METHOD AND SYSTEM FOR KEY RECOVERY USING GROUP STRUCTURE

CROSS-REFERENCES TO RELATED APPLICATIONS

[0001] This application claims the benefit of Greece Patent Application No. 20240100205 under 35 U.S.C. 119(a), filed March 20, 2024, which is herein incorporated by reference in its entirety for all purposes.

BACKGROUND

[0002] Private key management is important to digital wallet security, usability, and recoverability. A digital wallet's private keys are used to create digital signatures that prove ownership and authority over a digital wallet's contents. If a private key is lost or compromised, the contents under the digital wallet's control can be at risk of loss and theft.

[0003] Prior approaches to reducing the problems of private key loss have involved the use of multi-party computation (MPC), standard multi-signature policies, physical backups, smart contract rules, and as well as digital clouds. However current implementations of these technologies fall short across a variety of potential real-world scenarios.

[0004] For example, some prior approaches utilize a third party that can store a copy of the user's cryptographic keys. If the user loses their device that stores the cryptographic keys, then the user can reobtain their cryptographic keys from the third party. However, if the third party is compromised, then the user's account is compromised.

[0005] As another example, some prior approaches are susceptible to front-running attacks, where a malicious party can identify that a user has lost their private key (e.g., the malicious party steals the user's phone storing the private key) and request recovery of the private key prior to the user noticing that the private key is missing. In such a case, the malicious party can "front-run" and obtain the recovery key prior to the user.

[0006] Embodiments of the disclosure address this problem and other problems individually and collectively.

SUMMARY

[0007] Embodiments are related to methods and systems for private key management that enables a digital wallet or other application to remain secure despite the loss of a user device, key share, or full private key while also enabling a way for the digital wallet's owner to recover ownership of their digital wallet's contents.

[0008] One embodiment is related to a method comprising: generating, by a user device, a key recovery message comprising a public key or an identifier thereof associated with an account, the public key corresponding to a first private key associated with the user device; providing, by the user device, the key recovery message to a smart contract on a blockchain; receiving, by the user device, a recovery detected message from a server computer that monitors the smart contract; and after providing the key recovery message to the smart contract, generating, by the user device, a new private key to replace the first private key.

[0009] Another embodiment is related to a user device comprising: a processor; and a computer-readable medium coupled to the processor, the computer-readable medium comprising code executable by the processor for implementing a method comprising: generating a key recovery message comprising a public key or an identifier thereof associated with an account, the public key corresponding to a first private key associated with the user device; providing the key recovery message to a smart contract on a blockchain; receiving a recovery detected message from a server computer that monitors the smart contract; and after providing the key recovery message to the smart contract, generating a new private key to replace the first private key.

[0010] Another embodiment is related to a method comprising: detecting, by a server computer, a change to a smart contract on a blockchain, wherein the change is an inclusion of a key recovery message from a user device requesting recovery of a first private key; determining, by the server computer, whether or not to cancel key recovery; generating, by the server computer, a recovery detected message based

on the determining; and providing, by the server computer, the recovery detected message to the user device, wherein the user device generates a new private key to replace the first private key.

[0011] Further details regarding embodiments of the disclosure can be found in the Detailed Description and the Figures.

BRIEF DESCRIPTION OF THE DRAWINGS

[0012] FIG. 1 shows a block diagram of a key recovery system according to embodiments.

[0013] FIG. 2 shows a block diagram of components of a user device according to embodiments.

[0014] FIG. 3 shows a block diagram illustrating key shares according to embodiments.

[0015] FIG. 4 shows a block diagram of a blockchain according to embodiments of the present invention.

DETAILED DESCRIPTION

[0016] Prior to discussing embodiments of the disclosure, some terms can be described in further detail.

[0017] A “user device” may be a device that is operated by a user. Examples of user devices may include a mobile phone, a smart phone, a card, a personal digital assistant (PDA), a laptop computer, a desktop computer, a server computer, a vehicle such as an automobile, a thin-client device, a tablet PC, etc. Additionally, user devices may be any type of wearable technology device, such as a watch, earpiece, glasses, etc. The user device may include one or more processors capable of processing user input. The user device may also include one or more input sensors for receiving user input. As is known in the art, there are a variety of input sensors capable of detecting user input, such as accelerometers, cameras, microphones, etc. The user input obtained by the input sensors may be from a variety of data input types, including, but not limited to, audio data, visual data, or biometric data. The user device may comprise any electronic device that may be

operated by a user, which may also provide remote communication capabilities to a network. Examples of remote communication capabilities include using a mobile phone (wireless) network, wireless data network (e.g., 3G, 4G or similar networks), Wi-Fi, Wi-Max, or any other communication medium that may provide access to a network such as the Internet or a private network.

[0018] A “user” may include an individual. In some embodiments, a user may be associated with one or more personal accounts and/or mobile devices. The user may also be referred to as a cardholder, account holder, or consumer in some embodiments.

[0019] A “user identifier” can include any piece of data that can identify a user. A user identifier can comprise any suitable alphanumeric string of characters. In some embodiments, the user identifier may be derived from user identifying information. In some embodiments, a user identifier can include an account identifier associated with the user.

[0020] A "blockchain" can be a distributed database that maintains a continuously growing list of records secured from tampering and revision. A blockchain may include a number of blocks of interaction records. Each block in the blockchain can contain also include a timestamp and a link to a previous block. Stated differently, interaction records in a blockchain may be stored as a series of "blocks," or permanent files that include a record of a number of interactions occurring over a given period of time. Blocks may be appended to a blockchain by an appropriate node after it completes the block and the block is validated. Each block can be associated with a block header. In embodiments of the invention, a blockchain may be distributed, and a copy of the blockchain may be maintained at each full node in a verification network. Any node within the verification network may subsequently use the blockchain to verify interactions.

[0021] A “block header” can be a header including information regarding a block. A block header can be used to identify a particular block in a blockchain. A block header can comprise any suitable information, such as a previous hash, a Merkle root, a timestamp, and a nonce. In some embodiments, a block header can also include a difficulty value.

[0022] A “smart contract” can include distributed computer executable program code. A smart contract may contain a set of rules under which the parties to the smart contract agree to interact with each other. If and when the pre-defined rules are met, the program code is executed and the agreement implemented in the program code is automatically enforced. The smart contract code can be executed in a blockchain network to provide a decentralized automated mechanism to facilitate, verify, and enforce negotiation or performance of an agreement or transaction.

[0023] A “key” may include a piece of information that is used in a cryptographic algorithm to transform input data into another representation. A cryptographic algorithm can be an encryption algorithm that transforms original data into an alternate representation, or a decryption algorithm that transforms encrypted information back to the original data. Examples of cryptographic algorithms may include triple data encryption standard (TDES), data encryption standard (DES), advanced encryption standard (AES), etc.

[0024] A “public key” may include an encryption key that may be shared openly and publicly. The public key may be designed to be shared and may be configured such that any information encrypted with the public key may only be decrypted using a private key associated with the public key (i.e., a public/private key pair).

[0025] A “private key” may include any encryption key that may be protected and secure. A private key may be securely stored at an entity and may be used to decrypt any information that has been encrypted with an associated public key of a public/private key pair associated with the private key.

[0026] A “public/private key pair” may include a pair of linked cryptographic keys generated by an entity. The public key may be used for public functions such as encrypting a message to send to the entity or for verifying a digital signature which was supposedly made by the entity. The private key, on the other hand may be used for private functions such as decrypting a received message or applying a digital signature. In some embodiments, the public key may be authorized by a body known as a Certification Authority (CA) which stores the public key in a database and distributes it to any other entity which requests it. The private key can typically be kept in a secure storage medium and will usually only be known to the entity. Public

and private keys may be in any suitable format, including those based on Rivest-Shamir-Adleman (RSA) or elliptic curve cryptography (ECC).

