating, based on the retrieved key, a calculated key for unlocking the first function lock on the first blockchain; and

invoking a request to execute the AMS contract on the first blockchain with the calculated key.

2. A computer-implemented method for executing an anonymous multi-swap (AMS) contract between a first user and a second user, the method comprising:

recording, by the first user, a first plurality of parameters for the AMS contract on a first blockchain, the first plurality of parameters including: an initial value of a first function lock on the first blockchain, at least one action to be performed on the first blockchain when the AMS contract is executed, and a maximum number of times the AMS contract can be executed, the maximum number of times the AMS contract can be executed being greater than one;

recording, by the second user, a second plurality of parameters for the AMS contract on a second blockchain, the second plurality of parameters including: an initial value of a second function lock on the second blockchain, at least one action to be performed on the second blockchain when the AMS contract is executed, and the maximum number of times the AMS contract can be executed, the initial value of the second function lock being different from the initial value of the first function lock;

invoking, by the first user, of a request to execute the AMS contract on the second blockchain;

retrieving, by the second user, a key used by the first user to unlock the second function lock on the second blockchain;

calculating, by the second user, a key to be used by the second user to unlock the first function lock on the first blockchain; and

invoking, by the second user, a request to execute the AMS contract on the first blockchain with the calculated key.

3. The method of any preceding claim, wherein the at least one action to be performed on the first blockchain comprises at least one action to transfer a first asset recorded on the first blockchain from the first user to the second user, and the

at least one action to be performed on the second blockchain comprises at least one action to transfer a second asset recorded on the second blockchain from the second user to the first user.

4. The method of any preceding claim, wherein the at least one action to be performed on the first blockchain further comprises a first ordered list of actions to be performed on the first blockchain and the at least one action to be performed on the second blockchain further comprises a second ordered list of actions to be performed on the second blockchain.

5. The method of any preceding claim, wherein the initial value of the first function lock, H_N , is calculated based on a first successive application of a one-way function $g()$ to a first selected value, H_0 , the first successive application being applied N times to calculate $H_1 = g(H_0)$, $H_2 = g(H_1)$, ... $H_N = g(H_{N-1})$, where N is the maximum number of times the AMS contract can be executed.

6. The method of claim 5, wherein the initial value of the second function lock, $H_N + \Delta_N$, is calculated based on H_N and a second successive application of the one-way function $g()$ to a second selected value, Δ_0 , the second successive application being applied N times to calculate $\Delta_1 = g(\Delta_0)$, $\Delta_2 = g(\Delta_1)$, ... $\Delta_N = g(\Delta_{N-1})$.

7. The method of claim 6, further comprising:
receiving the second selected value, Δ_0 ;
successively applying the one-way function $g()$ to the second selected value, Δ_0 , to independently calculate $\Delta_1 = g(\Delta_0)$, $\Delta_2 = g(\Delta_1)$, ... $\Delta_N = g(\Delta_{N-1})$.

8. The method of claim 7, wherein the retrieved key used to unlock the second function lock on the second blockchain has a value of $H_{N-1} + \Delta_{N-1}$, and wherein the calculating, based on the retrieved key, the calculated key for unlocking the first function lock on the first blockchain further comprises:

calculating the calculated key by subtracting the independently calculated value of Δ_{N-1} from the retrieved key value of $H_{N-1} + \Delta_{N-1}$.

9. The method of claim 8, further comprising:

detecting an invocation of a second request to execute the AMS contract on the second blockchain;

retrieving, from the second request to execute the AMS contract on the second blockchain, a second retrieved key used to unlock the second function lock on the second blockchain, the second retrieved key having a value of $H_{N-2} + \Delta_{N-2}$;

calculating a second calculated key by subtracting the independently calculated value of Δ_{N-2} from the second retrieved key value of $H_{N-2} + \Delta_{N-2}$; and

invoking a second request to execute the AMS contract on the first blockchain with the second calculated key.

10. The method of any of claims 1 through 4, wherein the initial value of the second function lock, H_N , is calculated based on a first successive application of a one-way function $g()$ to a first selected value, H_0 , the first successive application being applied N times to calculate $H_1 = g(H_0)$, $H_2 = g(H_1)$, ... $H_N = g(H_{N-1})$, where N is the maximum number of times the AMS contract can be executed.

11. The method of claim 10, wherein the initial value of the first function lock, $H_N + \Delta_N$, is calculated based on H_N and a second successive application of the one-way function $g()$ to a second selected value, Δ_0 , the second successive application being applied N times to calculate $\Delta_1 = g(\Delta_0)$, $\Delta_2 = g(\Delta_1)$, ... $\Delta_N = g(\Delta_{N-1})$.

12. The method of claim 11, further comprising:

receiving the second selected value, Δ_0 ;

successively applying the one-way function $g()$ to the second selected value, Δ_0 , to independently calculate $\Delta_1 = g(\Delta_0)$, $\Delta_2 = g(\Delta_1)$, ... $\Delta_N = g(\Delta_{N-1})$.

13. The method of claim 12, wherein the retrieved key used to unlock the second function lock on the second blockchain has a value of H_{N-1} , and wherein the calculating, based on the retrieved key, the calculated key for unlocking the first function lock on the first blockchain further comprises:

calculating the calculated key by adding the independently calculated value of Δ_{N-1} to the retrieved key value of H_{N-1} .

14. The method of claim 13, further comprising:

detecting an invocation of a second request to execute the AMS contract on the second blockchain;

retrieving, from the second request to execute the AMS contract on the second blockchain, a second retrieved key used to unlock the second function lock on the second blockchain, the second retrieved key having a value of H_{N-2} ;

calculating a second calculated key by adding the independently calculated value of Δ_{N-2} to the second retrieved key value of H_{N-2} ; and

invoking a second request to execute the AMS contract on the first blockchain with the second calculated key.

15. A device for executing an anonymous multi-swap (AMS) contract, 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 14.
16. An apparatus for executing an anonymous multi-swap (AMS) contract, the apparatus comprising a plurality of modules for performing the method of any of claims 1 to 14.
17. 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 14.