[0027] A “key share” can include a part of a key. A key can be split into a plurality of key shares. A plurality of key shares can be used to construct a key. A key share can be used for cryptographic purposes. For example, a private key share can be used to digitally sign data.

[0028] A “commitment” may include cryptographic primitive that allows one to commit to a chosen value (or chosen statement) while keeping it hidden to others, with the ability to reveal the committed value later. Commitment schemes are designed so that a party cannot change the value or statement after they have committed to it: that is, commitment schemes are binding.

[0029] A “secure element” can include a component that can perform a function securely. A secure element may be a memory that securely stores data, such that its access is protected. An example of a “secure element” is a trusted execution environment (TEE), a secure area of a processor. Another example of a secure element is a Universal integrated-circuit card (UICC), a secure smart card. Yet another example of a secure element is an embedded secure element, an embedded hardware component in a larger mechanical or electrical system. Another example of a secure element is a hardware security module (HSM), a physical computing device that can safeguard and manage cryptographic keys for authentication and provide crypto-processing functions.

[0030] A “processor” may include a device that processes something. In some embodiments, a processor can include any suitable data computation device or devices. A processor may comprise one or more microprocessors working together to accomplish a desired function. The processor may include a CPU comprising at least one high-speed data processor adequate to execute program components for executing user and/or system-generated requests. The CPU may be a microprocessor such as AMD's Athlon, Duron and/or Opteron; IBM and/or Motorola's PowerPC; IBM's and Sony's Cell processor; Intel's Celeron, Itanium, Pentium, Xeon, and/or XScale; and/or the like processor(s).

[0031] A "memory" may be any suitable device or devices that can store electronic data. A suitable memory may comprise a non-transitory computer readable medium that stores instructions that can be executed by a processor to implement a desired method. Examples of memories may comprise one or more memory chips, disk drives, etc. Such memories may operate using any suitable electrical, optical, and/or magnetic mode of operation.

[0032] A "server computer" may include a powerful computer or cluster of computers. For example, the server computer can be a large mainframe, a minicomputer cluster, or a group of servers functioning as a unit. In one example, the server computer may be a database server coupled to a Web server. The server computer may comprise one or more computational apparatuses and may use any of a variety of computing structures, arrangements, and compilations for servicing the requests from one or more client computers.

[0033] Embodiments of the invention allow for private key management that enables a digital wallet (or other application) to remain secure despite the loss of a user device, key share, or full private key while also enabling a way for the digital wallet's owner to recover ownership of their digital wallet's contents.

[0034] A user and user device can be associated with a user account. The user account can be associated with private cryptographic keys. A first private key can be stored in a user device (e.g., within a secure element of the user's mobile phone). The second private key can be provided to the user separately from the first private key. The second private key can be securely written down, stored in a different device, stored in a cold storage device, etc. A third private key can be stored by a server computer associated with the user account.

[0035] In some embodiments, each private key can be formed of private key shares, where the private key is associated with the user account. The user device can perform interactions (e.g., transactions, access requests, etc.) using the private key shares in a k-out-of-3 signing process with a smart contract stored in a blockchain. Each interaction can dynamically have a threshold of a number of key shares needed to digitally sign data for the smart contract to complete the interaction (e.g., 1 private key share signature).

[0036] If the user loses access to the first private key or the second private key , then the user device can perform a key recovery process to obtain a replacement private key.

[0037] For example, the user device can generate a key recovery message. The key recovery message can include a public key or an identifier thereof associated with the user account. The public key can correspond to the first private key. The user device can then provide the key recovery message to the smart contract on the blockchain.

[0038] The server computer can monitor the smart contract for updates. The server computer can identify that the user device that provided the key recovery message to the smart contract. The server computer can generate and provide a recovery detected message to the user device.

[0039] After providing the key recovery message to the smart contract and/or after receiving the recovery detected message, the user device can generate a new private key to replace the first private key.

[0040] In embodiments of the invention, the user device can utilize key generation algorithms used in group signature schemes to generate new private keys to replace lost private keys while maintaining the same corresponding public key.

[0041] Group signature schemes are signature schemes associated with some set of parties called a group. Members of the group (either individually, or based on some group structure) are able to produce signatures on messages that can be verified as being generated by the group, using what is known as the group public key, or group verification key. Members generate signatures using their own group member signing keys, or group secret key (gsk). In embodiments of the invention, the user of the user device can have access to two group secret keys (e.g., a first private key and a second private key). A server computer can have access to a group secret key (e.g., a third private key). The group secret keys can be used to sign data for the members of the group, while the smart contract of the blockchain network can hold the group verification key to verify the signatures produced by the devices in the group.

[0042] In one example, the user registers their user devices, including a mobile device and smart watch, with the smart contract. The user devices may use a cryptographic key generation protocol to generate a first private key sk1 for the mobile device and to generate a second private key sk2 for the smart watch. Both the first private key sk1 and the second private key sk2 may be generated such that they can be authenticated by a group verification key gvk. A user device that generates or obtains a private key may securely store the private key and may ensure that the private key does not leave the user device.

[0043] Embodiments of the invention include dynamic group signature schemes that allow for a group structure to change with time. In embodiments of the invention, it is possible to add new members to the group, as well as revoke signing privileges of a user device or user (either individually or as a part of the group structure) and their membership within the group, without the actions of the smart contract. Further details regarding group signatures can be found in PCT Publication WO 2023/003582, published on January, 26 2023, which is incorporated by reference.

[0044] Embodiments of the invention can utilize the concept of expanding the space of signing keys. Embodiments of the invention can include many functionally equivalent signing keys for any given verification key. Embodiments of the invention can also have the property that given any one signing key corresponds to a verification key, it is possible to generate another random functionally equivalent signing key. To do so, the following generic ideal functionality in a re-randomizable function $\mathcal{F}_{\mathrm{ReRand}}$ can be described with respect to the pseudocode below.

$\mathcal{F}_{\mathrm{ReRand}}$

Device A has (vk, sk_A) ← Σ.KeyGen (1^λ) for a signature scheme Σ.

Device B wants a random signing key sk_B corresponding to vk.

Inputs: pub = vk, $inp_A = sk_A$, $inp_B = \perp$

Outputs: $outp_B = sk_B$ uniformly distributed subject to the constraint that, for all *m*, *r*:

Σ.Ver (*m*, Σ.Sign (*m*, sk_B; *r*), vk) = 1.

[0045] The pseudocode explains a scenario where a device A generates a new key for a device B. The device A would already have a verification key (vk) and a unique secret key of the device A (sk_A), which may have been generated using a key generation protocol (KeyGen). The device A would use its first secret key (sk_A) to generate a random signing key or secret key of the device B (sk_B) corresponding to the verification key (vk).

[0046] Embodiments of the invention can include modified key generation processes $\mathcal{F}_{\text{ReRand}-\text{Schnorr}}$ and $\mathcal{F}_{\text{ReRand}-\text{ElGamal}}$. These can be based on the Schnorr and El Gamal signature schemes, respectively. Further details can be found in C. Schnorr, "Efficient identification and signatures for smart cards," in *Advances in Cryptology – CRYPTO '89, 9th Annual International Cryptology Conference*, Santa Barbara, California, USA, August 20-24, 1989, Proceedings, ser. Lecture Notes in Computer Science, G. Brassard, Ed., vol. 435. Springer, 1989, pp. 239–252, and WO 2023/003582, which are incorporated by reference.

[0047] The following $\mathcal{F}_{\text{ReRand}-\text{Schnorr}}$ protocol can be used to generate a secret key for the new device B (sk_B) using the secret key for the device A (sk_A). The inputs can include the following:
$\mathbb{G}$ (a group), q(prime order), M(scaling parameter), and vk (verication key).

$\mathcal{F}_{\text{ReRand}-\text{Schnorr}^*}$

A has a key-pair (vk, sk_A) ← $\Sigma^*_{\text{Schnorr}}$. KeyGen($1^\lambda$)

B wants a random signing key sk_B corresponding to vk.

Inputs: pub = ($\mathbb{G}, q, M,$ vk), inp_A = sk_A, inp_B = $\perp$

Protocol:

1) A computes and sends x = (sk_A mod q) to B.
2) B samples random $y \leftarrow \mathbb{Z}_M$.
3) B outputs sk_B = x + yq.

Outputs: $outp_A$ = $\perp$, $outp_B$ = sk_B

[0048] The second user device (B) provides a request to be added to a group (G) to a first user device (A), which is part of the group (G). The first user device (A) stores a first secret key (sk_A).

[0049] Upon receiving the request from the second user device B, the first user device A forms a value (x) using the equation $sk_A \bmod q$. The value (x) may be characterized as being an altered value of the first secret key value (sk_A). The value (x) is then sent to the second user device (B).

[0050] Upon receiving the value (x) from the first user device A, the second user device (B) generates a nonce (y) based on the scaling parameter (M). The second user device (B) uses this nonce (y), along with the modulus value (q) and the value (x) to generate a second secret key (sk_B) using the equation $sk_B = x + yq$.

[0051] The second user device (B) stores the second secret key (sk_B), and the second user device (B) is then added as a member of the group. The first secret key (sk_A) and the second secret key (sk_B) are different and can sign the same data, which can be verified by the same group verification key (vk).

[0052] The following key generation function, $\mathcal{F}_{\mathrm{ReRand-ElGamal}}$, is based on the El Gamal process, and generates a new second secret key (sk_B) for a new second user device B using a first secret key (sk_A) for a first user device A.

$\mathcal{F}_{\mathbf{ReRand-ElGamal}^*}$

A has a key-pair (vk, sk_A) $\leftarrow \Sigma^*_{\mathrm{ElGamal}}.\mathrm{KeyGen}(1^\lambda)$

B wants a random signing key sk_B corresponding to vk.

Inputs: pub = $(\mathbb{G}, q, M,$ vk) $inp_A = sk_A$, $inp_B = \perp$

Protocol:

1) A computes and sends $x = sk_A \bmod (q - 1)$ to B.
2) B samples random $y \leftarrow \mathbb{Z}_M$.
3) B outputs $sk_B = x + y(q - 1)$.

Outputs: $outp_A = \perp$, $outp_B = sk_B$

[0053] The pseudocode above shows a process for adding a new second secret key (sk_B) for a second user device B using a first secret key (sk_A) of a first user device A.

[0054] FIG. 1 shows a system 100 according to embodiments of the disclosure. The system 100 comprises a user device 102, a server computer 106, and a blockchain network 110. The user device 102 comprises a secure element

104. The server computer 106 comprises a memory 108. The blockchain network 110 comprises a smart contract 112. The user device 102 can be in operative communication with the server computer 106 and the blockchain network 110. The server computer 106 can be in operative communication with the user device 102 and the blockchain network 110.

[0055] For simplicity of illustration, a certain number of components are shown in FIG. 1. It is understood, however, that embodiments of the invention may include more than one of each component. In addition, some embodiments of the invention may include fewer than or greater than all of the components shown in FIG. 1.

[0056] Messages between the devices included in the system 100 illustrated in FIG. 1 can be transmitted using a secure communications protocols such as, but not limited to, File Transfer Protocol (FTP); HyperText Transfer Protocol (HTTP); Secure Hypertext Transfer Protocol (HTTPS), SSL, ISO (e.g., ISO 8583) and/or the like. The communications network may include any one and/or the combination of the following: a direct interconnection; the Internet; a Local Area Network (LAN); a Metropolitan Area Network (MAN); an Operating Missions as Nodes on the Internet (OMNI); a secured custom connection; a Wide Area Network (WAN); a wireless network (e.g., employing protocols such as, but not limited to a Wireless Application Protocol (WAP), I-mode, and/or the like); and/or the like. The communications network can use any suitable communications protocol to generate one or more secure communication channels. A communications channel may, in some instances, comprise a secure communication channel, which may be established in any known manner, such as through the use of mutual authentication and a session key, and establishment of a Secure Socket Layer (SSL) session.

[0057] The user device 102 can include a device that is operated by a user. Examples of user devices may include a mobile phone, a smart phone, a laptop computer, a desktop computer, etc. The user device 102 can be process interactions and can digitally sign data in the smart contract 112 in the blockchain network 110.

[0058] The user device 102 can include the secure element 104. The secure element 104 can securely store data. The secure element 104 can store cryptographic keys, such as a first private key. The first private key can be associated with a user account.

[0059] Embodiments of the invention can include a number of protocols including managerial functions. Such functions can include key generation (KeyGen), add member (AddMember), signature (Sign), signature open (SigOpen), revoke member (RevokeMember), and verification (Ver). It is assumed that only the members (e.g., devices and/or authenticators) in the managerless group can use these protocols. These functions can be performed by the user device 102, the server computer, 106, or the smart contract 112. Further details regarding these functions can be found in PCT Publication WO 2023/003582, published on January, 26 2023, which is incorporated by reference.

[0060] When adding devices for the first time, an enrollment process is performed. The enrollment process can be a process of signing up to the authentication protocol offered by the smart contract 112 and introducing the first authenticator(s) or first user device(s) (e.g., the user device 102). For example, the (gsk_i gvk) key pair is generated on a user device 102 using the group signature KeyGen protocol. The user device 102 stores gsk_i securely and gvk is transferred to the smart contract. A list (Table) is created on the smart contract and a user device nickname for the user device 112 is added to the list. At the end of this phase, a round of authentication processing can take place to verify the enrollment. The authentication processing can be similar to enrolling in a FIDO (fast identity online) authentication service.

[0061] In some embodiments, KeyGen is a key generation protocol that determines a public group verification key (gvk), a public identity ledger, and private keys for each of the individual group members. It can be used when one or more devices first enroll into the group or when a new private key is generated. The key generation method is as follows:

- Sample pp←Com.Gen(1^λ), crs ← NIZK.CRSGen(1^λ), (vk,sk) ← Σ_{SR}.KeyGen(1^λ) and $(\mathrm{ek}, \mathrm{dk})$ ← Γ.KeyGen(1^λ).
- Sample random values r, ω and commitment com = Com$(r; \omega)$. Set gvk = (pp, crs, vk, ek, com, Table) where Table =⊥.
- For each $i \in [|G|]$,
    - Run Π_{ReRand} (playing the role of both parties) using input sk to obtain sk_i.

- Compute commitments $com_{i,1} = Com(r_i; \omega_i)$, $com_{i,2} = Com(\alpha_i; \cdot)$ using randomness r_i, ω_i, α_i. (The "$\cdot$" in $Com(\alpha_i; \cdot)$ can represent a random value which is not necessary for the operation of the KeyGen routine.)
- Compute ciphertext $ct_i = \Gamma.Enc((com_{i,1}, com_{i,2}); ek, \beta_i)$ using randomness β_i.
- Set
  - $gsk_i = (sk_i, r_i, \omega_i, com_{i,2}, \beta_i)$
  - $ik_i = (sk_i, dk, r, \omega)$
  - $ok_i = dk$
  - $rk_i = (r, \omega, r_i, \omega_i, com_{i,2}, \beta_i)$
- Set $Table_i = (ct_i, INIT, IN)$.
- Add $(\{i\}, add)$, $(\{i\}, sign)$, $(\{i\}, open)$ and $(\{i\}, revoke)$ to Π.

[0062] First, a user device in a group of user devices can run $\Gamma.KeyGen(1^\lambda)$ to obtain a public key ek and a secret key dk. Γ can be a CCA2-secure public encryption scheme. Then, the re-randomizable signature scheme $\Sigma_{SR}.KeyGen(1^\lambda)$ is run to obtain a verification key (vk) and a signing key (sk). A common reference string crs is then generated using $NIZK.CRSGen(1^\lambda)$, where NIZK is a non-interactive zero-knowledge argument, such as those described in ZKProof, "ZKProof Community Reference," Version 0.2, Ed. by D. Benarroch, L. T. A. N. Brandão, E. Tromer. Pub. by zkproof.org, Dec. 2019. After this, public parameters (pp) are sampled using $Com.Gen(1^\lambda)$.

[0063] The random values r, ω are then selected, and the commitment com is generated using r, ω ($com = Com(r; \omega)$). Then, a group verification key (gvk) is formed from the values (pp, crs, vk, ek, com, $Table$) where $Table = \perp$ (meaning that the Table is empty).

[0064] The protocol goes through each member device in the group (G) to enroll each device (e.g., the user device 102 and the server computer 106 can be enrolled in the group). The KeyGen protocol, which may be presented as software on the user device, uses the secret key (sk), generated previously along with the verification key (vk), in Π_{ReRand} to obtain a unique, random secret key (sk_i).

[0065] After obtaining the unique, random secret key (sk_i), commitments ($com_{i,1}$ and $com_{i,2}$) are computed using random values r_i, ω_i, α_i.

[0066] After computing the commitments ($com_{i,1}$ and $com_{i,2}$) and the unique secret key (sk_i), the user device computes the ciphertext (ct_i) using the encryption key (ek) and a random value (β_i).

[0067] Upon calculating the ciphertext (ct_i), as shown above, four private keys can be generated: group signing key (gsk_i), issuing key (ik_i), opening key (ok_i), and revocation key (rk_i).

[0068] The four private keys can have the following features. The group signing key (gsk_i) can be used to generate a signature on data (e.g., a challenge). The issuing key (ik_i) can be used to add new members to the group. The opening key (ok_i) can be used to identify a member or user device that generated a signature. The revocation key (rk_i) can be used to revoke members in the group.

[0069] After the generation of the ciphertext (ct_i), the member is enrolled in the system and is included in the table on the smart contract 112 with the ciphertext (ct_i). The enrollment is finalized when the user device 102 is added to the group structure (Π) to perform add, sign, open, and revoke actions.

[0070] AddMember is a protocol that adds a new member to the group and generates a private key for the new member of the group. After enrolling the first set of devices, this method can be used to add new devices to a group of devices.

[0071] To add additional user devices to a group, the user first runs a setup process to establish a connection between a first user device, which already exists in a group, and a new second user device that is not part of the group. On any of the previously enrolled user devices, the user selects services that the new device should be enrolled with and runs the AddMember protocol to add the new group member. The system can update the device list (Table) on the server computer during the first authentication attempt from the new user second device.

[0072] In some embodiments, the user may be in possession of a first private key and a second private key. If the first private key is lost, then the user can utilize the second private key to aid in generating a new private key to replace the lost first

private key. The user can add the second private key to the user device 102, where the second private key can be utilized to generate a new private key that adds the user device 102 as a new member. In some cases, the server computer 106 can store a third private key that and is a member of the group. The user device 102 can communicate with the server computer 106 to add the user device with a new private key to the group.

[0073] The pseudocode of AddMember protocol embodiment is as follows:

AddMember $(j; \text{ik}_i = (\text{sk}_i, \text{dk}, r, \omega))$:

- Parties i and j run Π_{ReRand} using sk_i as i's input with j obtaining sk_i. Party i also passes along (dk, r, ω) to party j.
- Party j computes commitment $\text{com}_{j,1} = \text{Com}(r_j; \omega_j)$ using random values r_j, ω_j.
- Party i computes commitment $\text{com}_{j,2} = \text{Com}(\alpha_j; r_{j,2})$ using random values $\alpha_j, r_{j,2}$.
- Parties i and j then exchange $\text{com}_{j,1}$ and $\text{com}_{j,2}$.
- Party j computes ciphertext $\text{ct}_j = \Gamma.\text{Enc}((\text{com}_{j,1}, \text{com}_{j,2}); \text{ek}, \beta_j)$ using randomness β_j. Sends (ct_j, β_j) to party i.
- Party i computes $\pi \leftarrow \text{NIZK.Prove}(\text{stmt}_{\text{add}}; [r_i, \omega_i, \text{com}_{i,2}, \beta_i, \text{com}_{j,1}, \alpha_j, r_{j,2}, \beta_j])$ where $\text{stmt}_{\text{add}} = (i, \text{ct}_j, \text{gvk})$ and the relation is defined by:

  $\exists\ [r_i, \omega_i, \text{com}_{i,2}, \beta_i, \text{com}_{j,1}, \alpha_j, r_{j,2}, \beta_j]$:

  - $\text{Table}_i = (\Gamma.\text{Enc}((\text{Com}(r_i; \omega_i), \text{com}_{j,2}); \text{ek}, \beta_i), \cdot, \text{IN})$
  - $\text{ct}_j = \Gamma.\text{Enc}((\text{com}_{j,1}, \text{Com}(\alpha_j; r_{j,2})); \text{ek}, \beta_j)$ and sets $\text{Table}_j = (ct_j, (i, \pi), \text{IN})$.
  - Party i adds j to G and $(\{j\}, \text{add}), (\{j\}, \text{sign}), (\{j\}, \text{open})$ and $(\{j\}, \text{revoke})$ to Π.
- Party j sets
  - $\text{gsk}_j = (\text{sk}_j, r_j, \omega_j, \text{com}_{j,2}, \beta_j)$
  - $\text{ik}_j = (\text{sk}_j, \text{dk}, r, \omega)$
  - $\text{ok}_j = \text{dk}$
  - $\text{rk}_j = (r, \omega, r_j, \omega_j, \text{com}_{j,2}, \beta_j)$

[0074] The server computer 106 can aid the user device 102 in digitally signing data in the smart contract 112. The server computer 106 can store another private key (e.g., a second private key) that is associated with the user account. In some embodiments, the server computer 106 can digitally sign data in the smart contract 112 in conjunction with the user device 102.

[0075] The AddMember protocol allows a member of a group of user devices to add a new user device to the group. This eliminates the need for the server computer to generate a new private key for a new user device that is to be added to the group. This has technical advantages including reducing the computational workload on a relying party computer, and reducing the need for the relying party computer to be online when a new device is added to the group. Further, greater system security is achieved, since the generation of keys for new devices is managed by different user devices in different groups. The compromise of the relying party computer could not cause the entire system to be compromised.

[0076] The server computer 106 and the user device 102 can sign data using the following signature protocol. Sign is a protocol that generates a signature for the members of the group. This signature, upon the verification by the smart contract, authenticates the user device. The algorithm for the sign protocol can be as follows:

$\mathrm{Sign}(m; \mathrm{gsk}_i = (\mathrm{sk}_i, r_i, \omega_i, \mathrm{com}_{i,2}, \beta_i))$:

- Generate signature $\sigma' \leftarrow \Sigma_{\mathrm{SR}}.\mathrm{Sign}(m; \mathrm{sk}_i)$.
- Compute ciphertext $\mathrm{ct} = \Gamma.\mathrm{Enc}([i, \sigma']; \mathrm{ek}, r)$ using randomness r.
- Compute proof $\pi \leftarrow \mathrm{NIZK.Prove}(\mathrm{stmt}_{\mathrm{sign}}; [i, \sigma', r, r_i, \omega_i, \mathrm{com}_{i,2}, \beta_i])$ where $\mathrm{stmt}_{\mathrm{sign}} = (m, \mathrm{ct}, \mathrm{gvk})$ and the relation is defined by $\exists\ [i, \sigma', r, r_i, \omega_i, \mathrm{com}_{i,2}, \beta_i]$:
  - $\Sigma_{\mathrm{SR}}.\mathrm{Ver}(m, \sigma'; \mathrm{vk}) = 1$
  - $\mathrm{ct} = \Gamma.\mathrm{Enc}([i, \sigma']; \mathrm{ek}, r)$
  - $\mathrm{Table}_i = (\Gamma.\mathrm{Enc}((\mathrm{Com}(r_i; \omega_i), \mathrm{com}_{i,2}); \mathrm{ek}, \beta_i), \cdot, \mathrm{IN})$ (The "$\cdot$" can represent a value which is not necessary for the operation of the Sign subroutine.)
- Set $\sigma = (\mathrm{ct}, \pi)$.

[0077] Where m is the message comprising the data to sign, (gsk_i) is a group private key, (sk_i) is a device's private key, (r) indicates randomness, ($com_{i,2}$) is a commitment, (β_i) is a random value, (ct) is a ciphertext, (π) is a proof, (ω_i) is a random value, (i) is an index, and (vk) is a verification key.

[0078] Using the secret key (sk_i), the device (i) (e.g., the user device) signs the message m, thereby generating a temporary signature (σ'). This temporary signature (σ') is encrypted using the encryption key (ek) and randomness (r), which can hide the identity of the device that created the signature. The encrypted signature (σ') produces a ciphertext (ct).

[0079] A statement $stmt_{sign}$ = (m; ct; gvk), which includes the ciphertext (ct), is verified using the zero knowledge proof (NIZK). The device then generates an official signature (σ) containing the ciphertext (ct) and the proof (π) upon the successful verification.

[0080] The server computer 106 can monitor the smart contract on the blockchain network 110. The server computer 106 can generate a recovery detected message that indicates that the server computer 106 has detected a recovery attempt for the user account associated with the user device. The recovery detected message can include a user account identifier. The server computer 106 can provide the recovery detected message to the user device 102 to notify the user of the recovery attempt.

[0081] In some embodiments, the user device 102 may not have initiated the recovery attempt. In such cases, the user device 102 can respond to the recovery detected message to notify the server computer 106 that the recovery attempt is fraudulent.

[0082] The blockchain network 110 can include a plurality of devices that can store blockchain data. The blockchain network 110 can include, for example, a plurality of full nodes that each stored the entirety of the blockchain. The plurality of full nodes can generate and store blocks on the blockchain. Each full node can be configured to store the blockchain in any suitable memory or database.

[0083] The blockchain maintained by the blockchain network 110 can store a smart contract 112. The smart contract 112 can include coded actions that execute

when predetermined conditions are satisfied. The computers in the blockchain network can execute the actions when the predetermined conditions have been met and verified. These actions can include releasing funds to the appropriate parties, registering a vehicle, sending notifications, issuing a ticket or pass, etc. The blockchain is then updated when the transaction is completed. As such the transaction cannot be changed. In some embodiments, devices that have been granted permission can see the results of the smart contract 112.

[0084] In some embodiments, the smart contract 112 can verify received signatures using the following exemplary verification protocol, or other verification protocol:

Ver$(m, \sigma = (\text{ct}, \pi); \text{gvk} = (\text{pp}, \text{crs}, \text{vk}, \text{ek}, \text{com}))$:

Output NIZK.Verify$(\pi, \text{stmt}_{\text{sign}} = (m, \text{ct}, \text{gvk}), \text{crs})$.

[0085] The verification protocol, uses as inputs, the signature (σ), the message that the signature is on (m), and the group verification key (gvk). It verifies the proof (π) to determine if the signature is valid. The smart contract 112 can maintain a group structure that indicates a list of private keys that correspond to a group verification key.

[0086] SigOpen is a protocol that determines the members of the group who generated the signature. The algorithm for the SigOpen protocol is as follows:

SigOpen$(m, \sigma = (\text{ct}, \pi); \text{ok}_i = \text{dk})$:

Output $\perp$ if Ver$(m, \sigma; \text{gvk}) = 0$. Else, compute plaintext $[i, \sigma'] = \Gamma.\text{Dec}(\text{ct}; \text{dk})$ and output i.

[0087] Methods can use the verification (Ver) protocol to make sure that the signature (σ) is valid or not by using the proof (π). After the verification of the signature (σ), the SigOpen protocol utilizes an opening key (ok), which is also a decryption key (dk), to decrypt the ciphertext (ct). By decrypting the ciphertext (ct), the protocol is able to identify the device, or the party i, that created the signature (σ).

[0088] In some embodiments, during an interaction (e.g., a transaction, an access request, etc.), the smart contract 112 may require a number of signatures be

present on interaction data provided to the smart contract 112. The signatures may need to be verified by a group verification key gvk that corresponds to the plurality of group private keys. The number of signatures needed can be a threshold k that is dynamically determined based on an interaction amount, interaction volume, interaction velocity, user settings, etc.

[0089] FIG. 2 shows a block diagram of user device 102 according to embodiments. The exemplary user device 102 may comprise a processor 204. The processor 204 may be coupled to a memory 202, a secure element 104, a network interface 206, a computer readable medium 208, input elements 210, and output elements 212. The computer readable medium 208 can comprise a signing module 208A, a recovery request module 208B, and a cryptographic key generation module 208C.

[0090] The memory 202 can be used to store data and code. For example, the memory 202 can store cryptographic keys, identifiers, messages, and/or other data. The memory 202 may be coupled to the processor 204 internally or externally (e.g., cloud based data storage), and may comprise any combination of volatile and/or non-volatile memory, such as RAM, DRAM, ROM, flash, or any other suitable memory device.

[0091] The computer readable medium 208 may comprise code, executable by the processor 204, for performing a method comprising: generating, by a user device, a key recovery message comprising a public key or an identifier thereof associated with an account, the public key corresponding to a first private key associated with the user device; providing, by the user device, the key recovery message to a smart contract on a blockchain; receiving, by the user device, a recovery detected message from a server computer that monitors the smart contract; and after providing the key recovery message to the smart contract, generating, by the user device, a third private key to replace the first private key.

[0092] The signing module 208A may comprise code or software, executable by the processor 204, for signing data using cryptographic keys. The signing module 208A, in conjunction with the processor 204 can sign interaction data using a private key stored in the secure element 104. In some embodiments, the signing module 208A can be located within the secure element 104.

[0093] The recovery request module 208B can include may comprise code or software, executable by the processor 204, for creating key recovery messages. The recovery request module 208B, in conjunction with the processor 204, can generate a key recovery message comprising an account identifier of the account.

[0094] The cryptographic key generation module 208 can include may comprise code or software, executable by the processor 204, for generating cryptographic keys. The cryptographic key generation module 208C, in conjunction with the processor 204 can generate private keys that correspond to the public key of the account. For example, the cryptographic key generation module 208C, in conjunction with the processor 204, can generate a private key using the AddMember function or the other key generation functions described above.

[0095] In some embodiments, the cryptographic key generation module 208C can be located within the secure element 104.

[0096] The network interface 206 may include an interface that can allow the user device 102 to communicate with external computers. The network interface 206 may enable the user device 102 to communicate data to and from another device (e.g., the server computer 106, the blockchain network 110, etc.). Some examples of the network interface 206 may include a modem, a physical network interface (such as an Ethernet card or other Network Interface Card (NIC)), a virtual network interface, a communications port, a Personal Computer Memory Card International Association (PCMCIA) slot and card, or the like. The wireless protocols enabled by the network interface 206 may include Wi-Fi™. Data transferred via the network interface 206 may be in the form of signals which may be electrical, electromagnetic, optical, or any other signal capable of being received by the external communications interface (collectively referred to as “electronic signals” or “electronic messages”). These electronic messages that may comprise data or instructions may be provided between the network interface 206 and other devices via a communications path or channel. As noted above, any suitable communication path or channel may be used such as, for instance, a wire or cable, fiber optics, a telephone line, a cellular link, a radio frequency (RF) link, a WAN or LAN network, the Internet, or any other suitable medium.

[0097] The input elements 210 may include any suitable device(s) capable of inputting data into the user device 102. Examples of input devices include buttons, touchscreens, touch pads, microphones, etc.

[0098] The one or more output elements 212 may comprise any suitable device(s) that may output data. Examples of output elements 212 may include display screens, speakers, data transmission devices, etc.

[0099] FIG. 3 shows a flow diagram illustrating a method according to embodiments. The method illustrated in FIG. 3 will be described in the context of a user that has lost access to a first private key that is associated with a user account.

[0100] The method illustrated in FIG. 3 provides for private cryptographic key recovery without involving a third party to generate a new private key. Rather, the user device 102 itself can generate the new private key.

[0101] In some embodiments, the method can utilize required collateral to disincentivize unauthorized account recovery requests. For example, a user can initiate an account recovery process, which invokes a timelock. This timelock provides time for an authorized user to dispute a recovery request (as notified by the server computer 106) if that request was unauthorized. If successfully disputed, the collateral deposited from the initiator (who is a malicious party) is confiscated.

[0102] The system can utilize a cryptographic group signature scheme which creates multiple private keys. A first private key is stored securely on the user device 102 as (sk1). A second private key (sk2) is stored in a location external to the user device (e.g., in a cold storage device, in a second user device, in a secure cloud backup service, etc.), and the third private key (sk3) is stored with the server computer 106. The three private keys are used within a group signature cryptographic scheme to create a single public key pk as well as a group structure Π.

[0103] The group structure Π can include information regarding the group verification key gvk that is a public key that is able to verify signatures created from private keys that are added to the group structure. Thus, the group verification key gvk can verify signatures created from a first private key, a second private key, and a third private key that are all added to the group structure.

[0104] In some embodiments, the group structure Π includes a static threshold parameter t (which is a value between one and the maximum number of keys or key shares). Therefore, the group structure Π can be replicated within the smart contract 112 as Π1, Π2, Π3, where Πt corresponds to a different threshold t. The smart contract's 112 logic dynamically chooses the group structure Πt (and therefore, t) depending on the conditions (e.g., interaction velocity, spending limits, etc.) at the time of interaction on-chain to determine how many signatures from the different private keys are needed to process the interaction.

[0105] At step 302, after losing access to a first private key (sk1) the user device 102 can generate a key recovery message. The key recovery message can indicate that a private key needs to be recovered for a particular account associated with the user. The key recovery message can comprise a public key or an identifier thereof associated with the account. The public key can correspond to the first private key (which has been lost).

[0106] In some embodiments, the key recovery message can also include an amount of collateral for the recovery process. The key recovery message can include a reference to a funding source for the collateral. For example, the key recovery message can include a reference to a second account associated with the user of the user device 102 and authorization from the user to transfer an amount of funding from the second account to the smart contract 112 as collateral.

[0107] At step 304, after generating the key recovery message, the user device 102 can provide the key recovery message to the blockchain network 110. Specifically, the user device 102 can provide the key recovery message to the smart contract 112 in the blockchain maintained by the blockchain network 110.

[0108] At step 306, after receiving the key recovery message, the blockchain network 110 can provide the key recovery message to the smart contract 112 maintained by the blockchain network 110. The smart contract 112 can store an indication that a recovery process has been attempted into the smart contract along with the key recovery message. The smart contract 112 can also obtain the collateral amount of funds from the funding source indicated in the key recovery message.

[0109] At step 308, the server computer 106 can continuously monitor the blockchain network 110 for updates to the smart contract 112. For example, the server computer 106 can check the smart contract 112 for updates every 1 second, 1 minute, 5 minutes, etc. The server computer 106 can identify changes to the smart contract 112 by looking at interactions/transactions being made with the contract that indicate that a recovery attempt has been initiated. The server computer 106 can identify if any cryptographic key recovery attempts have been made for accounts.

[0110] The server computer 106 can have some measure of authority over the user's account. For example, the server computer 106 may have issued or may maintain the account. The server computer 106 can store a private key associated with the account. The server computer 106 can store a third private key of the account, while the user may have access to the first private key.

[0111] At step 310, the server computer 106 can generate a recovery detected message. The recovery detected message can indicate that the server computer 106 has detected a key recovery attempt for a particular account. The recovery detected message can comprise an account identifier of the account associated with the key recovery attempt.

[0112] At step 312, after generating the recovery detected message, the server computer 106 can provide the recovery detected message to the user device 102. The server computer 106 can provide the recovery detected message to devices that are associated with the account. If the user had lost their original user device along with the original private key, then the user can update a user profile with an identifier of the new user device (e.g., the user device 102) with the server computer 106. As such, the server computer 106 can provide the recovery detected message and any other messages to the devices associated with the account, or to a server that is acting as a payment guardian/monitoring service.

[0113] At step 314, after receiving the recovery detected message, the user device 102 can determine whether or not the user device 102 initiated the key recovery process. If the user device 102 did initiate the key recovery process (e.g., at step 302), then the user device 102 can proceed to step 316.

[0114] If the user device 102 did not initiate the key recovery process, then the user device 102 can prompt the user of the user device 102 for input. The user device 102 can prompt the user to input whether or not the user initiated the key recovery process. For example, the user device 102 can prompt the user to input a selection of dispute recovery, cancel recovery, or accept recovery. If the user selects that they did indeed initiate the key recovery process (e.g., accept recovery), then the user device 102 can continue to step 316. If the user selects that they did not initiate the key recovery process (e.g., dispute recovery), then the user device 102 can respond to the server computer 106 with an indication that the recovery request is fraudulent.

[0115] In some embodiments, if the user does not recognize the recovery request, then the user device 102 can generate an interaction disputed message that can be signed using the first private key (which can still be the original private key). The user device 102 can provide the interaction displayed message to the smart contract 112 in the blockchain network 110 directly or via the server computer 106. In this case, the collateral amount can be deposited to the user's account.

[0116] In some embodiments, if the key recovery request is legitimate, then the user device 102 can wait for a predetermined period of time (e.g., 5 minutes, 10 minutes, 1 hour, etc.), after which the smart contract 112 of the blockchain network performs key replacement, as described in step 318. If the key recovery request is fraudulent, then the recovery may time out and not be processed after the predetermined period of time.

[0117] If the user finds the missing private key in the meantime, the user can utilize the user device 102 to invoke a cancel operation (e.g., cancel recovery), which can halt the recovery process and returns the collateral amount to the user. For example, the user device 102 can respond to the recovery detected message with an indication to cancel the recovery. The server computer 106 can notify the smart contract 112 to return the collateral to the funding source.

[0118] At step 316, the user device 102 can generate a new private key (e.g., a third private key) to replace the original private key (e.g., a first private key). The user device 102 can generate the new private key as described above. As such, the

new private key can be generated by a cryptographic key process that allows the new private key to be used with the public key.

[0119] At step 318, the user device 102 can generate a key update message. The key update message can include instructions to update the group structure to revoke the first private key (e.g., the missing private key) with a new private key. The key update message can include the public keys that correspond to the private keys. The key update message can be signed using the second private key known by the user and previously added to the user device 102. For example, the key update message m can be "1) Revoke key sk1, 2) Add key sk1'," where key sk1 is the first private key and key sk1' is the new private key. The user device 102 can generate a digital signature σ on the key update message m using the second private key sk2, for example, $\sigma = (m, sk2)$.

[0120] In some embodiments, the key update message may need two or other suitable number of signatures to be accepted by the smart contract 112. The user device 102 can sign the key update message m and provide the signature σ and the key update message m to the server computer 106 for the server computer 106 to sign the key update message m using the third private key sk3. As such, the signature σ can be formed from the second private key sk2 and the third private key sk3. For example, the signature σ can be formed as $\sigma = (m, sk2, sk3)$.

[0121] At step 320, the user device 102 can provide the key update message to the smart contract 112.

[0122] At step 322, the smart contract 112 of the blockchain network 110 can verify the signature σ on the key update message. If the signature is valid and the key update message is signed by the required number of private keys, then the smart contract 112 can update the stored group structure to add the new private key and remove the original first private key. As such, the smart contract 112 can revoke the private key member (e.g., the original private key) and can add the new private key member (e.g., the new private key) to the group structure.

[0123] In other embodiments, the user device 102 can generate the new private key prior to providing the key recovery message to the smart contract 112. For example, the user device can perform step 316 prior to step 302. The user

device 102 can then include the new private key as well as digital signature in the key recovery message that is sent to the smart contract 112. In such a situation the data that is included in the key update message can rather be included in the key recovery message during the initial communication with the smart contract. Doing so reduces the overall number of messages being sent between devices.

[0124] FIG. 4 shows a diagram of a blockchain according to embodiments of the present invention. FIG. 4 includes a blockchain 400 comprising a first block 410 and a second block 440. The blockchain 400 can include any suitable number of blocks (e.g., 10, 500, 2000, 500000, etc.).

[0125] Current blockchain technologies, such as Bitcoin and Ethereum, maintain an append-only ledger in a network. The ledger includes a list of blocks of transaction data, the blocks are cryptographically chained together. A block is created by a computationally intensive process called proof-of-work in which valid blocks need to demonstrate a sufficient “difficulty" (i.e., sufficient computation power to create on average). If there are more than one available chains of blocks, then network participants (i.e., nodes) can download all blocks in all chains and follow the chain which has the highest total difficulty. This mechanism guarantees that, eventually, the network will agree on a single and valid chain, see [Garay et al, The Bitcoin backbone protocol: Analysis and applications. In *Advances in Cryptology - EUROCRYPT 2015*, pages 281–310, 2015], [Bitcoin Website. bitcoin.org], and [Rafael Pass, Lior Seeman, and Abhi Shelat. Analysis of the blockchain protocol in asynchronous networks. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *Advances in Cryptology – EUROCRYPT 2017*, pages 643–673, Cham, 2017. Springer International Publishing.], which are all incorporated herein by reference for all purposes.

[0126] The blockchain 400 can create a history of data deposits, messages, or entries in a series of blocks where each block contains a mathematical summary, called a hash, of the previous block. This creates a chain where any changes made to a block will change that block's hash, which must be recomputed and stored in the next block. This changes the hash of the next block, which must also be recomputed and so on until the end of the chain.

[0127] Although the hash can be simple to compute, rules may be imposed, which require the value of the hash to be below a certain threshold value (i.e., a difficulty value). In addition, the hash is based on a type of mathematical function that is not reversible. One cannot predict what input can be used to produce the desired output. A valid hash is found by repeatedly adjusting a changeable value in the block, and recalculating the hash until it meets the validity requirements. The freely changeable value can be a nonce. The unpredictable nature of the hash considerably increases the difficulty of finding a nonce that produces a valid hash of the block.

[0128] As an example, the first block 410 can include a block header 420 and block entries 430. The block header 420 of the first block 410 can comprise a previous hash 412, a timestamp 414, a Merkle root 416, and a nonce 418.

[0129] The previous hash 412 can be a hash of the previous block's header. The previous hash 412 can be the result of a non-reversible mathematical computation using data from the previous block as the input. According to some embodiments, the computation used can include a SHA256 hash function. One of ordinary skill in the art would recognize that any suitable hash function could be used without departing from the spirit and scope of the present invention. The hash function can be designed so that any change to the data in the previous block results in an unpredictable change in the hash of that block. The previous hash 412 can be a link between blocks, chaining them together to form the blockchain 400.

[0130] When calculating the previous hash 412 for the previous block, a node can determine if the previous hash 412 can meet certain criteria defined by a difficulty value. In some embodiments, the difficulty value may include a number that the calculated hash must be less than. However, because the output of the hashing function is unpredictable, the output cannot be determined what input will result in an output that is less than the difficulty value before the hash is calculated. The nonce 418 can be used to vary the data content of the block, allowing for a large number of different outputs to be produced by the hash function in pursuit of an output that meets the difficulty value. This makes can make it computationally expensive to produce a valid block with a nonce 418 that produces a hash value meeting the criteria of the difficulty value.

[0131] The hash algorithms used for the previous hash 412 can include MD5, SHA-1, SHA-224, SHA-256, SHA-384, SHA-512, SHA-512/224, SHA-512/256, SHA-3 or any suitable hash function. There is also no requirement that a hash be computed only once. The results of a hash function may be reused as inputs into another or the same hash function again multiple times in order to produce a final result. One of ordinary skill in the art would recognize that any hash function could be used to compute the required hashing without departing from the spirit and scope of the present invention.

[0132] The Merkle root 416 can be a root of a Merkle tree, which can include a tree in which every leaf node is labelled with the hash of a data block, for example an entry. Each leaf of the Merkle tree can represent one of the entries. Each entry can be hashed together with a sibling node (i.e., entry) in the Merkle tree. Successively hashing sibling nodes in the Merkle tree can result in the Merkle root 416.

[0133] The block entries 430 can include interaction data and/or smart contracts. The block entries 430 can include any suitable number of entries. Example entries can include the data described above, such as witnesses, signed witnesses, digital signatures, interaction messages, unconditional payment data, proof of inclusions, other data related to the processing of an offline interaction between two user devices, etc. In some embodiments, an entry may be a null value, for example, in the case of the Merkle tree being a sparse Merkle tree.

[0134] In some embodiments, the number of entries in the block entries 430 may be limited by the overall size of the block (e.g., the first block 410). For example, the blocks on the blockchain may be limited by a certain amount of data (e.g., 1/2 MB, 1 MB, 2 MB, 5 MB, 10 MB, etc.). In other embodiments, the number of entries in the block entries 430 may be a predetermined number of entries. For example, the nodes in the verification network can determine that 1, 10, 150, 500, 1000, etc. entries can be included in the block entries 430.

[0135] The timestamp 414 can include a time that the block was created within a certain range of error. According to some embodiments of the present invention, the full nodes of the verification network can check the timestamp 414 against their own known time and can reject any block that seems to have an erroneous timestamp 414.

[0136] The nonce 418 can be a value adjusted by a full node while performing a proof-of-work process, as described herein. A nonce can be input into a hash function along with block data to determine the output hash value. A correct nonce (also referred to as a golden nonce) yields an output hash value that satisfies a predetermined criteria, such as being less than a difficulty value.

[0137] The second block 440 can be similar to the first block 410. For example, the second block 440 can include a block header 450 and block entries 460. The block header 450 of the second block 440 can comprise a previous hash 452, a timestamp 454, a Merkle root 456, and a nonce 458. The block entries 460 can include image data, interaction data, smart contracts, etc., and can be similar to the block entries 430.

[0138] Embodiments of the disclosure have a number of advantages. For example, embodiments eliminate the technical problem of front-running attacks. Embodiments are not susceptible to such attacks, because the threshold signature system gives the legitimate user the ability to dispute any attacker immediately by proving ownership of a second private key. In addition, the server computer monitors the smart contract and informs the user of any detected recovery attempts. This makes systems and methods according to embodiments much more efficient and secure than the prior cryptographic private key recovery systems.

[0139] Although the steps in the flowcharts and process flows described above are illustrated or described in a specific order, it is understood that embodiments of the invention may include methods that have the steps in different orders. In addition, steps may be omitted or added and may still be within embodiments of the invention.

[0140] Any of the software components or functions described in this application may be implemented as software code to be executed by a processor using any suitable computer language such as, for example, Java, C, C++, C#, Objective-C, Swift, or scripting language such as Perl or Python using, for example, conventional or object-oriented techniques. The software code may be stored as a series of instructions or commands on a computer readable medium for storage and/or transmission, suitable media include random access memory (RAM), a read only memory (ROM), a magnetic medium such as a hard-drive or a floppy disk, or an

optical medium such as a compact disk (CD) or DVD (digital versatile disk), flash memory, and the like. The computer readable medium may be any combination of such storage or transmission devices.

[0141] Such programs may also be encoded and transmitted using carrier signals adapted for transmission via wired, optical, and/or wireless networks conforming to a variety of protocols, including the Internet. As such, a computer readable medium according to an embodiment of the present invention may be created using a data signal encoded with such programs. Computer readable media encoded with the program code may be packaged with a compatible device or provided separately from other devices (e.g., via Internet download). Any such computer readable medium may reside on or within a single computer product (e.g., a hard drive, a CD, or an entire computer system), and may be present on or within different computer products within a system or network. A computer system may include a monitor, printer, or other suitable display for providing any of the results mentioned herein to a user.

[0142] The above description is illustrative and is not restrictive. Many variations of the invention will become apparent to those skilled in the art upon review of the disclosure. The scope of the invention should, therefore, be determined not with reference to the above description, but instead should be determined with reference to the pending claims along with their full scope or equivalents.

[0143] One or more features from any embodiment may be combined with one or more features of any other embodiment without departing from the scope of the invention.

[0144] As used herein, the use of "a," "an," or "the" is intended to mean "at least one," unless specifically indicated to the contrary.

WHAT IS CLAIMED IS:

1. A method comprising:

generating, by a user device, a key recovery message comprising a public key or an identifier thereof associated with an account, the public key corresponding to a first private key associated with the user device;

providing, by the user device, the key recovery message to a smart contract on a blockchain;

receiving, by the user device, a recovery detected message from a server computer that monitors the smart contract; and

after providing the key recovery message to the smart contract, generating, by the user device, a new private key to replace the first private key.

2. The method of claim 1, further comprising:

providing, by the user device, collateral to the smart contract with the key recovery message.

3. The method of claim 1, wherein the new private key is generated by a cryptographic key process that allows the new private key to be used with the public key.

4. The method of claim 1, further comprising:

after receiving the recovery detected message, prompting, by the user device, a user to input a selection of dispute recovery, cancel recovery, or accept recovery.

5. The method of claim 1, wherein the server computer determines whether or not to cancel key recovery.

6. The method of claim 1, wherein the first private key is a first private key share and the second private key is a second private key share, wherein the first private key share and the second private key share are shares of a private key that corresponds to the public key.

7. The method of claim 6, wherein the new private key is a new private key share to replace the first private key share.

8. The method of claim 1, wherein after generating the new private key, the method further comprises:

generating, by the user device, a key update message comprising the new private key and a request to revoke the first private key;

generating, by the user device, a digital signature on the key update message using a third private key obtained prior to generating the key recovery message; and

providing, by the user device, the key update message comprising the digital signature to the smart contract.

9. The method of claim 8, wherein the smart contract verifies the digital signature using a group public key, revokes the first private key, and adds the new private key.

10. The method of claim 1, wherein after generating the new private key, the method further comprises:

generating, by the user device, a key update message comprising the new private key and a request to revoke the first private key;

generating, by the user device, a digital signature on the key update message using a third private key obtained prior to generating the key recovery message; and

providing, by the user device, the key update message comprising the digital signature to the server computer.

11. The method of claim 10, wherein the server computer verifies the digital signature using a group public key and signs the key update message with the second private key.

12. The method of claim 11, wherein the server computer provides the key update message with the digital signature that is further signed by the server computer using the second private key to the user device, wherein the method further comprises:

providing, by the user device, the key update message with the digital signature that is further signed by the server computer to the smart contract.

13. A user device comprising:

a processor; and

a computer-readable medium coupled to the processor, the computer-readable medium comprising code executable by the processor for implementing a method comprising:

generating a key recovery message comprising a public key or an identifier thereof associated with an account, the public key corresponding to a first private key associated with the user device;

providing the key recovery message to a smart contract on a blockchain;

receiving a recovery detected message from a server computer that monitors the smart contract; and

after providing the key recovery message to the smart contract, generating a new private key to replace the first private key.

14. The user device of claim 13, wherein the smart contract maintains a group structure that indicates a list of private keys that correspond to a group verification key, wherein the server computer stores a second private key, and wherein transactions conducted using the blockchain need to be signed at least by the second private key to be valid.

15. The user device of claim 13, wherein prior to generating the key recovery message the method further comprises:

obtaining a third private key associated with the account, wherein the third private key is obtained from a device external to the user device and operated by user.

16. The user device of claim 15, wherein after generating the new private key, the method further comprises:

generating a key update message comprising the new private key and a request to revoke the first private key;

generating a digital signature on the key update message using the third private key; and

providing the key update message comprising the digital signature to the smart contract, wherein the smart contract verifies the digital signature using a group public key, revokes the first private key, and adds the new private key.

17. A method comprising:

detecting, by a server computer, a change to a smart contract on a blockchain, wherein the change is an inclusion of a key recovery message from a user device requesting recovery of a first private key;

determining, by the server computer, whether or not to cancel key recovery;

generating, by the server computer, a recovery detected message based on the determining; and

providing, by the server computer, the recovery detected message to the user device, wherein the user device generates a new private key to replace the first private key.

18. The method of claim 17, further comprising:

receiving, by the server computer, a key update message from the user device, wherein the key update message comprises the new private key and a digital signature formed by the user device using a third private key;

verifying, by the server computer, the digital signature using a group verification key; and

signing, by the server computer, the key update message using the second private key.

19. The method of claim 18, further comprising:

providing, by the server computer to the smart contract, the key update message that is signed by both the server computer using the second private key and by the user device using the third private key.

20. The method of claim 18, further comprising:

providing, by the server computer to the user device, the key update message that is signed by both the server computer using the second private key and by the user device using the third private key, wherein the user device provides the key update message to the smart contract.